DOI: 10.19884/j.1672-5220.202402003

Digital Twin Assisted Task Offloading for Maritime-UAV Integrated MEC Networks

ZOU Haozheng, ZHANG Wenqian*, YI Yuhan, ZHANG Guanglin
College of Information Science and Technology, Donghua University, Shanghai 201620, China

Abstract: With the growth of maritime activities, the number of computationally complex applications is growing exponentially. Mobile edge computing (MEC) is widely recognized as a viable option to address the substantial need for wireless communications and compute-intensive operations in maritime environments. To reduce the processing load and meet the demands of mobile terminals for high bandwidth, low latency and multiple access, MEC systems with unmanned aerial vehicles (UAVs) have been proposed and extensively explored. In this paper, a maritime MEC network that employs a top-UAV (T-UAV) for task offloading supported by digital twin (DT) is considered. To explore the task offloading strategy employed by the edge server, the flight trajectory and resource allocation strategy of the T-UAV is studied in detail. The objective of this study is to minimize latency costs while ensuring that the energy of the T-UAV is sufficient to fulfill services. In order to accomplish this objective, the joint optimization problem is described as a Markov decision process (MDP). To overcome this problem, the priority-based reinforcement learning (RL) algorithm for computation offloading and trajectory planning (PRL-COTP) is developed. The simulation results demonstrate that the proposed approach can significantly reduce the overall cost of the system in comparison to other benchmarks.

Key words: unmanned aerial vehicle (UAV); maritime mobile edge computing (MEC); digital twin; task offloading; resource management; reinforcement learning (RL)
CLC number: TN929.52 **Document code**: A
Article ID: 1672-5220(2024)06-0644-10

Open Science Identity (OSID)

0 Introduction

During the past several years, the development and the utilization of maritime resources have led to unprecedented prosperity, giving rise to innovative applications such as intelligent port marine monitoring systems and maritime rescue[1]. To keep up with the rapid development of emerging maritime applications, there is an urgent need for wireless communication and computing systems that provide less delay and optimal dependability. Among current methods of communication, satellite communications are expensive and time-consuming, while land-based base stations provide only limited access[2]. Furthermore, the non-fixed operating area of maritime users (MUs) creates a significantly different communication environment compared to land-based users, which includes long-distance communication and unstable channels[3]. To tackle these challenges, it is crucial to employ maritime communication systems that can effectively manage big data, guarantee high-speed data transmission, and maintain a low operational cost.

Mobile edge computing (MEC) has been recognized as a viable and efficient solution for fulfilling computing needs in resource-constrained and time-sensitive environments. MEC improves system capabilities by strategically positioning servers near the network edge, reducing communication distance and latency. With the increasing popularity and ease of deployment of unmanned aerial vehicles (UAVs)[4-5], integrating UAVs into MEC is considered a prospective approach to constructing maritime MEC systems[6].

The main advantage of using UAVs in MEC is that UAVs can establish line-of-sight (LoS) connections with sea terminals, which has the potential to improve network performance. In addition, UAVs are also suitable for a wide range of operational environments due to their flexibility and ease of deployment. UAVs can improve the communication speed by adjusting their distance from maritime terminals through effective trajectory planning. Peng et al.[7] utilized the feature of UAVs to maximize the number of connected edge devices. However, UAVs have limited energy capacity and require energy management strategies to extend their service time. Lin et al.[8] considered the energy efficiency of the UAV-assisted MEC and proposed a deep Q-network (DQN)-based reinforcement learning (RL) approach. Meanwhile, the concept of the digital twin (DT) has

Received date: 2024-02-09
Foundation items: National Natural Science Foundation of China (Nos. 62301307 and 62072096); Shanghai Pujiang Program, China (No. 23PJD041); Chenguang Program of Shanghai Education Development Foundation and Shanghai Municipal Education Commission, China (No. CGA60)
* Correspondence should be addressed to ZHANG Wenqian, email: wqzhang@dhu.edu.cn
Citation: ZOU H Z, ZHANG W Q, YI Y H, et al. Digital twin assisted task offloading for maritime-UAV integrated MEC networks [J]. *Journal of Donghua University (English Edition)*, 2024, 41(6): 644-653.

emerged as a significant technique for the 6G era. The virtual model generated by DT contains the capability to accurately represent physical entities within the network and actively observe the overall network's condition in real-time. The combining of DT and MEC enables the task offloading module in MEC systems to obtain the operational status of MEC servers without requiring continuous interaction with the real-time environment[9]. Zhang et al.[10] utilized DT to describe the communication of vehicles and develop a social model. Sun et al.[11] utilized DT technology in Internet of Things (IoTs) to achieve a dynamic balance between computing energy consumption and communication energy consumption in industrial production.

The preceding discussion has demonstrated the significant value of combining DT and UAV-MEC. We concentrate on the task offloading problem in maritime MEC assisted by a top-UAV (T-UAV). We use deep reinforcement learning (DRL) for task offloading management to minimize total latency. The main contributions can be summarized as follows.

1) We explore the concept of a UAV-assisted maritime MEC network with DT framework. Equipped with a MEC server, the T-UAV in our system provides computing capabilities to MUs with time-sensitive tasks.

2) We propose a DRL algorithm, i. e., the priority-based reinforcement learning (RL) algorithm for computation offloading and trajectory planning, named the PRL-COTP algorithm to handle the task offloading and trajectory planning problem of the T-UAV, ensuring that all tasks of MUs can be successfully completed while minimizing the overall service latency and improving the overall work efficiency.

3) The simulation experiments show that the PRL-COTP algorithm is highly effective in providing task offloading services for MUs. In terms of total time latency and convergence, the PRL-COTP algorithm outperforms other benchmark algorithms.

1 System Model

As shown in Fig. 1, we propose a UAV-assisted maritime MEC network assisted by DT that can fulfill the requirements of MUs' tasks. We assume that all the MUs move freely within the coverage range of the T-UAV, and their positions are randomly distributed. Each MU has a task to offload that is divisible and computation-intensive during each time slot. The T-UAV provides both split-mission, communication and computing services to all MUs, allowing for efficient coordination and collaboration among all parties involved. The utilization of digital representations of UAV, MUs and other components in the DT layer serves to augment the system's dynamic operational capabilities.

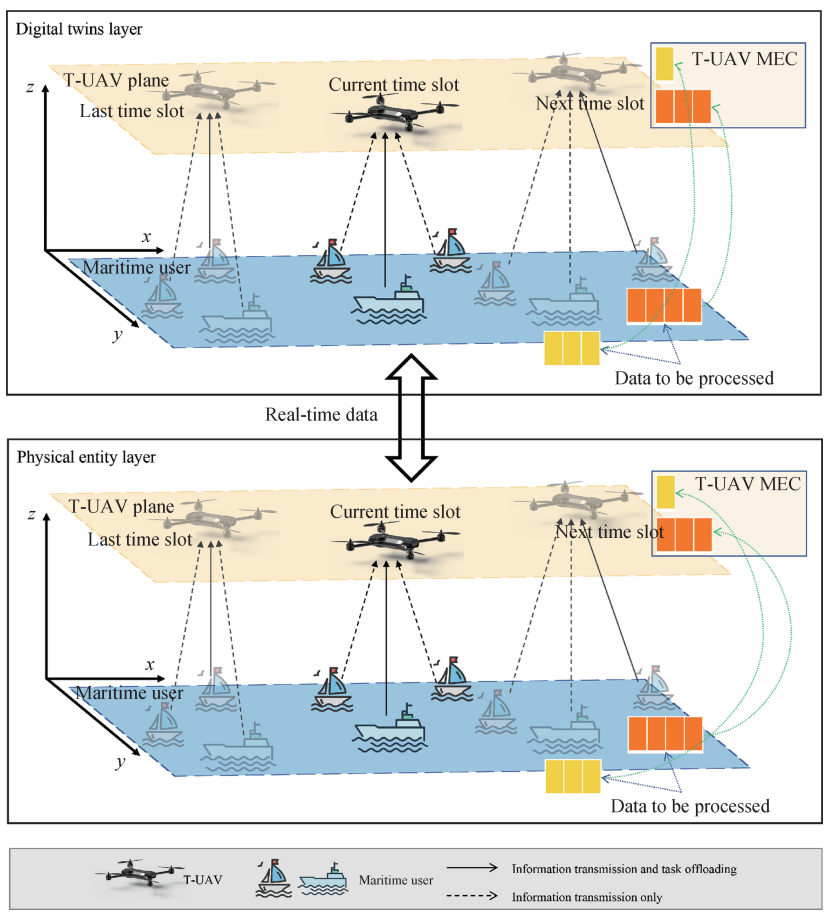

Fig. 1 System model of DT assisted maritime MEC networks

1.1 Network model

We investigate a UAV-assisted maritime MEC network with DT framework. Due to the limited processing power of the T-UAV, we assume that the T-UAV will only be able to connect to one MU at one time slot[12] and the connections follow the constraint:

$$\sum_{i\in\mathcal{I}} \alpha_{i,\mathrm{k}}(t) = 1, \tag{1}$$

where $i \in \mathcal{I} = \{1,2,\cdots,I\}$ denotes the set of MUs; k denotes the T-UAV; $\alpha_{i,\mathrm{k}}(t)$ indicates the MU i's connection established with the T-UAV.

The position of MU i at time slot t is denoted as $\boldsymbol{q}_i(t) = [x_i(t), y_i(t), 0]$ and the position of T-UAV at time slot t is denoted as $\boldsymbol{q}_{\mathrm{k}}(t) = [x_{\mathrm{k}}(t), y_{\mathrm{k}}(t), H]$, where H is the height of T-UAV. The distance between MU i and T-UAV at time slot t is calculated as

$$d_{i,\mathrm{k}}(t) = \sqrt{\|\boldsymbol{q}_{\mathrm{k}}(t) - \boldsymbol{q}_i(t)\|^2}, \tag{2}$$

where $\|\cdot\|$ is the Euclidean norm of a vector.

The flight direction of T-UAV is determined as $\theta \in (0, 2\pi]$, and the flight speed v is determined as $v \in [0, v_{\max}]$. We can calculate the location at next time slot as

$$x(t+1) = x(t) + v(t)t^{\mathrm{fly}}\cos(\theta(t)), \tag{3}$$

$$y(t+1) = y(t) + v(t)t^{\mathrm{fly}}\sin(\theta(t)), \tag{4}$$

where t^{fly} is the flight slot length of T-UAV.

1.2 Resource model

The demand for different task resources is increasing as the types of tasks for marine equipment become more diverse. The topology of the MEC network and the distribution of resource access requests will become more complicated and dynamic. It is essential to ensure the best performance and reliability of the whole system. This paper focuses on optimizing the storage resources of T-UAV to ensure that they meet the requirements of each task and are capable of handling task offloading demands.

The task created by MU i is modelled as a tuple $\{s_i, c_i\}$, where s_i means the whole data size needed to be processed and c_i represents the CPU cycle quantity required. The T-UAV creates resource allocation choices based on the task information received from the MU, with each decision containing the flight direction, the flight speed, the offloading ratio, and the computing resource allotted to the MU. We use $\{C_k^{\mathrm{co}}, C_k^{\mathrm{ca}}\}$ and $\{C_i^{\mathrm{co}}, C_i^{\mathrm{ca}}\}$ to denote the T-UAV's and the MU i's computing resource and cache resource. For the offloading ratio of each MU i, we have the constraint:

$$\beta_i^{\mathrm{lo}}(t) + \beta_i^{\mathrm{re}}(t) = 1, \beta_i^{\mathrm{lo}}(t), \beta_i^{\mathrm{re}}(t) \in [0,1], \tag{5}$$

where $\beta_i^{\mathrm{lo}}(t)$ and $\beta_i^{\mathrm{re}}(t)$ denote the offloading task proportions calculated by MU i and the T-UAV respectively.

Considering that the data size of all tasks should not exceed the maximum load of the T-UAV during offloading, we have

$$\sum_{i\in\mathcal{I}} \alpha_{i,\mathrm{k}}(t)\beta_i^{\mathrm{re}}(t)s_i(t) \leqslant C_{\mathrm{k}}^{\mathrm{ca}}. \tag{6}$$

1.3 DT model

In this paper, we examine two distinct forms of DTs, namely twin MUs and twin T-UAVs. The purpose of these DTs is to keep the raw information from each network member and to monitor the present operational condition of the network.

DTs are virtual clones of their physical counterparts, with whom they interact in real time. DTs are used to adjust the network architecture of the MEC system in response to current environmental conditions. There is a certain degree of estimation bias with respect to the real values of the physical entity and the values represented by the DT.

Similar to Ref. [9], the bias of DTs is denoted by a stochastic variable. The DT models D_i^{MU} and $D_{\mathrm{k}}^{\mathrm{T-UAV}}$ for MU i and T-UAV can be formulated as

$$D_i^{\mathrm{MU}}(t) = \Theta(C_i^{\mathrm{co}}(t), \widetilde{C}_i^{\mathrm{co}}(t)), \forall i \in \mathcal{I}, \tag{7}$$

$$D_{\mathrm{k}}^{\mathrm{T-UAV}}(t) = \Theta(C_{\mathrm{k}}^{\mathrm{co}}(t), \widetilde{C}_{\mathrm{k}}^{\mathrm{co}}(t)), \tag{8}$$

where $\widetilde{C}_i^{\mathrm{co}}(t)$ denotes the variation in CPU frequency between MU i and its DT; $\widetilde{C}_{\mathrm{k}}^{\mathrm{co}}(t)$ denotes the variation in the CPU frequency between the T-UAV and its DT; $\Theta(\cdot)$ denotes the actual value following the interplay between the estimated value of the ideal model and the deviation affected by the DT model.

1.4 Communication model

We suppose that the MU i and the T-UAV establish communication via a wireless link that exhibits LoS features while performing task offloading[13], and that the FDMA protocol is used to send data between them[14]. Using the free-space path loss model for the channel power gain $h_{i,\mathrm{k}}(t)$ that connects the MU i and the T-UAV delivers the following description:

$$h_{i,\mathrm{k}}(t) \triangleq \rho_o(d_{i,\mathrm{k}}(t))^{-2} = \frac{\rho_0}{\|\boldsymbol{q}_{\mathrm{k}}(t) - \boldsymbol{q}_i(t)\|^2}, \tag{9}$$

where ρ_0 is the power received at a distance of 1 m for the transmission level of 1 W in the free-space path loss model.

Based on the Shannon's capacity[15], the communication uplink rate $R_{i,\mathrm{k}}(t)$ from MU i to the T-UAV can be calculated as follows:

$$R_{i,\mathrm{k}}(t) = B_{i,\mathrm{k}}(t)\log_2\left(1 + \frac{h_{i,\mathrm{k}}(t)P_i}{\sigma^2}\right), \tag{10}$$

where $B_{i,\mathrm{k}}$, P_i and σ^2 denote the spectrum resource allocated to MU i by T-UAV, the power of the transmission at the MU i and the noise power at the T-UAV, respectively.

1.5 Delay analysis

The computing task is first determined to decide whether to solve it locally or partially send it to T-UAV for computation. Considering the tiny amount of task

results' size, we ignore the transmitting delay of the calculation results to MUs.

1.5.1 MU transmiting to T-UAV delay

The transmitting delay from the MU i to the T-UAV can be given by

$$T_{i,\mathrm{k}}^{\mathrm{tr}}(t)=\frac{\beta_i^{\mathrm{re}}(t)s_i(t)}{R_{i,\mathrm{k}}(t)}. \tag{11}$$

1.5.2 MU computing delay

Considering the part of the task to be computed locally, the estimated MU computing delay for the completion of the task is given by

$$\widetilde{T}_i^{\mathrm{co}}(t)=\frac{\beta_i^{\mathrm{lo}}(t)c_i(t)}{C_i^{\mathrm{co}}(t)}. \tag{12}$$

According to Ref. [16], we use $\Delta T_i^{\mathrm{co}}(t)$ to denote the MU computing delay difference between the actual value and the DT estimate, and it is calculated as follows:

$$\Delta T_i^{\mathrm{co}}(t)=\frac{\beta_i^{\mathrm{lo}}(t)c_i(t)\widetilde{C}_i^{\mathrm{co}}(t)}{C_i^{\mathrm{co}}(t)\left[C_i^{\mathrm{co}}(t)-\widetilde{C}_i^{\mathrm{co}}(t)\right]}. \tag{13}$$

The computing delay for local execution of MU i is

$$T_i^{\mathrm{co}}(t)=\widetilde{T}_i^{\mathrm{co}}(t)+\Delta T_i^{\mathrm{co}}(t). \tag{14}$$

1.5.3 T-UAV computing delay

Similar to MU, the estimated T-UAV computing delay can be described as

$$\widetilde{T}_{i,\mathrm{k}}^{\mathrm{co}}(t)=\frac{\beta_i^{\mathrm{re}}(t)c_i(t)}{C_{\mathrm{k}}^{\mathrm{co}}(t)}. \tag{15}$$

The delay in T-UAV's computation between the actual value and the estimated DT is defined as

$$\Delta T_{i,\mathrm{k}}^{\mathrm{co}}(t)=\frac{\beta_i^{\mathrm{re}}(t)c_i(t)\widetilde{C}_{\mathrm{k}}^{\mathrm{co}}(t)}{C_{\mathrm{k}}^{\mathrm{co}}(t)\left[C_{\mathrm{k}}^{\mathrm{co}}(t)-\widetilde{C}_{\mathrm{k}}^{\mathrm{co}}(t)\right]}. \tag{16}$$

Finally, the total computing delay for local execution of MU i to the T-UAV is expressed as

$$T_{i,\mathrm{k}}^{\mathrm{co}}(t)=\widetilde{T}_{i,\mathrm{k}}^{\mathrm{co}}(t)+\Delta T_{i,\mathrm{k}}^{\mathrm{co}}(t). \tag{17}$$

Considering the parallel execution of the local and remote T-UAV computing, the overall delay is the maximum value between the local and remote computations. It can be denoted as

$$T_i(t)=\max\left\{T_{i,\mathrm{k}}^{\mathrm{co}}(t),T_{i,\mathrm{k}}^{\mathrm{tr}}(t)+T_{i,\mathrm{k}}^{\mathrm{co}}(t)\right\}. \tag{18}$$

1.6 UAV energy consumption analysis

Controlling energy consumption is essential to extending the service period of the T-UAV since battery capacity is limited.

1.6.1 Flight energy consumption

According to Ref. [17], we denote the flight energy consumption by

$$E_{\mathrm{k}}^{\mathrm{fly}}(t)=\phi\left|v(i)\right|^2, \tag{19}$$

where $\phi=0.5M_{\mathrm{k}}t^{\mathrm{fly}}$, M_{k} means T-UAV's weight, and t^{fly} means the time consumed by the flight.

1.6.2 Computing energy consumption

In this paper, we use $\kappa(C_{\mathrm{k}}^{\mathrm{co}})^3$ to denote the CPU energy usage in T-UAV[18]. And the computation of the energy consumption $E_{\mathrm{k}}^{\mathrm{co}}(t)$ is described as

$$E_{\mathrm{k}}^{\mathrm{co}}(t)=\sum_{i\in\mathcal{I}}\alpha_{i,\mathrm{k}}(t)\kappa(C_{\mathrm{k}}^{\mathrm{co}})^3T_{i,\mathrm{k}}^{\mathrm{co}}(t), \tag{20}$$

where κ represents the effective switched capacitance, and it varies with different CPU architectures.

1.6.3 Transmission energy consumption

The T-UAV's receiving energy consumption when receiving task data from MU i, denoted as $E_k^{\mathrm{rx}}(t)$, can be provided by

$$E_{\mathrm{k}}^{\mathrm{rx}}(t)=\sum_{i\in\mathcal{I}}\alpha_{i,\mathrm{k}}(t)P_{\mathrm{k}}^{\mathrm{rx}}T_{i,\mathrm{k}}^{\mathrm{tr}}(t)=\sum_{i\in\mathcal{I}}\frac{P_{\mathrm{k}}^{\mathrm{rx}}\beta_i^{\mathrm{re}}(t)s_i(t)}{R_{i,\mathrm{k}}(t)}, \tag{21}$$

where $P_{\mathrm{k}}^{\mathrm{rx}}$ is the received power of the T-UAV.

The total energy consumption of the T-UAV is

$$E_{\mathrm{k}}(t)=E_{\mathrm{k}}^{\mathrm{co}}(t)+E_{\mathrm{k}}^{\mathrm{rx}}(t)+E_{\mathrm{k}}^{\mathrm{fly}}(t). \tag{22}$$

On the basis of the details given above, we can compute the remaining energy of the T-UAV in the following time slots as follows:

$$E_{\mathrm{k}}^{\mathrm{total}}(t+1)=E_{\mathrm{k}}^{\mathrm{total}}(t)-E_{\mathrm{k}}(t), \tag{23}$$

where $E_{\mathrm{k}}^{\mathrm{total}}(t)$ is the T-UAV's remaining energy.

1.7 Problem formulation

We jointly optimize the T-UAV trajectory θ and velocity v, as well as the communication resource management parameters $\alpha_{i,\mathrm{k}}$, β_i^{lo} and β_i^{re}, to minimize the processing latency for all MUs. The optimization problem is formulated as

$$\mathcal{P}_1:\min_{\substack{\theta(t),v(t),\alpha_{i,\mathrm{k}}(t),\\ \beta_i^{\mathrm{lo}}(t),\beta_i^{\mathrm{re}}(t)}}\sum_{i\in\mathcal{I}}\sum_{i\in\mathcal{I}}T_i(t), \tag{24}$$

$$\text{Constraint 1}\quad \text{s.t.}\sum_{i\in\mathcal{I}}\alpha_{i,\mathrm{k}}(t)=1,$$

$$\text{Constraint 2}\quad \theta(t)\in[0,2\pi),$$

$$\text{Constraint 3}\quad v(t)\in[0,v_{\max}],$$

$$\text{Constraint 4}\quad E_{\mathrm{k}}^{\mathrm{total}}(t)\geqslant E_k(t),$$

$$\text{Constraint 5}\quad \beta_i^{\mathrm{lo}}(t)+\beta_i^{\mathrm{re}}(t)=1,$$

$$\text{Constraint 6}\quad \sum_{i\in\mathcal{I}}\alpha_{i,\mathrm{k}}(t)\beta_i^{\mathrm{re}}(t)s_i(t)\leqslant C_{\mathrm{k}}^{\mathrm{ca}},$$

$$\text{Constraint 7}\quad D_i^{\mathrm{MU}}(t)=\Theta(C_i^{\mathrm{co}}(t),\widetilde{C}_i^{\mathrm{co}}(t)),\forall i\in I,$$

$$\text{Constraint 8}\quad D_{\mathrm{k}}^{\mathrm{T-UAV}}(t)=\Theta(C_{\mathrm{k}}^{\mathrm{co}}(t),\widetilde{C}_{\mathrm{k}}^{\mathrm{co}}(t)).$$

Constraint 1 guarantees that the T-UAV can only serve one MU at one time slot; Constraints 2 and 3 indicate the angle range and the speed range within which the T-UAV can fly; Constraints 4 means that the energy T-

UAV cannot be empty at every time slot; Constraint 5 represents the sum of the parts processed by T-UAV and MU is a complete task; Constraint 6 ensures that the size of the task offloaded to T-UAV at any time does not exceed the storage limit of T-UAV; Constraints 7 and 8 represent the deviation caused by DT at each time slot.

Based on the observation of problem $\mathcal{P}_1$, it is evident that the connection establishment decision is discrete, while the trajectory and computation offloading decisions are both continuous. And the battery level also changes dynamically in response to these decisions. This means that problem $\mathcal{P}_1$ is a non-convex and mixed integer nonlinear optimization problem, which has big challenges for conventional methodologies. To simplify the difficulty of solving the problem, the objective of minimizing $\sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{I}} T_i(t)$ can be converted to an MDP, and DRL is a powerful approach to solve it. In the following parts, we will propose a DRL algorithm to solve this problem.

2 Proposed Algorithm

In order to solve the problem of delayed convergence and the difficulty of fully examining the action space, we propose the PRL-COTP algorithm. The PRL-COTP algorithm framework is shown in Fig. 2.

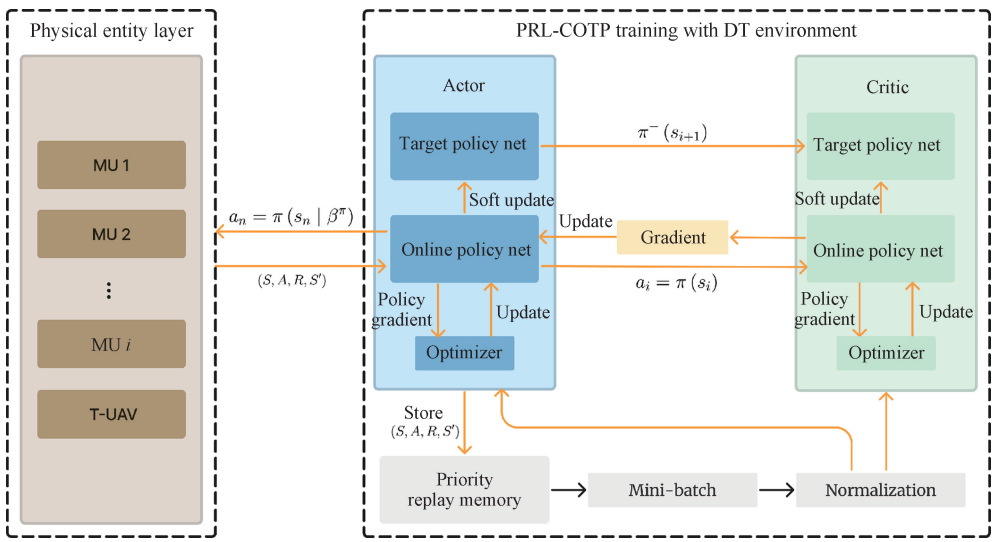

Fig. 2 PRL-COTP algorithm framework

2.1 Principle of PRL-COTP algorithm

Based on the actor-critic (AC) framework, the deep deterministic policy gradient (DDPG) algorithm trains a neural network policy function with two sub-networks, i.e., the double policy and the neural dynamics. DDPG is an approach that uses the deterministic strategy π to generate precise behaviors which can be described as

$$a_n = \pi(s_n \mid \beta^\pi), \tag{25}$$

where s_n represents both the state of environment and the input for the agent; β^π denotes the parameter of actor network in the AC framework, which means that the actor network represents the strategy π of the intelligent agent.

Once deterministic actions are obtained, DDPG employs the critic network to estimate the action value function corresponding to π, where the output of the actor network is combined with the state as the critic network input. It can be represented by

$$\mu_n = Q(s_n, a_n \mid \beta^Q), \tag{26}$$

where β^Q is the critic network parameter in the AC framework. It finally generates a value that functions as an evaluation of the actor network's output by executing mathematics on the action and state combinations.

In DDPG, the networks of actors and critics are constructed independently, and their respective target networks gradually converge towards the online network through a soft update mechanism. The update equation can be formulated as

$$\omega^- \leftarrow \tau\omega + (1-\tau)\omega^-, \tag{27}$$

where $\tau \in (0,1)$ denotes the update rate of the network; ω is the parameter in the neural network. In the actor network, $\omega = \beta^\pi$; in the critic network, $\omega = \beta^Q$.

The design of the objective function follows the deterministic policy gradient theorem, which can be expressed in the following formula:

$$\begin{aligned}\nabla_{\beta^\pi} J &\approx E_{s\sim\nu^{\pi_\beta}}\left[\nabla_{\beta^\pi} Q_\pi(s_n, \pi(s_n))\right] \\ &= E_{s\sim\nu^{\pi_\beta}}\left[\nabla_{\beta^\pi} Q(s, a\beta^Q)\big|_{s=s_n, a=\pi(s_t\mid\beta^\pi)}\right] \\ &= \frac{1}{N}\sum_i^N \nabla_a Q(s, a \mid \beta^Q)\big|_{s=s_i, a=\pi(s_i)} \nabla_{\beta^\pi}\pi(s \mid \beta^\pi)\big|_{s_i},\end{aligned} \tag{28}$$

where ν^{π_β} denotes the distribution of s_n. The ultimate

objective is to use π to determine the action a that maximizes the value $Q(s,a)$.

The target of updating the DDPG network is the minimization of the loss function, which is represented as the MSE formula:

$$L = \frac{1}{N}\sum_{i=1}^{N}\left(Q_i^{\text{target}} - Q(s_i,a_i \mid \beta^Q)\right)^2, \tag{29}$$

where $Q_i^{\text{target}} = r_i + \gamma Q^-(s_{i+1},\pi^-(s_i \mid \beta^{\pi^-}) \mid \beta^{Q^-})$ denotes the updating of the target; r_i denotes the reward.

2.2 Prioritized experience replay

An effective approach to optimize the performance of the DDPG algorithm is to establish an experience pool and use the experience replay technique to randomly extract samples from the pool. This random selection can reduce the likelihood of dependencies and correlations between experiences, which can help the algorithm converge more quickly.

In this paper, we present the prioritized experience replay technique, which involves selecting samples based on their priority rather than using random selection in experience replay. Unlike traditional DRL methods that employ an array to retain the agent's interactions with the environment, we use SumTree[19] as the data structure for storing these experiences. Due to this specific design, each experience has a priority, which will be ordered based on priority during the random sampling process.

According to our approach, a sample is more likely to be chosen for neural network parameter updates if it has a greater loss value. The probability of selecting a sample $P(j)$ can be mathematically expressed as

$$P(j) = \frac{p_j^v}{\sum_j^J p_j^v}, \forall j \in J, \tag{30}$$

where p_j denotes the probability of experience; v denotes the total quantity of prioritization used.

Through the use of this scheme, our algorithm can achieve enhanced efficiency in the training process and speed up the process of convergence. To address the risk of overfitting and achieve a more balanced sample selection, the sampling process has been augmented with a degree of randomness.

2.3 Problem transformation

Our system model suggests that the end position of the T-UAV at the previous time slot is the starting point of the next time slot, indicating that the T-UAV flight trajectory exhibits Markovian characteristics. Consequently, it is possible to optimize the trajectory of the T-UAV using the DRL method in both the continuous state and action spaces. In the DRL framework, the interaction between the T-UAV and its environment is represented by a four-tuple (S, A, R, S'), where S represents the state; A represents the action selected in this state; R and S′ represent the reward obtained and the next state when taking action A in the state S. Therefore, we transfer the problem $\mathcal{P}_1$ to the following modules.

2.3.1 State space

The state of our maritime MEC system contains three dimensions, i. e., the coordinates of the T-UAV, the coordinates of the MUs, and the task details associated with each MU.

The state of the maritime MEC can be formulated in a multi-dimensional space as the following vector representation:

$$\begin{aligned} S(t) = [&\boldsymbol{q}_k(t), \boldsymbol{q}_1(t), \cdots, \boldsymbol{q}_i(t), \\ &\{s_1(t), c_1(t)\}, \cdots, \{s_i(t), c_i(t)\}]. \end{aligned} \tag{31}$$

2.3.2 Action space

The action can be represented as a combination of the direction and speed of the T-UAV's flight, the T-UAV's choice of connection to MUs, and the proportion of tasks offloaded, and is denoted as

$$A(t) = [\theta(t), v(t), \alpha_{1,k}(t), \cdots, \alpha_{i,k}(t), \beta_1^{\text{re}}, \cdots, \beta_i^{\text{re}}]. \tag{32}$$

We only output $\beta_i^{\text{re}}(t)$ in $\boldsymbol{A}(t)$, because we could compute $\beta_i^{\text{lo}}(t)$ according to the constraint $\beta_i^{\text{lo}}(t) + \beta_i^{\text{re}}(t) = 1$.

2.3.3 Reward design

The reward $r(t)$ is used to estimate the effectiveness of strategy taken by the agent. The formula (24) accurately determines the main value in reward. The representation of the reward can be described as

$$r(t) = -\sum_i T_i(t) + p^1(t) + p^2(t), \tag{33}$$

where p^1 indicates that the storage space of the T-UAV is insufficient to finish the task offloading choice; p^2 indicates that the remaining power of the T-UAV is insufficient to support the completion of all operations.

2.3.4 Training process

In Algorithm 1, the operation of PRL-COTP algorithm depends on the input data provided by the environment. The utilization of $s(t)$ as the direct input to DNN could result in an unbalanced training set. Consequently, we employ the min-max scaler transformation method[20] to process all input data, thereby converting each element to the range of $[0,1]$.

Algorithm 1 Training process of PRL-COTP algorithm

Input: system state that requires to be normalized: $S(t)$

Output: the optimal PRL-COTP network for MEC-offloading

Initialize the critic network Q and the actor network π with random parameters β^{π} and β^{Q};
Initialize the target network through copying $\beta^{\pi^-} \leftarrow \beta^{\pi}$ and $\beta^{Q^-} \leftarrow \beta^{Q}$;
Initialize the priority replay memory $\mathcal{R}$ and the mini-batch size;
for episode = 1, 2, ⋯, E **do**
    Initialize the priority replay memory $\mathcal{R}$ and the mini-batch size;
    for t = 1, 2, ⋯, T **do**
        Get state and take the action $a_t = \pi(s_t \mid \beta^{\pi}) + \mathcal{N}$;
        Update the state $s(t+1)$ and get the reward $r(t)$;
        Calculate the priority probability according to Eq. (30);
        Store (S, A, R, S') and priority in the replay buffer $\mathcal{R}$;
        if $\mathcal{R}$'s size exceeds the mini-batch size **then**
            Sample random mini-batch of transitions from $\mathcal{R}$;
            Update the critic network β^{π} by minimizing the loss;
            Update the actor policy β^{Q} by using the sampled gradient;
            Update the total target networks according to Eq. (27);
        end if
    end for
end for

Note: $\mathcal{N}$ represents a random sequence with a value range of (0,1).

3 Simulation and Analyses

3.1 Experimental setup

This section focuses on evaluating the performance of the proposed PRL-COTP algorithm. The simulations are conducted using Python 3.7.16 and TensorFlow 1.14.0. The proposed PRL-COTP algorithm uses a neural network architecture with five layers, where the hidden layer consists of [400, 300, 50] neurons. The learning rate for the Actor network is configured at 1×10^{-4}, while the learning rate for the critic network is set to 2×10^{-4}. In our simulation, we created a marine working area of 100 m×100 m. The initial coordinates of the T-UAV are specified as [50, 50, 100]. The locations of MUs are randomly distributed within the working area. The other related parameters are set as Table 1.

Table 1 Simulation parameters

Parameter	Description	Value
I	The quantity of MUs	4
M_k /kg	The weight of T-UAV	9.5
ρ_0	The channel power gain	1.42×10^{-4}
σ^2 /dBm	The power spectrum density	−100
H/m	The height of T-UAV	100
κ	The effective switched capacitance	1×10^{-27}
t^{fly}	The flight slot length of T-UAV	1
$v_{\max}$ /(m/s)	The maximum flight speed of T-UAV	25
$E_k^{\mathrm{total}}(0)$ /kJ	The initial energy level of T-UAV	500
s_i /Mbits	The data quantity of MU i	[2.5, 3]
c_i /(cycle/bit)	The required CPU cycles per bit	[800, 1 200]
C_i^{co} /GHz	The computing resource of MU i	0.6
C_k^{co}/GHz	The computing resource of T-UAV	1.2

3.2 Performance analyses

As shown in Fig. 3, we compare the PRL-COTP algorithm with other benchmarks, i.e., DQN and DDPG, using cumulative rewards as a metric. The cumulative reward data indicates that all algorithms display random behavior at first, as they lacked prior

knowledge. This results in relatively low cumulative rewards at the beginning of the training episode. However, as the number of experience samples increases, all algorithms begin to use these samples to train their respective networks. Obviously, the total reward of all algorithms increases gradually, and eventually the PRL-COTP algorithm converges to the highest reward. This is because the PRL-COTP algorithm can utilize the AC framework to perform continuous operations, while DQN is unable to convert continuous operations into discrete operations. Meanwhile, the final training results of the PRL-COTP algorithm outperform those of DQN in high-dimensional output scenarios.

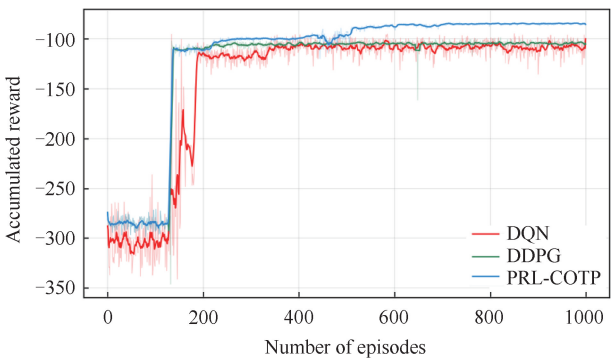

Fig. 3 Accumulated reward in training process

As shown in Fig. 4, DQN exhibits a more rapid convergence rate in comparison to the PRL-COTP algorithm. The DQN network achieves convergence after approximately 7 000 steps, while PRL-COTP requires around 125 000 steps to reach convergence. The reason for this is that DQN utilizes only two neural networks, and the PRL-COTP algorithm needs four neural networks. However, the PRL-COTP algorithm displays lower loss during convergence and ultimately attains superior performance. In summary, the PRL-COTP algorithm performs better in complex optimization problems while DQN is an effective choice for discrete scenarios with low precision needs.

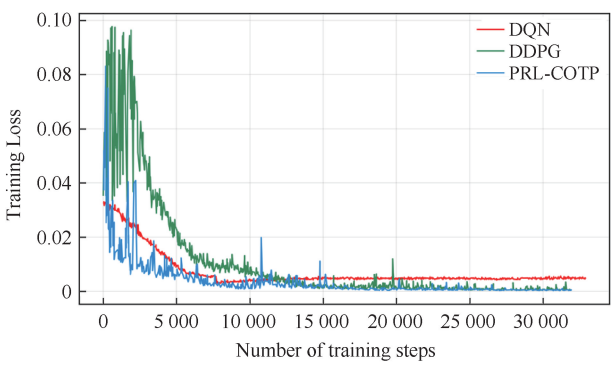

Fig. 4 Convergence performance in training process

As shown in Fig. 5, based on the experimental setup of the experiment, we vary the size of the memory pool and evaluate the effectiveness of priority memory pool by analyzing the algorithm's convergence. The graph illustrates the smooth results of the processing, indicating that the algorithm reaches a state of convergence after around 10 000 training steps. The graph illustrates the smooth results of the processing. Regardless of the size of the memory pool, whether it is 8 192 or 512, the training process can still maintain rapid convergence. Finally, we choose 4 096 as the final size of priority memory pool according to the experiment result.

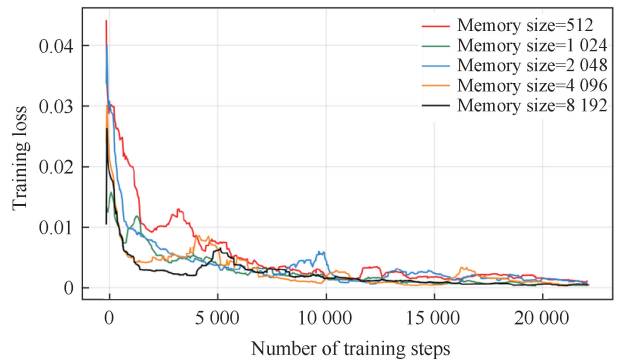

Fig. 5 Memory pool performance in the training process

In Fig. 6, we conduct a comparative analysis of the total average latencies related to the use of the PRL-COTP algorithm in comparison with alternative algorithms. Due to its foundation on the AC framework, the PRL-COTP algorithm provides a more detailed task offloading strategy. As the time slot increases, it is evident that the PRL-COTP algorithm consistently achieves the lower total average latency compared to other algorithms. The total task process latency of our algorithm is about 26. 7% less than that of the DQN algorithm. The constraints imposed by discrete space limitations cause the DQN algorithm to perform worse than the PRL-COTP algorithm. After comparing the results of the analyses, we can definitively determine that the PRL-COTP algorithm is both effective and superior for task offloading.

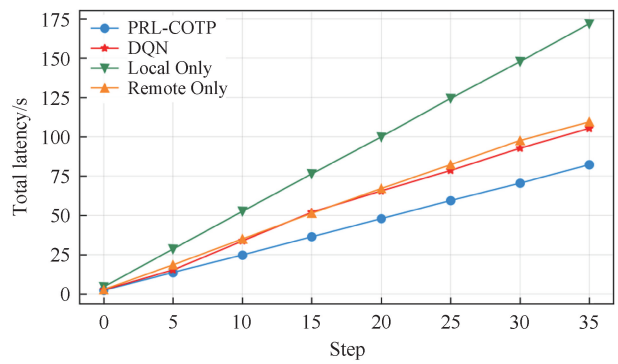

Fig. 6 Performance of total latency among episode steps

In Fig. 7, we analyze the total latency when the T-UAV has different computational resources. For the sake of fairness, all T-UAVs operated by the algorithms originate from the same takeoff location. With the

exception of the local-only method, all algorithms show a decrease in total latency when the processing resource of the T-UAV increases and the PRL-COTP algorithm consistently maintains the lowest total latency in every scenario. The reason for this is that the T-UAV has a greater capacity for computational capabilities. Furthermore, we observe that the PRL-COTP algorithm consistently maintains the lowest overall latency across every case. This result further confirms the efficiency and the scalability of the PRL-COTP algorithm.

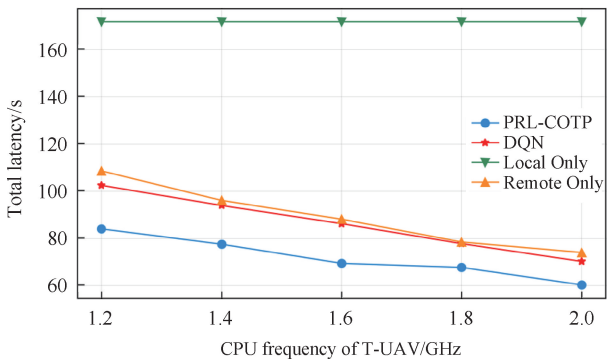

Fig. 7 Performance of total latency among different computing abilities

4 Conclusions

In this work, we primarily focused on solving the task offloading problem in maritime-UAV MEC assisted by DT. The T-UAV was used specifically for trajectory planning and task offloading, leading to expedited task completion within a given time slot. With the objective of minimizing overall latency, we proposed a DRL algorithm named the PRL-COTP algorithm which utilized the prioritized replay to accelerate the training process. This algorithm effectively handled the high-dimensional continuous action space challenge. The PRL-COTP algorithm exceeded other benchmark algorithms in simulation tests. With the effectiveness of this algorithm, it is expected to be applied to various maritime work situations in the future, and we will further investigate adding multiple UAVs to enhance the total maritime MEC system performance.

References

[1] NOMIKOS N, GKONIS P K, BITHAS P S, et al. A survey on UAV-aided maritime communications: deployment considerations, applications, and future challenges [J]. *IEEE Open Journal of the Communications Society*, 2022, 4: 56-78.

[2] ZHANG Y D, LU J, ZHANG H T, et al. Experimental study on low-altitude UAV-to-ground propagation characteristics in campus environment [J]. *Computer Networks*, 2023, 237: 110055.

[3] YANG T, FENG H L, YANG C M, et al. Multivessel computation offloading in maritime mobile edge computing network [J]. *IEEE Internet of Things Journal*, 2019, 6(3): 4063-4073.

[4] SUN Z Y, SHEN B, PAN A Q, et al. A modified self-adaptive sparrow search algorithm for robust multi-UAV path planning [J/OL]. *Journal of Donghua University (English Edition)*. (2024-10-09)[2024-11-01]. https://doi.org/10.19884/j.1672-5220.202312007.

[5] XU H Q, XING H X, LIU Y. Path planning of UAV by combing improved ant colony system and dynamic window algorithm [J]. *Journal of Donghua University (English Edition)*, 2023, 40(6): 676-683.

[6] FENG W, WANG J C, CHEN Y F, et al. UAV-aided MIMO communications for 5G Internet of Things [J]. *IEEE Internet of Things Journal*, 2019, 6(2): 1731-1740.

[7] PENG H X, SHEN X M. Multi-agent reinforcement learning based resource management in MEC- and UAV-assisted vehicular networks [J]. *IEEE Journal on Selected Areas in Communications*, 2021, 39(1): 131-141.

[8] LIN N, TANG H L, ZHAO L, et al. A PDDQNLP algorithm for energy efficient computation offloading in UAV-assisted MEC [J]. *IEEE Transactions on Wireless Communications*, 2023, 22(12): 8876-8890.

[9] LI B, LIU Y F, TAN L, et al. Digital twin assisted task offloading for aerial edge computing and networks [J]. *IEEE Transactions on Vehicular Technology*, 2022, 71(10): 10863-10877.

[10] ZHANG K, CAO J Y, MAHARJAN S, et al. Digital twin empowered content caching in social-aware vehicular edge networks [J]. *IEEE Transactions on Computational Social Systems*, 2022, 9(1): 239-251.

[11] SUN W, LEI S Y, WANG L, et al. Adaptive federated learning and digital twin for industrial Internet of Things [J]. *IEEE Transactions on Industrial Informatics*, 2021, 17(8): 5605-5614.

[12] XIONG J Y, GUO H Z, LIU J J. Task offloading in UAV-aided edge computing: bit allocation and trajectory optimization [J]. *IEEE Communications Letters*, 2019, 23(3): 538-541.

[13] LIU Y, YAN J J, ZHAO X H. Deep reinforcement learning based latency minimization for mobile edge computing with virtualization in maritime UAV communication network [J]. *IEEE Transactions on Vehicular Technology*, 2022, 71(4): 4225-4236.

[14] YU Z, GONG Y M, GONG S M, et al. Joint

task offloading and resource allocation in UAV-enabled mobile edge computing [J]. *IEEE Internet of Things Journal*, 2020, 7(4): 3147-3159.
[15] TIAN J, WANG D, ZHANG H X, et al. Service satisfaction-oriented task offloading and UAV scheduling in UAV-enabled MEC networks [J]. *IEEE Transactions on Wireless Communications*, 2023, 22(12): 8949-8964.
[16] DO-DUY T, VAN HUYNH D, DOBRE O A, et al. Digital twin-aided intelligent offloading with edge selection in mobile edge computing [J]. *IEEE Wireless Communications Letters*, 2022, 11(4): 806-810.
[17] HU Q Y, CAI Y L, YU G D, et al. Joint offloading and trajectory design for UAV-enabled mobile edge computing systems [J]. *IEEE Internet of Things Journal*, 2019, 6(2): 1879-1892.
[18] WANG Y T, SHENG M, WANG X J, et al. Mobile-edge computing: partial computation offloading using dynamic voltage scaling [J]. *IEEE Transactions on Communications*, 2016, 64(10): 4268-4282.
[19] SCHAUL T, QUAN J, ANTONOGLOU I, et al. Prioritized experience replay [EB/OL]. 2016 (2016-02-25) [2024-02-07]. https://arxiv.org/abs/1511.05952v4.
[20] CHEN M, SHEN Z R, WANG L, et al. Intelligent energy scheduling in renewable integrated microgrid with bidirectional electricity-to-hydrogen conversion [J]. *IEEE Transactions on Network Science and Engineering*, 2022, 9(4): 2212-2223.

基于数字孪生的无人机辅助海上边缘计算任务卸载

邹浩正，张文倩*，易雨菡，张光林
东华大学 信息科学与技术学院，上海 201620

摘　要：随着海事活动的增长，计算复杂的应用程序的数量呈指数级增长。移动边缘计算（mobile edge computing，MEC）被广泛认为是一种解决海上环境中对无线通信和计算密集型操作巨大需求问题的可行选择。为了减少终端计算负载并满足移动终端对高带宽、低延迟和多址的需求，目前业界已经提出并广泛探索了无人机辅助的边缘计算系统。该文采用海上边缘计算网络，使用由数字孪生（digital twin，DT）支持的顶层飞行无人机（top-unmanned aerial vehicle，T-UAV）完成任务卸载。重点研究了 T-UAV 的飞行轨迹和资源分配策略，旨在探索边缘服务器所采用的任务卸载策略，在确保 T-UAV 的能量足以提供服务的前提下最大限度地降低延迟成本。为了实现这一目标，将联合优化问题描述为马尔可夫决策过程（Markov decision process，MDP），提出了一项基于优先级的深度强化学习算法（priority-based reinforcement learning algorithm for computation offloading and trajectory planning，PRL-COTP）用于完成计算任务卸载和无人机轨迹规划。仿真结果表明，与其他基准算法相比，所提方法可以显著降低系统的总体成本。

关键词：无人机；海上移动边缘计算；数字孪生；任务卸载；资源管理；深度强化学习