Deep Reinforcement Learning-Based Joint Resource Allocation and Service Migration for Smart-Buoy-Enabled Maritime Multiaccess Edge Computing Networks

Hangqi Li, *Member, IEEE*, Yuzhang Wang, Mingyang Pan, Shaoxi Li, and Wei Guan, *Member, IEEE*

***Abstract*—Multiaccess edge computing (MEC) is a promising technology that can enhance the computational capabilities of maritime Internet of Things (MIoT) systems. By deploying smart buoys equipped with MEC servers (MSs) and base stations (BSs) along fairway boundaries, the distributed computing cluster can significantly improve the timeliness and efficiency in processing latency-sensitive and resource-intensive MIoT applications. However, due to the limited computational resources and service coverage of these smart buoys, mobile MIoT users (MUs), e.g., vessels, may experience degraded access latency and service discontinuity. Resource allocation (RA) and service migration serve as effective techniques for integrating and coordinating the communication, computing, and energy resources of MIoTs. This article investigates a joint RA and service migration (JRASM) optimization problem for smart-buoy-enabled maritime MEC networks to minimize the long-term task processing latency of MUs while ensuring service continuity. The JRASM optimization addresses the coupled challenges of resource competition, random task generation, and vessel mobility constraints, while integrating collision avoidance mechanisms. Subsequently, the mixed-integer nonlinear programming problem is reformulated as a Markov decision process (MDP) and solved by employing deep reinforcement learning (DRL) methods. A novel branching double deep Q network (Branching DDQN)-based JRASM algorithm is designed to decompose the hybrid discrete-continuous action space into independent subspaces, achieving enhanced learning efficiency while reducing parameters by 40%. Simulation results demonstrate that the proposed Branching DDQN-based JRASM algorithm achieves significantly superior performance in terms of latency reduction and task completion rate compared to other DRL-based JRASM baselines, particularly under high-workload scenarios.**

Received 15 July 2025; revised 1 September 2025; accepted 17 September 2025. Date of publication 22 September 2025; date of current version 8 December 2025. This work was supported in part by Guangxi Science and Technology Infrastructure and Talent Development Program under Grant GUIKE AD25069109; in part by the National Natural Science Foundation of China under Grant 52501436 and Grant 52171342; in part by the Department of Education of Liaoning Province about Fundamental Research Project under Grant LJKMZ20220376; and in part by the Fundamental Research Funds for the Central Universities under Grant 3132023152 and Grant 3132025801. *(Corresponding author: Shaoxi Li.)*

The authors are with the Navigation College, Dalian Maritime University, Dalian, Liaoning 116026, China (e-mail: lhq@dlmu.edu.cn; wyz1120231073@dlmu.edu.cn; panmingyang@dlmu.edu.cn; lishaoxi@dlmu.edu.cn; gwwtxdy@dlmu.edu.cn).

Digital Object Identifier 10.1109/JIOT.2025.3612960

***Index Terms*—Deep reinforcement learning (DRL), maritime multiaccess edge computing (MEC) networks, resource allocation (RA), service migration, smart buoy.**

NOMENCLATURE

$T_k^s(t)$	Task processing latency of vessel k.
$T_{k,n}^D(t)$	Transmission delay of task data from vessel k to MEC server (MS) n.
$T_{k,n}^C(t)$	Computation delay for MS n' to process the tasks of vessel k.
$T_{k,n'\to n}^{\text{mig}}(t)$	Service migration delay of vessel k.
$\rho_{k,n}(t)$	vessel-server association.
$Q_k^{\text{req}}(t)$	Task data size of vessel k.
$R_{k,n}^{\text{tran}}(t)$	Data transmission rate from vessel k to MS n.
W	Uplink channel bandwidth of MS.
N_0	Noise power spectral density.
$P_{k,n}^{\text{tx}}(t)$	Transmission power of vessel k.
$P_{k,n}^{\text{rx}}(t)$	Receive power at MS n.
λ	Carrier wavelength.
$d_{k,n}(t)$	Distance between vessel k and MS n.
g_k	Computation intensity of task from vessel k.
$f_{k,n}^{\max}(t)$	Maximum CPU cycles.
δ	Service migration time discount factor.
$\text{Dis}_{n'\to n}$	Distance between source MS n' and target MS n.
σ_n^{n+1}	Boundary lines of the overlapping coverage areas between adjacent servers.
Are_n	Circular service coverage area of MS n.
R_n	Service coverage radius of MS n.
$v_k(t)$	Navigation speed of vessel k.
$y_{\text{fair}}^{\max}$	Upper fairway boundary.
$y_{\text{fair}}^{\min}$	Lower fairway boundary.
d_{safe}	Minimum safety distance between vessels.
$d_{i,k}^{\text{long}}$	Longitudinal safety distance between vessels.
$d_{i,k}^{\text{late}}$	Transverse safety distance between vessels.
D_{fair}	Total width of fairway.
D_{sep}	Width of separation zone.
$V_k^{\min}$	Minimum navigation speed of vessel.
$V_k^{\max}$	Maximum navigation speed of vessel.
$E_{k,n}^D(t)$	Energy consumption of data transmission of vessel k.

I. Introduction

THE frequent and dynamic maritime activities, ranging from commercial shipping to marine scientific research, as well as search and rescue operations, are driving a growing demand for more efficient and reliable maritime communication systems. With the widespread deployment of unmanned surface vehicles (USVs), various emerging computation-intensive and latency-sensitive marine applications, e.g., real-time processing of navigation data, dynamic path planning, intelligent collision avoidance decisions, and autonomous navigation, are further intensifying requirements for next-generation maritime networks. As a specialized Internet of Things (IoT) paradigm tailored for marine scenarios, maritime IoTs (MIoTs) have been regarded as a promising technological paradigm for upgrading maritime networks [1]. To address the requirements of low-latency, energy-efficient, and reliable data processing for MIoT users (MUs), it is essential to enhance computing capabilities and optimize resource utilization in order to reduce service latency and ensure service continuity, thereby improving overall MIoT performance.

Multiaccess edge computing (MEC), a distributed near-source computing framework widely adopted in IoT domains, e.g., smart cities, industrial IoTs, vehicular networks, healthcare IoTs, and smart grids, can offer a viable solution for meeting low-latency and energy-efficient data processing needs of MUs [2], [3], [4], [5], [6]. However, traditional MEC access schemes predominantly employ mathematical modeling and optimization algorithms that heavily depend on precise environmental characterization, making them generally suitable only for static or low-mobility scenarios. When deployed in scenarios with highly dynamic task generation and burst data traffic, e.g., MEC-based MIoT scenarios, the optimization-based algorithms have fundamental limitation that prevents adaptive strategy adjustments due to their high computational complexity, ultimately resulting in system performance degradation. Furthermore, compared with terrestrial IoTs, deploying MEC servers (MSs) in the MIoT environment still faces unique challenges, such as the sparsity and limited coverage of marine infrastructures, resource constraints of shipborne equipment, and difficulties in power supply. These challenges are unfavorable for MUs in high traffic maritime zones, e.g., port vicinities, coastal waters, and narrow channels. Therefore, deep reinforcement learning (DRL) [7], [8] has been adopted and widely used in MEC networks, as a viable solution for addressing resource management and task offloading challenges in MEC networks, owing to its model-free learning capability and dynamic decision-making capacity without requiring precise environmental description [9], [10], [11].

Benefiting from the tremendous potential of MEC technology, its server deployment and network performance optimization in MIoT environments have attracted extensive research in recent years [12], [13], [14], [15], [16]. Among these, the task offloading energy consumption, service latency, and efficient resource allocation (RA) of edge servers constitute key research focuses. For maritime intelligent transportation applications, e.g., USV's autonomous navigation, service continuity is another crucial performance metric for data processing operations. However, research on service continuity for computational offloading in MEC-based MIoT remains notably absent in the existing literature. Service migration is widely adopted in high-mobility networks, though it introduces additional migration latency [17]. Since service migration alters the serving MS node, the RA decisions and limited computing capacity of the target MS determine postmigration task processing latency, which in turn influences migration decision-making. Consequently, upgrading or modernizing existing maritime communication and navigation systems to establish maritime MEC networks, while jointly optimizing RA and service migration strategies, is essential and beneficial for improving latency performance of MUs. To address the aforementioned challenges in MIoTs, we focus on reducing the offloading service latency while ensuring the service continuity of MUs in smart-buoy-enabled maritime MEC networks.

However, the mobility of MUs, the unknown channel state information (CSI), and the randomness of MUs' tasks in maritime environment make the proposed multivariable coupled task offloading problem difficult to solve using traditional optimization methods. Hence, we design a joint RA and service migration (JRASM) optimization framework with DRL in smart-buoy-enabled maritime MEC networks. This DRL-based task offloading scheme dynamically interacts with the complex and time-varying maritime communication environment, enabling autonomous parameter optimization through continuous learning without prior system knowledge. By integrating the low-latency processing of MEC with the adaptive learning capability of DRL, the proposed framework enables real-time analysis of network states and quality of service (QoS) metrics for MUs, dynamically determining vessel's transmission power allocation, vessel-server association parameters, and boundary lines of the overlapping coverage areas between adjacent servers. Moreover, our framework addresses the energy-latency tradeoff in task offloading under resource-constrained maritime environment through the long-term reward mechanism of DRL, effectively reducing total task processing latency of all MUs and improving resource utilization efficiency. Compared with the traditional optimization approaches, our proposed integrated framework provides greater flexibility and more efficient service in handling the high dynamism and complexity of MIoT environments. The main contributions of this article can be summarized as follows.

1) *System Model:* Unlike previous static coverage models, a smart-buoy-enabled maritime MEC architecture is constructed, where multiple channel-marker buoys equipped with integrated communication, computing, and dynamic coverage adjustment capabilities are deployed to form a distributed computing cluster. This architecture addresses communication coverage gaps resulting from sparse offshore base stations (BSs) while supporting dynamic task offloading and low-latency computing services for vessels. It is important for supporting vessel mobility yet remains largely unexplored in the existing buoy-assisted maritime MEC research. Specifically, a vessel mobility model following the traffic

separation scheme (TSS) and the International Regulations for Preventing Collisions at Sea (COLREGs) [18] is established. Furthermore, considering both direct and reflected paths in maritime communications, we adopt a dual-path propagation model [19] for maritime wireless channels to accurately characterize the deep fading effects in sparse scattering environments under dynamic weather conditions. Based on this characterization, we establish a reliable vessel information transmission framework.

2) *Problem Formulation:* To meet the latency-sensitive QoS requirements of maritime applications, a JRASM optimization problem is presented to minimize the total task processing latency of all MUs. The optimization framework incorporates multiple constraints on dynamic vessel trajectories, energy consumption of MUs during task offloading, interserver service migration requirements, minimum processing latency thresholds, and server load balancing conditions. Notably, the vessel trajectory constraints integrate a collision avoidance mechanism based on TSS rules and Fujii's ship domain model [20], enforcing constraints on both longitudinal and transverse safety distances, permissible speed ranges, and isolation zone, thereby ensuring high consistency with actual maritime scenarios. The proposed fine-grained vessel mobility and collision domain modeling establishes a practical foundation that has often been oversimplified in previous maritime and vehicle MEC service migration frameworks.
3) *Algorithm Design:* To address challenges, such as resource competition, network dynamics, random task generation, and the strong coupling of mixed-integer optimization variables in maritime environments, we reformulate the problem as a reinforcement learning (RL) task and employ a DRL-based JRASM algorithm for optimal decision-making. The proposed branching double deep Q network (Branching DDQN)-based JRASM algorithm decomposes the high-dimensional action space into discrete subspaces for transmission power selection, vessel-server association, and overlapping coverage boundary adjustment, reducing parameters by 40% and enhancing learning efficiency. Moreover, a reward function that jointly considers processing latency, migration delay, server load balancing, and power allocation guides the agent to dynamically optimize the tradeoff among low energy consumption, minimal latency, and fair resource utilization. This branching structure and explicit dynamic slicing optimization are absent in the existing DRL-based maritime offloading works, which mostly ignore real-time slice adaptation under vessel mobility.

The rest of this article is organized as follows. Section II reviews the related work. Section III describes a smart-buoy-enabled maritime MEC network model and formulates an optimization problem. The Branching DDQN-based JRASM algorithm is presented for solution in Section IV. Then, the proposed algorithm is compared with the baseline methods, with comprehensive simulation results presented in Section V. Finally, Section VI concludes this work and presents the future direction.

II. Related Works

A. RA and Computation Offloading in Maritime MEC Networks

To address the challenges of unstable maritime communication links, sparse node distribution, and limited device resources, various solutions have been proposed for RA and computation offloading in maritime MEC networks [21], [22], [23], [24], [25], [26], [27]. For example, Meng et al. [21] propose a joint perception-communication-computing-action co-optimization framework that enhances both communication quality and task execution in remote maritime mission-critical applications. Wu et al. [22] develop a multiple high-altitude platform (HAP)-assisted offloading architecture that reduces computational complexity through constraint relaxation. However, this approach is limited by its static nature and poor adaptability to mobility in dynamic maritime environments. Similarly, existing research on unmanned aerial vehicle (UAV)-assisted maritime MEC networks (e.g., Wang et al.'s [23] dual-edge strategy) remains constrained by static RA schemes, which fundamentally constrains their adaptability to maritime-specific mobility patterns and environmental dynamics. For mobile computation offloading systems involving USV or UAVs, deep Q-network (DQN) and deep deterministic policy gradient (DDPG)-based solutions [24], [25], [26] have emerged as a promising paradigm to optimize energy efficiency and offloading decisions. For example, Su et al. [27] propose a lightweight DRL algorithm that co-optimizes USV path planning and task offloading decisions, improving system practicality while reducing latency and energy consumption. However, this approach remains limited in addressing the complex dynamics of maritime environments. Moreover, the above DRL-based solutions typically neglect the need for dynamic network slicing, real-time adaptation to vessel mobility, and unpredictable sea-state variations.

B. Smart-Buoy-Based Edge Computing Solutions

Recently, smart buoys have evolved from simple navigational aids into intelligent and autonomous platforms with integrated sensing, communication, and computing capabilities [28], [29], [30]. For instance, Lyu et al. [29] deploy smart buoys in a maritime MEC network and develop a cloud-edge collaboration mechanism to facilitate data interaction between USVs, UAVs, and shore-based nodes. However, this work does not address task offloading energy consumption and dynamic RA optimization. Sun et al. [30] propose a joint optimization algorithm based on edge caching and power allocation, which enhances data processing efficiency. However, their approach still relies on static caching strategies without service migration capabilities. The Wi-Buoy system [31] introduces an energy-efficient, IEEE 802.11-based wireless buoy network for real-time, high-rate marine data acquisition. Although it achieves significant reductions in energy consumption and latency via a buoy power-saving backhaul scheme, its focus is primarily on communication backhaul

TABLE I
COMPARISON OF RELATED EDGE COMPUTING WORKS WITH OUR WORK

Works	Scenario	Migr.	Slicing	Mobility	Coll. Avoid.	Edge Deploy.	DRL	Compl.	Energy	Latency	RA Eff.
[22]	Space-Air-Ground-Sea	✗	✗	Not modeled	✗	Satellite+Buoy	✗	✗	✗	Medium	Low
[24]	Maritime UAV	✗	✗	Simple trajectory	✗	UAV only	DQN/DDPG	✗	Partial	Medium	Medium
[29]	Maritime MEC (Smart buoy)	✗	✗	Not modeled	✗	Buoy only	✗	✗	✗	High	Low
[30]	Maritime MEC (Smart buoy)	✗	✗	Not modeled	✗	Buoy only	✗	✗	Medium	Medium	Medium
[31]	Maritime MEC (Smart buoy)	✗	✗	Not modeled	✗	Buoy only	✗	✗	Low	Low	Low
[32]	Maritime MEC (Smart buoy)	✗	✗	Not modeled	✗	Buoy only	✗	✗	Medium	Low	Low
[33]	IoV	✓	✗	Simple trajectory	✗	RSU	✗	✗	Partial	Medium	Medium
[34]	IoV	✓	Static	Trajectory prediction	✗	RSUs	✗	✓	Partial	Medium	Medium
[35]	IoV	✗	Dynamic	Traffic-aware	✗	Core edge	✗	✓	Partial	Medium	High
Our Work	**Maritime MEC (Smart buoy)**	✓	**Dynamic**	**Fine trajectory + domain**	✓	**Smart buoy**	**Branching DDQN**	✓	**Low**	**Low**	**High**

✓: Feature present; ✗: Feature absent; Partial: Partially considered; Migr.: Migration; Coll. Avoid.: Collision Avoidance; Edge Deploy.: Edge Deployment; Compl.: Completeness; RA Eff.: RA Efficiency.

rather than on computation offloading or adaptive RA. Lately, Zhang et al. [32] integrate edge computing units on smart buoys to perform dynamic spatiotemporal fusion of automatic identification system (AIS) and video data for inland waterway surveillance, enhancing ship detection accuracy and improving real-time fusion robustness. However, this approach remains limited to sensing and fusion tasks and does not address key functionalities, such as service migration or network slicing.

Overall, the existing smart-buoy-enabled maritime systems primarily emphasize communication range extension, static caching, and perception tasks, leaving the potential for collaborative control, dynamic RA, and intelligent decision-making underexplored. Furthermore, these studies lack a comprehensive investigation of system performance, particularly regarding task offloading energy consumption, service latency, and RA efficiency in dynamic maritime environments. Consequently, our work incorporates these performance metrics into a comparison (see Table I), thereby establishing a clear superiority over prior smart-buoy-based research. This advancement is achieved through joint service migration, dynamic slicing, fine-grained vessel mobility modeling, and DRL-based optimization.

C. Service Migration and Network Slicing

The limited coverage range of maritime MSs often leads to QoS degradation or service interruptions when high-mobility vessels leave their associated MS coverage areas. How to maintain service continuity and improve resource utilization remains a challenging problem to be addressed in actual maritime MEC networks. Service migration and dynamic network slicing solutions developed for terrestrial and vehicular networks offer valuable insights for improving service continuity and resource adaptability in maritime scenarios [33], [34], [35], [36]. Li et al. [33] employ a divide-and-conquer optimization strategy that decouples multitask migration and offloading decisions based on vehicle mobility patterns. Liao et al. [34] introduce a distance-first greedy algorithm to achieve complete task migration and load balancing in vehicular edge computing networks. Meanwhile, Gabilondo et al. [35] propose a dynamic mobile network slicing framework that analyzes vehicular traffic patterns and adapts slicing policies in real time to optimize RA for autonomous driving applications. Minardi et al. [36] design a flexible virtual network embedding (VNE)-based network slicing solution for integrated satellite-terrestrial 6G networks, jointly optimizing slice migration and load balancing under dynamic mobility and handover scenarios. However, the existing service migration and dynamic network slicing solutions designed for terrestrial and Internet of Vehicles (IoV) environments cannot be directly transplanted to maritime scenarios due to fundamental differences, such as sparse BS distribution, vessel-specific mobility patterns, and time-varying maritime channel characteristics. Specifically, terrestrial MEC systems have predictable mobility models and relatively stable channel conditions for slicing management, while maritime MEC networks must account for unique factors, such as vessel collision avoidance, fairway navigation regulations, and dynamic sea-state conditions. Therefore, integrating service migration and network slicing into maritime MEC frameworks requires novel approaches that can simultaneously address these maritime-specific challenges.

Table I compares our work with the existing studies in the fields of maritime edge computing and IoV, focusing on service migration, network slicing, mobility modeling, and edge computing deployment. To further evaluate our work, we compare it in Table I with previous studies on key performance metrics for maritime MEC networks, namely, task offloading energy consumption, service latency, and RA efficiency, which have largely been omitted in previous works on smart buoys. Overcoming the limitations of existing maritime edge computing networks (e.g., [22], [24], [29], [31]), our method integrates dynamic boundary-aware slicing, vessel mobility modeling, and a DRL-driven optimization framework,

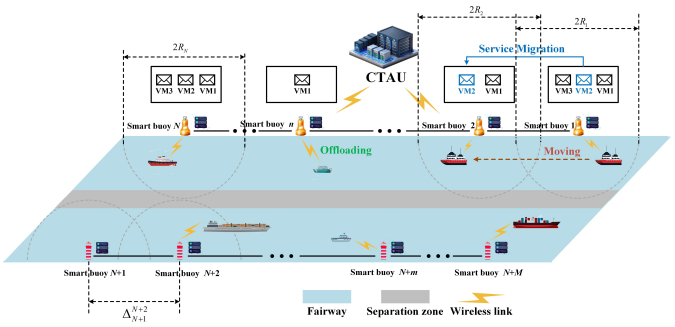


Fig. 1. Smart-buoy-enabled maritime MEC networks.

enabling real-time decision-making to address the unique challenges of maritime communications. This critical capability is not adequately addressed in the existing studies, which primarily focus on static caching and perception tasks (e.g., [22], [24], [29], [33], [34]). Furthermore, unlike service migration in IoV, which relies on traditional optimization methods or static RA (e.g., [33], [34], [35]), our DRL-based JRASM solution is specifically tailored for maritime environments. It incorporates fine-grained vessel trajectory constraints, including a collision avoidance mechanism based on TSS rules and Fujii's ship domain model, ensuring high consistency with real-world maritime scenarios.

III. System Model and Problem Formulation

In this section, we establish a maritime MEC framework comprising system model for describing vessel-edge interactions, vessel motion model, and a JRASM optimization problem formulation for minimizing the long-term task offloading latency of MUs. Hereafter, unless stated otherwise, "vessel" is used as a substitute for MU throughout this article. The key notations are provided in Nomenclature.

A. System Model

As illustrated in Fig. 1, our MIoT architecture integrates K vessels with N smart-buoy-based edge computing nodes deployed along fairway boundaries. Each node combines an MS and cellular BS, delivering computing services to vessels within its radio coverage area. The overall MIoT system operates over a cellular network, e.g., 4G/5G network, encompassing vessel-to-smart-buoys task offloading, computational result feedback, and peer-to-peer buoy communications. The sets of the vessels and MSs are denoted by $\mathcal{K} = \{1, 2, \ldots, k, \ldots, K\}$ and $\mathcal{N} = \{1, 2, \ldots, n, \ldots, N\}$, respectively. Smart buoy-enabled MSs provide computing, communication, and memory resources for the connected vessels through virtual machines (VMs) [37], where the VM set of each MS is defined as $\mathcal{Z} = \{1, 2, \ldots, z, \ldots, Z\}$. Each computational task exclusively occupies one VM instance [38], while a single MS can support multiple concurrent computational tasks simultaneously. The computational offloading tasks of the vessel can be processed in their connected MS or remotely in another smart buoy-based MS by traffic routing [37]. Specifically, the maritime MEC network incorporates a computational task allocation unit (CTAU), which, upon receiving offloading requests from vessels, dynamically assigns offloading tasks to MSs through joint evaluation of MS resource availability and vessel-to-server proximity. To address the contradiction between vessel mobility and restricted smart buoy-enabled MS coverage, the CTAU dynamically determines service migration decision for latency-sensitive tasks, minimizing task processing delays. The finite time domain is divided into T time slots of length Δt, i.e., the set of time slots is denoted by $\mathcal{T} = \{1, 2, \ldots, t, \ldots, T\}$, and vessels can initiate task computing requests to VM on MS within each time slot.

Compared to terrestrial communication systems, the maritime environment is characterized by sparser scatterers, dynamic sea surface conditions, and complex meteorological factors, making channel state prediction significantly more challenging. For this maritime MEC networks, we consider a dual-path propagation model incorporating both direct links and reflected paths for the communication between vessels and smart buoys. Referring to [19], the path loss of the two-path model between vessel k and MS n can be linearly described as

$$L^{\text{linear}}_{\text{2-ray},k,n}(t) = 10^{\frac{L^{\text{2-ray}}_{k,n}(t)}{10}} \tag{1}$$

where

$$L^{\text{2-ray}}_{k,n}(t) = -10 \lg \left\{ \left(\frac{\lambda}{4\pi d_{k,n}(t)} \right)^2 \left[2 \sin \left(\frac{2\pi H_1 H_2}{\lambda d_{k,n}(t)} \right) \right]^2 \right\}$$

λ is the wavelength, $d_{k,n}(t)$ represents the horizontal distance from the vessel k to the MS n at time slot t, and H_1 and H_2 denote the antenna heights of the vessel and the smart buoy, respectively.

B. Vessel Mobility Model

Based on the actual nearshore traffic separation [39] scenario, this article assumes that the smart buoys equipped with MS and cellular BS are deployed at the fairway boundary. For the vessel k (where $\forall k \in \mathcal{K}_U = \{1, 2, \ldots, I\}$ represents USV and $\forall k \in \mathcal{K}_M = \{I + 1, I + 2, \ldots, K\}$ denotes merchant vessels), we define its coordinate at time slot t as $(x_k(t), y_k(t))$. The trajectory of the vessel k can be given by

$$x_k(t+1) = x_k(t) + v_k(t) \cos(\theta_k(t)) \Delta t \tag{2}$$

$$y_k(t+1) = y_k(t) + v_k(t) \sin(\theta_k(t)) \Delta t \tag{3}$$

where $v_k(t)$ and $\theta_k(t)$ are the navigation speed (in knots) and the heading angle (in radians, relative to true north), respectively.

Noting that vessels navigating within TSS fairways must comply with the following navigation rules, e.g., fairway boundary clearance constraint and separation zone prohibition constraint, which can be, respectively, expressed as:

$$y^{\min}_{\text{fair}} + d_{\text{boundary}} \le y_k(t) \le y^{\max}_{\text{fair}} - d_{\text{boundary}} \tag{4}$$

$$y_k(t) > y^{\max}_{\text{sep}} \quad \text{or} \quad y_k(t) < y^{\min}_{\text{sep}} \tag{5}$$

where d_{boundary} is the safety distance, $y^{\max}_{\text{fair}}$ and $y^{\min}_{\text{fair}}$ are the upper and lower boundaries of the fairway, respectively, and $y^{\max}_{\text{sep}}$ and $y^{\min}_{\text{sep}}$ represent the upper and lower boundaries of the separation zone, respectively.

Furthermore, since overtaking constitutes the predominant encounter situation for vessels navigating within fairway, we establish the following fundamental safety constraint: all vessels must maintain a minimum safe distance to ensure navigational safety. This requirement is formally expressed as $\min d_{i,k}(t) \geqslant d_{\text{safe}}\ \forall i,k \in \mathcal{K}, i \neq k$, where d_{safe} is the safe distance between vessels and $d_{i,k}(t) = ((x_i(t) - x_k(t))^2 + (y_i(t) - y_k(t))^2)^{1/2}$ denotes the Euclidean distance between two vessels.

Specifically, the safe navigation distance between vessels is intrinsically linked to the concept of vessel domain, which has been widely adopted in maritime safety applications, such as collision risk assessment [40], collision avoidance systems, and maritime traffic simulation. In this study, we employ the vessel domain model originally proposed by Dr. Fujii [20] to determine vessel safety distances, which can be mathematically expressed as

$$\begin{cases} d_{i,k}^{\text{long}} = \alpha \mathcal{B} \\ d_{i,k}^{\text{late}} = \beta \mathcal{B} \end{cases} \tag{6}$$

where $d_{i,k}^{\text{long}}$ and $d_{i,k}^{\text{late}}$ represent the longitudinal and the lateral safe distances between the vessels i and k, respectively. α and β denote the longitudinal and the lateral safety coefficients with the values of 8 and 3.2, respectively. $\mathcal{B}$ is the length of the front vessel (in meters). It should be noted that the lateral safe distance $d_{i,k}^{\text{late}}$ for vessel navigation is simultaneously affected by fairway boundary clearance constraint and separation zone prohibition constraint. That is, if the lateral safety distance exceeds the width of the fairway, vessels cannot overtake. The limitations can be mathematically expressed as

$$d_{i,k}^{\text{late}} \leq \frac{D_{\text{fair}} - D_{\text{sep}}}{2} \tag{7}$$

$$\begin{cases} D_{\text{fair}} = y_{\text{fair}}^{\max} - y_{\text{fair}}^{\min} \\ D_{\text{sep}} = y_{\text{sep}}^{\max} - y_{\text{sep}}^{\min} \end{cases} \tag{8}$$

where D_{fair} represents the total width of the fairway and D_{sep} represents the total width of the separation zone.

Considering the unique maneuvering characteristics of vessels, to prevent potential loss of control at excessively low speeds and ensure adequate stopping capability during emergencies at high speeds, we propose the navigational speed range based on fairway conditions and other external factors, i.e., $V_k^{\min} \leqslant v_k(t) \leqslant V_k^{\max}$, where $V_k^{\min}$ ensures minimal steerage and controllability and $V_k^{\max}$ guarantees safe stopping distance in emergencies.

C. Service Migration Model

In order to optimize the latency performance of long-term tasks in maritime scenarios, our proposal aims to manage transmission delay through RA while considering VM migration between MSs to further decrease task processing latency for vessels. Notably, service migration is inherently tied to MSs coverage range. To detail the service migration mechanism, we introduce a topology-aware partitioning [41] framework for efficient interserver migration, as illustrated in Fig. 2.

In Fig. 2, we assume that Are_n and Are_{n+1} represent the circular service areas of MSs n and $n+1$ with coverage

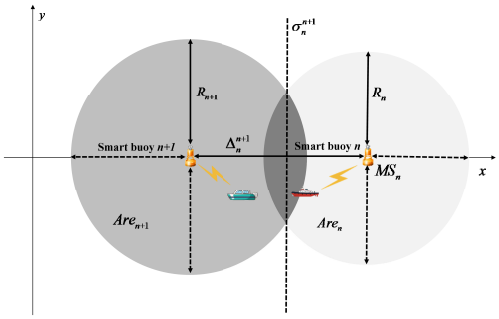


Fig. 2. Topology-aware partitioning framework.

radii R_n and R_{n+1}, respectively. These two adjacent MSs are separated by a distance $\Delta_n^{n+1} = x_n - x_{n+1}$, where x_n and x_{n+1} denote the coordinates of the MS n and $n+1$ in the horizontal axis, respectively. Let $\rho_{k,n}(t)$ denote a binary indicator where $\rho_{k,n}(t) = 1$ if and only if the vessel k is connected to the MS n at time t. The vessel-server association rule is defined as follows.

1) If the vessel k is located within the nonoverlapping coverage area of a single MS, it connects to that MS, i.e., $\rho_{k,n}(t) = 1$ for the serving MS n.
2) If the vessel k is within the overlapping coverage region of the two adjacent MSs n and $n+1$, the association follows the rule

$$\rho_{k,m}(t) = \begin{cases} 1, & \text{if } \left(m = n \wedge x_k(t) > \sigma_n^{n+1}\right) \\ 1, & \text{if } \left(m = n+1 \wedge x_k(t) \leq \sigma_n^{n+1}\right) \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

where σ_n^{n+1} is the boundary line of the overlapping coverage areas between the MSs n and $n+1$, which defines the service migration boundary. Let $R_n^{n+1} = R_n + R_{n+1}$, σ_n^{n+1} can be expressed by

$$\begin{cases} (x_n - R_n) < \sigma_n^{n+1} < \left(x_{n+1} + R_{n+1}\right), & \text{if } R_n^{n+1} > \Delta_n^{n+1} \\ \sigma_n^{n+1} = x_{n+1} + R_{n+1}, & \text{if } R_n^{n+1} = \Delta_n^{n+1}. \end{cases} \tag{10}$$

Thus, the binary variable $\rho_{k,n' \to n}(t) \in \{0,1\}$ indicates whether a service migration occurs, i.e., $\rho_{k,n' \to n}(t) = 1$ denotes a migration from the MS n' to n at time slot t, and $\rho_{k,n' \to n}(t) = 0$ denotes no migration. The decision of service migration is determined by computation offloading according to the following.

1) If $\rho_{k,n}(t-1) = 0$ and $\rho_{k,n}(t) = 1$, the computational service from the vessel k is migrated from the MS n' to n at time slot t, i.e., $\rho_{k,n' \to n}(t) = 1$.
2) If $\rho_{k,n}(t-1) = 1$ and $\rho_{k,n}(t) = 0$, the computational service from the vessel k is migrated from the MS n to another MS n' at time slot t, i.e., $\rho_{k,n \to n'}(t) = 1$.
3) If $\rho_{k,n}(t-1) = 0$ and $\rho_{k,n}(t) = 0$, the vessel k is not connected to the MS n at both the current and previous time slots. Thus, no service migration occurs, i.e., $\rho_{k,n' \to n}(t) = 0$.
4) If $\rho_{k,n}(t-1) = 1$ and $\rho_{k,n}(t) = 1$, the vessel k remains connected to the MS n at both the current and previ-

ous time slots, and consequently, no service migration occurs, i.e., $\rho_{k,n'\to n}(t) = 0$.

D. Problem Formulation

1) Task Processing Latency: Task processing latency is significant for the navigation safety of USVs and the service qualities for commercial vessel crews, which has been regarded as an important performance metric for evaluating maritime MEC networks. According to the previous description of the system model, it can be known that the vessel k initiates a computing offloading request to the VM on the nearest MS allocated by CTAU. Therefore, the transmission rate of task data from the vessel k to the MS n can be given by the Shannon theory [19] as follows:

$$R_{k,n}^{\text{tran}}(t) = W \log_2\left(1 + \frac{P_{k,n}^{\text{rx}}(t)}{N_0 W}\right) \tag{11}$$

where W is the communication bandwidth and N_0 represents the power spectral density of additive white Gaussian noise (AWGN). $P_{k,n}^{\text{rx}}(t)$ denotes the received power from the vessel k at the MS n, which can be expressed by the ratio of the transmission power of the vessel $kP_{k,n}^{\text{tx}}(t)$ to the path loss

$$P_{k,n}^{\text{rx}}(t) = \frac{P_{k,n}^{\text{tx}}(t)}{L_{\text{2-ray},k,n}^{\text{linear}}(t)}. \tag{12}$$

Then, the vessel k receives the result returned by the server after processing is completed. As described in the previous system model, the vessel initiates an offloading request to the VM hosted on the MS allocated by CTAU. Upon completion of the task processing, the vessel receives the computed results returned from the server. The task processing latency for the vessel k at time slot $tT_k^s(t)$ is composed of three key components: the uplink data transmission delay $T_{k,n}^D(t)$, the computation delay at MS $T_{k,n}^C(t)$, and the conditional service migration delay $T_{k,n'\to n}^{\text{mig}}(t)$ (if migration is triggered).

1) *Transmission Delay:* The transmission delay of task data from the vessel k to the MS n can be expressed by

$$T_{k,n}^D(t) = \frac{Q_k^{\text{req}}(t)}{R_{k,n}^{\text{tran}}(t)} \tag{13}$$

where $Q_k^{\text{req}}(t)$ denotes the data size of task from the vessel k at time slot t.

2) *Computation Delay:* Let the computational intensity of the offloaded computational task k be denoted as g_k, then the server computation delay for the task can be given by the following:

$$T_{k,n}^C(t) = \frac{Q_k^{\text{req}}(t)\, g_k}{f_{k,n}^{\text{co}}(t)} \tag{14}$$

where $f_{k,n}^{\text{co}}(t)$ represents the effective CPU cycles of the VM executing computational task from the vessel k on MS n at time slot t. If other computational tasks running on MS are not considered [42], [43], [44], the effective CPU cycles are equal to the maximum CPU cycles $f_{k,n}^{\max}(t)$.

3) *Service Migration Delay:* The service migration delay from the source MS n' to the target MS n is given by

$$T_{k,n'\to n}^{\text{mig}}(t) = \delta \text{Dis}_{n'\to n} Q_k^{\text{req}}(t) \tag{15}$$

where δ represents the time discount factor for service migration and $\text{Dis}_{n'\to n}$ denotes the distance between the source MS n' and the target MS n.

The download delay for computing result is excluded as it is negligible for minimal edge computing results. Therefore, the long-term task processing latency for the vessel k at time slot t can be expressed as

$$T_{\text{service}} = \lim_{T\to\infty} \sum_{t\in\mathcal{T}} \sum_{k\in\mathcal{K}} \rho_{k,n}(t) \times \left[T_{k,n}^D(t) + T_{k,n}^C(t) + \rho_{k,n'\to n}(t)\, T_{k,n'\to n}^{\text{mig}}(t)\right]. \tag{16}$$

2) Energy Consumption: Let $E_{k,n}^D(t)$ indicates the transmission energy consumption of the vessel k for task uploading to the MS n, namely

$$E_{k,n}^D(t) = P_{k,n}^{\text{tx}}(t)\; T_{k,n}^D(t). \tag{17}$$

To ensure the normal operation and extended mission duration of USVs, we establish the following transmission energy constraint for the vessel k, i.e.,

$$E_k^{\text{total}}(t) \le E_k^{\max} \tag{18}$$

where $E_k^{\max}$ represents the maximum allowable energy stored in the vessel k.

3) Problem Formulation: As proposed in Section for the computational task offloading scheme where vessels offload tasks to MSs deployed at smart buoys along the navigation channel, with service migration occurring during vessel movement, our objective is to minimize the total task processing latency for K vessels by jointly optimizing vessel's transmission power allocation $P_{k,n}^{\text{tx}}$, vessel-server association parameters $\rho_{k,n}(t)$, and boundary lines of the overlapping coverage areas between adjacent servers σ_n^{n+1}. The optimization problem can be formulated as

$$\textbf{OP:} \min_{P_{k,n}^{\text{tx}}(t),\rho_{k,n}(t),\sigma_n^{n+1}} \lim_{T\to\infty} \frac{1}{T} \sum_{t\in\mathcal{T}} \sum_{k\in\mathcal{K}} T_k^s(t) \tag{19}$$

$$\text{s.t. } y_{\text{fair}}^{\min} + d_{\text{boundary}} \le y_k(t) \le y_{\text{fair}}^{\max} - d_{\text{boundary}} \tag{19a}$$

$$y_k(t) > y_{\text{sep}}^{\max} \text{or} \quad y_k(t) < y_{\text{sep}}^{\min} \tag{19b}$$

$$d_{i,k}(t) \ge d_{\text{safe}} \tag{19c}$$

$$d_{i,k}^{\text{long}} = \alpha \mathcal{B} \tag{19d}$$

$$d_{i,k}^{\text{late}} = \beta \mathcal{B} \tag{19e}$$

$$d_{i,k}^{\text{late}} \le \frac{D_{\text{fair}} - D_{\text{sep}}}{2} \tag{19f}$$

$$V_k^{\min} \le v_k(t) \le V_k^{\max} \tag{19g}$$

$$E_k^{\text{total}}(t) \le E_k^{\max} \tag{19h}$$

$$Q_k^{\text{req}}(t)\, g_k \le f_{n,z}^C(t) \tag{19i}$$

$$T_k^s(t) \le T_k^{\max} \tag{19j}$$

$$\sum_{n\in\mathcal{N}} \rho_{k,n}(t) = 1 \tag{19k}$$

$$\rho_{k,n}(t) \in \{0,1\} \tag{19l}$$

$$\begin{cases} (x_n - R_n) < \sigma_n^{n+1} < \left(x_{n+1} + R_{n+1}\right) \\ \qquad\qquad\qquad \text{if } R_n^{n+1} > \Delta_n^{n+1} \\ \sigma_n^{n+1} = x_{n+1} + R_{n+1}, \ \text{if } R_n^{n+1} = \Delta_n^{n+1} \end{cases} \tag{19m}$$

$$P_{k,\text{tx}}^{\min} \le P_{k,n}^{\text{tx}}(t) \le P_{k,\text{tx}}^{\max} \tag{19n}$$

$$\forall i \neq k \quad \forall i,\ k \in \mathcal{K}\ \forall n \in \mathcal{N}\ \forall z \in \mathcal{Z} \tag{19o}$$

where constraints (19a)–(19g) jointly characterize the vessels' trajectories with random speeds in the fairway channel and the intervessel collision avoidance mechanism. Constraint (19h) restricts the vessels' energy consumption to not exceed their available energy budget. Constraint (19i) ensures that the computational demand of each task running on the VM does not exceed the maximum computational capacity. Constraint (19j) enforces the task processing delay requirement for each vessel. Constraints (19k) and (19l) establish the vessel-server association, specifying that each task can be processed by at most one MS per time slot. Constraint (19m) defines the boundary thresholds for triggering service migration within spatial regions, where adjacent MS handovers may occur. Constraint (19n) limits the transmission power of vessels during computation offloading.

Due to the presence of discrete decision variables (e.g., $\rho_{k,n}(t)$) and tightly coupled optimization parameters, the proposed JRASM problem formulated in (19) is a mixed-integer nonlinear programming (MINLP) problem. Such problems are NP-hard and cannot be solved efficiently using conventional optimization techniques [45]. Moreover, vessels' mobility, random task generation, and time-varying maritime channel conditions result in a highly dynamic and uncertain environment. To address these challenges, our DRL-based solution methodology is designed in Section IV.

IV. DRL-Based JRASM Algorithm

RL is well suited for decision-making under uncertainty, as it can learn optimal policies through the trial-and-error interaction without requiring prior knowledge of the environment. However, when applied to large-scale and complex state-action spaces, RL algorithms face challenges, including slow convergence, inefficient exploration, and high computational overhead. The DQN algorithm integrates RL with deep learning, enabling it to handle high-dimensional state spaces via techniques, such as experience replay and target networks. Nevertheless, the DQN is susceptible to Q value overestimation, degrading policy stability and performance. To address this issue, the DDQN [46] approach was developed, which decouples action selection from evaluation. While the DDQN mitigates Q value overestimation, it still operates under the assumption of a 1-D action space, limiting its effectiveness in scenarios requiring multidimensional discrete actions. To overcome the abovementioned limitations and efficiently solve the proposed JRASM problem, we develop a Branching DDQN-based JRASM algorithm. By factorizing the high-dimensional composite action space into independent subspaces, our approach significantly reduces the number of parameters and improves learning efficiency.

In the following, the Markov decision process (MDP) framework for DRL is described. Subsequently, the Branching DDQN-based JRASM algorithm is proposed to find the optimal solution to the original optimization problem.

A. MDP Formulation

The MDP is widely regarded as one of the most effective frameworks for solving sequential decision-making problems [24]. Due to the high dynamics of vessels, the varying transmission power levels of users, the fluctuating load on MS, and the computational tasks randomly generated in each time slot, all of which exhibit the Markov property, we formulate the proposed optimization problem as an MDP. The MDP is formally defined as MDP $= (\mathcal{S}, \mathcal{A}, \mathcal{P}_{\text{sa}}, \mathcal{R})$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ denotes the action space, $\mathcal{R}$ signifies the immediate reward function, and $\mathcal{P}_{\text{sa}}$ represents the state transition probability. Considering that both the statistical properties and state transition probabilities of the optimization problem are unknown in the constructed maritime MEC environment, we hereby provide the concrete physical definitions of $(\mathcal{S}, \mathcal{A}, \mathcal{R})$ corresponding to (19) as follows.

1) State Space $\mathcal{S}$: The observed state space for the computational task generated by the vessel k at time slot t is defined as

$$s_t = \{\mathbf{P}(t), \mathbf{V}(t), \mathbf{F}(t), \mathbf{G}(t), \mathbf{Q}(t), \mathbf{g}(t), \mathbf{D}(t)\} \tag{20}$$

where $\mathbf{P}(t)$ represents the real-time positions of all vessels at time slot t, formulated as a $2K$-dimensional vector. $\mathbf{V}(t)$ denotes the velocity information of all vessels, including both magnitude $v_k(t)$ and heading direction $\theta_k(t)$, and is also represented as a $2K$-dimensional vector. $\mathbf{F}(t)$ captures the load status of all MSs at time slot t, forming a $(N+M)$-dimensional vector. $\mathbf{G}(t)$ is a vector of dimensions $K(N+M)$ that contains the CSI between all vessels and MSs. $\mathbf{Q}(t)$ specifies the data sizes of computation tasks to be offloaded to MSs at time slot t, represented as a K-dimensional vector. $\mathbf{g}(t)$ indicates the computational intensity associated with each task at time slot t and is likewise a K-dimensional vector. Lastly, $\mathbf{D}(t)$ contains the distances between vessels and MSs, forming another $K(N+M)$-dimensional vector.

2) Action Space $\mathcal{A}$: At each time slot t, the agent determines a joint action $a_t \in \mathcal{A}$ based on the observed system state $s_t \in \mathcal{S}$. This composite action vector comprises three components: transmission power allocation of vessels $P_{k,n}^{\text{tx}}$, vessel-server association $\rho_{k,n}(t)$, and boundary lines of the overlapping coverage areas between adjacent servers σ_n^{n+1}. Since the vessel-server association is a discrete variable, whereas both transmission power and service migration decisions are continuous, we apply uniform quantization to discretize the latter two. Accordingly, we adopt a joint action encoding scheme, wherein a single action index encapsulates the vessel-server association, the adjustment of boundary lines of the overlapping coverage areas between adjacent servers,

and the transmission power selection. The joint action index at time slot t is defined as follows:

$$a_t = h_t\,(\mathrm{YB}) + w_t B + b_t \tag{21}$$

where h_t denotes the index of the vessel-server association, with $h_t \in \{0, 1, \ldots, N-1\}$. w_t represents the index of the discretized transmission power level, where $w_t \in \{0, 1, \ldots, Y-1\}$. b_t denotes the index of the boundary-adjustment action for overlapping server-coverage regions, with $b_t \in \{0, 1, \ldots, B-1\}$. The parameter Y is the total number of quantized transmission power levels, and B is the number of quantization intervals for the boundary $\sigma_n^{n+1}(t)$. Accordingly, the vessel-server association action is calculated as $a_k^n(t) = \lfloor a_t/(\mathrm{YB}) \rfloor$, the transmission power action as $a_k^{\mathrm{tx}}(t) = \lfloor (a_t \bmod (\mathrm{YB}))/B \rfloor$, and the boundary-adjustment action as $a_n^{n+1}(t) = a_t \bmod (B)$, where mod(.) represents the modulo operation, i.e., the remainder after division. Consequently, the joint action space has a cardinality of $|\mathcal{A}| = (N+1)\mathrm{YB}$.

3) Immediate Reward Function $\mathcal{R}$: To minimize the total task processing latency, we define an immediate reward function that is negatively correlated with task processing latency. To ensure compliance with constraints (19i), (19m), and (19n), three penalty terms are incorporated. Accordingly, the immediate reward function at time slot t is formulated as follows:

$$r_t\,(s_t, a_t) = \sum_{k \in \mathcal{K}} \Big(\omega_1 J_k^s\,(t) + \omega_2 J_k^{\mathrm{mig}}\,(t) + \omega_3 J_k^{\mathrm{res}}\,(t) + \omega_4 J_{k,n}^{\mathrm{tx}}\,(t) \Big) \tag{22}$$

where

1) $J_k^s(t)$ reward associated with task processing latency. A positive reward is assigned when $T_k^s(t) < T_k^{\max}$, i.e., $J_k^s(t) > 0$; otherwise, a negative reward (penalty) is applied when $T_k^s(t) > T_k^{\max}$, i.e., $J_k^s(t) < 0$;
2) $J_k^{\mathrm{mig}}(t)$ reward related to service migration delay. If migration occurs, i.e., $T_{k,n' \to n}^{\mathrm{mig}}(t) > 0$, a penalty is imposed $J_k^{\mathrm{mig}}(t) < 0$; otherwise, if no migration takes place, i.e., $T_{k,n' \to n}^{\mathrm{mig}}(t) = 0$, the reward is set to zero $J_k^{\mathrm{mig}}(t) = 0$;
3) $J_k^{\mathrm{res}}(t)$ reward associated with the load of MS. When the MS's load exceeds 70% of its maximum capacity, a penalty is applied $J_k^{\mathrm{res}}(t) < 0$; otherwise, $J_k^{\mathrm{res}}(t) = 0$;
4) $J_{k,n}^{\mathrm{tx}}(t)$ reward associated with the transmission power of vessel k. A negative reward (penalty), i.e., $J_{k,n}^{\mathrm{tx}}(t) < 0$, is imposed if the current power allocation exceeds 80% of the maximum threshold or falls below 1.2 times the minimum threshold, mathematically expressed as $P_{k,n}^{\mathrm{tx}}(t) > 0.8\ P_{k,\mathrm{tx}}^{\max}$ or $P_{k,n}^{\mathrm{tx}}(t) < 1.2\ P_{k,\mathrm{tx}}^{\min}$. Otherwise, $J_{k,n}^{\mathrm{tx}}(t) = 0$.

In addition, ω_1, ω_2, ω_3, and ω_4 are the weighting coefficients. To prioritize minimizing the total task processing latency, ω_1 should be assigned a significantly higher value than other weighting coefficients, ensuring that the immediate reward function $r_t(s_t, a_t)$ lies in the range $[-1, 1]$. Furthermore, in practical maritime MEC scenarios, the relative importance of each reward component varies with the training phase and environmental state. Therefore, in order to achieve adaptive optimization, we dynamically update the weighting coefficients $\omega_1 \sim \omega_4$ by

$$\omega_j\,(e, \mathcal{E}) = \begin{cases} \omega_j^{(0)} + \eta_j^{\mathrm{early}}, & e < E_{\mathrm{th1}}, \quad j = 1 \\ [3pt]\, \omega_j^{(0)} + \eta_j^{\mathrm{mid}}, & E_{\mathrm{th1}} \le e < E_{\mathrm{th2}}, \ \ j \in \{2, 3\} \\ [3pt]\, \omega_j^{(0)} + \eta_j^{\mathrm{env}}\, g_j\,(\mathcal{E}), & \text{if resource-limited}, \quad j = 4 \\ [3pt]\, \omega_j^{(0)}, & \text{otherwise} \end{cases} \tag{23}$$

where e denotes the training episode index and $\mathcal{E}$ represents the observed environment state. The terms $\omega_j^{(0)}$ and $\eta_j^{(\cdot)}$ are the initial weight and adaptation rate for ω_j, respectively, while E_{th1} and E_{th2} are training phase thresholds. Finally, the function $g_j(\mathcal{E})$ maps environmental indicators (e.g., available bandwidth) to an adjustment factor.

4) Optimal Policy π^*: The MDP aims to derive an optimal policy $\pi^* : \mathcal{S} \to \mathcal{A}$ that maximizes the expected cumulative discounted long-term reward of the system in processing maritime tasks, given by

$$\pi^* = \arg\max_{\pi^*} \lim_{T \to \infty} \mathbb{E}_{\pi^*} \left[\sum_{t=0}^{T} \gamma r_t\,(s_t, a_t) \right] \tag{24}$$

where $\mathbb{E}_{\pi^*}[\cdot]$ denotes the expected value obtained under policy π^*. The discount factor $\gamma \in [0, 1]$ determines the agent's temporal preference, specifying the influence of future expected rewards on the current decision-making process.

B. Branching DDQN-Based JRASM Algorithm

To tackle the challenges of solving the proposed JRASM optimization problem characterized by a high-dimensional state space and a discrete action space, we adopt the Branching DDQN algorithm to train the agent and derive optimal strategies for computation offloading and task migration. The framework of the Branching DDQN-based JRASM algorithm is illustrated in Fig. 3.

The Branching DDQN algorithm retains the dual-network architecture of DDQN, consisting of an online network $Q(s_t, a_t; \theta_t)$ and a target network $Q(s_{t+1}, a_{t+1}; \theta_t^-)$. Here, θ_t denotes the parameters of the online network, which are updated at each training step, whereas θ_t^- represents the parameters of the target network, which are either updated periodically or via a soft update strategy. In contrast to conventional approaches that estimate Q-values over the entire joint action space, the Branching DDQN algorithm employs individual subnetworks for each action dimension, thereby enabling a distributed representation of the action space. Specifically, at time slot t, for each action dimension i, the algorithm defines an advantage function branch $Q_i(s_t, a_t^{(i)}; \theta_t)$, which is integrated into the overall Q value using the following aggregation mechanism, i.e.,

$$Q\,(s_t, a_t) = V_t\,(s) + \sum_{i=1}^{d} Q_i\left(s_t, {a_t}^{(i)}\right) - \frac{1}{\left|\mathcal{A}^{(i)}\right|} \sum_{i=1}^{d} \sum_{a^{(i)} \in \mathcal{A}^{(i)}} Q_i\left(s_t, {a_t}^{(i)}\right) \tag{25}$$

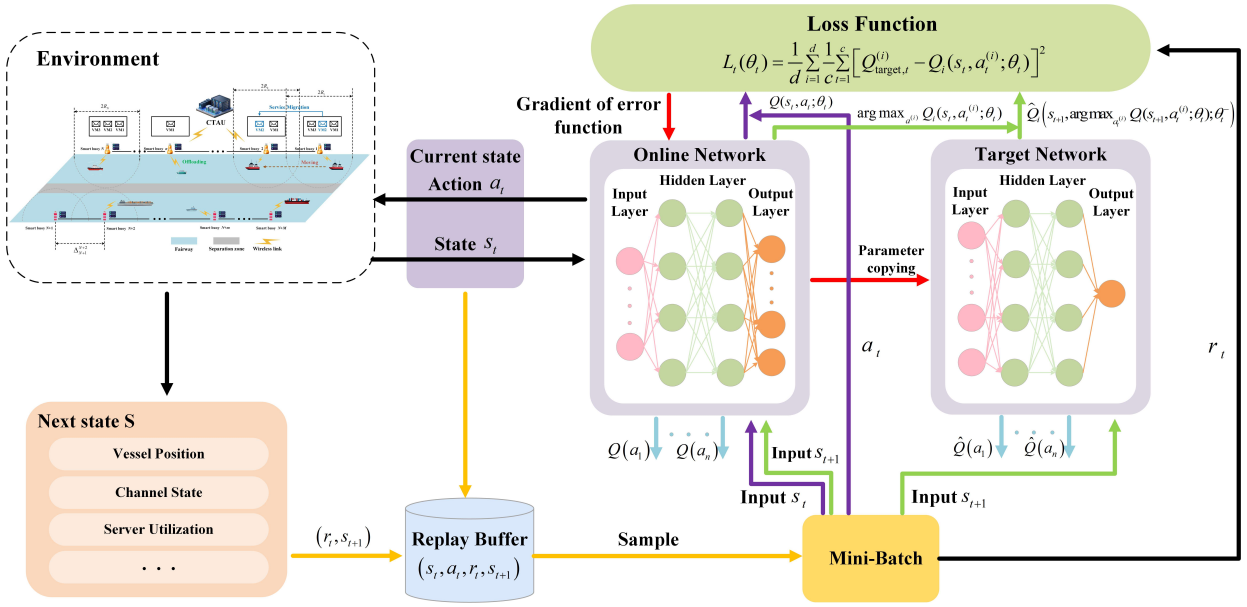


Fig. 3. Framework of Branching DDQN-based JRASM algorithm.

where $V_t(s)$ denotes the state-value function, d is the number of action dimensions, and $Q_i(s_t, a_t^{(i)})$ represents the advantage value for the ith action dimension. The final term applies normalization to maintain identifiability. This architecture enhances the efficiency of action selection while promoting faster convergence and more stable training.

During the training phase, the DDQN strategy is employed to alleviate the overestimation bias of Q-values. Specifically, the target Q value for each action branch is calculated by

$$Q^{(i)}_{\text{target},t} = r_t + \gamma Q_i\left(s_{t+1}, \arg\max_{a_t^{(i)}} Q_i\left(s_{t+1}, a_t^{(i)}; \theta_t\right); \theta_t^-\right) \tag{26}$$

where $Q^{(i)}_{\text{target},t}$ denotes the target Q value corresponding to the ith action branch. r_t represents the immediate reward obtained after executing action a_t in state s_t. The variable s_{t+1} denotes the subsequent state. The action $a_t^{(i)}$ refers to the action selected by the online network for the ith action dimension. The final joint action a_t^* is constructed by combining the optimal actions from all action branches, which is formulated as

$$a_t* = \left(a_t*^{(1)}, a_t*^{(2)}, \ldots, a_t*^{(d)}\right) \tag{27}$$

where $a_t*^{(i)}$ denotes the optimal action selected for the ith dimension, i.e., $a_t*^{(i)} = \arg\max_{a^{(i)}} Q_i(s_t, a_t^{(i)}; \theta_t)$.

The target network is employed to evaluate the Q value of the selected action. The objective of the training process is to minimize the average loss across all action branches, which can be expressed by

$$L_t(\theta_t) = \frac{1}{d}\sum_{i=1}^{d}\frac{1}{c}\sum_{t=1}^{c}\left[Q^{(i)}_{\text{target},t} - Q_i\left(s_t, a_t^{(i)}; \theta_t\right)\right]^2 \tag{28}$$

where c represents the batch size sampled from the experience replay buffer. The experience replay mechanism breaks the temporal correlations between consecutive samples and improves training stability.

During training, the Branching DDQN algorithm introduces an ε-greedy policy to select actions, balancing exploration and exploitation. Meanwhile, the target network parameters are updated using a soft update strategy. The specific update formula is given by

$$\theta^- \leftarrow \tau\theta + (1-\tau)\theta^- \tag{29}$$

where $\tau \in (0, 1)$ denotes the update rate.

In the customized environment constructed in this article, the agent is required to simultaneously determine the vessel-server association, adjust the boundary line of overlapping server coverage areas (i.e., service migration), and select the transmission power level. Let a_1, a_2, and a_3 denote the number of discrete options available for each action, respectively. Utilizing a conventional DQN architecture would result in a joint action space of size $|\mathcal{A}| = a_1 \times a_2 \times a_3$, which significantly hampers learning efficiency. In contrast, the Branching DQN algorithm decomposes this composite action space into three independent dimensions corresponding to vessel-server association, boundary adjustment for service migration, and power control. This decomposition reduces the effective output dimension from $a_1 \times a_2 \times a_3$ to $a_1 + a_2 + a_3$, substantially decreasing both model complexity and training difficulty. However, the Branching DQN algorithm is prone to Q value overestimation. The Branching DDQN algorithm addresses this issue by decoupling action selection from Q value evaluation, thereby enabling more efficient JRASM optimization. The proposed Branching DDQN-based JRASM algorithm is outlined as Algorithm 1. To validate this design choice, the next subsection will provide a theoretical analysis of the algorithms' complexity.

C. Algorithm Complexity Analysis

In this subsection, we conduct a theoretical analysis of the computational complexity of the proposed Branching DDQN-based JRASM algorithm and compare it with the following

Algorithm 1 Branching DDQN-Based JRASM Algorithm With Dynamic Reward Weighting

Input: Smart-buoy-enabled maritime MEC environment
Output: Optimal JRASM strategy

Randomly initialize online Branching Q-network Q_θ and target network Q_{θ^-}
Initialize experience replay buffer $\mathcal{RB}$
Set maximum episodes E, mini-batch size m_{bs}
for episode $e = 1$ to E **do**
  Initialize environment with vessels and MSs
  for timestep $t = 1$ to T **do**
    Vessel k observes state $s_k(t)$
    Dynamic reward weight adjustment:
    Update weights ω_1, ω_2, ω_3, and ω_4 at episode e by applying the dynamic weighting rule from (23)
    Vessel k selects action $a_k(t)$ via ε-greedy:
    $a_k(t) = [a_{k,1}(t), a_{k,2}(t), a_{k,3}(t)]$
    $a_{k,1}$: vessel-server association
    $a_{k,2}$: transmission power of vessel k
    $a_{k,3}$: adjust boundary of overlapping coverage
    Execute $a_k(t)$, get reward $r_k(t)$, go to $s_k(t+1)$
    Store (s_t, a_t, r_t, s_{t+1}) into $\mathcal{RB}$
    if $|\mathcal{RB}| \geq m_{bs}$ **then**
      Sample mini-batch (s_t, a_t, r_t, s_{t+1})
      Compute target Q value:

$$Q^{(i)}_{\text{target},t} = r_t + \gamma Q_i(s_{t+1}, \arg\max_{a_t^{(i)}} Q_i(s_{t+1}, a_t^{(i)}; \theta_t); \theta_t^-), \quad i = 1, 2, 3$$

      Compute loss:

$$L_t(\theta_t) = \frac{1}{d}\sum_{i=1}^{d}\frac{1}{c}\sum_{t=1}^{c}[Q^{(i)}_{\text{target},t} - Q_i(s_t, a_t^{(i)}; \theta_t)]^2$$

      Update Q_θ by gradient descent
      if update interval reached **then**
        $\theta^- \leftarrow \tau\theta + (1-\tau)\theta^-$
      end if
    end if
  end for
end for

four baseline approaches to demonstrate its effectiveness and efficiency in dynamic maritime MEC scenarios.

1) *DQN-Based JRASM Algorithm*: The DQN algorithm employs a single Q-network to uniformly evaluate all discrete actions. It features a relatively simple network structure, with parameter settings including experience replay and a target network update mechanism.
2) *DDPG-Based JRASM Algorithm*: The DDPG algorithm adopts an actor-critic architecture, utilizing two separate networks to estimate action-value functions and output policy. It has a more complex network structure, with parameter settings that include dual networks and a soft update mechanism for the target networks.
3) *SAC-Based JRASM Algorithm*: The soft actor-critic (SAC) algorithm adopts an entropy-regularized actor-critic with a stochastic policy and two Q-critics, trained off-policy with experience replay and soft target updates to balance reward maximization and exploration.
4) *PPO-Based JRASM Algorithm*: The proximal policy optimization (PPO) algorithm employs a stochastic policy and a value function optimized via a clipped surrogate objective for multiple episodes per on-policy batch, providing stable updates at the cost of collecting fresh trajectories.

The computational complexity of the aforementioned algorithms can be analyzed as follows.

1) *DQN-Based JRASM Algorithm*: For the DQN architecture, during each training episode, the agent must perform Q value computations for all possible combinations in the composite action space. Let m_{bs} denote the mini-batch size for experience replay, $|\mathcal{S}|$ represent the state dimension, and $|\theta|$ indicate the total number of parameters in the Q-network. Thus, the computational complexity per episode of the DQN-based JRASM algorithm can be expressed as

$$T^{\text{CC}}_{\text{DQN}} = \mathcal{O}\left(m_{\text{bs}}\,|\mathcal{A}|\,|\mathcal{S}|\,|\theta|\right). \tag{30}$$

Its space complexity is primarily determined by the storage of the Q-network and the maintenance of the empirical replay buffer, i.e.,

$$S^{\text{CC}}_{\text{DQN}} = \mathcal{O}\left(|\theta|\right) + \mathcal{O}\left(m_{\text{bs}}\left(|\mathcal{S}| + |\mathcal{A}|\right)\right). \tag{31}$$

2) *DDPG-Based JRASM Algorithm*: The DDPG architecture is typically applied to continuous action spaces, where its actor–critic architecture requires simultaneous updates of both actor and critic networks per episode, along with policy gradient backpropagation. Consequently, the computational complexity of the DDPG-based JRASM algorithm can be expressed as

$$T^{\text{CC}}_{\text{DDPG}} = \mathcal{O}\left(m_{\text{bs}}\,|\mathcal{S}|\left(|\theta_a| + |\theta_c|\right)\right). \tag{32}$$

Its space complexity primarily stems from storing the actor–critic networks and maintaining the experience replay buffer, i.e.,

$$S^{\text{CC}}_{\text{DDPG}} = \mathcal{O}\left(|\theta_a| + |\theta_c|\right) + \mathcal{O}\left(m_{\text{bs}}\left(|\mathcal{S}| + |\mathcal{A}|\right)\right). \tag{33}$$

Although DDPG circumvents the enumeration of composite discrete actions, its dual-network architecture introduces additional computational overhead and tends to exhibit poor convergence stability in high-dimensional discrete composite variable scenarios.

3) *SAC-Based JRASM Algorithm*: The SAC updates two critics and one stochastic policy per gradient step (plus a lightweight temperature update). Denoting policy parameters by $|\theta_\pi|$ and the two critics' parameters by $|\theta_{Q1}|$ and $|\theta_{Q2}|$, respectively. Then, the time complexity per-update of the SAC-based JRASM algorithm is

$$T^{\text{CC}}_{\text{SAC}} = \mathcal{O}\left(m_{\text{bs}}\left(|\theta_\pi| + \left|\theta_{Q1}\right| + \left|\theta_{Q2}\right|\right) + m_{\text{bs}}\,|\mathcal{A}|\right) \tag{34}$$

and its space complexity is given by

$$S_{\mathrm{SAC}}^{\mathrm{CC}} = \mathcal{O}\left(|\theta_\pi| + \left|\theta_{Q1}\right| + \left|\theta_{Q2}\right|\right) + \mathcal{O}\left(m_{\mathrm{bs}}\left(|\mathcal{S}| + |\mathcal{A}|\right)\right). \tag{35}$$

4) *PPO-Based JRASM Algorithm*: Let H_{ppo} denote the number of on-policy samples collected per-update (roll-out length) and K_{ppo} denote the number of optimization epochs performed on that batch. Then, for a PPO-based JRASM algorithm with policy and value parameters $|\theta_\pi|$ and $|\theta_V|$, the time complexity per update can be expressed as

$$T_{\mathrm{PPO}}^{\mathrm{CC}} = \mathcal{O}\left(\left(1 + K_{\mathrm{ppo}}\right) H_{\mathrm{ppo}}\left(|\theta_\pi| + |\theta_V|\right) + H_{\mathrm{ppo}}\,|\mathcal{A}|\right) \tag{36}$$

and its space complexity is given by

$$S_{\mathrm{PPO}}^{\mathrm{CC}} = \mathcal{O}\left(|\theta_\pi| + |\theta_V|\right) + \mathcal{O}\left(H_{\mathrm{ppo}}\left(|\mathcal{S}| + |\mathcal{A}|\right)\right). \tag{37}$$

5) *Branching DDQN-Based JRASM Algorithm*: The proposed Branching DDQN-based JRASM algorithm factorizes the high-dimensional composite action space into multiple independent subaction branches. All branches share a common trunk Q-network and only diverge at the output layer. Through this factorization, the action space complexity reduces from exponential to linear summation over subaction dimensions. Consequently, the computational complexity of our algorithm can be expressed as

$$T_{\mathrm{B-DDQN}}^{\mathrm{CC}} = \mathcal{O}\left(m_{\mathrm{bs}}\left(a_1 + a_2 + a_3\right)|\mathcal{S}|\,|\theta|\right). \tag{38}$$

The space complexity of our algorithm is dominated by storing the trunk Q-network and maintaining the experience replay buffer, i.e.,

$$S_{\mathrm{B-DDQN}}^{\mathrm{CC}} = \mathcal{O}\left(|\theta|\right) + \mathcal{O}\left(m_{\mathrm{bs}}\left(|\mathcal{S}| + \left(a_1 + a_2 + a_3\right)\right)\right). \tag{39}$$

Since $a_1 + a_2 + a_3 \ll a_1 \times a_2 \times a_3$ obviously holds in practical maritime scenarios, this factorization effectively mitigates the combinatorial action space explosion, thereby significantly improving training efficiency.

Based on the above theoretical analysis, the simulation results presented in Section V will further verify the efficiency advantages of the proposed algorithm.

V. Simulation Results

This section presents a customized DRL environment that simulates agent–environment interactions in smart-buoy-enabled maritime MEC scenarios. Based on this environment, we conduct simulation experiments to evaluate the performance of the proposed Branching DDQN-based JRASM algorithm and compare it with other baseline approaches.

A. Simulation Setup

The simulation scenarios are constructed in accordance with International Maritime Organization (IMO) TSS specifications and COLREGs, ensuring realistic vessel traffic patterns and navigational safety requirements [18]. To evaluate the universality and effectiveness of the proposed Branching DDQN-based JRASM algorithm, three distinct TSS configurations with varying fairway widths, separation zone widths, and smart buoy spacing are considered. The layouts of these scenarios are illustrated in Fig. 4, and the key configuration parameters are summarized in Table II. In all scenarios, two vessel types are modeled: 1) large commercial ships, depicted as blue isosceles triangles and 2) USVs, shown as orange

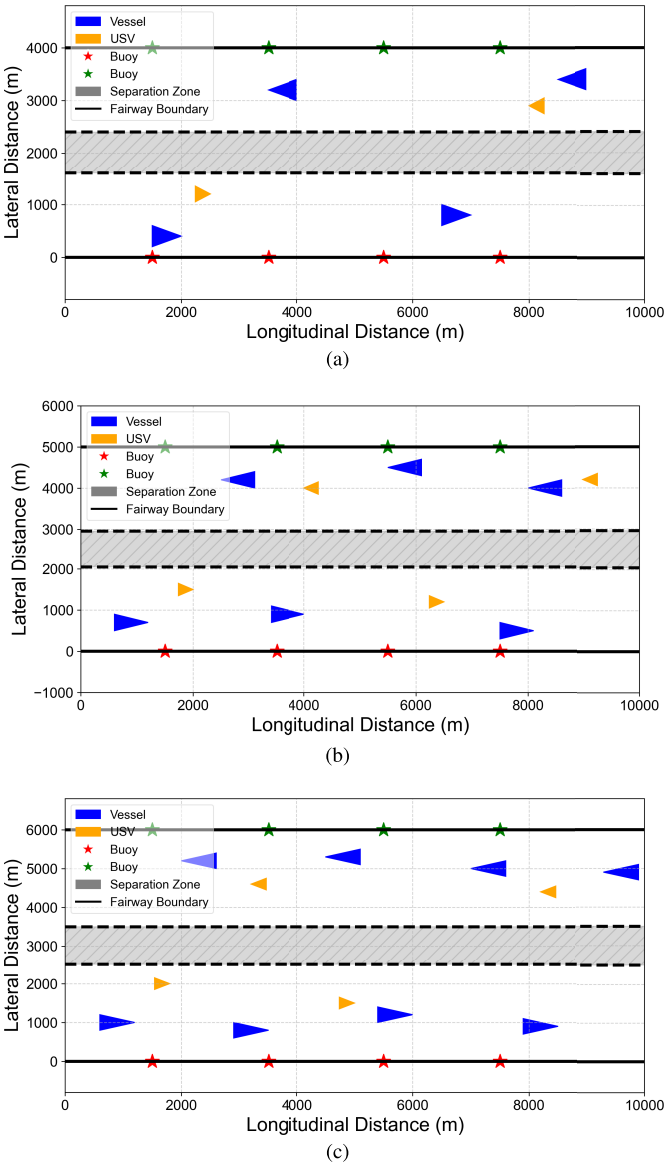


Fig. 4. Illustration of customized simulation environments. (a) Scenario 1 (FA1600/SE800). (b) Scenario 2 (FA2050/SE900). (c) Scenario 3 (FA2500/SE1000).

TABLE II
Configurations of Simulation Scenarios

Parameters	Scenario 1	Scenario 2	Scenario 3
TSS area dimensions (km)	10×4	10×5	10×6
D_{fair} (FA) (km)	1.60	2.05	2.50
D_{sep} (SE) (km)	0.80	0.90	1.00
Buoy spacing (km)	1.0	1.5	2.0
Number of fairways	2 (Opposing)	2 (Opposing)	2 (Opposing)
Navigation direction	90° / 270°	90° / 270°	90° / 270°

TABLE III
COMMON SIMULATION PARAMETERS

Simulation parameters	Value chosen
Number of server N	$[4, 8]$
Number of vessel K	$[6, 10]$
Longitudinal safety factor α	8 [20]
Lateral safety factor β	3.2 [20]
Safety distance from fairway boundary d_{boundary}	100 m
Navigation speed range of vessel V_k	$[5, 8]$ knots
Wavelength λ	0.375 m
Transmission power level	3
Noise power spectral density N_0	-174 dBm/Hz
System bandwidth W	$[20, 40]$ MHz
Task arrival rate μ	0.51
Hidden layer dimension	256
Discount factor γ	0.99
Learning rate α	1×10^{-3}
Target network soft update rate τ	0.01
Initial exploration rate	1.0
Minimum exploration rate	0.1
Exploration decay rate	0.998
Target network update frequency	200
Replay buffer size	10000
Batch size	256

equilateral triangles. Both vessel types navigate bidirectionally with constrained heading angles (90° or 270°) and random speeds of 5-8 knots. Task generation follows a Poisson process with an arrival rate $\mu = 0.51$ to capture the bursty and nonuniform nature of real maritime communication traffic.

We implement the proposed Branching DDQN-based JRASM algorithm and the four baseline methods on a system running Linux 5.4.0-200-generic, featuring an Intel[1] Xeon[1] E5-2698 v4 at 2.20-GHz processor and an NVIDIA Tesla V100-DGXS GPU (32-GB memory). The algorithms are implemented in Python 3.10.15 with PyTorch 2.5.0 and CUDA 12.4 for GPU acceleration. Unless otherwise specified, the remaining simulation parameters, which are common to all scenarios, are listed in Table III.

The proposed Branching DDQN-based JRASM algorithm is evaluated against the DQN-based, DDPG-based, SAC-based, and PPO-based JRASM baselines. The evaluation covers key performance metrics include algorithm convergence and stability, task processing latency, server load balancing, and optimal transmission power selection of vessels. The simulation is conducted as follows: 1) environment initialization according to Tables II and III; 2) vessel deployment subject to safety constraints; 3) task generation following the Poisson process; 4) decision-making performed by the tested algorithms for vessel-server association, the boundary lines adjustment of the overlapping coverage areas between adjacent servers, and transmission power control; 5) recording of performance metrics; and 6) repetition over multiple episodes for statistical averaging. In particular, in all experiments, the training for each DRL algorithm consists of 800 episodes with $T = 200$ time steps per episode, resulting in a total of 160 000 agent-environment interactions per scenario. This training scale is determined based on preliminary experiments to ensure convergence of the average reward and stability of the learned

[1]Registered trademark.

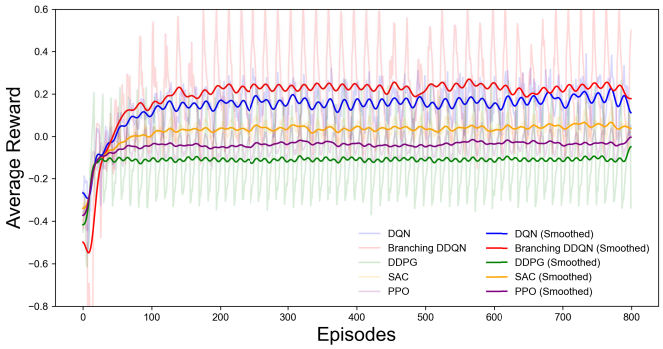


Fig. 5. Average reward curves under different algorithms.

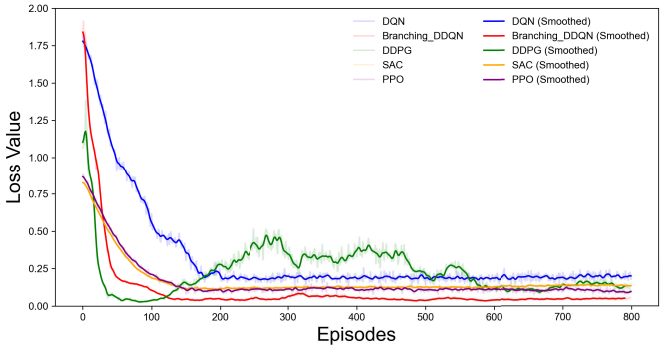


Fig. 6. Loss curves under different algorithms.

policy, as evidenced in Fig. 5. For performance evaluation, all reported results are averaged over ten independent runs with different random seeds to account for the stochasticity in task generation and vessel mobility, thereby ensuring that the amount of training data is sufficient to support the validity of the experimental conclusions. For brevity, the aforementioned five DRL-based JRASM algorithms will hereafter be denoted as the Branching DDQN, DQN, DDPG, SAC, and PPO algorithms, respectively.

B. Convergence and Computational Complexity

As shown in Fig. 5, we compare the average reward curves of the five DRL-based JRASM algorithms over 800 training episodes. The Branching DDQN algorithm achieves the highest average reward convergence value (approximately 0.25-0.3), followed by the DQN algorithm, with the SAC and the PPO algorithms trailing in the middle, and the DDPG algorithm performing the worst, near zero. Although the DDPG algorithm converges fastest around the 65th episode, it suffers from poor final performance due to suboptimal convergence under discrete action settings. In contrast, both the Branching DDQN and the DQN algorithms exhibit steeper early-stage improvements, implying higher sample efficiency. The SAC and the PPO algorithms yield stable but limited gains, constrained by their difficulty in modeling multidimensional discrete actions. The superior performance of the Branching DDQN algorithm stems from its branched architecture, which enables fine-grained optimization over joint action components. Furthermore, this reward trend is consistent with

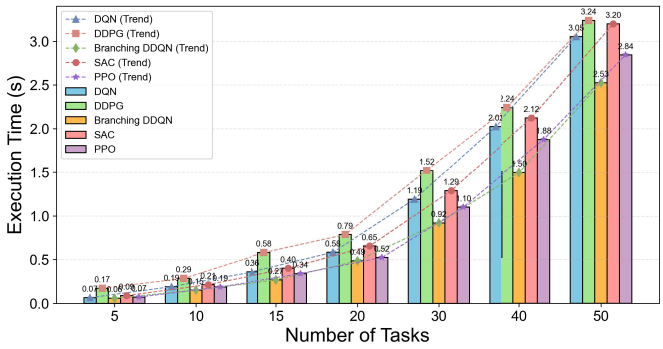


Fig. 7. Comparison of execution time versus number of tasks under different algorithms.

the loss dynamics shown in Fig. 6, validating our algorithm's effectiveness and training stability.

Fig. 6 presents the training loss trajectories and their smoothed trends over 800 episodes. The Branching DDQN algorithm achieves the lowest steady-state loss with minimal fluctuations, followed by the SAC and the PPO algorithms. The DQN algorithm exhibits visible oscillations in the later training stage, while the DDPG algorithm shows a secondary rise and increased instability between episodes 200 and 500. These differences originate in the challenges of high-dimensional discrete joint actions, which amplify Q value noise and induce overestimation. The Branching DDQN algorithm alleviates this by aggregating a shared state value with per-branch advantages, reducing target variance. The SAC and the PPO algorithms improve stability via entropy regularization and clipping but are limited in representing multidimensional discrete actions. The DDPG algorithm's deterministic policy gradients are prone to target drift under discrete constraints. Overall, this comparison indicates that the Branching DDQN algorithm offers better numerical stability and robustness, forming the basis for its superior downstream performance.

C. Performance Under Varying Scenarios

Fig. 7 illustrates the impact of the number of tasks on the execution time under different DRL-based JRASM algorithms. As the number of tasks increases from 5 to 50, all DRL-based JRASM algorithms exhibit convex upward trends, indicating that as concurrence grows, the compounded effects of resource contention and scheduling overhead lead to superlinear latency escalation. Under the same task load, the Branching DDQN algorithm consistently achieves the lowest execution time, reaching approximately 2.53 s when the number of tasks reaches 50, significantly outperforming the other baseline algorithms. Combined with the observation of Fig. 8, this advantage originates in its branching architecture, which increases the number of branches without significantly raising the per-step inference cost, while enabling more fine-grained optimization of vessel-server association, boundary lines of the overlapping coverage areas between adjacent servers, and transmission power control. The branching structure exhibits a computational complexity exponent comparable to that of other algorithms, but with a smaller constant factor due to reduced redundant computations, thereby resulting in lower runtime overhead at the same scale and offering greater scalability and practicality.

TABLE IV
PER-STEP DECISION-MAKING LATENCY UNDER DIFFERENT ALGORITHMS

Algorithms	Mean (s)	Median (s)	P95 (s)
DQN	0.128	0.096	0.285
Branching DDQN	0.143	0.106	0.296
DDPG	1.187	1.093	1.807
SAC	1.836	1.827	2.561
PPO	2.337	2.248	3.696

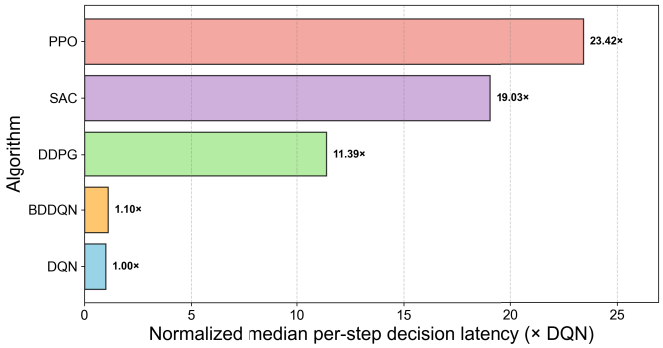


Fig. 8. Per-step inference latency normalized to DQN algorithm.

Fig. 8 and Table IV compare the per-step inference latency under different algorithms. Both the Branching DDQN and DQN algorithms maintain the lowest latency, with median latencies of 0.106 and 0.096 ms and the 95th percentile latency (P95) within 0.3 ms. In contrast, the SAC and PPO algorithms show much higher latencies, and the DDPG algorithm also exceeds 1 ms. The normalized results in Fig. 8 show that the Branching DDQN algorithm adds only 10% overhead over the DQN algorithm but is 10×-20× faster than both the SAC and PPO algorithms. This minimal cost supports more efficient boundary lines adjustment of the overlapping coverage areas between adjacent servers and power control while still meeting real-time requirements, making it well-suited for latency-sensitive maritime MEC deployments.

Fig. 9 compares the total task processing latency under three fairway configurations with varying numbers of MSs. The latency for all algorithms decreases monotonically as the number of MSs increases, with steeper reductions observed in narrower fairways or denser deployments. The Branching DDQN algorithm consistently achieves the lowest latency across all scenarios, while the DDPG algorithm performs the worst. This improvement stems from two factors: 1) a higher MS density shortens the average transmission distance and alleviates server-side load and 2) the Branching DDQN algorithm enables a "slightly preferential yet globally balanced" association strategy across heterogeneous nodes, effectively leveraging better links and compute capacity without triggering overloads.

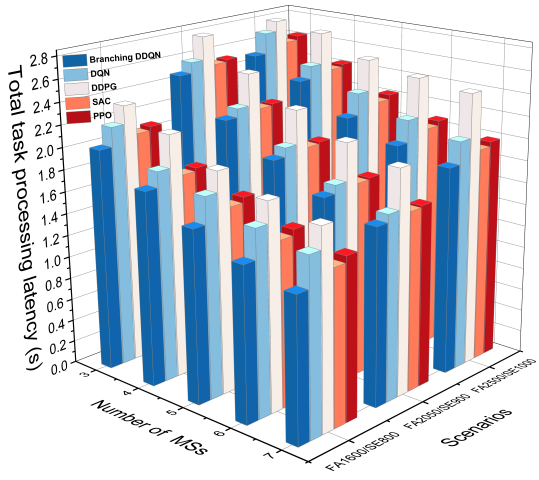
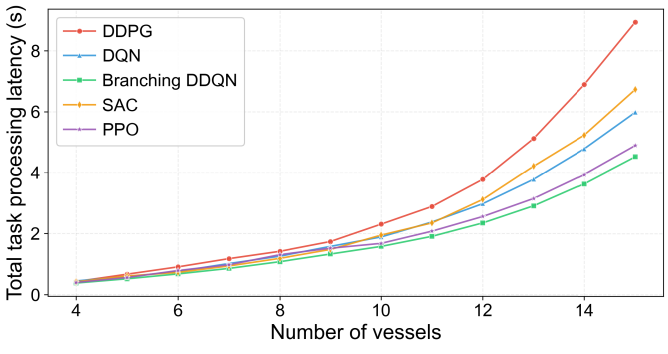
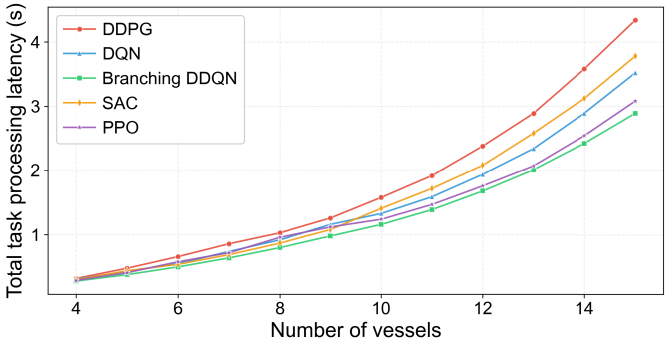


Fig. 9. Performance of total task processing latency with different fairway and MSs configurations.



(a)



(b)

Fig. 10. Performance of total task processing latency versus number of vessels for different bandwidths. (a) W = 20 MHz. (b) W = 40 MHz.

Fig. 10 illustrates the impact of the number of vessels and the system bandwidths on the total task processing latency. As the number of vessels increases from 4 to 15, the latency curves for both $W = 20\,\text{MH}z$ and $W = 40\,\text{MH}z$ bandwidths exhibit a monotonically increasing and convex trend, with the curve for $W = 40\,\text{MH}z$ consistently shifted downward. This phenomenon stems from the Poisson-distributed task generation process, where an increasing mean arrival rate with more vessels raises the probability of bursty arrivals. This intensifies the contention for limited communication and computing resources in maritime MEC scenarios, resulting in nonlinear latency growth. Increasing bandwidth effectively enhances the available transmission capacity, thereby reducing the overall latency. Under heavy-load conditions, performance gaps among the five DRL-based JRASM algorithms become more pronounced. The Branching DDQN algorithm demonstrates a clear advantage by achieving more balanced vessel-server associations and medium-power selections, which alleviate global resource contention and improve scalability in dense scenarios.

Fig. 11 investigates the influence of task data size D (in MB) and computational capacity on the total task processing latency under the five DRL-based JRASM algorithms. All algorithms exhibit significant latency reduction as computation capacity increases, but the improvement diminishes, while larger data sizes uniformly shift the curves upward. This behavior occurs because the total task processing latency can be decomposed into three phases: uplink transmission delay, computation processing delay, and service migration delay. Increasing computation capacity primarily reduces computation processing delay, but its benefit weakens once computation is no longer the dominant bottleneck in processing, and transmission or migration delays become more significant. In contrast, larger task data sizes increase both transmission and computation processing delays, thereby increasing overall latency. The Branching DDQN algorithm consistently achieves the lowest latency across all configurations, demonstrating its ability to select more appropriate migration and power allocation strategies that balance "transmission-computation-migration" and maintain low latency even as the system bottleneck shifts from computation latency to transmission latency.

D. RA and Service Migration Behavior

In maritime MEC scenarios, the TCR serves as a significant indicator that directly affects QoS of task offloading service. High TCR levels demonstrate an algorithm's capability to complete reliably assigned missions. The reliability of the proposed Branching DDQN algorithm with TCR at different training stages is shown in Fig. 12. As shown in Fig. 12, the training process is divided into early, middle, and late stages to evaluate TCRs under different algorithms. The TCR performance of all algorithms improves monotonically across stages, but with distinct trends and final values. In the late stage, the ranking stabilizes with the Branching DDQN algorithm achieving the best performance, followed by the SAC, PPO, DQN, and DDPG algorithms in descending order. This is because, as training advances, the policies progressively learn to make more effective decisions on vessel-server association and to adjust the boundary lines of the overlapping coverage areas between adjacent servers, based on servers' load and link quality. Simultaneously, power selection converges to an efficient energy-latency tradeoff. Furthermore, such coordinated improvements in "association-migration-power" directly increase the proportion of tasks completed on time. In other words, this figure illustrates how the convergence differences observed in Figs. 5 and 6 ultimately lead to measurable improvements in TCR.

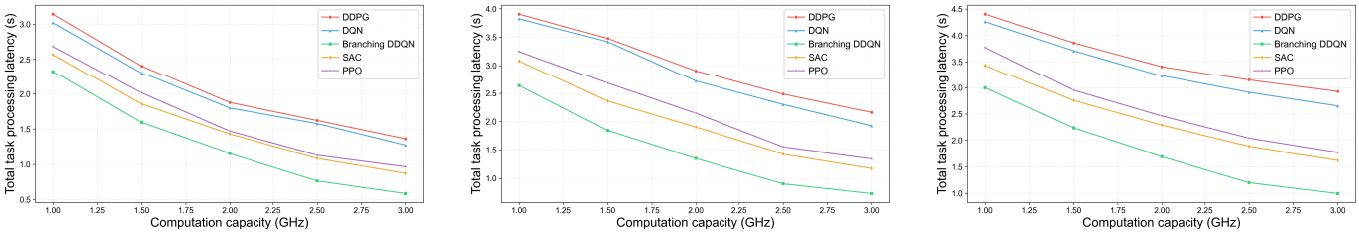


Fig. 11. Performance of total task processing latency under different algorithms with varying computation capacities and task sizes. (a) D = 7 MB. (b) D = 7.5 MB. (c) D = 8 MB.

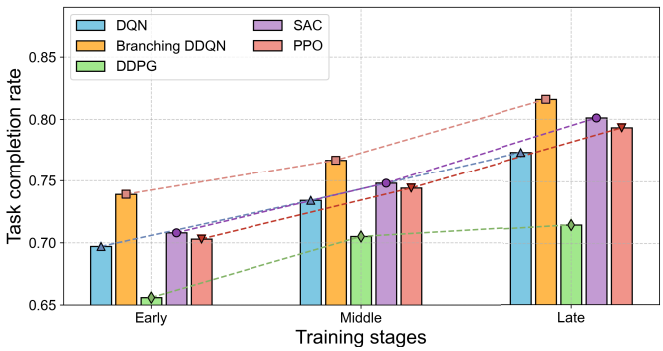


Fig. 12. Comparison of TCR under different algorithms at different training stages.

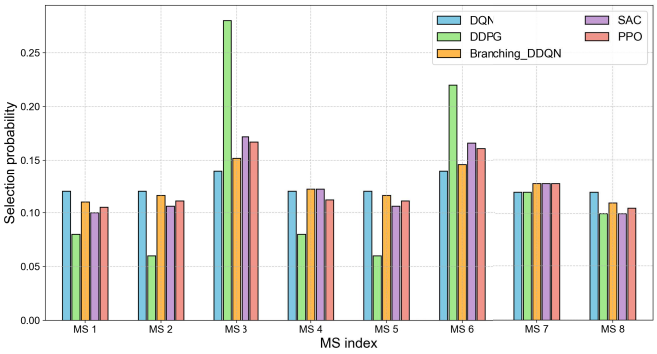


Fig. 13. MS selections under different algorithms.

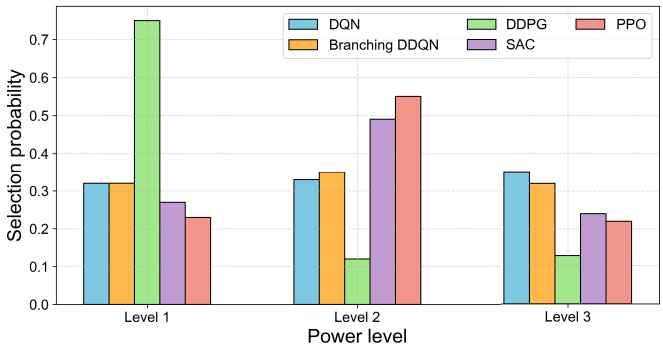


Fig. 14. Transmission power level selections under different algorithms.

Figs. 13 and 14 illustrate the differences among the five DRL-based JRASM algorithms in terms of MS selections and transmission power-level selections, respectively. The server selection distribution with N = 8 in Fig. 13 reveals that

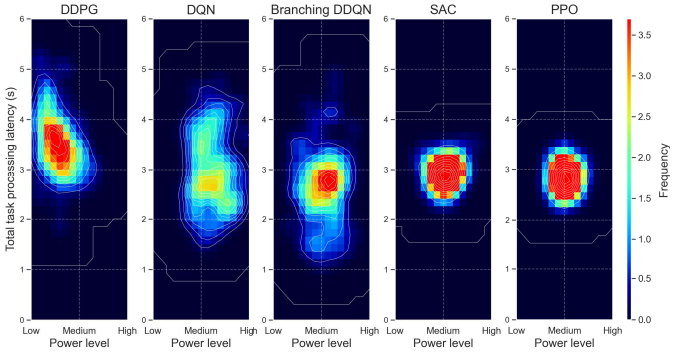


Fig. 15. Heatmaps of transmission power level-latency distribution under different algorithms.

the DDPG algorithm suffers from severe load imbalance, excessively favoring MS 3 and MS 6 and risking overload. The DQN, SAC, and PPO algorithms yield more uniform distributions in transmission power-level selections, while our Branching DDQN algorithm achieves overall balance while showing a mild preference for high-performance nodes. This occurs because its reward function includes a latency penalty term, which drives the policy to avoid pushing traffic onto a single node. Moreover, the branched architecture of our algorithm that decouples "which node to select" from "whether to migrate and how much power to use" in separate branches, enabling it to exploit marginal computational and link gains of stronger nodes without causing excessive concentration.

Fig. 14 presents the selection distribution across three transmission power levels. The DDPG algorithm shows a clear bias toward low power level, while the SAC and the PPO algorithms predominantly select medium power level. In contrast, the DQN and Branching DDQN algorithms exhibit more balanced distributions in transmission power-level selections, with the latter having the highest proportion near the medium power level. This distribution is shaped by the tradeoff between energy consumption penalties and latency constraints: low power saves energy but leads to insufficient transmission rates and unreliable task completion; high power improves transmission rates but incurs high energy costs and offers limited gains under resource contention; medium power typically represents an effective operating region with acceptable rate and energy efficiency. These results are consistent with the 2-D latency-power heatmap in Fig. 15 and the low-latency performance observed in Figs. 10 and 11, indicating that the Branching DDQN algorithm has learned to focus its policy on the most efficient power range.

Fig. 15 illustrates the 2-D density distribution between transmission power and total task processing latency. The DDPG algorithm's hot spot is concentrated in the "low power-high latency" region, while the DQN algorithm, on the other hand, exhibits a more dispersed distribution. This is because the deterministic policies or single-head Q-learning tend to either favor energy-biased strategies or produce higher latency variance under the same transmission power level. The SAC and PPO algorithms also center around medium power levels but exhibit greater latency dispersion at the same power level. The Branching DDQN algorithm shows a compact hotspot in the "medium power-low latency" area, along with a long tail extending toward even lower latency. This distribution highlights the direct source of performance differences. The branching update mechanism in the Branching DDQN algorithm can more responsively adapt to the joint state of signal-to-noise ratio and task data size, guiding migration decisions and transmission power-level selections toward the low-latency core region.

E. Discussion

This study compares the Branching DDQN, DQN, SAC, PPO, and DDPG algorithms in maritime MEC scenarios in terms of convergence, complexity, and end-to-end latency. Results show that the branching value function reduces target variance and Q value overestimation in high-dimensional discrete action spaces, leading to smoother convergence and higher steady-state rewards. System-level evaluations indicate consistent optimal or near-optimal performance across varying system bandwidth, computational capacity, and load conditions, with TCR increasing from 73.8% to 81.8% and latency being significantly reduced under heavy loads. The per-step decision latency of our Branching DDQN algorithm is slightly higher than that of the DQN algorithm and significantly lower than that of the policy-gradient methods, demonstrating an efficient complexity-performance balance. Power-latency analysis confirms effective power allocation without sacrificing energy efficiency, avoiding the issue of server overconcentration seen in the DDPG algorithm. Unlike prior DRL-based approaches that ignore dynamic migration or use single-action-space, our method jointly optimizes vessel-server association, the boundary lines of the overlapping coverage areas between adjacent servers, and power control, improving convergence speed, energy-latency tradeoffs, and high-load stability. Moreover, observed effects, i.e., diminishing marginal gains from added resources, convex latency growth, and balanced mid-power strategies, form a causal explanation supporting the generality and robustness. However, the current framework relies on accurate state input, lacks validation on constrained buoy platforms, and excludes multiagent coordination issues to be addressed in future work. Overall, the proposed method ensures robust, real-time performance and offers superior adaptability compared to existing DRL-based solutions in maritime MEC environments.

F. Real-World Deployment Considerations

To facilitate real-world deployment of the proposed JRASM framework, we outline the following key engineering considerations under maritime conditions, including hardware feasibility, energy supply, latency control, and more. We plan to conduct field trials in representative maritime regions to evaluate the performance of task offloading and service migration under real-world conditions. These efforts will guide subsequent large-scale deployment and system refinement.

1) *Buoy-Based MEC Infrastructure*: All edge computing modules are integrated into smart buoys deployed along fairways. Given maritime constraints, low-power industrial edge computing platforms (e.g., NVIDIA Jetson Orin NX 16 GB) are suitable for real-time task inference. Each buoy independently executes offloaded tasks from nearby vessels, without requiring onboard sensing, thus simplifying installation and reducing maintenance overhead.
2) *Energy Autonomy*: The buoys are designed for off-grid operation, incorporating solar panels and battery storage for power. This system, combined with quantized inference and operator fusion, maintains an average power consumption below 15 W. Dynamic power scheduling and sleep-wake cycles are applied, ensuring long-term energy sustainability under variable environmental conditions.
3) *Real-Time Responsiveness*: Per-step inference latency is maintained at approximately $1.1\times$ that of standard DQN, as shown in Fig. 8. To satisfy subsecond control cycle requirements, the system employs 4G/5G communication modules and performs policy updates asynchronously or via offline replay, thereby minimizing resource contention during peak load periods.
4) *Deployment Strategy and Reliability*: The system supports container-based deployment, A/B testing, automatic rollback, and quarterly policy retraining. It integrates AIS/Electronic Navigational Chart (ENC) data with environmental priors into its decision-making pipeline to enhance adaptability. Under link degradation, the system reverts to fallback policies (e.g., nearest buoy selection with moderate power) to preserve baseline performance.

VI. Conclusion

This article investigates the JRASM problem in smart-buoy-enabled maritime MEC networks by modeling it as an MDP and proposing a Branching DDQN-based JRASM algorithm to decompose the high-dimensional discrete action space and mitigate Q value overestimation. The proposed DRL-based JRASM framework overcomes the challenges introduced by vessel mobility, real-time constraints, and sparse edge infrastructure typical of maritime environments. Moreover, our approach addresses the gaps of prior works, which primarily focus on static offloading, communication coverage extension, or caching strategies, while neglecting the dynamic interplay among association decisions, service migration, and power control. Comparative simulations with the DQN, DDPG, SAC, and PPO-based baselines demonstrate that our algorithm achieves faster convergence, higher steady-state rewards, and lower latency while ensuring balanced server utilization and adaptive transmission power allocation. Specifically, the TCR under our algorithm improves from 73.8% to 81.8%, and the

total task processing latency is reduced by up to 15% under heavy loads. In terms of computational overhead, the per-step decision-making latency of our algorithm is slightly higher than that of the DQN-based JRASM algorithm but is significantly lower than that of policy-gradient methods, reflecting an efficient complexity-performance tradeoff. In addition to these efficiency considerations, our approach offers a key advantage over prior maritime MEC studies, which often neglect dynamic service migration or rely on single-action-space DRL. The proposed method jointly optimizes vessel-server association, boundary lines of the overlapping coverage areas between adjacent servers, and power allocations of vessels by adopting the Branching DDQN algorithm to handle high-dimensional discrete action spaces. This integration leads to faster convergence, better energy-latency tradeoffs, and enhanced stability under high-load conditions.

Given the aspects not yet addressed in this study, including unconsidered emergency server handover failures, the lack of validation on resource-constrained buoy platforms, and the absence of multiagent coordination, future work will extend the framework to multiagent DRL and conduct small-scale field trials in real maritime environments to further evaluate robustness, energy efficiency, and latency performance.

REFERENCES

[1] F. S. Alqurashi, A. Trichili, N. Saeed, B. S. Ooi, and M.-S. Alouini, "Maritime communications: A survey on enabling technologies, opportunities, and challenges," *IEEE Internet Things J.*, vol. 10, no. 4, pp. 3525–3547, Feb. 2023.

[2] H. Huang, W. Zhan, G. Min, Z. Duan, and K. Peng, "Mobility-aware computation offloading with load balancing in smart city networks using MEC federation," *IEEE Trans. Mobile Comput.*, vol. 23, no. 11, pp. 10411–10428, Nov. 2024.

[3] L. Wang, X. Deng, J. Gui, X. Chen, and S. Wan, "Microservice-oriented service placement for mobile edge computing in sustainable Internet of Vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 9, pp. 10012–10026, Sep. 2023.

[4] J. Chi, T. Qiu, F. Xiao, and X. Zhou, "ATOM: Adaptive task offloading with two-stage hybrid matching in MEC-enabled industrial IoT," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 4861–4877, May 2024.

[5] J. Fan, Q. Junwei, L. Lei, and T. Hui, "Associative tasks computing offloading scheme in Internet of Medical Things with deep reinforcement learning," *China Commun.*, vol. 21, no. 4, pp. 38–52, Apr. 2024.

[6] X. Zhigang, S. Xin, X. Siyang, and C. Jing, "Providing robust and low-cost edge computing in smart grid: An energy harvesting based task scheduling and resource management framework," *China Commun.*, vol. 22, no. 2, pp. 226–240, Feb. 2025.

[7] S. A. Chaudhry et al., "A lightweight authentication scheme for 6G-IoT enabled maritime transport system," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 2, pp. 2401–2410, Feb. 2023.

[8] F. Xu, F. Yang, C. Zhao, and S. Wu, "Deep reinforcement learning based joint edge resource management in maritime network," *China Commun.*, vol. 17, no. 5, pp. 211–222, May 2020.

[9] C. Li, Y. Zhang, X. Gao, and Y. Luo, "Energy-latency tradeoffs for edge caching and dynamic service migration based on DQN in mobile edge computing," *J. Parallel Distrib. Comput.*, vol. 166, pp. 15–31, Aug. 2022.

[10] C. Sun, X. Li, C. Wang, Q. He, X. Wang, and V. C. M. Leung, "Hierarchical deep reinforcement learning for joint service caching and computation offloading in mobile edge-cloud computing," *IEEE Trans. Services Comput.*, vol. 17, no. 4, pp. 1548–1564, Jul. 2024.

[11] M. Tang and V. W. S. Wong, "Deep reinforcement learning for task offloading in mobile edge computing systems," *IEEE Trans. Mobile Comput.*, vol. 21, no. 6, pp. 1985–1997, Jun. 2022.

[12] S. C. Ghoshal et al., "VESBELT: An energy-efficient and low-latency aware task offloading in Maritime Internet-of-Things networks using ensemble neural networks," *Future Gener. Comput. Syst.*, vol. 161, pp. 572–585, Dec. 2024.

[13] C. Zhang, B. Lin, L. Lyu, X. Hu, and S. Qi, "Energy-efficient transmission and computation resource optimization for UAV-assisted NOMA-based maritime Internet of Things," in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC)*, Shanghai, China, Aug. 2023, pp. 1–6, doi: 10.1109/ICCC57788.2023.10233651.

[14] T. Yang et al., "Multi-armed bandits learning for task offloading in maritime edge intelligence networks," *IEEE Trans. Veh. Technol.*, vol. 71, no. 4, pp. 4212–4224, Apr. 2022.

[15] M. Dai, Z. Luo, Y. Wu, L. Qian, B. Lin, and Z. Su, "Incentive oriented two-tier task offloading scheme in marine edge computing networks: A hybrid Stackelberg-auction game approach," *IEEE Trans. Wireless Commun.*, vol. 22, no. 12, pp. 8603–8619, Dec. 2023.

[16] Y. Pang, D. Wang, D. Wang, L. Guan, C. Zhang, and M. Zhang, "A space-air-ground integrated network assisted maritime communication network based on mobile edge computing," in *Proc. IEEE World Congr. Services (SERVICES)*, Beijing, China, Oct. 2020, pp. 269–274, doi: 10.1109/SERVICES48979.2020.00060.

[17] X. Chen et al., "Dynamic service migration and request routing for microservice in multicell mobile-edge computing," *IEEE Internet Things J.*, vol. 9, no. 15, pp. 13126–13143, Aug. 2022.

[18] D. Yu, Y. He, X. Zhao, J. Chen, J. Liu, and L. Huang, "Dynamic adaptive autonomous navigation decision-making method in traffic separation scheme waters: A case study for Chengshanjiao waters," *Ocean Eng.*, vol. 285, Oct. 2023, Art. no. 115448.

[19] Y. Zhao, J. Ren, and X. Chi, "Maritime mobile channel transmission model based on ITM," in *Proc. 2nd Int. Symp. Comput., Commun., Control Autom.*, 2013, pp. 378–383, doi: 10.2991/3ca-13.2013.93.

[20] Z. Jingsong, W. Zhaolin, and W. Fengchen, "Comments on ship domains," *J. Navigat.*, vol. 46, no. 3, pp. 422–436, Sep. 1993.

[21] S. Meng, S. Wu, H. Wu, A. Li, and Q. Zhang, "Effectiveness-oriented SAGSIN: Unveiling a unified metric and a comprehensive framework," *IEEE Netw.*, vol. 38, no. 6, pp. 39–47, Nov. 2024.

[22] W. Wu, W. Feng, Y. Fang, Z. Lin, and X. Lu, "Multi-HAP-assisted computation offloading in space-air-ground-sea integrated network," *IEEE Internet Things J.*, early access, Mar. 5, 2025, doi: 10.1109/JIOT.2025.3548156.

[23] Z. Wang, B. Lin, and Q. Ye, "Double-edge-assisted computation offloading and resource allocation for space-air-marine integrated networks," *IEEE Trans. Veh. Technol.*, early access, 2025, doi: 10.1109/TVT.2025.3561346.

[24] Y. Liu, J. Yan, and X. Zhao, "Deep reinforcement learning based latency minimization for mobile edge computing with virtualization in maritime UAV communication network," *IEEE Trans. Veh. Technol.*, vol. 71, no. 4, pp. 4225–4236, Apr. 2022.

[25] L. Yan, H. Chen, Y. Tu, and X. Zhou, "A task offloading algorithm with cloud edge jointly load balance optimization based on deep reinforcement learning for unmanned surface vehicles," *IEEE Access*, vol. 10, pp. 16566–16576, 2022.

[26] H. Zeng et al., "USV fleet-assisted collaborative computation offloading for smart maritime services: An energy-efficient design," *IEEE Trans. Veh. Technol.*, vol. 73, no. 10, pp. 14718–14733, Oct. 2024.

[27] N. Su, J.-B. Wang, C. Zeng, H. Zhang, M. Lin, and G. Y. Li, "Unmanned-surface-vehicle-aided maritime data collection using deep reinforcement learning," *IEEE Internet Things J.*, vol. 9, no. 20, pp. 19773–19786, Oct. 2022.

[28] X. Hu, Y. Wang, B. Lin, and H. Wei, "Stochastic geometry based modeling and performance analysis of buoy-assisted offshore Internet of Things," in *Proc. IEEE Int. Conf. Internet Things (iThings) IEEE Green Comput. Commun. (GreenCom) IEEE Cyber, Phys. Social Comput. (CPSCom) IEEE Smart Data (SmartData) IEEE Congr. Cybermatics (Cybermatics)*, Danzhou, China, Dec. 2023, pp. 26–29, doi: 10.1109/ITHINGS-GREENCOM-CPSCOM-SMARTDATA-CYBERMATICS60724.2023.00029.

[29] T. Lyu, X. Dou, X. Ren, T. Ma, and B. Lin, "A novel maritime edge computing system for Internet of Vessels," in *Proc. IEEE 23rd Int. Conf. Commun. Technol. (ICCT)*, Shenzhen, China, Oct. 2023, pp. 735–739, doi: 10.1109/ICCT59356.2023.10419619.

[30] S. Sun, Y. Dai, and L. Lyu, "Cache placement and power allocation in offshore maritime wireless networks," in *Proc. IEEE 98th Veh. Technol. Conf. (VTC-Fall)*, Hong Kong, Oct. 2023, pp. 1–6, doi: 10.1109/VTC2023-FALL60731.2023.10333725.

[31] V. A. Reddy and G. L. Stüber, "Wi-buoy: An energy-efficient wireless buoy network for real-time high-rate marine data acquisition," *IEEE Access*, vol. 9, pp. 130586–130600, 2021.

[32] R. Zhang, C. Zhao, Y. Liang, J. Hu, and M. Pan, "Edge-based dynamic spatiotemporal data fusion on smart buoys for intelligent surveillance of inland waterways," *J. Mar. Sci. Eng.*, vol. 13, no. 2, p. 220, Jan. 2025.

[33] L. Xujie, T. Jing, X. Yuan, and S. Ying, "Mobility-aware multi-task migration and offloading scheme for Internet of Vehicles," *Chin. J. Electron.*, vol. 32, no. 6, pp. 1192–1202, Nov. 2023.
[34] Z. Liao, S. Xu, J. Huang, and J. Wang, "Task migration and resource allocation scheme in IoV with roadside unit," *IEEE Trans. Netw. Service Manage.*, vol. 20, no. 4, pp. 4528–4541, Dec. 2023.
[35] Á. Gabilondo, Z. Fernández, Á. Martín, M. Zorrilla, P. Angueira, and J. Montalbán, "Dynamic mobile network slicing through vehicular traffic analysis," *IEEE Open J. Veh. Technol.*, vol. 6, pp. 1464–1480, 2025.
[36] M. Minardi, Y. Drif, T. X. Vu, and S. Chatzinotas, "SAST-VNE: A flexible framework for network slicing in 6G integrated satellite-terrestrial networks," *IEEE J. Sel. Areas Commun.*, vol. 43, no. 1, pp. 234–244, Jan. 2025.
[37] Y. Sun, Z. Wu, K. Meng, and Y. Zheng, "Vehicular task offloading and job scheduling method based on cloud-edge computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 12, pp. 14651–14662, Dec. 2023.
[38] F. Zhang, G. Liu, X. Fu, and R. Yahyapour, "A survey on virtual machine migration: Challenges, techniques, and open issues," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 2, pp. 1206–1243, 2nd Quart., 2018.
[39] K. Peters, "Deep routeing and the making of 'maritime motorways': Beyond surficial geographies of connection for governing global shipping," *Geopolitics*, vol. 25, no. 1, pp. 43–64, Jan. 2020.
[40] J. Montewka, T. Hinz, and P. Kujala, "Probability modeling of vessel collisions," *Reliab. Eng. Syst. Saf.*, vol. 95, pp. 573–589, May 2010.
[41] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "Hybrid method for minimizing service delay in edge cloud computing through VM migration and transmission power control," *IEEE Trans. Comput.*, vol. 66, no. 5, pp. 810–819, May 2017.
[42] Q. Yuan, J. Li, H. Zhou, T. Lin, G. Luo, and X. Shen, "A joint service migration and mobility optimization approach for vehicular edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 9041–9052, Aug. 2020.
[43] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2333–2345, Oct. 2018.
[44] C. Liu, F. Tang, Y. Hu, K. Li, Z. Tang, and K. Li, "Distributed task migration optimization in MEC by extending multi-agent deep reinforcement learning approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 7, pp. 1603–1614, Jul. 2021.
[45] S. Wang and Y. Gong, "Joint power control and task offloading in collaborative edge–cloud computing networks," *IEEE Internet Things J.*, vol. 10, no. 17, pp. 15197–15208, Sep. 2023.
[46] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," *Proc. AAAI Conf. Artif. Intell.*, vol. 30, no. 1, Mar. 2016, pp. 2094–2100, doi: 10.1609/aaai.v30i1.10295.

Hangqi Li (Member, IEEE) received the B.E., M.S., and Ph.D. degrees in communication engineering from Jilin University, Changchun, China, in 2014, 2017, and 2020, respectively.

She is currently a Lecturer with the Navigation College, Dalian Maritime University, Dalian, China. Her research interests include intelligent resource allocation in maritime communications, green communication technologies for marine surface vessels, smart buoys, unmanned aerial vehicle (UAV)-assisted wireless communications, and mobile edge computing.

Yuzhang Wang received the B.E. degree in navigation technology from the Navigation College, Dalian Maritime University, Dalian, China, in 2023, where he is currently pursuing the M.S. degree in transportation.

His research interests include deep reinforcement learning approaches and their application to maritime MEC networks, smart-buoy-enabled maritime MEC networks, and intelligent resource allocation in maritime communications.

Mingyang Pan received the Ph.D. degree in traffic information engineering and control from Dalian Maritime University, Dalian, China, in 2004.

He is currently a Professor with the Navigation College, Dalian Maritime University. He is also the Director of the Technical Institute of Navigation, Dalian Maritime University. His research interests include electronic chart display and information systems (ECDISs), digital waterway systems, and intelligent waterway transportation systems.

Shaoxi Li received the M.S. degree in traffic information engineering and control from Dalian Maritime University, Dalian, China, in 2004.

He is currently an Associate Professor with the Navigation College, Dalian Maritime University. His research interests focus on electronic chart display and information systems (ECDISs), intelligent waterway transportation systems, machine learning algorithms, ocean mapping, cartography, and marine geographic information systems (MGISs).

Wei Guan (Member, IEEE) received the Ph.D. degree from Dalian Maritime University, Dalian, China, in 2010.

He is currently a Professor and a Ph.D. Supervisor with the Navigation College, Dalian Maritime University. His research interests are in concise robust control and its application to marine vessels, autonomous marine surface vessels, and intelligent behavior decision-making.