; accepted Nov 16.

# MARHO: Hybrid Task Offloading in Maritime MEC via Multi-Agent Reinforcement Learning

**Jiahong Ning**[1,3] ***(Student Member, IEEE)***, **Aimin Li** [2] ***(Member, IEEE)***, **Gary C.F. Lee** [3] ***(Member, IEEE)***, **Sumei Sun** [3] ***(Fellow, IEEE)***, **Tingting Yang** [1,4] ***(Member, IEEE)***

[1]Dalian Maritime University
[2]Harbin Institute of Technology (Shenzhen)
[3]Institute for Infocomm Research (I²R), A*STAR
[4]Pengcheng Laboratory (Shenzhen)

CORRESPONDING AUTHOR: Tingting Yang (e-mail: yangtingting820523@163.com).

**ABSTRACT** This paper presents MARHO, a **M**ulti-**A**gent **R**einforcement learning-based **H**ybrid task **O**ffloading framework, designed for maritime mobile edge computing (MEC) environments characterized by time-varying wireless channels, heterogeneous workloads, and stringent quality of service (QoS) requirements. The considered MEC architecture integrates *unmanned surface vessels (USVs)*, *unmanned aerial vehicles (UAVs)*, and a *ship platform* with high-performance edge servers. USVs generate sensing and computing tasks that can be (i) executed locally, (ii) offloaded to UAVs for aerial edge processing, or (iii) relayed through UAVs to the ship under line-of-sight (LoS) links. The system model jointly captures queueing dynamics, wireless transmission latency, computation delay, and battery constraints. The hybrid offloading problem is formulated as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP), where each USV acts as an *agent* that decides its offloading mode under partial observations. To solve this, MARHO employs a centralized training and decentralized execution (CTDE) scheme, enabling agents to learn resource-aware strategies that effectively balance communication and computation. A Gym-based simulation environment is developed, integrating realistic maritime signal propagation, queue dynamics, and mixed offloading scenarios. The experimental results under different task loads demonstrate that MARHO consistently achieves higher throughput and has a lower average latency compared to the existing benchmark.

**INDEX TERMS** Mobile Edge Computing (MEC), Hybrid Offloading, Multi-Agent Reinforcement Learning (MARL), Quality of Service (QoS)

## I. INTRODUCTION

THE maritime environment, covering over 71% of the Earth's surface, is of strategic importance for resource exploration, ecological monitoring, and maritime security. Efficient data collection and processing are essential for harnessing this potential. However, the harsh and dynamic characteristics of the marine environment pose serious challenges to real-time perception and computation, necessitating a distributed and intelligent network of complementary nodes. Unmanned surface vessels (USVs) are commonly deployed for tasks such as water quality monitoring, patrolling, and inspection, but their functionality is constrained by limited computing and battery capacity. Unmanned aerial vehicles (UAVs) extend operational coverage and offer preliminary computation and relay services, yet they also face restrictions in energy and onboard resources. To address these limitations, high-performance computational nodes such as ships and offshore platforms are often introduced, providing abundant computing power and energy supply, though at the cost of higher deployment complexity and reduced mobility. These complementary yet resource-constrained roles make centralized cloud processing impractical at sea, calling for computation to be pushed closer to where data are produced.

This constellation of heterogeneous yet interdependent nodes highlights a core difficulty: unlike terrestrial networks, maritime links are highly dynamic, with frequent Line-of-Sight (LoS) or Non-Line-of-Sight (NLoS) transitions, long-range fading, and strong coupling between communication and energy states. Under such conditions, predefined task placements quickly become fragile. Purely centralized cloud strategies are infeasible, and rigidly binding tasks to a single node type is equally inadequate. What is fundamentally

required is a mechanism that can flexibly assign and migrate computation among USVs, UAVs, and ships, ensuring latency and completion targets are met under resource constraints.

MEC provides a principled way to reduce backhaul pressure and end-to-end latency by processing data near its source. Classical optimization approaches, such as alternating direction method of multipliers (ADMM) and block-coordinate descent (BCD), have been widely used for resource allocation and task scheduling [1]. But they typically rely on static models or predefined structures, that are brittle under rapid channel changes and strong multipath. The coupled, high-dimensional decision space also challenges problem decomposition and iterative solvers at scale. In addition, distributed coordination methods, including matching algorithms [2] and game theoretic formulations [3], often incur substantial message passing whose overhead grows with the fleet size, reducing effective throughput and aggravating queueing delays. These limitations motivate approaches that can orchestrate resources online and remain aligned with quality of service (QoS) objectives under non-stationary maritime dynamics.

Deep reinforcement learning (DRL) has gained traction in MEC because it enables real-time decision-making. In [4], DRL has been used to optimize task offloading together with power control, learning environment policies that reduce service delay and improve effective throughput. The multi-agent reinforcement learning (MARL) paradigm extends DRL to cooperative–competitive environments where multiple agents must share spectrum, energy, and compute resources under partial observability. In MEC, this formulation captures the coupling among offloading decisions, queue dynamics, and power control across distributed nodes. Actor–critic methods provide stable policy updates by combining policy learning with value estimation, making them suitable for latency-sensitive scheduling [5]. Building on this, the Deep Deterministic Policy Gradient (DDPG) algorithm introduces deterministic policy gradients for continuous control, which is effective for fine-grained resource allocation such as transmit power and bandwidth [6]. Further, the Multi-agent Deep Deterministic Policy Gradient (MADDPG) algorithm incorporates Centralized Training with Decentralized Execution (CTDE), alleviating non-stationarity and enabling stable coordination when agents interact and compete in shared MEC environments [7].

In particular, maritime MEC presents three challenges for resource orchestration: (i) non-stationarity, as mobility and fluctuating links continually alter interaction dynamics, complicating stable learning. (ii) Hybrid, high-dimensional actions, since discrete offloading modes must be decided jointly with continuous power allocations. And (iii) partial observability with queuing effects, because each node relies on local measurements while the sojourn time depends on coupled processing backlogs. These challenges necessitate a problem formulation that explicitly incorporates hybrid offloading modes, queue-aware QoS constraints, and decentralized execution under uncertainty.

Compared with existing MARL-based MEC methods that often assume full observability, purely discrete or purely continuous actions, and ignore queue coupling. Our framework models the problem as a Dec-POMDP with hybrid discrete–continuous actions and adopts queue-aware CTDE. This combination targets UAVs, USVs, and ship collaboration under dynamic topology and heterogeneous resources. This formulation leads to a hybrid offloading mechanism that jointly selects modes and allocates power and bandwidth.

In this paper, we propose a novel approach named **M**ulti-**A**gent **R**einforcement learning **H**ybrid task **O**ffloading (MARHO) framework. The main contributions of our work are summarized as follows:

- **System Model:** We design a hybrid maritime edge computing model that integrates UAVs, USVs, and ship-based platforms. The model explicitly captures the coupling between channel dynamics, queue evolution, and resource limitations, and reveals how these interactions affect perception, learning, and QoS evaluation.
- **Solutions and Algorithms:** We develop MARHO, a queue-aware DRL framework for the maritime MEC problem. The problem is cast as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) with hybrid actions that couple discrete offloading mode selection with continuous power and bandwidth allocation. We adopt CTDE for decentralized execution on USVs with QoS by maximizing the number of completed tasks under an average-delay budget. The framework provides an interpretable decomposition of sojourn time into transmission, queuing, and computation under realistic LoS/NLoS channels and resource constraints.
- **Experiments and Insights:** We develop a comprehensive experimental framework that integrates key components of the maritime MEC setting, including wireless transmission under LoS/NLoS conditions, queue dynamics, and hybrid task offloading mechanisms. MARHO is evaluated against several benchmark approaches under diverse conditions, including varying meteorological scenarios, task loads, field-of-view (FoV) ranges, and fleet sizes. Experimental results demonstrate that MARHO consistently achieves higher throughput while maintaining comparable or lower average sojourn time. Moreover, the end-to-end delay remains well-controlled, even in highly dynamic environments. These findings highlight the scalability and robustness of MARHO for real-world maritime edge computing applications.

The rest of this paper is organized as follows. Section II reviews related work on resource orchestration. Section III presents the system model, including the maritime channel, task structure, and offloading modes. Section IV details the MARHO formulation. Section V reports simulation results

and analysis, and section VI concludes the paper and outlines future directions.

## II. Related Work

### A. Mobile Edge Computing in Marine

Research on maritime MEC can be broadly classified into static optimization, heuristic algorithms, and simulation-based methods, with more recent works reporting QoS improvements in practical deployments.

Early studies aimed to enhance performance under the assumption of static conditions. By using relay-assisted design [8], the operability of the network and the effective service capacity could be improved. Iterative schemes could reduce link delay [9], and the joint trajectory-power strategy could increase the service rate and per-slot delay in a controllable environment [10]. However, when specific dynamic factors related to the ocean, such as evaporation waveguides, surface reflection, mobility, and obstacle obstruction, are considered, the QoS becomes unreliable. In [11], system-level evidence indicates that the end-to-end delay is underestimated by approximately 25% to 40%, while the increase in throughput under actual fading conditions is overestimated. Therefore, although the static form provides a clear benchmark, its static channels limit the QoS in maritime tasks.

To achieve the expansion of QoS control, researchers have explored distributed matching and fairness methods, such as heterogeneous collaboration [12], maximum-minimum resource allocation [13], and two-layer coordination [14], which can reduce signal volume and stabilize the rate of the link. However, when the fleet size increases, the communication overhead becomes dominant, leading to queue backlogs and thus reducing system throughput and prolonging the stay time. In the case of rapid channel changes, heuristic schedulers also exhibit significant service quality inefficiencies [15]. Fundamentally, explicit message passing usually grows quadratically with the number of nodes [16], which reduces effective throughput and worsens the queuing delay of large maritime fleets.

High-fidelity simulators are helpful for quantifying the throughput-delay curves under sea propagation and workload stress. For instance, the studies in [17] and [18] investigated the system behavior and the adaptability of UAV relays. In [19], QoS was further evaluated under adverse weather and NLoS conditions. These studies are realistic, but they are compute-intensive and typically limited to small fleets, which restricts QoS scaling assessment. More importantly, the evaluations based on simulations are offline. The strategies are tested under preset conditions rather than learning online, so they lack real-time QoS adaptability.

Existing work typically targets static channels, limited fleet sizes, or simplified queues. Static formulations fail under time-varying fading, while heuristic and game-theoretic designs introduce large signalling overhead as the fleet expands. Simulation-only approaches do not adjust policies online and overlook real-time coupling. Consequently, the joint management of throughput, delay, and energy in realistic maritime MEC systems is still open.

### B. Reinforcement Learning in resource allocation

DRL has been increasingly applied in MEC problems, aiming to achieve online resource scheduling in time-varying channels and heterogeneous workloads. Representative single-agent research combines trajectory planning with task offloading, where DRL learns environment-aware strategies to reduce service latency [20]. Besides trajectory offloading coupling, DRL is also applied to bandwidth and power allocation, edge association, and task scheduling.

It is obvious that in real scenarios, there are often multiple decision-makers that interact with each other. The MARL technology is applied to distributed decision-making problems. In [21], [22], MARL has been applied to the collaboration of UAVs and USVs, as well as the research on secure cooperation in distributed networks. Under the CTDE model, the policies are trained in a limited global context and executed locally, enabling task scheduling and computational coordination. Learning-based design is also used for routing and relay selection [23], as well as for handling Non-Orthogonal Multiple Access (NOMA) uplink links to improve effective throughput [18], thereby enhancing the deployment capability [24]. At the same time, DRL is combined with optimization modules to simplify the offloading scheduling models. In terms of algorithms, the Proximal Policy Optimization (PPO) is widely adopted as a stable and general benchmark for MEC resource allocation, usually combined with reward normalization and exploration enhancement to improve sample efficiency [25], [26]. These advances demonstrate the applicability of DRL in MEC resource orchestration.

Compared with representative maritime MEC studies, our contribution is threefold: (i) A comprehensive UAVs, USVs, and ship system modeling that enables multi-tier hybrid offloading specific to maritime operations. (ii) A hybrid offloading strategy that remains flexible and stable under dynamics. (iii) An explicit queueing formulation and implementation, yielding a realistic evaluation environment for sojourn time analysis.

Previous studies typically only covered a portion of these aspects: multi-hop drone relays or maritime-air hybrid systems can enhance coverage and latency, but often lack hybrid decision coupling or complete queue dynamics. Digital twin task placement can improve the accuracy of planning, but still maintains a centralized architecture and the runtime queues are relatively simplified.

## III. System Model

In this chapter, we present a system model of hybrid MEC task offloading in maritime networks. By considering realistic LoS or NLoS channel conditions, queueing delays, energy and bandwidth constraints, and battery consumption.

### A. Scenario Description

We consider a hybrid task offloading scenario, including multiple USVs, a UAV, and one ship serving platform. With the characteristics of small size, high speed and easy operation, USVs are often used in marine scenarios, especially for distress search and rescue missions and oil spill detection.

The power-domain NOMA is adopted for short distance USVs to UAV uplinks to exploit LoS and bursty arrivals with SIC at the UAV. And FDMA is used for long-distance transmission between USV and Ship, links to isolate heterogeneous-SNR paths and stabilize service rates under NLoS-dominant maritime channels.

We discretize the operational timeline into time slots indexed by $t \in \{1, 2, \ldots, T\}$. At the beginning of each time slot $t$, each USVs collects and processes data. We denote the set of USVs as $\mathcal{M} = \{1, 2, \ldots, M\}$, and each USV generates tasks $n$ represented as $\mathcal{N} = \{1, 2, \ldots, N\}$. The computational workload of task $n$ generated by USV $m$ at time slot $t$ is denoted as $S_{n,m}(t)$ (bits), with computational intensity $W_{n,m}$ (cycles/bit).

As illustrated in Fig. 1, the transmission channel is significantly influenced by factors such as atmospheric refraction and the Earth's curvature. Under these realistic maritime conditions, we assume that the direct communication path from USVs to the ship is primarily characterized by NLoS transmission. Such conditions result in large-scale channel fading, increased transmission delay, and higher energy consumption. To optimize offloading efficiency, a UAV is introduced to improve channel conditions. USVs can offload their tasks to the UAV by NOMA technology, benefiting from better communication links. However, due to the UAV's limited computational capacity and battery constraints, it is not feasible to rely entirely on UAV computing. Alternatively, USVs can offload tasks directly to the ship platform utilizing Frequency Division Multiple Access (FDMA) for communication, leveraging the ship's substantial computational resources to minimize processing latency. Nevertheless, the effectiveness of this approach is limited by NLoS conditions, which degrade communication quality and increase energy consumption. An efficient alternative is to utilize the UAV as a relay between the USVs and the ship. The UAVs' high-altitude deployment establishes a LoS channel with the ship, creating a two-hop offloading strategy that enhances transmission reliability and efficiency.

In this hybrid task offloading model, each task generated by a USV must be executed using one of the following four mutually exclusive strategies:

- Local computing on the USVs.
- Direct offloading to the UAV by NOMA.
- Direct offloading to the ship by FDMA.
- Relay offloading through the UAV to ship.

To formalize this, we introduce four binary decision variables at each time slot $t$, denoted as $x^{\text{local}}_{n,m}(t), x^{\text{uav}}_{n,m}(t), x^{\text{ship}}_{n,m}(t), x^{\text{relay}}_{n,m}(t) \in \{0, 1\}$, where each

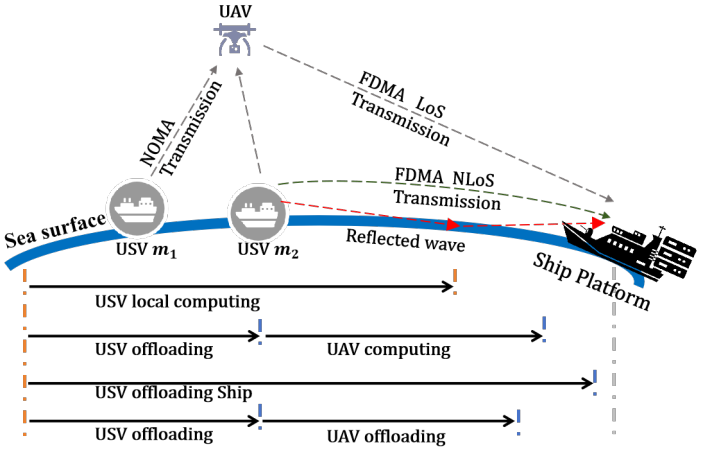



**FIGURE 1. The scenario of hybrid MEC task offloading in maritime networks. Each USV must select one of the following policies to balance job completion and latency: (i) local computing on the USV itself, (ii) offloading to a UAV for edge computing via NOMA uplink, (iii) direct offloading to the ship platform through FDMA under NLoS conditions, or (iv) relay offloading through the UAV to the ship platform under LoS conditions.**

variable indicates whether the full workload of task $n$ from USV $m$ at time slot $t$ is assigned to the corresponding offloading method. Specifically, if $x^{\text{local}}_{n,m}(t) = 1$, the task is fully executed on the USVs. If $x^{\text{uav}}_{n,m}(t) = 1$, it is entirely offloaded to the UAV for computation, If $x^{\text{ship}}_{n,m}(t) = 1$, it is directly processed on the ship. And if $x^{\text{relay}}_{n,m}(t) = 1$, it is first transmitted to the UAV and then relayed to the ship. Since each task must be fully processed through exactly one of these methods, the decision variables satisfy

$$x^{\text{local}}_{n,m}(t) + x^{\text{uav}}_{n,m}(t) + x^{\text{ship}}_{n,m}(t) + x^{\text{relay}}_{n,m}(t) = 1, \quad \forall m \in \mathcal{M},\ n \in \mathcal{N}. \tag{1}$$

For bookkeeping, the (indivisible) task size can be written as

$$S_{n,m}(t) = x^{\text{local}}_{n,m}(t)\, S_{n,m}(t) + x^{\text{uav}}_{n,m}(t)\, S_{n,m}(t) + x^{\text{ship}}_{n,m}(t)\, S_{n,m}(t) + x^{\text{relay}}_{n,m}(t)\, S_{n,m}(t), \quad \forall m \in \mathcal{M},\ n \in \mathcal{N}. \tag{2}$$

Once chosen in slot $t$, the offloading mode remains fixed for task $n$ until completion. This allocation across all tasks and time slots serves as the decision basis for the optimization problem, whose specific objectives and constraints will be presented in the following subsection.

We adopt a three-dimensional (3D) *Cartesian coordinate* system to represent the spatial positions of the USVs, the UAV, and the ship platform. The positions of these nodes are assumed to remain static during data transmission, and their relative distances determine the quality of the communication links. The coordinate of the ship is expressed as $\mathbf{q}_s = [x_s, y_s, 0]$, since the ship is assumed to operate on the sea surface. The coordinate of USV $m$ in time slot $t$ is denoted as $\mathbf{q}_m = [x_m, y_m, z_m]$, where $z_m$ represents the potential variation in the USV by sea waves. For the UAV, its elevated position is given as $\mathbf{q}_u = [x_u, y_u, z_u]$, where $z_u$ is the UAVs' altitude above sea level.

Building on the discrete task offloading decisions, we now characterize the communication links between USVs, the UAV, and the ship, as well as their impact on task execution. Given the UAV's high-altitude deployment, the communication channels between each USV and the UAV, as well as between the UAV and the ship, generally operate under LoS conditions, benefiting from unobstructed visibility. Conversely, the direct link between a USV and the ship is often subject to NLoS transmission due to Earth's curvature or environmental obstructions, resulting in increased path loss, higher transmission delays, and greater energy consumption.

For **local computing**, where tasks executed directly on the USVs, there is no dependency on communication links at time slot $t$, making it an independent and low-latency option. However, its efficiency is limited by the USVs' constrained computing resources. For **UAV Offloading**, the transmission link between USVs and UAV follows an LoS model, ensuring stable communication. In [27], [28], the large scale path loss for LoS transmission is

$$\mathrm{PL}_{\mathrm{LoS}}(d) = 20\log_{10}\left(\frac{4\pi df}{c}\right) + X_{\mathrm{shadow,LoS}}(t), \quad (3)$$

where $d$ is the transceiver distance, $f$ is the carrier frequency, and $c$ is the speed of light. The first term represents free-space path loss, while $X_{\mathrm{shadow,LoS}}(t)$ accounts for shadow fading, modeled as a normal random distribution $X_{\mathrm{shadow,LoS}}(t) \sim \mathcal{N}(0, \sigma^2_{\mathrm{LoS}})$.

For **ship offloading**, where tasks are transmitted from USVs to ship for processing, the NLoS condition introduces additional attenuation due to Earth's curvature, as captured by the round-earth loss model in [29]. The corresponding path loss is expressed as:

$$\mathrm{PL}_{\mathrm{NLoS}}(d) = 20\log_{10}\left(\frac{4\pi df}{c}\right) + \Delta_{\mathrm{curv}}(d) + X_{\mathrm{shadow,NLoS}}(t), \quad (4)$$

where $\Delta_{\mathrm{curv}}(d)$ represents the additional attenuation caused by Earth's curvature, and the shadow fading follows $X_{\mathrm{shadow,NLoS}}(t) \sim Normal(0, \sigma^2_{\mathrm{NLoS}})$.

To capture the impact of resource contention on task execution delay, a slot-based fluid queueing model is adopted for each node in the system. For each node, two types of backlogs are maintained. The reception queue backlog $b^{\mathrm{rx}}_{\mathrm{node}}(t)$ (bits) and the processing queue backlog $c^{\mathrm{proc}}_{\mathrm{node}}(t)$ (CPU cycles). Let $a^{\mathrm{rx}}_{\mathrm{node}}(t)$ and $a^{\mathrm{proc}}_{\mathrm{node}}(t)$ denote the arrivals into the reception and processing queues at slot $t$, and $s^{\mathrm{rx}}_{\mathrm{node}}(t)$ and $s^{\mathrm{proc}}_{\mathrm{node}}(t)$ denote their corresponding service capacities. The queue dynamics follow the standard arrival service balance:

$$b^{\mathrm{rx}}_{\mathrm{node}}(t+1) = \left[b^{\mathrm{rx}}_{\mathrm{node}}(t) + a^{\mathrm{rx}}_{\mathrm{node}}(t) - s^{\mathrm{rx}}_{\mathrm{node}}(t)\right]^+, \quad (5)$$

$$c^{\mathrm{proc}}_{\mathrm{node}}(t+1) = \left[c^{\mathrm{proc}}_{\mathrm{node}}(t) + a^{\mathrm{proc}}_{\mathrm{node}}(t) - s^{\mathrm{proc}}_{\mathrm{node}}(t)\right]^+, \quad (6)$$

where $[x]^+ = \max\{x, 0\}$ ensures non-negativity of queue lengths. The service capacities are determined by the physical-layer transmission rate and computing frequency:

$$s^{\mathrm{rx}}_{\mathrm{node}}(t) = R_{\mathrm{link}}(t)\,\tau \quad (\mathrm{bits/slot}), \quad (7)$$

$$s^{\mathrm{proc}}_{\mathrm{node}}(t) = f_{\mathrm{node}}(t)\,\tau \quad (\mathrm{cycles/slot}), \quad (8)$$

where $\tau$ is the slot duration, $R_{\mathrm{link}}(t)$ is the effective data rate of the corresponding link, and $f_{\mathrm{node}}(t)$ is the CPU frequency of the processing node.

The instantaneous queueing delays are then defined as:

$$Q^{\mathrm{rx}}_{\mathrm{node}}(t) = \frac{b^{\mathrm{rx}}_{\mathrm{node}}(t)}{s^{\mathrm{rx}}_{\mathrm{node}}(t)}, \quad Q^{\mathrm{proc}}_{\mathrm{node}}(t) = \frac{c^{\mathrm{proc}}_{\mathrm{node}}(t)}{s^{\mathrm{proc}}_{\mathrm{node}}(t)}, \quad (9)$$

whenever the denominators are positive. This deterministic, fluid definition aligns with the discrete-time simulation framework used in our experiments and avoids introducing additional probabilistic assumptions.

**Queue Discipline and Stability:** We assume a First-In-First-Out (FIFO) discipline at all nodes. While alternative policies could be incorporated, the above dynamics remain valid regardless of scheduling rule. System stability can be enforced via:

$$\limsup_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}\left(a^{\mathrm{rx}}_{\mathrm{node}}(t) - s^{\mathrm{rx}}_{\mathrm{node}}(t)\right) \le 0, \quad (10)$$

$$\limsup_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}\left(a^{\mathrm{proc}}_{\mathrm{node}}(t) - s^{\mathrm{proc}}_{\mathrm{node}}(t)\right) \le 0, \quad (11)$$

ensuring that the long-term average arrival rate does not exceed the service rate for both reception and processing queues.

The **relay offloading**, where tasks are first transmitted to the UAV and then relayed to the ship, takes advantage of the UAV's ability to establish LoS communication with the USVs and ship. This two-hop approach mitigates the high attenuation associated with NLoS transmission, reducing overall transmission delays and energy consumption. Since relay offloading introduces a dependency between the UAV's processing and its subsequent transmission to the ship, it ensures more reliable communication while leveraging the ship's superior computing resources.

The path loss characteristics of each offloading method play a fundamental role in determining overall efficiency. The high attenuation of NLoS transmission often makes direct ship offloading less favorable under adverse conditions, however, the drone-assisted strategy benefits from the LoS link, thereby reducing energy consumption and latency. The Table 1 presents the key parameters involved in this paper.

### B. Offloading Model

**Local Computing:** In this mode, any task $n$ generated by USV $m$ is fully computed on USV in binary variable $x^{\mathrm{local}}_{n,m} = 1$. Assuming each USV operates on a single core and processes assigned tasks sequentially, the total computation time at USV $m$ is:

$$t^{\mathrm{comp}}_{m} = \frac{\sum_{n\in\mathcal{N}}\left(x^{\mathrm{local}}_{n,m}\, S_{n,m}(t)\, W_{n,m}\right)}{f_m}, \quad (12)$$

where $S_{n,m}(t)$ denotes the data size of task in $t$ slot, and $f_m$ is the CPU frequency of USV $m$. The corresponding local

**TABLE 1. List of notation parameters.**

| Notation | Description |
|---|---|
| $S_{n,m}(t)$ | Input data size of task $n$ generated by USV $m$ (bits). |
| $W_{n,m}$ | Computational intensity of task $n$ (cycles per bit). |
| $x_{n,m}^{(\bullet)}$ | Binary indicator of task $n$ processed locally/ UAV/ ship/ relay $m$. |
| $\mathcal{M}$ | Set of all USVs in the system. |
| $\mathcal{N}_m(t)$ | Set of tasks generated by USV $m$. |
| $P_m^{\text{uav}}(t)$ | Transmit power of USV $m$ when offloading to the UAV (W) in slot $t$. |
| $p_m^{\text{ship}}$ | Transmit power of USV $m$ when offloading to the ship (W) in slot $t$. |
| $p_{\text{uav}}(t)$ | Transmit power of the UAV (W) in slot $t$. |
| $\text{PL}_{\text{LoS}}(d)$ | Path loss in db for LoS channel at distance $d$. |
| $\text{PL}_{\text{NLoS}}(d)$ | Path loss in db for NLoS channel at distance $d$. |
| $T_{n,m}^{(\bullet)}$ | Sojourn time (arrival-to-completion latency) of task $n$ generated by USV $m$. |
| $\tau$ | Slot duration (s) |
| $\{_{(\bullet)}$ | CPU frequency of USV $m$, UAV, and ship (Hz) |
| $\mathcal{K}_{(\bullet)}$ | CPU dynamic power coefficient used. |
| $Q_j^{rx}, Q_j^{proc}$ | Queueing delays (slots or seconds). |

energy consumption is:

$$E_m^{\text{local}} = \kappa_m \sum_{n\in\mathcal{N}} x_{n,m}^{\text{local}}(t)\, S_{n,m}(t)\, W_{n,m}\, f_m^2, \tag{13}$$

where $\kappa_m$ is a hardware coefficient capturing the energy efficiency of USV $m$. This formulation ensures that if $x_{n,m}^{\text{local}} = 1$, the entire workload of task $n$ is processed locally, whereas if $x_{n,m}^{\text{local}} = 0$, no portion of the task is executed on that USV. Therefore, the end-to-end latency is:

$$T_{n,m}^{\text{local}}(t) = \frac{S_{n,m}(t)W_{n,m}}{f_m} + Q_m^{\text{proc}}(t)\tau. \tag{14}$$

**Offloading UAV:** Tasks are fully offloaded to the UAV platform for remote computation. For each task $n$ generated by USV $m$, we define a binary decision variable $x_{n,m}^{uav}(t) \in \{0,1\}$ to represent the discrete offloading decision. Due to the simultaneous uplink communication from multiple USVs, NOMA is employed to enhance spectrum utilization. Considering successive interference cancellation (SIC), USVs are ordered according to their channel gains in descending order such that $|h_1^{\text{uav}}(t)|^2 > |h_2^{\text{uav}}(t)|^2 > \cdots > |h_m^{\text{uav}}(t)|^2$, where the channel gain between USV $m$ and the UAV is expressed as:

$$|h_m^{\text{uav}}(t)|^2 = G_m + G^{\text{uav}} - \text{PL}_{\text{LoS}}(d_m^{\text{uav}}), \tag{15}$$

where $G_m$ and $G^{\text{uav}}$ are the antenna gains of the USV and the UAV, respectively, $PL_{LoS}(\cdot)$ is the path loss, under LoS conditions at distance $d_m^{\text{uav}}$. While USV $m$ offloads task via NOMA, the achievable rate $C_m^{uav}(t)$ is:

$$C_m^{\text{uav}}(t) = \tag{16}$$
$$B_m^{\text{uav}}(t) \cdot \log_2\left(1 + \frac{p_m^{\text{uav}}(t)|h_m^{\text{uav}}(t)|^2}{\sum_{j=m+1}^{M} P_j(t)|h_j^{\text{uav}}(t)|^2 + N_0 B(t)}\right),$$

where $B_m^{\text{uav}}(t)$ is the bandwidth allocated to USV, $P_m^{\text{uav}}(t)$ is the transmit power of USV $m$. The denominator reflects the interference caused by other USVs, which is addressed by successive interference cancellation at the UAV. The communication time for offloading in time slot $t$ is:

$$t_{n,m}^{\text{com,uav}}(t) = \frac{x_{n,m}^{\text{uav}}(t)S_{n,m}}{C_m^{\text{uav}}(t)}. \tag{17}$$

Upon reception, the UAV computes the offloaded tasks. Let $\mathcal{N}^{uav}(t) = \{(n,m)|x_{n,m}^{uav}(t) = 1$ be the set of offloaded tasks and define the total computational load

$$\mathcal{C}^{\text{uav}}(t) = \sum_{(n,m)\in\mathcal{N}^{uav}} S_{n,m}(t) \cdot W_{n,m}, \tag{18}$$

with $W_{n,m}$ representing the computational intensity. Suppose the UAV has $R^{\text{uav}}$ identical processing cores each running at frequency $f^{\text{uav}}$. Then the computing time required to process $C^{uav}(t)$ is

$$t^{\text{comp,uav}}(t) = \frac{\mathcal{C}^{\text{uav}}(t)}{R^{\text{uav}} f^{\text{uav}}}. \tag{19}$$

To capture congestion, we define the queueing model as $Q_{UAV}^{proc}(t)$ by (9). Using the slot length $\tau$ to convert slots to seconds, the end-to-end latency under UAV offloading is

$$T_{n,m}^{\text{uav}}(t) = \frac{S_{n,m}(t)}{C_m^{\text{uav}}(t)} + Q_{\text{UAV}}^{\text{proc}}(t)\tau + \frac{S_{n,m}(t)W_{n,m}}{f^{\text{uav}}} \tag{20}$$

The energy consumption in UAV offloading and computation energy at UAV

$$E^{\text{com,uav}}(t) = \sum_{m\in\mathcal{M}} P_m^{\text{uav}}(t) \cdot \frac{\sum_{n\in\mathcal{N}} x_{n,m}^{\text{uav}}(t) S_{n,m}(t)}{C_m^{\text{uav}}(t)}, \tag{21}$$

$$E^{\text{comp,uav}}(t) = \kappa_{\text{uav}} \cdot (f^{\text{uav}})^2 \cdot t^{\text{comp,uav}}(t), \tag{22}$$

where $\kappa_{\text{uav}}$ is a hardware-related coefficient representing the UAV's computation energy efficiency.

**Offloading ship:** In the ship offloading mode, tasks generated by USVs are entirely transmitted to the ship platform for computation.

The communication channel between USVs and the ship operates under NLoS conditions, characterized by higher path loss and additional attenuation due to the Earth's curvature and potential environmental obstructions. We define the comprehensive NLoS channel gain between USV $m$ and the ship as

$$|h_m^{\text{ship}}(t)|^2 = G_m + G^{\text{ship}} - \text{PL}_{\text{NLoS}}(d_m^{\text{ship}}), \tag{23}$$

where $G_m$ and $G^{\text{ship}}$ represent the antenna gains of the USV and the ship, respectively, and $\text{PL}_{\text{NLoS}}(d_m^{\text{ship}})$ denotes the path loss under NLoS conditions at distance $d_m^{\text{ship}}$.

FDMA is adopted to allocate orthogonal bandwidth resources to each USV, thus eliminating interference. The achievable data rate for USV $m$ transmitting to the ship is then expressed as:

$$C_m^{\text{ship}}(t) = B_m^{\text{ship}}(t)\log_2\left(1 + \frac{p_m^{\text{ship}}(t)\left|h_m^{\text{ship}}(t)\right|^2}{N_0 B_m^{ship}(t)}\right), \tag{24}$$

where $B_m^{\text{ship}}(t)$ represents the allocated bandwidth, $p_m^{\text{ship}}(t)$ is the transmit power of USV, and $N_0$ denotes the noise power at the ship receiver.

If the task $n$ from USV $m$ is fully offloaded to the ship, $x^{\text{ship}}_{n,m}(t) = 1$, the communication delay for transferring the entire workload $S_{n,m}$ is:

$$t^{\text{com,ship}}_{n,m}(t) = \frac{x^{\text{ship}}_{n,m}(t) S_{n,m}(t)}{C^{\text{ship}}_{m}(t)} = \frac{S_{n,m}(t)}{C^{\text{ship}}_{m}(t)}. \quad (25)$$

Upon reception, the ship initiates execution of the offloaded workloads. Let $\mathcal{N}^{\text{ship}}(t) = \{(n,m) \mid x^{\text{ship}}_{n,m}(t) = 1\}$ denote the task set assigned to the ship. Given its abundant computing cores operating at frequency $f^{ship}$ the ship performs parallel processing, while task congestion is captured by the processing-queue backlog $Q^{proc}_{ship}(t)$. The total latency of task $(n,m)$ offloaded to the ship is thus expressed as

$$T^{\text{ship}}_{n,m}(t) = \frac{S_{n,m}(t)}{C^{\text{ship}}_{m}(t)} + Q^{\text{proc}}_{\text{Ship}}(t)\tau + \frac{S_{n,m}(t) W_{n,m}}{f^{\text{ship}}}. \quad (26)$$

The energy expenditure in this mode stems exclusively from the USVs' transmission, since the ship is assumed to possess sufficient onboard energy resources. The total communication energy is formulated as

$$E^{\text{com,ship}}(t) = \sum_{m \in \mathcal{M}} p^{\text{ship}}_{m}(t) \cdot \frac{\sum_{n \in \mathcal{N}} x^{\text{ship}}_{n,m}(t) S_{n,m}(t)}{C^{\text{ship}}_{m}(t)}, \quad (27)$$

where $p_m(t)$ is the transmit power of USVs $m$.

**Relay Offloading:** In the relay offloading strategy, tasks generated by USVs are transmitted to the ship via a UAV, which functions purely as a relay node without executing any computation. We introduce the binary decision variable $x^{\text{relay}}_{n,m}(t) \in \{0,1\}$, where $x^{\text{relay}}_{n,m}(t) = 1$ denotes that task $n$ generated from USV $m$ is transmitted to the ship through the UAV relay.

This offloading method consists of two serial communication hops. In the first hop, tasks are transmitted from each USV to the UAV under NOMA. Since the UAV performs only forwarding, the achievable rate remains consistent with the NOMA uplink channel model, expressed as

$$C^{\text{uav}}_{m}(t) = \quad (28)$$
$$B^{\text{uav}}_{m}(t) \log_2 \left( 1 + \frac{p^{\text{uav}}_{m}(t) \left| h^{\text{uav}}_{m}(t) \right|^2}{\sum_{j=m+1}^{M} p^{\text{uav}}_{j}(t) \left| h^{\text{uav}}_{j}(t) \right|^2 + N_0 B^{\text{uav}}_{m}(t)} \right),$$

where $B^{\text{uav}}_{m}(t)$, $P^{\text{uav}}_{m}(t)$, and $|h^{\text{uav}}_{m}(t)|^2$ denote bandwidth allocation, transmission power, and LoS channel gain from USV $m$ to the UAV, and $N_0$ is the noise power. The corresponding first-hop transmission delay for tasks relayed by USV $m$ is

$$t^{\text{com,uav}'}_{m}(t) = \frac{\sum_{n \in \mathcal{N}} x^{\text{relay}}_{n,m}(t) S_{n,m}(t)}{C^{\text{uav}}_{m}(t)}. \quad \forall m \in \mathcal{M}. \quad (29)$$

In the second hop, the UAV forwards the tasks received from all USVs. Leveraging its high-altitude position, the UAV establishes an LoS link with the ship, characterized by the channel gain:

$$|h^{\text{ship}}_{\text{uav}}(t)|^2 = G_{\text{uav}} + G^{\text{ship}} - \text{PL}_{\text{LoS}}(d^{\text{ship}}_{\text{uav}}), \quad (30)$$

where $G^{\text{uav}}$ and $G^{\text{ship}}$ represent the antenna gains of the UAV and the ship, respectively. And $\text{PL}_{\text{LoS}}(\cdot)$ denotes LoS path loss at distance $d^{\text{ship}}_{\text{uav}}$. Utilizing FDMA for interference-free transmission, the achievable data rate from UAV to ship is expressed as:

$$C^{\text{ship}}_{\text{uav}}(t) = B^{\text{ship}}_{\text{uav}}(t) \log_2 \left( 1 + \frac{P_{uav}(t) \left| h^{\text{ship}}_{\text{uav}}(t) \right|^2}{N_0 B^{ship}_{uav}} \right), \quad (31)$$

where $B^{\text{ship}}_{\text{uav}}(t)$ is the allocated bandwidth and $P_{uav}(t)$ is the transmit power of the UAV. Thus, the second-hop communication delay for all relayed tasks is:

$$t^{\text{com,ship}'}_{\text{uav}}(t) = \frac{\sum_{m \in \mathcal{M}} \sum_{n \in \mathcal{N}_m(t)} x^{\text{relay}}_{n,m}(t) S_{n,m}(t)}{C^{\text{ship}}_{\text{uav}}(t)}. \quad (32)$$

Due to the serial nature of the relay transmission, the total relay offloading communication delay is the sum of delays from both hops, given as:

$$t^{\text{com, relay}}(t) = \max_{m \in \mathcal{M}} t^{\text{com,uav}\prime}_{m}(t) + t^{\text{com,ship}\prime}_{\text{uav}}(t). \quad (33)$$

In addition, to capture potential buffering at the UAV during the forwarding process, a relay-queue backlog $b^{\text{fwd}}_{\text{uav}}(t)$ is maintained. Its evolution follows

$$b^{\text{fwd}}_{uav}(t+1) = \Big[ b^{\text{fwd}}_{uav}(t) + \sum_{m \in \mathcal{M}} \min \Big\{ \sum_{n \in \mathcal{N}_m(t)} x^{\text{relay}}_{n,m}(t) S_{n,m}(t), C^{\text{uav}}_{m}(t)\,\tau \Big\} - C^{\text{ship}}_{uav}(t)\,\tau \Big]^{+}, \quad (34)$$

and the associated waiting delay is incorporated as $Q^{\text{fwd}}_{\text{uav}}(t)\tau$, where $Q^{\text{fwd}}_{\text{uav}}(t) = b^{\text{fwd}}_{\text{uav}}(t) / \left( C^{\text{ship}}_{\text{uav}}(t)\tau \right)$. This ensures that the second-hop transmission does not exceed the first-hop arrival rate, thereby maintaining strict flow consistency.

Upon successful data reception, the ship immediately processes these tasks in parallel, assuming sufficient onboard computational resources. Thus, the computation delay for tasks relayed to the ship is primarily constrained by the task with the heaviest workload, and can be modeled as:

$$t^{\text{comp,ship}'}(t) = \max_{(n,m) \in \mathcal{N}^{\text{relay}}(t)} \frac{S_{n,m}(t)\, W_{n,m}}{f^{\text{ship}}}. \quad (35)$$

where $\mathcal{N}^{\text{relay}}(t) = \Big\{ (n,m) \mid x^{\text{relay}}_{n,m}(t) = 1, \Big\}$, $W_{n,m}$ is the computational intensity (cycles/bit), and $f^{\text{ship}}$ denotes the ship's computing frequency.

The total energy consumption under relay offloading involves only communication-related energy from both hops, since the UAV does not perform computation. Formally, it is given by

$$E^{\text{com,relay}}(t) = \sum_{m \in \mathcal{M}} P^{\text{uav}}_{m}(t) \frac{\sum_{n \in \mathcal{N}} x^{\text{relay}}_{n,m}(t) S_{n,m}(t)}{C^{\text{uav}}_{m}(t)} + P_{\text{uav}}(t) \frac{\sum_{m \in \mathcal{M}} \sum_{n \in \mathcal{N}} x^{\text{relay}}_{n,m}(t) S_{n,m}(t)}{C^{\text{ship}}_{\text{uav}}(t)}. \quad (36)$$

where the first term represents the energy consumed by USVs in the first hop, and the second term denotes UAV energy consumption during the second hop. Additionally, to ensure logical consistency in data transmission, the total amount of data relayed by the UAV to the ship must not exceed the data received from all USVs:

$$C_{uav}^{\text{ship}}(t)\,\tau \;\le\; b_{uav}^{\text{fwd}}(t) + a_{uav}^{\text{fwd}}(t), \quad \forall t. \tag{37}$$

Furthermore, the total system bandwidth $B_{\text{total}}$ must be carefully allocated among different communication links. Specifically, the bandwidth allocated to the NOMA-based communication, the FDMA transmission way, and the relay communication must satisfy the following constraint:

$$\sum_{m\in\mathcal{M}} B_m^{\text{ship}}(t) + B_{\text{uav}}^{\text{ship}}(t) \le B_{\text{total}}(t) - B^{\text{NOMA}}, \tag{38}$$

where $B^{\text{NOMA}}$ denotes the shared bandwidth for NOMA-based uplink transmission from USVs to UAV, $\sum_{m\in\mathcal{M}} B_m^{\text{ship}}(t)$ represents the total FDMA sub-bandwidth allocation for direct ship offloading, and $B_{\text{uav}}^{\text{ship}}(t)$ is the bandwidth assigned for the relay link. This ensures efficient and logically consistent use of the available spectrum resources, preventing interference and excessive bandwidth allocation.

The end-to-end sojourn delay under relay offloading is the sum of queueing, first-hop transmission, UAV forwarding, and ship-side computation delays, which can be expressed as:

$$\begin{aligned} T_{n,m}^{\text{relay}}(t) = {} & Q_{\text{uav}}^{\text{fwd}}(t)\,\tau + \frac{x_{n,m}^{\text{relay}}(t) S_{n,m}(t)}{C_m^{\text{uav}}(t)} \\ & + \frac{\sum_{(n,m)\in\mathcal{N}^{\text{relay}}(t)} S_{n,m}(t) W_{n,m}}{f^{\text{ship}}} + \frac{x_{n,m}^{\text{relay}}(t) S_{n,m}(t)}{C_{\text{uav}}^{\text{ship}}(t)}. \end{aligned} \tag{39}$$

Here $Q_{\text{uav}}^{\text{fwd}}(t)$ denotes the UAV relay backlog in (34), $C^{\text{uav}}m(t)$ and $C^{\text{ship}}\text{uav}(t)$ are the per-hop link rates, and $f^{\text{ship}}$ is the ship's computing frequency. This expression completes the latency characterization and ensures consistency across all offloading modes.

Since each task must be executed through exactly one mode, the total latency at slot $t$ is governed by the longest sojourn time among all tasks, incorporating both communication and computation delays at their respective service nodes. The system latency is defined as

$$\begin{aligned} t_{\text{total}}(t) = \max_{m\in\mathcal{M},\, n\in\mathcal{N}} \Big\{ & x_{n,m}^{\text{local}}(t)\, T_{n,m}^{\text{local}}(t), x_{n,m}^{\text{uav}}(t)\, T_{n,m}^{\text{uav}}(t), \\ & x_{n,m}^{\text{ship}}(t)\, T_{n,m}^{\text{ship}}(t), x_{n,m}^{\text{relay}}(t)\, T_{n,m}^{\text{relay}}(t) \Big\}. \end{aligned} \tag{40}$$

Energy consumption in each slot is consistently accounted for on a mode basis, as previously defined in (36), (21), (22), (13), (27). Specifically, the energy consumed by USV $m$ at slot $t$ is obtained as

$$\begin{aligned} E_m(t) = {} & E_m^{\text{local}}(t) + E_m^{\text{com,uav}}(t) + E_m^{\text{com,ship}}(t) \\ & + E_m^{\text{com,relay}}(t). \end{aligned} \tag{41}$$

Here, each component corresponds to the contributions already defined in the mode-specific models. The ship's computation energy is excluded owing to its abundant onboard resources. The battery dynamics of each node then follow

$$\mathcal{E}_m(t{+}1) = \mathcal{E}_m(t) - E_m(t), 0 \le \mathcal{E}_m(t) \le \mathcal{E}_m^{\max}, \tag{42}$$

$$\mathcal{E}_{\text{uav}}(t{+}1) = \mathcal{E}_{\text{uav}}(t) - E_{\text{uav}}(t), 0 \le \mathcal{E}_{\text{uav}}(t) \le \mathcal{E}_{\text{uav}}^{\max}. \tag{43}$$

Whenever the residual energy of a node reaches zero, its associated transmission or computation service becomes unavailable. This ensures that the system operation remains within feasible energy budgets over the entire horizon.

### C. Problem Formulation

We now formulate the hybrid task offloading control as a finite-horizon optimization problem under stringent energy and resource constraints. Over a time horizon of $T$ slots, each USV $m\in\mathcal{M}$ generates tasks $n\in\mathcal{N}$ of size $S_{n,m}(t)$. At each slot, an admission decision $a_{n,m}(t)\in\{0,1\}$ indicates whether task $(n,m)$ is scheduled for processing. Conditional on admission, a binary offloading variable $x_{n,m}^{(o)}(t)\in\{0,1\}$ specifies the offloading mode $o\in\{\text{local},\text{uav},\text{ship},\text{relay}\}$. Transmit power $p_m^{(o)}(t)$ and bandwidth $B_m^{(o)}(t)$ are continuous control variables jointly optimized with $x_{n,m}^{(o)}(t)$. The system states include the UAV forwarding queue backlog $b_u^{\text{fwd}}(t)$ and the residual energies $\mathcal{E}_m(t)$ and $\mathcal{E}_{\text{uav}}(t)$. The optimization problem is then expressed as:

$$\max \quad \sum_{t=1}^{T} \sum_{m\in\text{M}} \sum_{n\in\text{N}} a_{n,m}(t)\, S_{n,m}(t) \tag{44}$$

$$\textbf{s.t.} \quad x_{n,m}^{\text{local}}(t) + x_{n,m}^{\text{uav}}(t) + x_{n,m}^{\text{ship}}(t) + x_{n,m}^{\text{relay}}(t) = a_{n,m}(t), \tag{a}$$

$$\sum_{m,n} x_{n,m}^{\text{uav}}(t)\, S_{n,m}(t)\, W_{n,m} \le R^{\text{uav}} f^{\text{uav}} \tau, \tag{b}$$

$$\sum_{m,n} \left(x_{n,m}^{\text{ship}}(t) + x_{n,m}^{\text{relay}}(t)\right) S_{n,m}(t)\, W_{n,m} \le R^{\text{ship}} f^{\text{ship}} \tau, \tag{c}$$

$$b_u^{\text{fwd}}(t{+}1) = \left[ b_u^{\text{fwd}}(t) + a_u^{\text{fwd}}(t) - C_u^{\text{ship-rel}}(t)\,\tau \right]^+, \tag{d}$$

$$\mathcal{E}_m(t{+}1) = \mathcal{E}_m(t) - E_m(t), \quad 0 \le \mathcal{E}_m(t) \le \mathcal{E}_m^{\max},$$

$$\mathcal{E}_{\text{uav}}(t{+}1) = \mathcal{E}_{\text{uav}}(t) - E_{\text{uav}}(t), \quad 0 \le \mathcal{E}_{\text{uav}}(t) \le \mathcal{E}_{\text{uav}}^{\max}, \tag{e}$$

$$\sum_{m\in\mathcal{M}} B_m^{\text{ship}}(t) + B_{\text{uav}}^{\text{ship}}(t) \le B_{\text{total}} - B^{\text{NOMA}}, \tag{f}$$

$$0 \le p_m^{\text{uav}}(t),\ p_m^{\text{ship}}(t) \le p_{\max}, \quad 0 \le p_{uav}(t) \le p_{\max}^{\text{uav}}, \tag{g}$$

$$x_{n,m}^{(\bullet)}(t) \in \{0,1\}, \quad o \in \{\text{local},\text{uav},\text{ship},\text{relay}\},\ \forall m,n,t. \tag{h}$$

The above formulation comprehensively captures the operational limitations of the maritime MEC system. Constraint (a) guarantees that each admitted task is exclusively assigned

to one offloading mode, ensuring the discreteness of decision variables. Constraints (b) and (c) impose per-slot computational capacity limits on the UAV and the ship, thereby preventing overload. Constraint (d) enforces flow conservation for relay offloading by explicitly modeling the forwarding buffer at the UAV, ensuring that the second-hop transmission never exceeds the first-hop arrivals. Constraint (e) incorporates energy causality and battery capacity limitations, ensuring that both USVs and UAV remain within feasible energy budgets throughout the horizon. Constraint (f) enforces the total bandwidth budget, accounting for reserved NOMA spectrum and coherent allocation across USV-to-Ship and UAV-to-Ship FDMA channels. Constraint (g) restricts the transmit powers of both USVs and UAV to remain within permissible ranges. Finally, constraint (h) specifies the binary nature of the offloading variables, reaffirming the discrete structure of the problem.

The resulting problem is inherently a mixed-integer nonlinear program (MINLP) due to the binary offloading decisions, nonlinear communication rates, and the queue–energy dynamics. These properties significantly increase computational complexity and render exact convex optimization intractable. To address this, we subsequently reformulate the system as a Dec-POMDP and adopt reinforcement learning techniques to approximate near-optimal solutions in a scalable manner.

We deliberately avoid imposing a hard per–task latency constraint $T^{(\bullet)}_{n,m}(t) \le T_{\max}$, which would time–couple the MINLP and further increase online complexity. Instead, in our learning-based solution, we *implicitly* discourage long sojourn times by maximizing sliding-window throughput

$$R_t \triangleq \sum_{k=0}^{w-1} N_{\text{finish}}(t-k), \qquad J \triangleq \sum_{t=1}^{T} R_t, \tag{45}$$

where $N_{\text{finish}}(\tau)$ is the number of tasks that complete in slot $\tau$ and $w$ is a small window length. Exchanging the sums yields

$$J = \sum_{s=1}^{T} \beta_s \, N_{\text{finish}}(s), \quad \beta_s = \min\big(w,\, T-s+1\big), \tag{46}$$

so earlier completions receive a larger weight. Together with the stability, bandwidth, compute, energy and relay-causality constraints in (44), maximizing (45) reduces the average backlog. Moreover, for any budget $T_{\max} > 0$, $\mathbb{P}\{T > T_{\max}\} \le \bar{T}/T_{\max}$. So reducing $\bar{T}$ directly tightens an upper bound on the violation probability. Hence latency is treated as a *soft constraint*: we strictly enforce resource/energy causality, and steer the policy to low-delay regimes without explicit dual updates or virtual delay queues.

## IV. Problem Formulation and Learning Algorithm

### A. Dec-POMDP Formulation

We model the dynamic hybrid offloading in a maritime MEC network with multiple USVs, one UAV, and a ship as a Dec-POMDP under a CTDE environment. Each USV acts as an agent that selects an offloading mode for task based on partial observations, While the environment enforces energy, bandwidth, and relay constraints with dynamic channels.

Formally, the Dec-POMDP is

$$\langle \mathcal{I}, \mathcal{S}, \{\mathcal{O}_m\}_{m\in\mathcal{I}}, \{\Omega_m\}_{m\in\mathcal{I}}, \{\mathcal{A}_m\}_{m\in\mathcal{I}}, \mathcal{P}, \mathcal{R}, \gamma \rangle, \tag{47}$$

where $\mathcal{I} = \{1, \ldots, M\}$ indexes USV agents, $\mathcal{S}$ is the global state space, $\mathcal{O}_m$ and $\Omega_m : \mathcal{S} \to \mathcal{O}_m$ are the local observation space and observation map of agent $m$. The $\mathcal{A}_m$ is its action space, $\mathcal{P}$ is the Markov transition induced by the dynamics environment. $\mathcal{R}$ is the sliding-window throughput reward, and $\gamma \in (0, 1)$ is the discount factor. Constraint feasibility at execution time is ensured via action masking provided by the environment. In our setting:

**State space** $\mathcal{S}$. The global state at slot $t$ aggregates the Markov variables that drive queue, energy, and channel dynamics:

$$s_t = \Big( \{\, Q_m^{\text{proc}}(t),\ \mathcal{E}_m(t),\ |h_m^{\text{uav}}(t)|^2,\ |h_m^{\text{ship}}(t)|^2 \,\}_{m\in\mathcal{M}}, \tag{48}$$
$$Q_{\text{UAV}}^{\text{proc}}(t),\ Q_{\text{Ship}}^{\text{proc}}(t),\ b_{\text{uav}}^{\text{fwd}}(t),\ \mathcal{E}_{\text{uav}}(t),\ B_{\text{res}}(t),\ \xi(t) \Big).$$

Here $Q_m^{\text{proc}}(t), Q_{\text{UAV}}^{\text{proc}}(t)$, and $Q_{\text{Ship}}^{\text{proc}}(t)$ are processing tasks defined by (9). $\mathcal{E}_m(t)$ and $\mathcal{E}_{\text{uav}}(t)$ are battery energies, and $b_{\text{uav}}^{\text{fwd}}(t)$ is the relay buffer from constraint (d). $B_{\text{res}}(t)$ denotes the system-level residual FDMA bandwidth after NOMA reservation (distinct from the per-agent allocation $b_m(t)$ in the action space), and $\xi(t)$ indexes the propagation regime. Therefore the process remains Markov when LoS probability and path-loss parameters switch across scenarios.

**Observation space** $\mathcal{O} = \prod_{m\in\mathcal{M}} \mathcal{O}_m$, where $\mathcal{O}_m$ denotes the local observation space of USV. Under decentralized execution, USV $m$ receives a partial observation

$$O_m(t) = \Omega_m(s_t) = \big( \mathcal{E}_m(t),\ Q_m^{\text{proc}}(t), \ \{\, |h_m^{(j)}(t)|^2 \,\}_{\mathbf{j}\in\mathcal{J}},\ \mathbf{z}_{\text{pub}}(t) \big), \tag{49}$$

where $\mathcal{J} = \{\text{uav}, \text{ship}\}$ indexes mode, so that $\{\, |h_m^{(j)}|^2 \,\}_{j\in\mathcal{J}}$ compactly represents both $|h_m^{\text{uav}}|^2$ and $|h_m^{\text{ship}}|^2$. The public broadcast context is

$$\mathbf{z}_{\text{pub}}(t) = \big( \mathcal{E}_{\text{uav}}(t),\ b_{\text{uav}}^{\text{fwd}}(t), \ Q_{\text{UAV}}^{\text{proc}}(t),\ Q_{\text{Ship}}^{\text{proc}}(t),\ B_{\text{res}}(t),\ \xi(t) \big), \tag{50}$$

which is shared over a low-rate control channel so that agents can satisfy energy, bandwidth, and relay feasibility while acting independently. Other agents' private queues remain unobserved.

**Action space**. At each slot $t$, each USV $\mathcal{M}$ selects a action that couples the offloading mode of its head task with continuous resource control. The discrete component is a one-hot vector

$$\mathbf{x}_m(t) = \big( x_m^{\text{local}}(t), x_m^{\text{uav}}(t), x_m^{\text{ship}}(t), x_m^{\text{relay}}(t) \big) \in \{0,1\}^4, \mathbf{1}^\top \mathbf{x}_m(t) = 1, \tag{51}$$

corresponding to local computing, direct UAV offloading, direct ship offloading, and UAV-relay to ship. If USV $\mathcal{M}$ has no task at slot $t$, the discrete action is ignored. Conditioned

on the selected mode, the continuous component specifies the transmit power $P_m(t)$ and bandwidth allocation $b_m(t)$, which are constrained by the physical-layer limits of NOMA/FDMA channels and per-node energy budgets.

Hence the overall action of agent $\mathcal{M}$ is expressed as

$$a_m(t) = \big(\mathbf{x}_m(t),\, P_m(t),\, b_m(t)\big), \tag{52}$$

which integrates the mode selection with joint power and bandwidth allocation. This hybrid action formulation ensures that $\sum_o x^o_{n,m}(t) = 1$ for the scheduled task while satisfying the constraints of (d)–(g), and aligns with the Dec-POMDP framework adopted in MARHO.

**Observation function.** Since the execution is decentralized, USV $m$ receives only a partial view of the global state. Its local observation at slot $t$ is

$$O(t) = \Omega_m(s_t) = \big(\mathcal{E}_m(t),\, Q^{\mathrm{proc}}_m(t),\, |h^{\mathrm{uav}}_m(t)|^2,\, |h^{\mathrm{ship}}_m(t)|^2\big) \cup z_{\mathrm{pub}}(t), \tag{53}$$

where $\mathcal{E}_m(t)$ is the residual battery energy of USV $m$, $Q^{\mathrm{proc}}_m(t)$ is its processing backlog defined in (9), and $|h^{\mathrm{uav}}_m(t)|^2, |h^{\mathrm{ship}}_m(t)|^2$ are its instantaneous channel gains to the UAV and the ship. The broadcast context $z_{\mathrm{pub}}(t)$, defined in (50), includes the UAV's residual energy $\mathcal{E}_{\mathrm{uav}}(t)$ and its relay-buffer backlog $b^{\mathrm{fwd}}_{\mathrm{uav}}(t)$, which are periodically disseminated to all USVs to enable feasible coordination while preserving decentralized execution. Therefore, $o_m(t)$ is a deterministic projection of the global state $s_t$ that contains exactly the information needed for decentralized decision-making under the constraints in (44).

**State Transition Function**: Given state $s_t$, the joint mode selections $\{x^o_{n,m}(t)\}$ and exogenous randomness determine $s_{t+1}$. We write the compact transition as

$$s_{t+1} \;=\; F\big(s_t,\; \{a_m(t)\},\; \xi(t)\big), \tag{54}$$

where $\xi_t$ collects task arrivals and channel/shadowing draws governed by (3) and (4). The transition is realized via the following recursions:

- **Battery updates**. The time slot energy expenditures $E_m(t)$ and $E_{\mathrm{uav}}(t)$ are assembled from the chosen modes using (13), (21), (22), (27), and (36), and aggregated as (41). Residual energies then evolve according to the unified battery laws (42).
- **Queue updates**. Reception and processing backlogs at each node follow the slot-based balances in (9). Link services use the capacities in (16), (24), and (31), under the resource budgets in constraints (d)-(h). The relay pipeline obeys the arrival, buffer, and causality relations (33), (34), and (37). Processing arrivals equal admitted bits times the computation intensity $W_{n,m}$, and the instantaneous queueing delays used in latency expressions are the fluid measures in (9).

This construction ensures the Dec-POMDP transition respects all feasibility constraints (44) in the optimization model, with the mode choices $\{x^o_{n,m}(t)\}$ driving energy use, link services, relay flow, and hence the next state.

**Reward Function:** Let $N_{\mathrm{fin}}(t)$ be the number of tasks that finish at slot $t$. With window length $w$, define the windowed throughput

$$\overline{\Theta}_t \;=\; \frac{1}{\min(w,\; t{+}1)} \sum_{i=\max(0,\, t-w+1)}^{t} N_{\mathrm{fin}}(i). \tag{55}$$

Let $F_t$ be the set of tasks finished by the end of slot $t$, and let the global average sojourn delay be

$$\overline{D}_t \;=\; \begin{cases} \dfrac{1}{|F_t|} \sum_{j\in F_t} \big(t^{\mathrm{finish}}_j - t^{\mathrm{arrive}}_j\big), & |F_t| > 0, \\ 0, & |F_t| = 0. \end{cases} \tag{56}$$

The instantaneous reward is the sliding-window throughput modulated by a delay discount:

$$r(t) \;=\; \overline{\Theta}_t \cdot \big(\gamma_d\big)^{\overline{D}_t}, \tag{57}$$

where $\gamma_d \in (0,1)$ is a fixed delay-discount factor ($\gamma_d \approx 0.9565$). Note that $\gamma_d$ is not the RL discount $\gamma$ in the return, but a QoS-oriented shaping parameter introduced in the reward function. It exponentially penalizes long sojourn delays: tasks completed earlier contribute more since $\gamma_d^{D_t}$ remains closer to one, whereas late completions either fall outside the sliding window or incur a smaller multiplicative weight.

**Policy objective and CTDE:** We optimize a joint policy $\pi = \{\pi_m\}_{m=1}^{M}$ under centralized training and decentralized execution:

$$\max_{\pi}\; \mathbb{E}_{\pi}\left[\sum_{t=1}^{T} \gamma_{\mathrm{rl}}^{\,t-1}\, r(t)\right], \tag{58}$$

with DRL discount $\gamma_{\mathrm{rl}} \in (0,1)$ ($\gamma_{\mathrm{rl}} \in [0.95, 0.99]$). During training, the critic may access the global state $s_t$ to improve value estimation, while at execution each USV acts only on its local observation $o_m(t)$. All agents receive the same team reward $r(t)$.

### *B. The MARHO Algorithm*

We develop MARHO as a multi-agent reinforcement learning framework based on PPO under the CTDE paradigm. In our environment, the USVs are modelled as agents, while the UAV and the ship are treated as part of the environment whose resources enter the centralized critic and the global reward calculation. All USVs' actors share parameters, which improves sample efficiency and ensures permutation invariance across agents.

Each actor $\pi_\theta(a_m|o_m)$ maps the local observation $o_m(t)$ of USV $m$ to a discrete action $x^o_{n,m}(t)$, determining the offloading mode of the task. The centralized critic $V_\psi(s_t)$ is trained with access to the global state $s_t$, thereby providing accurate value estimation during training. At execution time, each USV selects its offloading mode solely based on its partial observation, ensuring feasible decentralized decision-making in dynamic maritime environments.

The reward function integrates throughput and delay in a unified form. Specifically, we adopt a sliding-window

**Algorithm 1:** MARHO: Training processing

**Input:** Episode length $T$, sliding window size $w$, discount factor $\gamma$
**Output:** Trained actor $\pi_\theta(a_m|o_m)$, critic $V_\psi(s)$

Initialize actor parameters $\theta$, critic parameters $\psi$;
**for** *each episode* **do**
  Reset environment, initialize state $s_0$;
  **for** $t = 1$ **to** $T$ **do**
    **foreach** *USV* $m$ *in parallel* **do**
      Observe $o_m(t)$; select mode $x_m(t) \sim \pi_\theta(\cdot|o_m(t))$;
    Execute joint actions $\{x_m(t)\}_{m\in\mathcal{M}}$;
    Environment updates queues, batteries, and relay buffers;
    Compute reward $r_t = \overline{\Theta}_t \cdot \gamma_d^{\overline{D}_t}$;
    Store $(o_{1:M}, x_{1:M}, r_t, s_t, s_{t+1})$;
  Compute advantages $\hat{A}_t$ using critic $V_\psi$;
  Update actor $\pi_\theta$ via PPO objective with entropy regularization;
  Update critic $V_\psi$ by minimizing value loss;

**Algorithm 2:** MARHO: Decentralized Execution

**Input :** Trained actor $\pi_\theta(a_m|o_m)$, episode length $T$
**Output :** Action sequence $\{x_m(t)\}$

Reset environment; initialize $s_0$; **for** $t \leftarrow 1$ **to** $T$ **do**
  ▷ USVs act *independently* with local observations
  **foreach** *USV* $m \in \mathcal{M}$ ▷ in parallel **do**
    observe $o_m(t)$
    mask infeasible modes in feasibility oracle
    select mode $x_m(t) \sim \pi_\theta(\cdot \mid o_m(t))$
  ▷ Environment transition from (44)
  Execute joint modes $\{x_m(t)\}_{m\in\mathcal{M}}$;
  Update link services, relay buffer, queues and batteries
  Log instantaneous metrics: finished tasks $\Theta_t$, average sojourn delay $\bar{D}_t$;

throughput $\overline{\Theta}_t$ of length $w$ as the primary reward signal and incorporate a delay-discount factor $\gamma^{\overline{D}_t}$ to suppress long delays in (57).

As shown in Alg. 1, every USV observes its local state, masks infeasible actions according to energy and bandwidth constraints, and samples an offloading mode from its actor. The environment then updates link capacities, relay buffers, processing queues, and battery states, after which the global reward $r_t$ is computed. Trajectories are collected over multiple rollouts and used to update both the shared actor and the centralized critic. The critic leverages the full state $s_t$ to improve value estimation, while the actor is trained only on partial observations.

After training, each USV executes its actor independently using only local observations. In Alg. 2, the USVs select their offloading modes in parallel, while the environment enforces bandwidth allocation, relay causality, and energy feasibility. No critic or global coordination is required at runtime, ensuring decentralized execution.

The proposed MARHO adopts the CTDE paradigm where the centralized critic assists training but is discarded at execution. The overall per-iteration computational cost scales as $\mathcal{O}(N_{\text{agents}} \cdot N_{\text{steps}} \cdot \text{FLOPs}_{\text{AC}})$, which grows linearly with the number of USVs and training steps. Since actors operate independently during execution, no inter-agent communication is required, ensuring scalability in fleet size.

## V. Experimental Result and Analysis

In this section, a series of progressively in-depth experiments and observations are conducted to verify the effectiveness of the MARHO strategy. In Section A, we first clearly define the simulation environment design, hyperparameter selection, benchmark algorithms, and evaluation metrics. In Section B, we present the core comparison under a medium load and short running time, showing that MARHO simultaneously achieves the maximum number of completed tasks and the lowest average delay. Finally, Section C verifies that these advantages still hold when the task duration is extended and the ocean channel deteriorates under higher load, highlighting the robustness of the learned strategy.

**TABLE 2.** The Hyper-parameters of experiment.

| Parameter | Value |
|---|---|
| $\lambda$ (task arrival rate) | 0.5 tasks/slot |
| $\mathcal{M}$ (number of USVs) | 5 – 20 |
| $T$ (episode length) | 1200, 1800 slots |
| $\gamma$ (discount factor) | 0.95 – 0.99 |
| $\gamma_d$ (QoS-oriented shaping) | 0.9565 |
| $\lambda_{\text{GAE}}$ (GAE parameter) | 0.95 – 0.97 |
| $\epsilon_{\text{clip}}$ (clipping epsilon) | 0.2 – 0.3 |
| $S_{\min}, S_{\max}$ (task size) | 0.5 – 10.0 Mb |
| cycles/bit (CPU density) | 1000 cycles/bit |
| $W_{\text{usv\_uav}}$, $W_{\text{usv\_ship}}$, $W_{\text{uav\_ship}}$ | 20, 20, 15 MHz |
| $P_{\max}$ (transmit power) | USV:1 W, UAV:5 W, |
| Queue capacity | 60 tasks |
| $w$ (sliding-window length) | 10 (5 – 15) |
| $T_{\max}$ (delay budget) | 60 slots (40 – 100) |
| `actor_lr` (PPO actor LR) | $1 \times 10^{-4} - 3 \times 10^{-4}$ |
| `critic_lr` (PPO critic LR) | $5 \times 10^{-4} - 1 \times 10^{-3}$ |
| **PSO** | swarm_size = 3, max_iter = 2 |

### A. Evaluation Setup

We designed a Hybrid Offloading Gym environment, which simulates an offshore MEC network consisting of $\mathcal{M}$ USVs, a single UAV, and a ship equipped with a multi-access edge server. During the initialization stage, the USVs are randomly distributed within an internal square area, the UAV hovers at the center of this area, and the ship is randomly placed at a certain point within the external square area. Each node is assigned a limited CPU capacity and energy budget, and

implements tail-drop queue policies to limit memory usage. The USVs maintain an arrival queue, a local computing queue, and a transmission queue. The UAV also has computing and relay queues, while the ship has only a computing queue.

The task generation process follows independent Poisson arrivals at each USV with a rate of $\lambda = 0.5$ tasks per slot. Task sizes are uniformly distributed between 0.5 MB and 10 MB, reflecting the diversity of maritime sensing and monitoring applications. The computation intensity is fixed at 1000 cycles per bit across all tasks. Network parameters are configured to represent realistic maritime communication conditions, with 20 MHz shared NOMA bandwidth for USV–UAV uplink, 20 MHz FDMA for USV–ship, and 15 MHz FDMA for UAV–ship relay. Initial transmission powers are set to $1W$ for USVs and $5W$ for the UAV, representing typical constraints of battery-powered maritime devices. And the arrival timestamp $t_{\text{arrive}}$. At each time slot $t$, completed tasks satisfy remain_bits $= 0$ and are marked as $t_{\text{finish}}$. The sojourn delay is $D = t_{\text{completion}} - t_{\text{arrival}}$, all hyper parameters are presented in Table 2. We adopt the reward function by (57), balancing throughput and delay:

$$r(t) = \bar{\Theta}_t(\gamma_d)^{D_t}, \tag{59}$$

where $\bar{\Theta}_t$ is the normalized windowed throughput and $D_t$ the average sojourn delay of completed tasks. We set $\gamma_d = 0.9565$ to penalize long delays while keeping $r(t) \in [0, 1]$, as listed in Table 2.

After a task arrives, its size, the CPU cycles required per bit, and the arrival timestamp are recorded. Subsequently, it remains in the input queue of the USV until the strategy selects one of the four offloading modes. Within each discrete time interval, the environment makes routing decisions, performs local CPU processing, wireless transmission, UAV relay and computing, and ship-side computing, while simultaneously tracking energy consumption and marking task completion.

We compare MARHO against four baselines, targeting maximal task throughput within the sampling horizon $T$.

- **MARHO** (ours): A CTDE method based on the improvement of PPO for hybrid action offloading algorithm. Each USV selects a hybrid offloading action under queue, bandwidth, and energy feasibility masks, while a centralized critic uses the global state to improve value estimation during training. The reward integrates sliding-window throughput and delay (44).
- **Random**: Each USV uniformly samples an offloading mode.
- **Greedy-Delay**: Each USV selects the mode that minimizes the instantaneous transmission delay within the slot.
- **IQL** (Independent Q-Learning): Each USV learns a separate Q-network and executes independently. During training, joint transitions are collected into a centralized replay buffer for fair data exposure. Target networks are used for stabilization. Actions follow $\varepsilon$-greedy with linear decay. Optimizer settings, discount factor, and batch size follow Table 2. This baseline ignores inter-agent coupling at update time and reflects a simple MARL approach with low coordination cost.
- **PSO**: A centralized particle-swarm optimizer that allocates transmission power and bandwidth per slot under identical budgets, solving a relaxed continuous allocation to minimize one-step delay.

**Evaluation setup.** We vary the number of USVs $\mathcal{M}$ from 5 to 20 and run episodes of 1200 and 1800 slots. Channel robustness is assessed under clear, rain, and storm profiles by adjusting LoS probability and path loss using the ITU-R maritime model. Performance is reported by three QoS metrics: (i) cumulative reward $J$, (ii) task throughput $N_{\text{task}} = |\mathcal{F}|$ where $\mathcal{F}$ is the set of tasks whose remaining bits reach zero by the end of the episode, and (iii) global average sojourn delay

$$\bar{T} = \frac{1}{|\mathcal{F}|} \sum_{i \in \mathcal{F}} \left(t_{\text{finish},i} - t_{\text{arrive},i}\right). \tag{60}$$

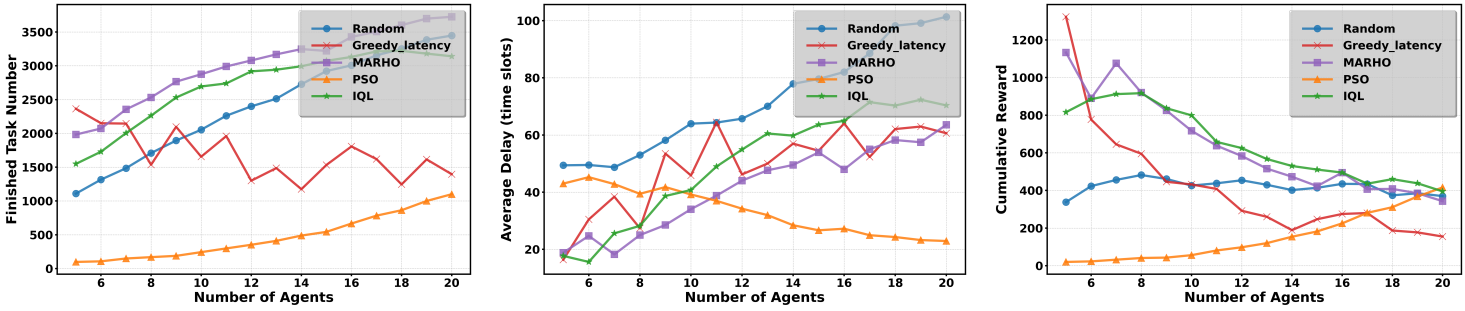
For fairness, IQL uses the same sampling budget, evaluation cadence, discount factor, and optimizer class as MARHO, and shares the training weather schedules and traffic arrival processes. Each configuration is executed 10 times. We report means with 95% confidence intervals. Experiments run on an Intel i7-10700K, an NVIDIA RTX 3080, and 16 GB RAM with Ubuntu 20.04.

### B. Core Performance Analysis Under Short Period

We define core performance as the policy's behavior over a 1200-slot horizon, representing a typical mission duration under favorable channel states. Under the base hyperparameter settings ($\lambda = 0.5$ tasks/slot, sliding-window length $w = 10$, delay budget $T_{\max} = 60$ slots), Fig. 2 compares the MARHO policy with four baselines, including Random, Greedy-Latency, IQL, and PSO as the number of USVs $\mathcal{M}$ varies from 5 to 20.

As $M$ increases from 5 to 20, MARHO completes about $1.5 \times 10^3$ to $3.6 \times 10^3$ tasks. The average delay remains close to 60 slots, which matches the reward design. Random also benefits from a larger fleet and reaches about $1.2 \times 10^3$ to $3.4 \times 10^3$ finished tasks, but its average delay rises quickly and soon exceeds 60 slots. This shows that Random can exploit parallel agents but cannot maintain delay when the load grows. Greedy-Latency reaches nearly $1.7 \times 10^3$ tasks at $M = 10$ and shows little improvement when $M > 15$. The policy reduces delay within each slot but does not prevent queue build-up, so the total gain is limited.

The greedy algorithm demonstrates intermediate performance with notable limitations at higher fleet sizes. Greedy-Latency achieves reasonable task completion rates, reaching approximately $1.7 \times 10^3$ tasks at $N = 10$ and saturates beyond $\mathcal{N} = 15$ due to its myopic decision-making. The algorithm's focus on instantaneous delay minimization prevents it from considering longer-term system dynamics and queue manage-




(a) Number of finished tasks vs. USV fleet size. (b) Average sojourn delay vs. $M$; (c) Cumulative reward over 1200 slots.

**FIGURE 2. The purpose of the core performance experiment is to evaluate the scalability of the system and the trade-off between delay and throughput under 1200 time slots. When $M$ increases from $5$ to $20$, the number of tasks completed by MARHO increases from approximately $1.5 \times 10^3$ to $3.6 \times 10^3$, while the average latency remains around $60$ time slots. Moreover, MARHO achieves the highest cumulative reward, outperforming the random algorithm, the greedy delay algorithm, the IQL algorithm, and the PSO algorithm.**

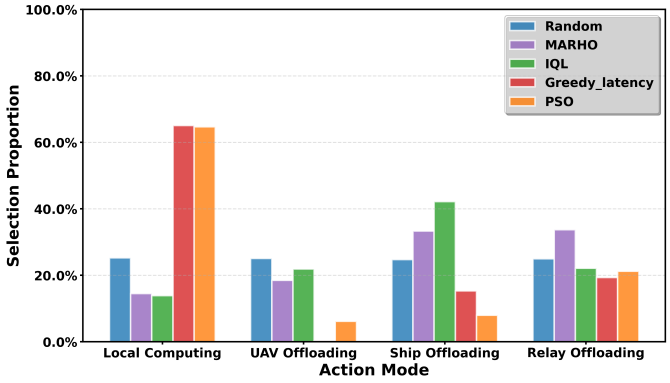



**FIGURE 3. The results of the action selection percentage under the condition of a system with $1200$ time slots and $N = 10$ ($10$ runs, mean counts). MARHO predominantly selects ship and relay offloading, whereas Greedy baselines favor Local. The proportion of local computation in IQL is similar to that of MARHO, but the selection of the offloading strategy is different. PSO leans to Local then offloading. Random is roughly uniform.**

ment, leading to suboptimal resource allocation as system complexity increases.

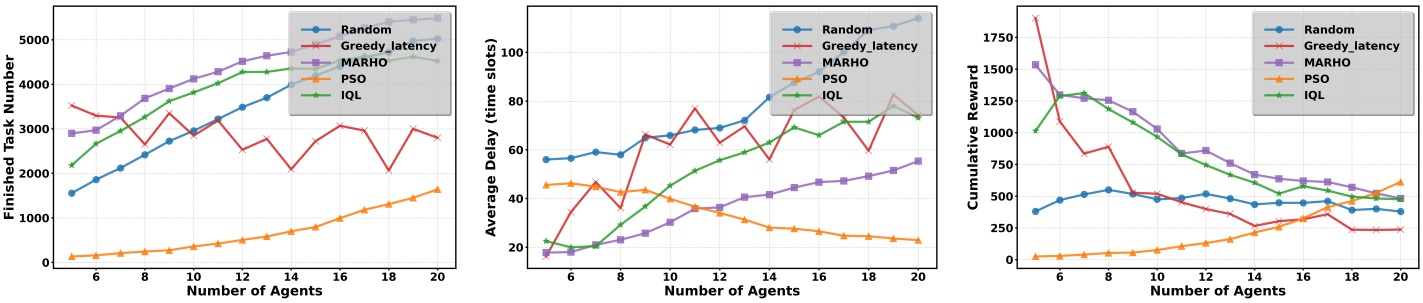
The IQL baseline, which follows a decentralized MARL structure, shows steady but not optimal performance relative to MARHO. As illustrated in Fig. 2, IQL performs better than Random and Greedy-Latency across most fleet sizes. This indicates that independent learning under the CTDE framework provides some degree of coordinated behavior. When the fleet size exceeds $\mathcal{M} = 10$, the gap between IQL and MARHO becomes more apparent. The average delay rises with $\mathcal{M}$, while the growth in completed tasks slows and the cumulative reward drops. These trends stem from the use of separate Q-networks based only on local observations, which restricts the understanding of queue interactions and link coupling among agents. As a result, IQL adapts less effectively to congestion and varying channel quality, and its performance decreases under larger-scale settings.

The PSO method provides a contrasting example of centralized optimization in this setting. Although PSO can search for solutions that are close to the continuous optimum, the total number of completed tasks remains below that of MARHO and IQL. The reduction in throughput is mainly related to the communication load required to coordinate a centralized solver and the computational effort of optimizing continuous variables across multiple nodes. These costs grow quickly when channels fluctuate and the number of agents increases. The results suggest that centralized schemes face practical difficulties in maritime environments where link reliability varies and node resources are limited.

To provide detailed performance insight, we examine the $\mathcal{M} = 10$ configuration as a representative anchor point that balances system complexity with practical deployment considerations. At this operating point, MARHO achieves approximately $2.8 \times 10^3$ completed tasks compared to $2 \times 10^3$ for Random, $1.7 \times 10^3$ for Greedy-Latency, $2.6 \times 10^3$ for IQL, and $1.1 \times 10^3$ for PSO. These results correspond to relative improvements of 40% over Random, 64.7% over the greedy approach, and 154.5% over PSO, demonstrating substantial performance gains across all baseline comparisons.

The average task delay analysis shows MARHO achieves low latency while sustaining high throughput. MARHO stays near $60$ slots across $\mathcal{M}$, close to the target $60.7$ slots. PSO attains lower delay around $40$ slots but trades off substantial throughput. Greedy-Latency lies in the mid-range (about $50$–$65$ slots) and degrades as $\mathcal{M}$ increases. IQL tracks MARHO at small $\mathcal{M}$ but its delay grows steadily with fleet size, settling between MARHO and Random at large $\mathcal{M}$. Random yields the highest delay, often above $80$ slots. These outcomes match the window-based reward, where earlier completions contribute more windows and discourage long sojourns.

From $\mathcal{M} = 6$ to $\mathcal{M} = 15$, MARHO attains the highest cumulative rewards. As $\mathcal{M}$ increases, rewards decrease for all learned methods due to tighter power budgets and intensified queuing that reduce 10-step window throughput. IQL consistently exceeds Random but lags MARHO, with a widening gap beyond $\mathcal{M} \approx 10$ where coordination becomes more critical. Despite this downward trend, MARHO keeps




(a) Number of finished tasks vs. USVs fleet size. (b) Average sojourn delay vs. USVs fleet size. (c) Cumulative reward over 1800 slots.

**FIGURE 4. This experiment aims to evaluate the robustness of the algorithm over a longer duration (1800 time slots, rainy conditions, LoS $\approx 0.65$), especially after exceeding the energy limit of approximately 1500 time slots. As $\mathcal{M}$ increases from 5 to 20, MARHO completes approximately $2.7 \times 10^3$ to $5.3 \times 10^3$ tasks, maintaining an average delay between 18 and 55 time slots, and maintaining the highest cumulative reward. While Random and Greedy baselines suffer larger delays or reward collapse. PSO attains lower delay but with much lower throughput.**

a clear margin up to $\mathcal{M} = 20$, indicating robust multi-agent cooperation under constrained resources.

Fig. 3 reports action counts under 1200 slots. MARHO favors offloading, especially ship and relay, indicating a preference to exploit MEC capacity rather than local processing, consistent with the window-throughput objective and system constraints. IQL also increases ship offloading relative to Random and Greedy-Latency, but uses less relay than MARHO, reflecting partial exploitation without full queue-aware coordination. Greedy-Latency concentrates on local computing. PSO leans to local then offloading. Random remains roughly uniform across modes.

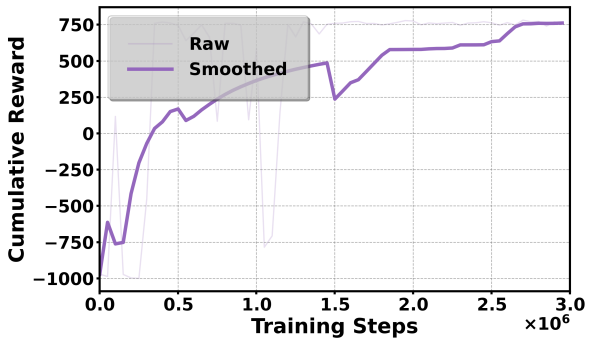



**FIGURE 5. Cumulative evaluation episode rewards of MARHO during training. The randomized weather training stars at $1.5 \times 10^6$ steps.**

### C. Robustness to Extended Missions

This experiment evaluates policy robustness on a long mission where the system crosses the energy boundary. In our experiment setup, the fleet maintains nominal endurance for about 1500 slots. Beyond this point, some USVs and the UAV begin to hit the energy floor, which restricts actions and increases the cost of offloading. We extend the horizon to 1800 slots to test whether the policy can adapt its offloading mix once energy becomes scarce.

During training, each episode samples a weather profile from the set: sun, night, rain, and storm, with LoS probability and path loss drawn according to the ITU-R maritime model, which provides data augmentation and prevents overfitting to a single visibility level. The 1,800-slot evaluation is conducted under rain. The LoS probability under rain is about 0.65, markedly lower than the 0.95 observed in sun, which reduces effective link rates.

Under these harsher conditions, MARHO continues to outperform all baselines. From Fig. 4(a), MARHO's finished-task count increases from about $2.7 \times 10^3$ at $\mathcal{M} = 5$ to $5.3 \times 10^3$ at $\mathcal{M} = 20$, maintaining near-linear scaling despite degraded channels. IQL performs second best, ranging from $2.2 \times 10^3$ to $4.7 \times 10^3$, showing improved adaptability over Random and Greedy-Latency but still lagging behind MARHO. The Random baseline grows slowly from $1.6 \times 10^3$ to $4.5 \times 10^3$, while PSO remains the lowest, capped below $3.0 \times 10^3$. These results confirm MARHO's ability to sustain throughput even when energy and communication resources become scarce.

Fig. 4(b) shows that MARHO keeps moderate average delay, increasing from roughly 18 to 55 slots as $\mathcal{M}$ grows, while maintaining the highest cumulative reward, as shown in Fig. 4(c). IQL maintains delay and reward between MARHO and Random, proving its stable yet less coordinated adaptation under constrained energy. Random and Greedy-Latency experience rapid delay escalation and reward degradation once energy constraints activate. PSO achieves the lowest delay but at a severe cost in throughput. The sliding-window reward with $w = 10$ continues to penalize long queues, enabling MARHO to preserve QoS even beyond the 1500-slot energy boundary.

### D. Convergence and Training Stability Analysis

We evaluate the convergence and stability of MARHO under the same configuration as the main experiments, with $\mathcal{M} = 20$ USVs and an episode length of 1800 time slots. All network, channel, and weather settings remain consistent with Section V, ensuring comparability between convergence and performance results.

Figure 5 illustrates the cumulative evaluation rewards. During the early phase below $1.5 \times 10^6$ steps, the agents


**FIGURE 6.** **Value loss convergence of the critic network. The loss stabilizes after $2.0 \times 10^6$ steps, indicating consistent value estimation under hybrid maritime conditions.**

learn stably under fixed weather conditions. At $1.5 \times 10^6$ steps, random weather transitions including clear, rain, and storm profiles are introduced. A temporary performance drop appears, followed by rapid recovery, indicating adaptive policy adjustment. The cumulative reward later stabilizes near $7.5 \times 10^2$, showing that MARHO reaches a steady operating point and maintains robust policy behavior under hybrid LoS and NLoS channel dynamics.

The value loss reflects the accuracy of long-term return estimation by the critic and is a direct indicator of convergence in CTDE-based reinforcement learning. In MARHO, the critic minimizes

$$L_V = \mathbb{E}_t\left[\left(V_\psi(s_t) - \hat{R}_t\right)^2\right]. \tag{61}$$

Where $V_\psi(s_t)$ denotes the predicted state value and $\hat{R}_t$ the empirical discounted return. A monotonic decline in $L_V$ implies reduced estimation bias and improved stability of the policy gradient update.

As shown in Figure 6, the value loss steadily decreases throughout training and converges below 0.1 after $2.0 \times 10^6$ steps. The small oscillation observed near $1.5 \times 10^6$ steps corresponds to the introduction of weather variability, consistent with the transient drop in cumulative reward. The aligned convergence of reward and value loss demonstrates that MARHO achieves stable learning and reliable value approximation under realistic maritime MEC conditions.

## VI. CONCLUSION

This paper studied hybrid task offloading in maritime MEC networks with USVs, a UAV relay, and a ship server operating under time-varying LoS and NLoS channels. We formulated the problem with queue dynamics and heterogeneous links, and developed the MARHO framework based on CTDE to adapt offloading decisions to channel and workload variations. Across different weather profiles and traffic levels, MARHO completed more tasks and kept delay near the desired range, outperforming heuristic methods under both nominal and low-LoS conditions.

The current design is limited to a single UAV-ship pair and relies on offline simulations, which do not capture large-scale mobility or hardware constraints. Future work will extend MARHO to multi-UAV and multi-ship settings, consider online channel estimation and safety restrictions, and evaluate the framework in real maritime experiments.

## REFERENCES

[1] M. Jahanbakht, W. Xiang, L. Hanzo, and M. Rahimi Azghadi, "Internet of Underwater Things and Big Marine Data Analytics—A Comprehensive Survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 904–956, 2021.

[2] H. Zhang, Y. Qiu, K. Long, G. K. Karagiannidis, X. Wang, and A. Nallanathan, "Resource Allocation in NOMA-Based Fog Radio Access Networks," *IEEE Wireless Communications*, vol. 25, no. 3, pp. 110–115, 2018.

[3] M. M. Esfahani, A. Hariri, and O. A. Mohammed, "A multiagent-based game-theoretic and optimization approach for market operation of multimicrogrid systems," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 1, pp. 280–292, 2019.

[4] Z. Cao, P. Zhou, R. Li, S. Huang, and D. Wu, "Multiagent deep reinforcement learning for joint multichannel access and task offloading of mobile-edge computing in industry 4.0," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6201–6213, 2020.

[5] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html

[6] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *International Conference on Learning Representations (ICLR 2016)*, 2016. [Online]. Available: https://arxiv.org/abs/1509.02971

[7] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*, 2017, pp. 6379–6390. [Online]. Available: https://arxiv.org/abs/1706.02275

[8] R. Kumar, C. K. Singh, P. K. Upadhyay, A. M. Salhab, A. A. Nasir, and M. Masood, "IoT-inspired cooperative spectrum sharing with energy harvesting in UAV-assisted NOMA networks: Deep learning assessment," *IEEE Internet of Things Journal*, vol. 10, no. 24, pp. 22 182–22 196, 2023.

[9] D. Malak, F. V. Mutlu, J. Zhang, and E. M. Yeh, "Transmission delay minimization via joint power control and caching in wireless hetnets," in *2021 19th International Symposium on Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks (WiOpt)*, 2021, pp. 1–8.

[10] A. H. A. Bafghi, M. Mirmohseni, F. Ashtiani, and M. Nasiri-Kenari, "Joint optimization of power consumption and transmission delay in a cache-enabled c-ran," *IEEE Wireless Communications Letters*, vol. 9, no. 8, pp. 1137–1140, 2020.

[11] C. Zeng, J.-B. Wang, C. Ding, H. Zhang, M. Lin, and J. Cheng, "Joint optimization of trajectory and communication resource allocation for unmanned surface vehicle enabled maritime wireless networks," *IEEE Transactions on Communications*, vol. 69, no. 12, pp. 8100–8115, 2021.

[12] B. Tian, L. Wang, L. Xu, W. Pan, H. Wu, L. Li, and Z. Han, "UAV-Assisted Wireless Cooperative Communication and Coded Caching: A Multiagent Two-Timescale DRL Approach," *IEEE Transactions on Mobile Computing*, vol. 23, no. 5, pp. 4389–4404, 2024.

[13] W. Hao, M. Zeng, G. Sun, and P. Xiao, "Edge cache-assisted secure low-latency millimeter-wave transmission," *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 1815–1825, 2020.

[14] Y. Du, J. Li, L. Shi, T. Liu, F. Shu, and Z. Han, "Two-tier matching game in small cell networks for mobile edge computing," *IEEE Transactions on Services Computing*, vol. 15, no. 1, pp. 254–265, 2022.

[15] A. Cabrera, A. Acosta, F. Almeida, and V. Blanco, "A heuristic technique to improve energy efficiency with dynamic load balancing," *The Journal of Supercomputing*, vol. 75, pp. 1610–1624, 2019.

[16] J. Huang, P. Majumder, S. Kim, A. Muzahid, K. H. Yum, and E. J. Kim, "Communication algorithm-architecture co-design for distributed deep learning," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2021, pp. 181–194.

[17] R. Chen, R. Chen, L. Zhang, L. Li, R. Hou, R. Hou, T. Yang, T. Yang, T. Yang, T. Yang, L. Wang, L. Wang, M. Pan, and M. Pan, "Data-driven

optimization for resource provision in non-cooperative edge computing market," *null*, 2020.
[18] M. Dai, Y. Wu, L. Qian, Z. Su, B. Lin, and N. Chen, "UAV-Assisted Multi-Access Computation Offloading via Hybrid NOMA and FDMA in Marine Networks," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 1, pp. 113–127, Jan. 2023.
[19] T. Yang, T. Yang, T. Yang, Z. Jiang, Z. Jiang, R. Sun, R. Sun, R. Sun, R. Sun, N. Cheng, N. Cheng, H. Feng, H. Feng, and H. Feng, "Maritime search and rescue based on group mobile computing for unmanned aerial vehicles and unmanned surface vehicles," *IEEE Transactions on Industrial Informatics*, 2020.
[20] F. Lu, G. Liu, W. Lu, Y. Gao, J. Cao, N. Zhao, and A. Nallanathan, "Resource and trajectory optimization for UAV-relay-assisted secure maritime mec," *IEEE Transactions on Communications*, vol. 72, no. 3, pp. 1641–1652, 2023.
[21] X. Li, W. Huangfu, X. Xu, J. Huo, and K. Long, "Secure offloading with adversarial multi-agent reinforcement learning against intelligent eavesdroppers in UAV-Enabled mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 13 914–13 928, 2024.
[22] Y. Wang and Y. Zhao, "Multiple ships cooperative navigation and collision avoidance using multi-agent reinforcement learning with communication," 2024. [Online]. Available: https://arxiv.org/abs/2410.21290
[23] G. Sun, L. He, Z. Sun, Q. Wu, S. Liang, J. Li, D. Niyato, and V. C. M. Leung, "Joint task offloading and resource allocation in aerial-terrestrial UAV networks with edge and fog computing for post-disaster rescue," *IEEE Transactions on Mobile Computing*, vol. 23, no. 9, pp. 8582–8600, 2024.
[24] V. Niazmand and Q. Ye, "Joint task offloading, dnn pruning, and computing resource allocation for fault detection with dynamic constraints in industrial iot," *IEEE Transactions on Cognitive Communications and Networking*, pp. 1–1, 2025.
[25] M. Lee and C. W. Anderson, "Relevance vector sampling for reinforcement learning in continuous action space," in *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2016, pp. 774–779.
[26] H. Tang, Y. Hu, F. Zhao, J. Yan, T. Dong, and W. Ding, "M3arl: Moment-embedded mean-field multi-agent reinforcement learning for continuous action space," in *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024, pp. 7250–7254.
[27] A. Al-Hourani, S. Kandeepan, and A. Jamalipour, "Modeling air-to-ground path loss for low altitude platforms in urban environments," in *2014 IEEE Global Communications Conference*, 2014, pp. 2898–2904.
[28] 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.
[29] K. Yang, A. F. Molisch, T. Ekman, T. Røste, and M. Berbineau, "A round earth loss model and small-scale channel properties for open-sea radio propagation," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 9, pp. 8449–8460, 2019.

**Jiahong Ning**(Graduate Student Member, IEEE) received the M.S. degree in electrical engineering at Minnesota State University, Mankato, MN, USA, and he is currently pursuing Ph.D degree at Dalian Maritime University. His research interests encompass machine learning, large-scale distributed learning, wireless distributed learning, and wireless edge computing. And he is visiting in the Institute for Infocomm Research ($I^2R$), Agency for Science, Technology, and Research (A*STAR), Singapore.

**Aimin Li**(Student Member, IEEE) received his B.S. degree in electronics and information engineering from Harbin Institute of Technology (Shenzhen) in 2020, where he won the Best Thesis Award. He is currently pursuing the Ph.D. degree with the Department of Electronic Engineering, Harbin Institute of Technology (Shenzhen). From 2023 to 2024, he has visited the Institute for Infocomm Research ($I^2R$), Agency for Science, Technology, and Research (A*STAR), Singapore. His research interests include aerospace communications, advanced channel coding techniques, information theory, and wireless communications. He has served as a reviewer for IEEE TIT, IEEE JSAC, IEEE TCOM, IEEE TWC, IEEE TNNLS, IEEE TCCN, IEEE ISIT, etc. He has also served as a session chair for IEEE Information Theory Workshop 2024 and IEEE Globecom 2024.

**Gary C.F. Lee**(Member, IEEE) received his B.S. degree in Electrical Engineering in Stanford University in 2016, his M.S. degree in Electrical Engineering in Massachusetts Institute of Technology (MIT) in 2019, and his P.hD. degree in Electrical Engineering in Massachusetts Institute of Technology (MIT) in 2023. He is currently a senior scientist in the Institute for Infocomm Research ($I^2R$), Agency for Science, Technology, and Research (A*STAR), Singapore. His current research interest is in wireless communications, machine learning, and signal processing.

**Sumei Sun**(Fellow, IEEE) is a Principal Scientist, Distinguished Institute Fellow, and Acting Executive Director of the Institute for Infocomm Research ($I^2R$), Agency for Science, Technology, and Research (A*STAR), Singapore. She is also holding a joint appointment with the Singapore Institute of Technology, and an adjunct appointment with the National University of Singapore, both as a full professor. Her current research interests are in next-generation wireless communications, cognitive communications and networks, industrial internet of things, communications-computing-control integrative design, joint radar-communication systems, and signal intelligence. She is Editor-in-Chief of the IEEE Open Journal Vehicular Technology, and Steering Committee Chair of the IEEE Transactions on Machine Learning in Communications and Networking. She is also Member-at-Large of the IEEE Communications Society, a member of the IEEE Vehicular Technology Society Board of Governors (2022-2024), Fellow of the IEEE and the Academy of Engineering Singapore.

**Tingting Yang**(M'13) received the Ph.D. degrees from Dalian Maritime University, China, in 2010. She is currently a professor in the School of Electrical Engineering and Intelligentization, Dongguan University of Technology, China. Since September 2012, she has been a visiting scholar at the Broadband Communications Research (BBCR) Lab at the Department of Electrical and Computer Engineering, University of Waterloo, Canada. Her research interests are in the areas of maritime wideband communication networks, DTN networks, and green wireless communication. She serves as the associate Editor-in-Chief of the IET Communications, as well as the advisory editor for SpringerPlus. She also serves as a TPC Member for IEEE ICC'14, ICC'15.