DRL-based Task Offloading for Networked UAVs with Random Mobility and Collision Avoidance*

Xixin Zhang[a,b,1], Dongge Jia[b,c,1], Junfei Xie[b,1,*], Xiaobai Liu[b,1]

[a]*Department of Electrical and Computer Engineering, University of California San Diego, 9500 Gilman Dr, La Jolla, 92092, California, USA*
[b]*Department of Electrical and Computer Engineering, San Diego State University, 5500 Campanile Drive, San Diego, 92182, California, USA*
[c]*Department of Civil and Environmental Engineering, University of Pittsburgh, 3700 O'Hara Street, Pittsburgh, 15261, Pennsylvania, USA*

Abstract

Unmanned Aerial Vehicles (UAVs) have gained widespread use across various fields due to their flexibility and multifunctionality. However, their limited onboard computing capacity and energy budget are often criticized for hindering their ability to execute complex tasks in real time. To address this challenge, Networked Airborne Computing (NAC) has emerged as a promising solution, which harnesses the collective computing power of multiple UAVs interconnected through air-to-air communication links. This approach enables efficient large-scale data processing, real-time analytics, and complex mission coordination. Despite its potential, research in this area is still in its infancy. In this paper, we consider a typical NAC scenario where multiple UAVs with collision avoidance capabilities share computation and communication resources while moving stochastically within an area. Without prior knowledge of the system models, we aim to optimize the allocation of computation tasks and network resources among UAVs with uncertain mobility. To achieve this, we propose a Deep RL (DRL)-based framework and evaluate its performance using three state-of-the-art DRL algorithms including DDPG, TD3, and PPO. Simulation results demonstrate that our approach signifi-

*This work is partially supported by National Science Foundation (Grant No. 2048266, 2402689, 2106991) and Office of Naval Research (Grant No. N00014-21-1–2567).

*Corresponding author

Email addresses: xiz166@ucsd.edu (Xixin Zhang), djia@sdsu.edu (Dongge Jia), jxie4@sdsu.edu (Junfei Xie), xiaobai.liu@sdsu.edu (Xiaobai Liu)

cantly speeds up task execution and enhances energy efficiency compared to existing methods.

Keywords: Computation Offloading, Deep RL, Unmanned Aerial Vehicle, Edge Computing

1. Introduction

In recent years, unmanned aerial vehicles (UAVs), or drones, have seen rapid advancements and growing popularity in areas such as precision agriculture, disaster response, aerial photography, and environmental monitoring [1, 2, 3, 4]. As UAV applications become increasingly complex, the use of multiple cooperative UAVs has become more common. Nevertheless, their limited onboard computational resources often become a bottleneck. One solution that naturally follows is to offload computationally intensive tasks to external resources.

Extensive research has focused on efficiently utilizing resources on edge servers or remote clouds to support multi-UAV applications. For instance, Liu et al . [5] proposed to utilize a UAV-Edge-Cloud computing model and formulate a joint optimization of workflow assignment and multi-hop routing scheduling for UAV swarms to minimize computation cost and latency. Bai et al . [6] investigated delay-aware cooperative task offloading for multi-UAV enabled edge-cloud computing, proposing an algorithm to balance task distribution and minimize completion delay. In these studies, UAVs in the swarm are typically treated as relays that bring edge servers or remote clouds closer, rather than serving as computing nodes themselves. While this strategy provides access to ample computing resources, it incurs high data transmission delays, which are unacceptable for time-sensitive UAV applications, particularly real-time tasks. Moreover, edge servers typically rely on stable local network infrastructure, which is costly to deploy and difficult to scale, especially in underdeveloped regions or post-disaster scenarios [7].

Recent technological advancements have led to the development of small, lightweight, yet powerful micro-computers, significantly enhancing the onboard computing capacity of UAVs. These advancements enable UAVs to handle computationally intensive tasks directly onboard, while also inspiring researchers to explore their potential as edge servers. In [8], Hu *et al*. leveraged the computing resources of a moving UAV to serve ground users, aimed to minimize the total maximum delays among users by jointly optimizing of-

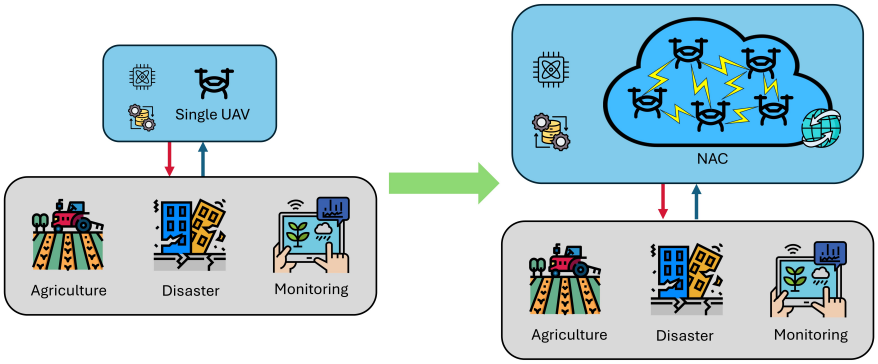

Figure 1: Networked Airborne Computing (NAC)

floading ratios, user scheduling, and UAV trajectory in a UAV-aided mobile edge computing system. Miao *et al*. [9] proposed a multi-UAV-assisted mobile edge computing (MEC) offloading algorithm that maximizes the access quantity and minimizes the task completion latency by cluster path planning based on user mobility and communication coverage. Although UAVs have proven promising in providing on-demand computing resources, these existing studies treat them as isolated servers without interaction or collaboration.

To fully unlock the computational potential of multi-UAV systems, a new paradigm called Networked Airborne Computing (NAC) has been proposed [10, 11, 12]. NAC allows multiple UAVs to share resources with each other through air-to-air communication links, forming a collaborative and interconnected computing network, as illustrated in Fig.1. The fast deployment, infrastructure-free operation, and cost-effective nature of UAV-based NAC make it a highly promising technique. Nevertheless, research in this field is still in its early stages. In our previous studies, we have developed a ROS-based simulator and a hardware testbed that consists of multiple UAVs to facilitate NAC research [12]. In [11], we introduced a coded distributed computing scheme based on deep RL (DRL) for optimally partitioning and allocating tasks to multiple networked UAVs.

In this paper, we address a challenging yet common NAC scenario where all UAVs, including both offloaders and offloadees, move randomly during task execution while actively avoiding collisions. The UAVs operate without prior knowledge of the environment or system models, and their movement

patterns or future trajectories are not shared among one another. Additionally, computation tasks are considered as general functions or operations that can be partitioned into arbitrary subtasks for parallel computation. The **main contributions** are summarized as follows:

- *New NAC Scenario*: To the best of our knowledge, this study is the first to address a NAC scenario that incorporates random UAV mobility, collision avoidance, and no prior knowledge of system models or UAV future trajectories.

- *Novel Random Mobility Models for Multi-UAVs with Collision Avoidance*: To model the random movement patterns of UAVs and capture their collision avoidance interactions, we develop two novel random mobility models (RMMs). One is an extension of the Random Direction (RD) model [13], capturing movement patterns of multirotor UAVs, and the other is an extension of the Smooth Turn (ST) model [14], capturing movement dynamics of fixed-wing UAVs. Both the RD and ST models are originally designed for individual agents.

- *Generic DRL-based Framework for Optimal Task Offloading and Resource ALlocation*: We formulate a nonlinear optimization problem for the allocation of computation tasks and network resources, aiming to balance computation efficiency and energy consumption. To address this, we develop a generic DRL-based framework (show in Fig. 2) that effectively solves the problem and is adaptable to various DRL algorithms.

- *Comprehensive Comparison Studies:* To evaluate the performance of the proposed framework, we implemented three state-of-the-art DRL algorithms within the proposed framework, including Deep Deterministic Policy Gradient (DDPG) [15], Twin Delayed Deep Deterministic Policy Gradient (TD3) [16], and Proximal Policy Optimization (PPO) [17]. These were compared against multiple benchmark methods. The results, evaluated across diverse experimental settings, demonstrate the promising performance of the proposed framework in optimizing task offloading and resource allocation while addressing the challenges posed by uncertainties in UAV movement.

Notably, this journal article extends upon a previously published short conference paper [18]. Compared to the conference version, this paper in-

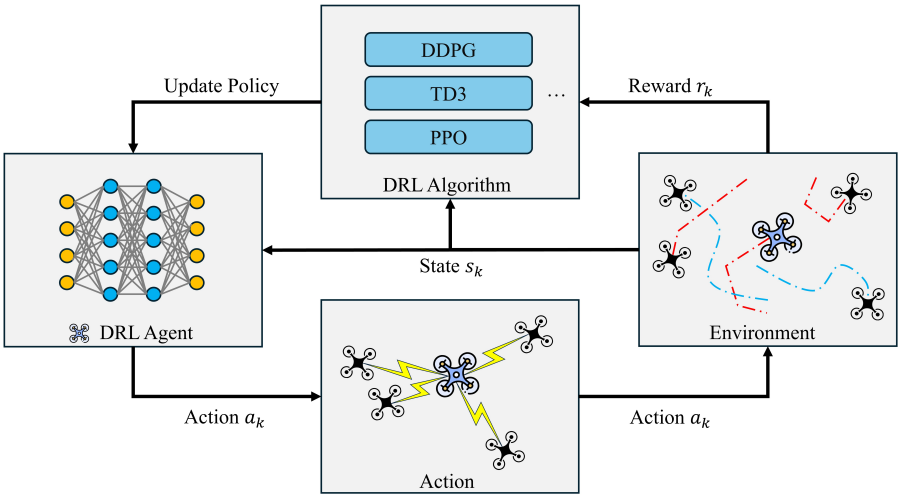

Figure 2: DRL-based Framework for Optimal Task Offloading and Resource Allocation of Multi-UAV system under Random Mobility Model

cludes a new Multi-UAV mobility model with collision avoidance strategy based on Smooth-Turn model[14] and update the system to be more realistic by pose constraint on Quality of Service. Instead of focusing only on reducing time latency through data partition and offloading, in this paper we take the network condition and energy efficiency into consideration, by allocating task workload and network resources(transmission power, and channel bandwidth), we jointly optimize the time latency and energy consumption. We develop and evaluate more DRL-based methods that adapted from DDPG and PPO to serve as component of our framework for joint optimization problem.

In the rest of this paper, Sec 2 reviews existing studies relevant to this work, Sec. 3 details the system models and formulates the optimization problem. Sec. 5 describes the proposed DRL algorithm. In Sec. 6, simulation results are presented and discussed. We conclude in Sec. 7.

2. Related Work

This section reviews existing studies most relevant to this work.

2.1. DRL-based Task Offloading

Since task offloading requires interaction with dynamic environments, DRL has been widely applied to optimize task allocation strategies across various networks such as Internet of Things (IoT) networks, UAV-assisted edge computing, and NAC.

For task offloading in IoT, [19] proposed a DRL-based distributed task offloading framework to achieve more stable and affordable task execution. Experiments revealed significant improvements in energy efficiency, response time, and system utility. Similarly, [20] introduced a distributed DRL tool to improve energy savings while satisfying tasks' latency requirements. This tool distributes multiple actor-critic algorithms across different user equipments, with each device running its own RL algorithm but sharing the same environment. This model outperforms other analyzed baselines.

In the context of UAV-assisted edge computing, studies such as [21] explored the performance of deep Q-learning to optimize offloading decisions and energy consumption in a multi-UAV-enabled MEC network. Simulations reveal better performance of deep Q-learning compared to Q-learning, full offloading, and full local computing. Ref. [22] proposed a deep Q-Network-based resource allocation policy to identify the optimal offloading relay node (UAV) for the current task, thereby maximizing the utilization of UAV resources for data relaying. Such an algorithm improves efficiency in data processing and resource utilization.

Recent advancements have further extended DRL-based offloading mechanisms to NAC. In [11], two optimization problems are solved to minimize the total task completion time under a batch-processing-based coded computation framework. Given controllable and uncontrollable UAVs, batch size is optimized using DDPG, outperforming other representative distributed computing schemes.

2.2. Multi-UAV-Assisted Mobile Edge Computing

With the rapid growth of IoT, complex mobile applications have emerged in areas such as smart cities, autonomous driving, smart agriculture, healthcare monitoring, and energy management [23]. These applications are highly sensitive to both latency and computational demands. However, IoT devices, with their limited processing power and battery capacity, struggle to deliver optimal performance [24]. A traditional approach is offloading tasks from mobile devices to cloud servers. However, this method requires high-speed

and stable network links, making it unsuitable for time-sensitive applications and regions with limited or no internet coverage [25].

To counter these practical challenges, MEC emerged by bringing computational resources closer to the network edge [26]. However, traditional MEC systems rely on static base stations, limiting their transmission capabilities in non-line-of-sight communication links, which perform poorly in mountainous and rugged environments. Furthermore, deploying terrestrial MEC units is challenging in inaccessible locations or during emergencies due to infrastructure constraints [27].

UAVs provide a convenient solution to extend the applicability of MEC. UAVs, acting as mobile stations, enable flexible and easy deployment of mobile edge computing services while maintaining line-of-sight communication links, offering a promising alternative to terrestrial MEC systems. Research has demonstrated significant advantages of UAV-assisted MEC networks in terms of energy efficiency, adaptability, and reduced latency [28, 29]. Generally, UAVs can enhance the coverage and transmission efficiency of traditional MEC systems, improving service delivery in challenging environments.

2.3. Networked Airborne Computing

NAC emerges in response to the long-lasting challenges in transmission latency and the increasing computing capabilities of modern UAVs. Unlike Multi-UAV-Assisted MEC, NAC employs UAVs as both transmitters and processors, allowing computing tasks to be completed on-site, significantly reducing data transmission costs. Moreover, to alleviate the constraints of a single UAV's computing resources, NAC aims to boost its network's computing capacity as a whole by constructing an efficient distributed computing framework. The airborne network has the advantage of unblocked line-of-sight communication. As introduced in [10], NAC augments UAV capabilities beyond traditional tasks like positioning and image processing.

Recent advances in NAC feature distributed computing schemes. In [30], coded computation was introduced to solve matrix multiplication, improving robustness to uncertain system noise. This strategy utilizes idle computing resources throughout the network to provide computation redundancy, and thus enhance computation reliability and drastically lower the need for data transfer. Additionally, a batch processing procedure was adopted to loosen the requirement for sending complete results to the master node, further enhancing efficiency [31]. Simulations and experiments showcased superior

computational efficiency and robustness to uncertainty attributed to this novel integration.

In a more recent study, a dynamic batch-processing-based coded computation framework, DRL-based load allocation, UAV mobility control strategies, and a NAC simulator were integrated for the first time to investigate the performance of NAC on task offloading and UAV trajectory management [11]. Benchmarks demonstrated the performance advantage of this integrated NAC. Later, hardware development for NAC was further studied, featuring a hardware platform equipped with NVIDIA Jetson TX2 processors for onboard computing and communication [32]. Moreover, a ROS [33] and Gazebo-based simulator [34] integrating the message passing interface for distributed computing [32] was developed. Extensive simulations and hardware tests revealed key factors influencing NAC performance, such as task size, UAV mobility, and communication quality, offering valuable insights for further development of NAC systems.

3. System Models

In this section, we will introduce the system models. Consider a group of $N+1$ heterogeneous UAVs with varying physical configurations, indexed as $i \in \mathcal{N} = \{0, 1, 2, ..., N\}$. Each UAV is equipped with computing and communication modules, enabling resource sharing and onboard computation. Their characteristics of computing, communication, energy consumption, and mobility within the system are comprehensively described and modeled as follows.

3.1. Computing Model

We describe the computing capability of each UAV i as idle CPU cycle frequency $f_i(t)$ in Hz, which is variable dependent on time and can fluctuate due to some ongoing computation tasks. For a general computation task k, its input data size is denoted as S_k (bits), and its required computation intensity is represented as ξ_k (cycles/bit)[35]. The total CPU cycles required to compute task k is hence $\xi_k S_k$, and the time required for UAV i to execute this task is:

$$T_{k,i}^{comp} = \frac{\xi_k \cdot S_k}{f_i(t)} \tag{1}$$

The corresponding energy consumption during computation is given as:

$$E_{k,i}^{comp} = \epsilon \cdot f_i(t)^3 \cdot T_{k,i}^{comp} \tag{2}$$

where ϵ represents an energy consumption parameter associated with the effective switched capacitance, which is determined by the underlying CPU architecture. To simplify the analysis, it is assumed that this capacitance remains uniform across all devices [36].

3.2. Communication Model

At time t, let the distance between UAV i and UAV j be denoted as $d_{ij}(t)$. UAV-to-UAV links are assumed Line of Sight (LoS), with propagation speed approaching the speed of light. Hence, the transmission latency can be approximated using the transmission time. Here, we model the transmission rate (bits/s) based on the Simplified Path Loss Model [37] as follows:

$$\nu_{ij}(t) = \begin{cases} B_{ij}(t) \log_2 \left(1 + \frac{G(d_r/d_{ij}(t))^{\theta} \psi_{ij}(t)}{N_0 B_{ij}(t)}\right), & \text{if } \nu_{ij} \geq \nu_{\min}, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

where $B_{ij}(t)$ is the bandwidth (Hz) of the channel between UAV i and j, d_r is a constant reference distance (meters), G is a unitless constant representing the path gain at the reference distance d_r, θ is the path loss exponent, $\psi_{ij}(t)$ is the transmitted power (mW), and N_0 is the constant noise power spectral density (dBm/Hz). Additionally, ν_{min} represents the threshold to suppress the data rates that are too low, which can be regarded as a constraint on the Quality of Service (QoS).

Unlike our previous work [18], which assumed the channel bandwidth and transmitted power to be constants, here we treat both B_{ij} and ψ_{ij} as variables to be determined for a specific task k. Once the bandwidth B_{ij}^k and transmission power ψ_{ij}^k allocated for task k are determined, the data rate ν_{ij}^k can be calculated using Eq. (3). Subsequently, the overall transmission time is given by:

$$T_{k,ij}^{trans} = \frac{S_k}{\nu_{ij}^k} \tag{4}$$

The energy (Joule) consumed for offloading task k from UAV i to UAV j is a combination of transmission energy and reception energy, expressed as follows [38]:

$$E_{k,ij}^{trans} = (\psi_{ij} + \tilde{\psi}_j) \cdot 10^{-3} \cdot T_{k,ij}^{trans} \tag{5}$$

Here, $\tilde{\psi}_j$ (mW) represents the reception power of UAV j.

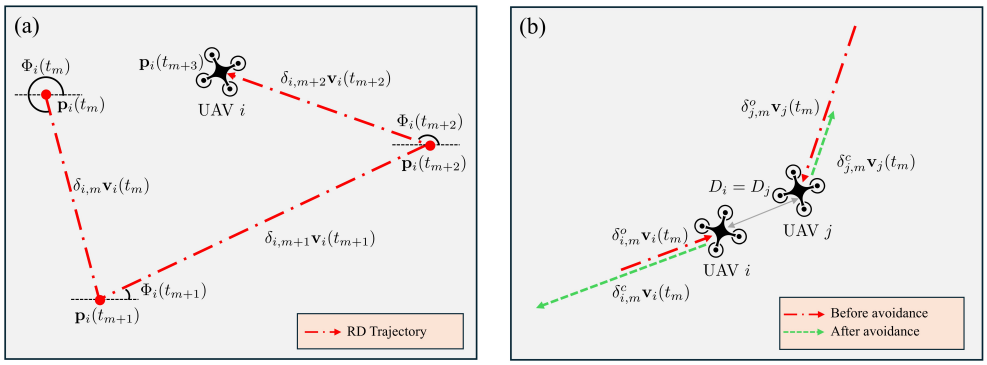

Figure 3: Random Direction Model for Multirotor UAVs . (a) RD trajectory of single UAV i, starting from $\mathbf{p}_i(t_m)$ to $\mathbf{p}_i(t_{m+3})$. (b) Trajectories of UAV i and j before and after collision avoidance in the RD multi-UAV system.

3.3. Mobility Model

We assume that all UAVs fly at the same altitude. Therefore, the position of each UAV i at time t can be depicted as $\mathbf{p}_i(t) = (x_i(t), y_i(t)) \in \mathbb{R}^2$, subject to the constraints $0 \leq x_i(t) \leq E$ (m), $0 \leq y_i(t) \leq E$ (m), ensuring the position is bounded within an area of $E \times E(m^2)$. The velocity of UAV i at time t is denoted as $\mathbf{v}_i(t) = (v_{i,x}(t), v_{i,y}(t)) \in \mathbb{R}^2$, where $v_{i,x}(t)$, $v_{i,y}(t)$ are velocity components in the x- and y-directions, respectively. The heading direction of UAV i is represented as $\Phi_i(t) \in [0, 2\pi]$.

To model uncertain UAV mobility, we adapt and extend two widely used Random Mobility Models (RMMs). The first is the Random Direction (RD) model [13], commonly applied to capture the movement patterns of multirotor UAVs. The second is the Smooth Turn (ST) model [14], designed to capture the movement dynamics of fixed-wing UAVs. Since both models were originally designed for independent single entities, which ignore the spatiotemporal correlations of trajectory across entities, we extend them to accommodate multiple UAVs, incorporating collision avoidance mechanisms to capture the interactions required for safe navigation among UAVs.

3.3.1. RD Model for Multirotor UAVs with Collision Avoidance

In the traditional RD model [13], a UAV randomly picks a velocity and moves in a straight line at this velocity for a randomly chosen duration. Afterward, it selects a new velocity and duration, repeating the process as illustrated in Fig. 3a. To describe this process mathematically, let the start

time of the m-th duration be denoted as $t_m = \sum_{l=0}^{m-1} \delta_{i,l}$ which corresponds to the m-th instance when UAV i changes its velocity. Here, each duration $\delta_{i,l}$ is randomly sampled from an exponential distribution with parameter λ_{rd}, and t_0 is set to be 0. At time t_m, UAV i select a speed magnitude uniformly between 0 and $v_{max} \in \mathbb{R}$, such that $0 < |\mathbf{v}_i(t_m)| < v_{max}$. It also selects a heading direction $\Phi_i(t_m)$ in radians, sampled uniformly from the interval $[0, 2\pi)$. These parameters determine the new velocity $\mathbf{v}_i(t_m)$ for the m-th duration, such that:

$$
\begin{aligned}
v_{i,x}(t_m) &= |\mathbf{v}_i(t_m)| \cos(\Phi_i(t_m)) \\
v_{i,y}(t_m) &= |\mathbf{v}_i(t_m)| \sin(\Phi_i(t_m))
\end{aligned}
$$

Since the velocity remains constant during each duration, the UAV i's position at time t, where $t_m \leq t < t_{m+1}$, can be expressed as:

$$
\mathbf{p}_i(t) = \mathbf{p}_i(0) + \sum_{l=0}^{m-1} \delta_{i,l} \mathbf{v}_i(t_l) + (t - t_m)\mathbf{v}_i(t_m) \tag{6}
$$

Here, $\mathbf{p}_i(0) = (x_i(0), y_i(0))$ is the initial position, which is given.

To adapt the RD model for multi-UAV systems, we integrate a simple collision avoidance mechanism. Specifically, if the distance between UAV i and any other UAV j falls below a threshold D_i, UAV i is assumed to turn around and move in the opposite direction while maintaining its speed until the current duration $\delta_{i,l}$ is completed. If both UAVs have the same threshold, *i.e.*, $D_i = D_j$, UAV j will also reverse its direction, as illustrated in Fig. 3b. Moreover, by treating the boundaries of the area as obstacles, we ensure that all UAVs move within the designated area. The mobility of each UAV i with collision avoidance can then be described as:

$$
\begin{aligned}
\mathbf{p}_i(t) = \mathbf{p}_i(0) &+ \sum_{l=0}^{m-1} (\delta_{i,l}^o - \delta_{i,l}^c)\mathbf{v}_i(t_l) \\
&+ (\tilde{\delta}_{i,m}^o - \tilde{\delta}_{i,m}^c)\mathbf{v}_i(t_m)
\end{aligned} \tag{7}
$$

where $0 \leq \delta_i^o \leq \delta_{i,l}$ is the time spent moving at velocity $\mathbf{v}_i(t_l)$ in the l-th instance before triggering collision avoidance and $\delta_{i,l}^c = \delta_{i,l} - \delta_{i,l}^o$. Likewise, $0 \leq \tilde{\delta}_{i,m}^o \leq t - t_m$ is the time spent moving at velocity $\mathbf{v}_i(t_m)$ before collision avoidance in the m-th instance, and $\tilde{\delta}_{i,m}^c = (t - t_m) - \tilde{\delta}_{i,m}^o$.

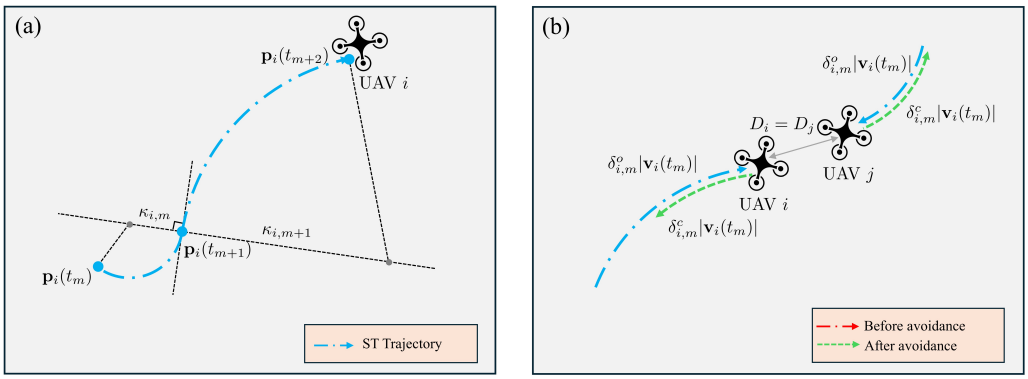

Figure 4: Smooth Turn Model for Multirotor UAVs . (a) ST trajectory of single UAV i, starting from $\mathbf{p}_i(t_m)$ to $\mathbf{p}_i(t_{m+2})$. (b) Trajectories of UAV i and j before and after collision avoidance in the ST multi-UAV system.

3.3.2. ST Model for Fixed-Wing UAVs with Collision Avoidance

In the traditional ST model [14], UAV i randomly picks a turning center located along the line perpendicular to its current heading direction. The UAV then moves at a constant speed following the circular trajectory around the turning center for a randomly chosen duration, repeating this process as illustrated in Fig. 4a. Mathematically, given that the speed $|\mathbf{v}_i(t)|$ remains constant for all time t, UAV i changes its turning center at the start time $t_m = \sum_{l=0}^{m-1} \delta_{i,l}$ of the m-th duration, where each $\delta_{i,l}$ is randomly sampled from an exponential distribution with parameter λ_{st}, and t_0 is set to 0. Additionally, at time t_m, UAV i also randomly picks a turning radius $\kappa_{i,m} \in \mathbb{R}$, where its inverse follows a standard normal distribution with zero mean and variance of σ^2. The sign of $\kappa_{i,m}$ determines the turning direction, with positive values indicating clockwise rotations and negative values indicating counterclockwise rotations. Once the turning radius is determined, the center of the circular trajectory is uniquely determined along the line perpendicular to the heading angle $\Phi_i(t_m)$. As $\kappa_{i,m} \in \mathbb{R}$ remains unchanged during the m-th duration, the heading angle $\Phi_i(t)$ of UAV i at time t, where $t_m \leq t < t_{m+1}$, is given by:

$$\Phi_i(t) = \Phi_i(t_m) - \frac{|\mathbf{v}_i(t)|}{\kappa_{i,m}}(t - t_m) \tag{8}$$

and the velocity follows as:

$$v_{i,x}(t) = |\mathbf{v}_i(t)| \cos(\Phi_i(t))$$
$$v_{i,y}(t) = |\mathbf{v}_i(t)| \sin(\Phi_i(t))$$

The UAV i's position at time t can be computed with $\mathbf{v}_i(t) = [v_{i,x}, v_{i,y}]$ as follows:

$$\mathbf{p}_i(t) = \mathbf{p}_i(0) + \sum_{l=0}^{m-1} \int_{t_l}^{t_{l+1}} \mathbf{v}_i(\tau)\, d\tau + \int_{t_m}^{t} \mathbf{v}_i(\tau)\, d\tau \tag{9}$$

Similarly, to extend the ST model to multi-UAV systems, we incorporate a collision avoidance mechanism similar to that described in the previous subsection. Specifically, when the distance between two UAVs falls below a threshold, UAVs retrace their prior circular trajectory in the reverse direction to restore safe inter-UAV spacing and maintain the spatial relationship, as illustrated in Fig. 4b. As a result, the mobility of UAV i is modified to:

$$\begin{aligned} \mathbf{p}_i(t) = \mathbf{p}_i(0) + \sum_{l=0}^{m-1} & \int_{t_l}^{t_l+\delta_{i,l}^o-\delta_{i,l}^c} \mathbf{v}_i(\tau)\, d\tau \\ & + \int_{t_m}^{t_m+\tilde{\delta}_{i,m}^o-\tilde{\delta}_{i,m}^c} \mathbf{v}_i(\tau)\, d\tau \end{aligned} \tag{10}$$

where $\delta_{i,l}^o$ represents the duration during which UAC i moves along the designated arc trajectory in its original direction (determined by the sign of $\kappa_{i,l}$), and $\delta_{i,l}^c$ denotes the duration during which UAV i moves in the opposite direction along the circular trajectory as a result of the collision avoidance maneuver within the l-th duration. These durations satisfy the relationship $\delta_{i,l}^o + \delta_{i,m}^c = \delta_{i,m}$. Additionally, let $\tilde{\delta}_{i,m}$ be the elapsed duration in m-th instance up to time t. $\tilde{\delta}_{i,m}^o$, $\tilde{\delta}_{i,m}^c$ represent the splits of this duration for the original direction and the reversed direction, respectively, such that $\tilde{\delta}_{i,m} = \tilde{\delta}_{i,m}^o + \tilde{\delta}_{i,m}^c$.

4. Problem Formulation

Without loss of generality, we let UAV $i = 0$ be the master (or offloader) and treat the remaining N UAVs as potential offloadees with idle computing resources. The master UAV must decide how to partition its computational workload by determining the fraction of tasks to offload to each worker UAV

and the fraction to process locally. At the same time, the master UAV must allocate available communication resources, including subcarriers (or channel bandwidth) and transmission power, while operating within the constraints of its energy and bandwidth budgets. Suppose a sequence of computation tasks $\mathcal{K} = \{1, 2, ..., K\}$ is generated at the master, and each task k can be divided into $L_k \in \mathbb{Z}^+$ atomic tasks, each of size $\ell_k = \frac{S_k}{L_k}$, which can be computed in parallel. Consider a UAV network utilizing Orthogonal Frequency Division Multiple Access (OFDMA)[39, 40] that divides the available bandwidth into multiple orthogonal subcarriers to dynamically allocate them to users based on service requirements, where the total available communication bandwidth is represented by B (Hz). This bandwidth is evenly divided into $W \in \mathbb{Z}^+$ orthogonal subcarriers, with each subcarrier having a bandwidth of $b = \frac{B}{W}$ (Hz). The master UAV, responsible for managing resources and task allocation, operates within a total power budget denoted by $\Psi \in \mathbb{R}$.

4.1. Joint Optimization Problem

The goal of the master UAV is to minimize the total task completion time while ensuring minimal energy consumption. To achieve this, it strategically allocates the L_k atomic tasks across all available UAVs, including itself. Furthermore, the master UAV dynamically assigns network resources to balance computational and communication loads, thereby ensuring efficient utilization of the network's bandwidth and power resources.

Suppose the workload offloaded to UAV $j \in \mathcal{N}$ for task k is $c_j^k \ell_k$, where c_j^k is a non-negative integer that satisfies $0 \leq c_j^k \leq L_k$. The total workload allocation must satisfy $\sum_{j=0}^{N} c_j^k = L_k$. Of note, when no workload is offloaded to UAV j, then $c_j^k = 0$.

Regarding network resources, let w_j^k represent the number of subcarriers allocated to the channel between master and UAV $j \in \mathcal{N} \backslash \{0\}$. w_j^k is a non-negative integer such that $0 \leq w_j^k \leq W$, resulting in a channel bandwidth of $B_{0j}^k = w_j^k b$. Additionally, the transmission power allocated by the master UAV to this channel is denoted as $\psi_{0j}^k \in \mathbb{R}$, which is subject to the constraint $\sum_{j=1}^{N} \psi_{0j}^k = \Psi$.

Let T_k denote the time taken to compute task k, and $T_{k,j}$ represent the time taken by UAV j to receive the data, process it and return the result back to the master. We then have

$$T_k = \max_j \ T_{k,j} \tag{11}$$

For simplicity, we assume that the result size is small and the latency of sending it back to the master is negligible, as often assumed in existing works [41]. Therefore, $T_{k,j}$ can be expressed as

$$T_{k,j} = T_{k,j}^{comp} + T_{k,0j}^{trans} \tag{12}$$

According to the computing and communication models described in the previous section, we can derive that $T_{k,j}^{comp} = \frac{\xi_k c_j^k \ell_k}{f_j(t)}$ and $T_{k,j}^{trans}$ satisfies

$$\begin{cases} \int_0^{T_{k,0j}^{trans}} \nu_{0j}^k(\tau)\, d\tau = c_j^k \ell_k & \text{if } j \neq 0 \\ T_{k,0j}^{trans} = 0 & \text{if } j = 0 \end{cases} \tag{13}$$

Regarding the energy consumption E_k for offloading task k, we focus on the energy consumption of the entire NAC system. This includes energy consumed for both computation and transmission across all the computing nodes, encompassing the master and workers. Specifically, E_k is given by:

$$E_k = \sum_{j=0}^{N} E_{k,j}^{comp} + \sum_{j=1}^{N} E_{k,0j}^{trans} \tag{14}$$

The Joint Optimization of Task Offloading and Resource Allocation for Networked UAV Systems focuses on efficiently distributing computational tasks and communication resources across a UAV network comprising a master UAV and multiple worker UAVs. In this system, the master UAV must decide how to partition its computational workload by determining the fraction of tasks to offload to each worker UAV and the fraction to process locally. At the same time, the master UAV must allocate available communication resources, including subcarriers (or channel bandwidth) and transmission power, while operating within the constraints of its energy and bandwidth budgets.

The objective of the joint optimization problem is to minimize the weighted sum of the system's total energy consumption and overall task completion

time, which is mathematically formulated as follows:

$$\underset{c_j^k, w_j^k, \psi_j^k, \forall k \in \mathcal{K}, j \in \mathcal{N}}{\text{Minimize}} \quad \sum_{k=1}^{K} (1-\eta) T_k + \eta E_k \tag{15a}$$

$$\text{subject to} \quad \sum_{j=0}^{N} c_j^k \ell_k = S_k, \ \sum_{j=1}^{N} w_j^k = W, \ \sum_{j=1}^{N} \psi_j^k = \Psi, \tag{15b}$$

$$c_j^k \in \mathbb{Z}, \ 0 \leq c_j^k \leq L_k, \ \forall j \in \mathcal{N}, k \in \mathcal{K} \tag{15c}$$

$$w_j^k \in \mathbb{Z}, \ 0 \leq w_j^k \leq W, \ \forall j \in \mathcal{N}, k \in \mathcal{K} \tag{15d}$$

$$\psi_j^k \in \mathbb{R}, \ 0 \leq \psi_j^k \leq \Psi, \ \forall j \in \mathcal{N}, k \in \mathcal{K} \tag{15e}$$

Here, $\eta \in [0, 1]$ is a weighting parameter that balances the trade-off between energy consumption and task completion time. A smaller η emphasizes minimizing task completion time, while a larger η prioritizes reducing energy consumption.

5. DRL-based Framework for Joint Task Offloading and Resource Allocation

To solve the optimization problem formulated in the previous section, the greatest challenge lies in the unknown relationship between T_k, E_k and the decision variables c_j^k, w_j^k, ψ_j^k, as UAVs lack knowledge of the system models. Additionally, the randomness of UAVs' mobility presents another significant challenge. In this section, we introduce a model-free DRL-based framework to address these challenges.

5.1. Markov Decision Process

We first convert the optimization problem defined in Eq. (15) as a Markov Decision Process (MDP), represented by the tuple $(\mathcal{S}, \mathcal{A}, r, P)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, P is the state transition model, and r is the reward function. This conversion provides a framework for modeling the sequential decision-making process in a dynamic environment. By employing RL algorithms, particularly model-free approaches, the master UAV (as the agent) can iteratively learn an optimal policy. This eliminates the need for complete system knowledge and allows the agent to adapt to stochastic network dynamics, providing an effective solution for minimizing the weighted objective function. In the following subsections, we provide a detailed definition of each component of the MDP tuple.

5.1.1. State

We assume that the master agent has access only to the real-time positions of all UAVs, denoted as $\mathbf{p}_j(t)$ for all $j \in \mathcal{N}$, at each time t. Additionally, the agent possesses prior knowledge of the observed idle computational resources $f_j(t)$ available on each UAV at time t. These inputs provide the foundational information for dynamic task offloading and resource allocation.

To capture the temporal dynamics of the UAV network, continuous time is discretized into intervals of fixed length Δt. Let t^k denote the start time of task k. At this point, the state is defined as a combination of the task's characteristics, the recent trajectories of all UAVs, and their respective configurations. Formally, the state at t^k is expressed as:

$$s_k = \left[S_k, \left(\boldsymbol{\rho}_0(k), f_0(t^k)\right), \left(\boldsymbol{\rho}_1(k), f_1(t^k)\right), \ldots, \left(\boldsymbol{\rho}_N(k), f_N(t^k)\right)\right] \in \mathcal{S},$$

where S_k represents the size of the incoming task k, $\boldsymbol{\rho}_j(k)$ denotes the recent trajectory of UAV j, and $f_j(t^k)$ corresponds to the observed idle computational resources of UAV j at the start of task k.

The trajectory $\boldsymbol{\rho}_j(k)$ is constructed by stacking the $M+1$ most recent positions of UAV j, including its position at t^k, to capture mobility patterns critical for estimating task execution and communication feasibility in the future. Mathematically, $\boldsymbol{\rho}_j(k)$ is defined as:

$$\boldsymbol{\rho}_j(k) = \left[\mathbf{p}_j(t^k - M\Delta t), \mathbf{p}_j(t^k - (M-1)\Delta t), \ldots, \mathbf{p}_j(t^k)\right].$$

Here, $\mathbf{p}_j(t)$ represents the coordinates of UAV j at time t.

This comprehensive state representation provides a detailed view of the system at time t^k, integrating spatial and computational aspects of the UAV network. The inclusion of task size S_k, UAV trajectories $\boldsymbol{\rho}_j(k)$, and idle computational resources $f_j(t^k)$ allows the master agent to make informed decisions regarding task partitioning and resource allocation. By effectively capturing the pattern of system's dynamics, this state formulation ensures the agent can optimize task offloading policies in a highly dynamic and stochastic environment.

5.1.2. Action

Every time a task k arrives, the master UAV partitions it into sub-tasks and distributes them among the UAVs while allocating communication resources including subcarriers and transmission power. Here, we introduce the notations $\zeta_j^{k,d} \geq 0$, $\zeta_j^{k,s} \geq 0$, and $\zeta_j^{k,p} \geq 0$ to represent the fraction of task

k assigned to UAV $j \in \mathcal{N}$, the fraction of subcarriers allocated to UAV j for task k, and the fraction of transmission power allocated to UAV j for task k, respectively. The superscripts d, s, and p indicate task allocation, subcarrier allocation, and transmission power allocation, respectively. These fractions satisfy the following constraints:

$$\sum_{j=0}^{N} \zeta_j^{k,d} = 1, \quad \sum_{j=1}^{N} \zeta_j^{k,s} = 1, \quad \text{and} \quad \sum_{j=1}^{N} \zeta_j^{k,p} = 1, \tag{16}$$

ensuring that the entire task, all subcarriers, and the total transmission power are fully allocated. The workload offloaded, expressed as the number of atom tasks, and the subcarrier allocations, expressed as the number of subcarriers, are computed by rounding the fractional values as follows:

$$c_j^k = \text{round}\left(\zeta_j^{k,d} L_k\right),$$

$$w_j^k = \text{round}\left(\zeta_j^{k,s} W\right).$$

The transmission power allocated to UAV j is retained as a fractional value:

$$\psi_j^k = \zeta_j^{k,p} \Psi.$$

The action taken by the master UAV at the start time t^k is then defined as:

$$a_k = \left[\zeta_0^{k,d}, \ldots, \zeta_N^{k,d}; \zeta_1^{k,s}, \ldots, \zeta_N^{k,s}; \zeta_1^{k,p}, \ldots, \zeta_N^{k,p}\right] \in \mathcal{A},$$

where a combination of an N-simplex and two $(N-1)$-simplices forms the action space $\mathcal{A}$.

5.1.3. Reward

To jointly minimize the total task completion time and energy consumption, we define the reward function as a weighted sum of the improvement ratios in task completion time and energy consumption for each task k. Specifically, the reward function is defined as:

$$r(s_k, a_k) = (1-\eta)\left(\frac{\tilde{T}_k - T_k}{\tilde{T}_k}\right) + \eta\left(\frac{\tilde{E}_k - E_k}{\tilde{E}_k}\right), \tag{17}$$

where $\eta \in [0, 1]$ is the weighting parameter that controls the trade-off between energy consumption and task completion time.

In this formulation, T_k represents the actual task completion time achieved through offloading, and $\tilde{T}_k = \frac{\xi_k S_k}{f_0(t)}$ denotes the time required to execute task k locally at the master UAV. The term $\frac{\tilde{T}_k - T_k}{\tilde{T}_k}$ quantifies the relative improvement in task completion time achieved by offloading. Similarly, E_k is the energy consumed for task k with offloading, and $\tilde{E}_k$ is the energy required to execute the task locally at the master UAV. The term $\frac{\tilde{E}_k - E_k}{\tilde{E}_k}$ quantifies the relative improvement in energy consumption. By combining these two ratios, the reward function ensures that both latency and energy efficiency are considered.

5.1.4. Transition

As discussed in Sec. 3, all UAVs move randomly according to the RMMs that incorporate collision avoidance schemes to ensure safe operation. The task at time t^k is completed within a total wall time T_k, a random variable dependent on the current state s_k of all UAVs and the offloading action a_k. Consequently, the next state s_{k+1}, starting at $t^{k+1} = t^k + T_k$, is governed by the transition dynamics of the mobility model and the actions taken by the master UAV.

The transition model, which describes state transitions based on the current state and action, depends on the specific RMM. It is abstracted as:

$$s_{k+1} \sim \begin{cases} P(s_{k+1}|s_k, a_k; v_{\max}, \lambda_{\mathrm{rd}}) & (\mathrm{RD}) \\ P(s_{k+1}|s_k, a_k; \sigma, \lambda_{\mathrm{st}}) & (\mathrm{ST}) \end{cases} \tag{18}$$

Here, $P(\cdot)$ represents the transition probability, whose explicit form is unknown and depends on the selected RMM. Specifically, for UAVs modeled by the RD model, $v_{\max}$ determines the maximum velocity of the UAVs, and λ_{rd} governs the frequency of direction changes. For UAVs modeled by the ST model, σ determines the radius of the circular trajectory, and λ_{st} specifies the frequency of direction changes, influencing how often the UAV reorients its trajectory.

Although the explicit form of $P(\cdot)$ is generally unknown, it can be inferred or approximated using observed UAV trajectories and environmental conditions. This abstraction provides a robust framework for effective decision-making in task allocation and resource management in dynamic, networked UAV systems.

5.2. DRL Methodology

Deterministic policy gradient methods, such as DDPG [15] and TD3 [16], compute a specific optimal action for each state, excelling in precision-critical tasks like resource allocation. Stochastic policy gradient methods, such as PPO [17], learn a probability distribution over actions, offering robustness to uncertainty and enhanced exploration in dynamic environments. To address the unique challenges of NAC systems, we adapt these methods to handle the continuous action spaces and simplex constraints required for joint task offloading, subcarrier allocation, and power distribution, while preserving their core advantages.

5.2.1. Deterministic Policy Gradient Methods

In deterministic policy gradient methods, the actor learns a deterministic mapping from states to specific actions, enabling precise decision-making, while the critic evaluates these state-action pairs using a value function to guide the actor's policy refinement.

Actor. The actor in both DDPG and TD3 is is implemented as a neural network parameterized by ϕ, denoted as $\pi_\phi : \mathcal{S} \to \mathcal{A}$. The policy maps each state $s \in \mathcal{S}$ to an action $a \in \mathcal{A}$, specifying the fractions of data, subcarriers, and transmission power allocated to each UAV. The objective of the actor is to determine an optimal policy π^* that maximizes the expected return:

$$J = \mathbb{E}_{s_k \sim P, a_k \sim \pi}\left[R_1\right], \quad R_k = \sum_{i=k}^{K} \gamma^{i-k} r(s_i, a_i),$$

where γ is the discount factor, and $r(s_i, a_i)$ represents the reward at step i. A target actor network parameterized by ϕ' is also maintained, from which reliable reference action can be taken deterministically to facilitate stable learning and ensure smooth updates.

The neural network for the actor consists of fully connected layers. The input layer encodes the state features, while the intermediate layers apply non-linear activations (e.g., ReLU) to extract latent features that capture the movement patterns of individual UAVs and their interactions, such as collision avoidance. The output layer produces the following logits:

$$[\mathbf{z}^{k,d}, \mathbf{z}^{k,s}, \mathbf{z}^{k,p}] \in \mathbb{R}^{3N+1}.$$

where $\mathbf{z}^{k,d} = [z_j^{k,d}]_{j=0}^{N}$ contains logits representing the reliability scores where the UAV with higher score will be allocated more workload of UAVs for task

offloading, $\mathbf{z}^{k,s} = [z_j^{k,s}]_{j=1}^N$ corresponds to logits prioritizing subcarrier allocation, and $\mathbf{z}^{k,p} = [z_j^{k,p}]_{j=1}^N$ represents logits for transmission power allocation. Each logit reflects resource demands or reliability based on the UAV's proximity, computational resources, and movement stability. The logits are then transformed into valid actions using the softmax function:

$$\zeta_j^{k,\chi} = \text{Softmax}(z_i^{\chi}) = \frac{e^{z_j^{k,\chi}}}{\sum_{j=0}^{N} e^{z_j^{k,\chi}}} \tag{19}$$

where $\chi \in d, s, p$ represents the different components for data, subcarriers, or transmission power. The resulting fractions $\zeta_j^{k,\chi}$ satisfy the simplex constraints in Eq. (16), ensuring that resource allocations are both valid and normalized.

To align exploration with the simplex constraints of the action space, we introduce Dirichlet-distributed noise in place of the Gaussian noise traditionally used in DDPG and TD3. The Dirichlet distribution ensures that the exploration noise remains within the simplex, making it inherently compatible with our resource allocation problem. The noisy actions during exploration are defined as:

$$\begin{aligned} \tilde{\zeta}_j^{k,\chi} =& (1 - \beta_{explore})\zeta_j^{k,\chi} + \beta_{explore}\epsilon_j \\ \epsilon_j \sim& \text{Dir}(\alpha_{explore}) \end{aligned} \tag{20}$$

where $\beta_{explore}$ controls the noise intensity, and ϵ_j is sampled from a Dirichlet distribution with a uniform concentration parameter $\alpha_{explore}$. This adaptation not only preserves the validity of the allocations but also ensures smooth exploration, allowing the agent to learn efficiently within the structured action space.

Critic. In our optimization problem, the critic evaluates the quality of task, subcarrier, and power allocations, enabling the actor to optimize these decisions effectively in dynamic networked UAV environments. The critic is based on the state-action value function, $Q^{\pi}(s_k, a_k)$, which maps a state s_k and an action a_k to their expected return. Mathematically, the function is defined as:

$$Q^{\pi}(s_k, a_k) = \mathbb{E}_{s_{i>k} \sim P, a_{i>k} \sim \pi}[R_k | s_k, a_k],$$

which can be estimated with immediate reward and the value of the next state-action pair according to the Bellman equation:

$$Q^{\pi}(s_k, a_k) = r(s_k, a_k) + \gamma \mathbb{E}_{s_{k+1} \sim P}[Q^{\pi}(s_{k+1}, \pi(s_{k+1}))] \tag{21}$$

In both DDPG and TD3, the critic is implemented as a neural network parameterized by weights θ. The network learns to approximate $Q^{\pi}(s_k, a_k)$ by minimizing the temporal difference (TD) error with the loss function given as:

$$\mathcal{L}(\theta) = \mathbb{E}\left[\left(Q_{\theta}(s_k, a_k) - y_k\right)^2\right], \tag{22}$$

where y_k is the target Q-value computed from the Bellman equation in Eq. (21). Given one pair of critic networks parameterized with θ_1 and the target critic network parameterized with θ_1', our adapted DDPG method for energy-efficient offloading and network resource allocation, estimate the target Q value as:

$$y_k = r(s_k, a_k) + \gamma Q_{\theta_1'}(s_{k+1}, a_{k+1}), \tag{23}$$

where $a_{k+1} = \pi_{\phi'}(s_{k+1})$ is the target action in the next step. In our adapted TD3 algorithm, the target Q-value would be estimated with one additional pair of critic networks and target critic networks parameterized by (θ_2, θ_2') as follows:

$$y_k = r(s_k, a_k) + \gamma \min_{j=1,2} Q_{\theta_j'}(s_{k+1}, \tilde{a}_{k+1}) \tag{24}$$

where $\tilde{a}_{k+1} = \pi_{\phi'}(s_{k+1}) + \epsilon$ is a noised target action. To prevent overestimation of action values, we introduce Dirichlet noise, similar to the exploration noise in Eq. (20), to the actions produced by the target policy $\pi_{\phi'}$, enhancing the robustness of policy learned and improving stability. The smoothed target action is defined as:

$$\begin{aligned} \tilde{\zeta}_j^{k,\chi} =& (1 - \beta_{policy})\zeta_j^{k,\chi} + \beta_{policy}\epsilon_j \\ \epsilon_j \sim& \mathrm{Dir}(\alpha_{policy}) \end{aligned} \tag{25}$$

where β_{policy} controls the noise intensity, $\zeta_j^{k,\chi}$ represents the action produced by the target actor, and α_{policy} is the concentration parameter for policy. This modification ensures that the smoothed target actions remain valid with the structural requirements of our problem.

Training. The training procedure for the DDPG and TD3 follows an off-policy learning paradigm to decouple the learning process from the agent's interaction with the environment. As shown in Alg. 1, for both DDPG and TD3, training starts with the initialization of critic networks, actor networks,

Algorithm 1 Training for task offloading (DDPG, TD3)

1: Initialize the critic networks $Q_{\theta_1}, Q_{\theta_2}$(TD3) by randomly assigning values to θ_1, θ_2, and initialize the target critic networks $Q_{\theta_1'}, Q_{\theta_2'}$(TD3) by setting $\theta_1' \leftarrow \theta_1, \theta_2' \leftarrow \theta_2$
2: Initialize the actor network π_ϕ and the target actor $\pi_{\phi'}$ by randomly assigning values to ϕ and setting $\phi' \leftarrow \phi$
3: Initialize the replay buffer by $\mathcal{D} \leftarrow \emptyset$
4: **for** iteration= 1 to U **do**
5: Select action $a \sim (1-\beta)\pi_\phi(s) + \beta\epsilon$, with exploration noise $\epsilon \sim \text{Dir}(\alpha_{explore})$, observe reward r and new state s'
6: Store the transition tuple (s, a, r, s') in $\mathcal{D}$
7: **if** iteration$> H$ **then**
8: Sample a batch of $|\mathcal{B}|$ transitions (s, a, r, s') from buffer $\mathcal{D}$
9: Update the critics by:
10: $\theta_i \leftarrow \arg\min_{\theta_i} \frac{1}{|\mathcal{B}|} \sum (y - Q_{\theta_i}(s,a))^2$, $i = 1, 2$
11: **if** iteration $\mod u = 0$ **then**
12: Update the actor by:
13: $\phi \leftarrow \arg\max_\phi \frac{1}{|\mathcal{B}|} \sum Q_{\theta_1}(s, \pi_\phi(s))$
14: Update the target network by:
15: $\theta_i' \leftarrow \theta_i + (1-\tau)(\theta_i' - \theta_i), i = 1, 2$
$\phi' \leftarrow \phi + (1-\tau)(\phi' - \phi)$
16: **end if**
17: **end if**
18: **end for**

and their corresponding target networks. The critic networks, Q_{θ_1} and Q_{θ_2} (for TD3), are initialized with random parameters θ_1 and θ_2, while the target critics $Q_{\theta_1'}$ and $Q_{\theta_2'}$ are set to match their primary counterparts. Similarly, the actor network π_ϕ and its target network $\pi_{\phi'}$ are initialized with $\phi' \leftarrow \phi$. An empty replay buffer $\mathcal{D}$ is prepared to store transition samples (s, a, r, s') (Lines 1-3). At each training iteration h, the master agent interacts with the environment to collect transition data and store them in the buffer $\mathcal{D}$ until the number of collected transitions exceeds H (Lines 5-6). After that, the agent utilizes a batch $\mathcal{B}$ sampled from the replay buffer to update the parameters of the critics (Lines 9) and actor (Lines 10-11). Particularly, the parameters θ_i, $i \in \{1, 2\}$, of each primary critic are updated by minimizing the following loss function, which approximates the expectation in Eq. (22)

using the average over the sampled batch:

$$l_i = \frac{1}{|\mathcal{B}|} \sum_{(s,a,r,s') \in \mathcal{B}} (y - Q_{\theta_i}(s,a))^2 \tag{26}$$

Here, y is derived from Eq. (23) for DDPG or Eq. (24) for TD3.

We update the policy function every u iterations by maximizing the expected return in the direction of the batch gradient of the policy, where the gradient is computed by:

$$\nabla_\phi J = \frac{1}{|\mathcal{B}|} \sum_{(s,a,r,s') \in \mathcal{B}} [\nabla_a Q_{\theta_1}(s,a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)] \tag{27}$$

Of note, TD3 has a lower update frequency with $u > 1$ than DDPG with $u = 1$. Likewise, the parameters of the target networks are gradually adjusted towards the weights of the primary networks through weighted soft updates every u iteration as follows:

$$\begin{aligned} &\theta_i' \leftarrow \theta_i + (1-\tau)(\theta_i' - \theta_i), i = 1, 2 \\ &\phi' \leftarrow \phi + (1-\tau)(\phi' - \phi) \end{aligned} \tag{28}$$

where $\tau \in [0, 1]$ is the weight.

5.2.2. Stochastic Policy Gradient Method

In contrast to deterministic approaches like DDPG and TD3, Proximal Policy Optimization (PPO) employs a stochastic policy that learns a probability distribution over possible actions which naturally satisfies our simplex constraints for resource allocation., making it particularly well-suited for handling the uncertainty inherent in NAC task offloading environments.

Actor. The actor in PPO is implemented as a neural network parameterized by ϕ, which maps states to the concentration parameters of three Dirichlet distributions, one each for data offloading, subcarrier allocation, and transmission power allocation. The network architecture consists of fully connected layers that process state features and output concentration parameters $\alpha \in \mathbb{R}_+^{3N+1}$ for the Dirichlet distributions:

$$\pi_\phi(a|s) = \prod_{\chi \in d,s,p} \mathrm{Dir}(\zeta^\chi | \alpha^\chi(s)) \tag{29}$$

where $\alpha^{\chi}(s)$ are the concentration parameters for each resource type χ, and ζ^{χ} represents the corresponding allocation fractions on a simplex, thereby enforcing valid fractions for data offloading, subcarrier allocation, and power allocation. By sampling from the learned Dirichlet distribution, we ensures that the sampled actions naturally satisfy the simplex constraints of our resource allocation problem. In addition, exploration on simplex is automatically managed through sampling, without the need to manually setting the noise intensity in DDPG and TD3.

Critic. The critic evaluates state values function, $V^{\pi}(s_k)$, which maps a state s_k to its expected return when following policy π. Mathematically, the function is defined as:

$$V^{\pi}(s_k) = \mathbb{E}_{s_{i>k} \sim P, a_{i \geq k} \sim \pi}[R_k | s_k], \tag{30}$$

rather than state-action values $Q^{\pi}(s_k, a_k)$ in DDPG or TD3. In our UAV task offloading scenario, this state-value estimation helps evaluate the overall system efficiency for a given network state.

The critic is implemented as a neural network parameterized by a separate parameter θ that's not shared with the actor[42]. For our resource allocation problem, this network processes state features to estimate the long-term value of the system configuration. The advantage function, which measures how much better or worse specific resource allocation decisions are compared to the average performance, is computed using Generalized Advantage Estimation (GAE)[43]:

$$\hat{A}_k = \Delta_k + (\gamma\Lambda)\Delta_{k+1} + ... + (\gamma\Lambda)^{K-k+1}\Delta_{K-1} \tag{31}$$

where $\Delta_k = r_k + \gamma V_{\theta}(s_{k+1}) - V_{\theta}(s_k)$ is the temporal difference error, γ is the discount factor, and Λ is the GAE parameter. This advantage estimation is particularly crucial for our problem as it helps differentiate between resource allocation decisions that might appear similarly beneficial in the short term but have different long-term impacts on system performance.

Training. The training procedure adapted from PPO follows an on-policy learning paradigm that simultaneously optimizes three key objectives: efficient task offloading and resource allocation through policy improvement, accurate value estimation for system states, and sufficient exploration of the allocation strategy space. As shown in Alg. 2, training starts with the initialization of critic network and actor network(Lines 1). At each iteraion,

the master agent collects trajectory data τ by executing the current policy across K sequential task offloading decisions for $\mathcal{T}$ times(Lines 4-5).

To ensure stable learning of resource allocation strategies, we employ a clipped surrogate objective that prevents excessive policy updates by limiting the ratio between new and old policy probabilities, the policy gradient loss of each trajectory follows:

$$L^{\mathrm{PG}}(\phi) = \min\Big(r(\phi)\hat{A}, \mathrm{clip}\big(r(\phi), 1-\epsilon, 1+\epsilon\big)\hat{A}\Big) \tag{32}$$

where the probability ratio between the new and old policies is defined as:

$$r(\phi) = \frac{\pi_\phi(a|s)}{\pi_{\phi_{\mathrm{old}}}(a|s)},$$

$\hat{A}$ represents the advantage estimate defined in Eq. (31), and ϵ is a hyperparameter that determines the clipping threshold. This clipping mechanism prevents drastic changes in resource allocation policies that could lead to unstable system behavior or violation of service guarantees.

The value function, essential for evaluating the long-term impact of resource allocation decisions, is trained by minimizing the squared error between predicted values and computed target returns as follows:

$$L^{\mathrm{VF}}(\theta) = \big(V_\theta(s) - V^{\mathrm{targ}}(s)\big)^2, \quad V^{\mathrm{targ}}(s) = \hat{A} + V_{\theta_{\mathrm{old}}}(s) \tag{33}$$

To promote exploration of diverse resource allocation strategies and prevent premature convergence to suboptimal policies, we incorporate an entropy bonus term. The complete loss function combines all three objectives:

$$L(\phi, \theta) = -L^{\mathrm{PG}}(\phi) + c_1 L^{\mathrm{VF}}(\theta) - c_2 L^{H}(\phi), \tag{34}$$

where c_1, c_2 are coefficients, and $L^H(\phi)(s_k) = -H(\pi_\phi(\cdot|s))$ denotes an entropy bonus. In our UAV task offloading context, this entropy term encourages the exploration of different resource allocation patterns across multiple computing nodes and helps discover robust strategies that can adapt to varying task loads and network conditions.

The policy and value networks are then updated through multiple epochs of gradient descent on the averaged composite loss function, using sampled trajectories (Lines 8-10).

Algorithm 2 Training for PPO

Initialize the critic networks V_θ and the actor network π_ϕ by randomly assigning values to θ and ϕ
for iteration= 1 to $\mathcal{I}$ **do**
    for episode= 1 to $\mathcal{T}$ **do**
        Run policy π_ϕ for K tasks to collect episode trajectory:
        $\tau = \{(s_k, a_k, r_k, s_{k+1})\}_{k=1}^{K}$
        Compute $L^{\mathrm{PG}}(\phi)$, $L^{\mathrm{VF}}(\theta)$ and $L^{H}(\phi)$ for the episode trajectory
    end for
    for epoch= 1 to O **do**
        Update the critics and actors by:
        $\theta, \phi \leftarrow \arg\min_{\theta,\phi} \frac{1}{\mathcal{T}} L(\theta, \phi)$
    end for
end for

6. Experiments

In this section, we conduct experiments to evaluate the effectiveness and scalability of our proposed DRL algorithms.

6.1. Experimental Setup

Details of the experimental setup are provided in this subsection.

6.1.1. Environment Configuration

We evaluate our methods in simulated scenarios involving a master UAV and $N =$ offloadee UAVs, where N is set to 3, 6, 9, or 12. All UAVs move following either the multi-RD model or multi-ST model. For the multi-RD mobility model, we set its parameters as $v_{max} = 20$ m/s, $\lambda_{rd} = \frac{1}{15}$. For the multi-ST mobility model, we set its parameters as $\sigma = 0.001$, $\lambda_{st} = \frac{1}{15}$. The collision threshold is set to $D_j = 5$, $\forall j \in \mathcal{N}$ in both mobility models. The length of each discrete time step is set to $\Delta t = 1$ second. We also vary the size of the flying zone and consider two configurations, $E = 200$m and $E = 300$m.

The initial idle computing power $f_j(0)$ of each UAV j is randomly assigned within the range of $[1, 1.6]$ GHz. Over time, $f_j(t)$ fluctuates dynamically, following a uniform distribution within the range $[0.9f_j(0), 1.1f_j(0)]$. The task list consists of $K = 25$ tasks, each of which can be completed locally in $\tilde{T}_k \in [20, 60]$ seconds by the master UAV. Each task can be divided into

$L_k = 1000$ atomic tasks, with a constant computation intensity of $\xi_k = 10^6$ cycles/kB. The size of each task is determined by $S_k = \frac{f_0(t^k)\tilde{T}_k}{\xi_k}$.

For the communication model, as a case of a networked UAV swarm utilizing millimeter-wave antennas for air-to-air communication [44], we set the total available bandwidth as $B = 2$ GHz, divided it into subcarriers each with a bandwidth of 15kHz. Other model parameters are configured as follows: relative distance $d_r = 1$ meter, path gain $G = 40$, path loss exponent $\theta = 4$, and noise power spectral density $N_0 = -174$ dBm/Hz. The transmitted power budget Ψ of the master UAV offloading to other UAVs is set to 1W. The QoS requirement for the transmission data rate is set to $\nu_{min} = 30$ kb/s. For simplicity, the power consumed by each receiver j is assumed to be a constant with a value of $\tilde{\psi}_j = 100$ mW.

6.1.2. DRL Configuration and Training Setup

We employ a 3-layer Multi-layer Perceptron (MLP) architecture [45] for all three DRL algorithms. The actor network takes observations as input, whose dimension is determined by the number of computing nodes $(N + 1)$, and outputs actions of dimension N. Each UAV trajectory has a length of $M+1$, where $M = 20$. The critic network takes as input the concatenation of observations and actions and outputs a scalar representing the state-action value in DDPG and TD3 or the state-value in PPO. The width of the hidden layer in both the actor and critic networks is set to twice the dimension of the input. All parameters are initialized using Kaiming initialization [46].

To train DDPG and TD3, the learning rates are set to 0.0001 and 0.0002, respectively. The threshold H is set to 1000 and the batch size is $|\mathcal{B}| = 512$. Gradient norms are clipped within the range $[0, 0.2]$. The discount factor γ is 0.9, and the soft update weight τ for the target networks is 0.005. The actor network in TD3 is updated every $u = 25$ iterations, while in DDPG, updates occur at $u = 1$ iteration. The exploration and policy noises in both DDPG and TD3 are sampled from a Dirichlet distribution with parameters $\alpha_{explore} = \alpha_{policy} = 1$. The corresponding noise weights are $\beta_{explore} = 0.1$ and $\beta_{policy} = 0.01$. In PPO, we use a learning rate of 0.00003. The other hyperparameters in PPO are configured as follows: $\gamma = 0.9$, $\lambda = 0.95$, $\epsilon = 0.2$, $c_1 = 0.9$, and $c_2 = 0$. Gradient clipping is used during training with a maximum norm of 0.5. The policy is updated over $O = 15$ epochs in each iteration.

In each scenario, we train the three DRL agents with the same amount of data sampled from $\mathcal{E} = 4000$ episodes (with $\mathcal{T} = 1$ in PPO), *i.e.*, $U =$

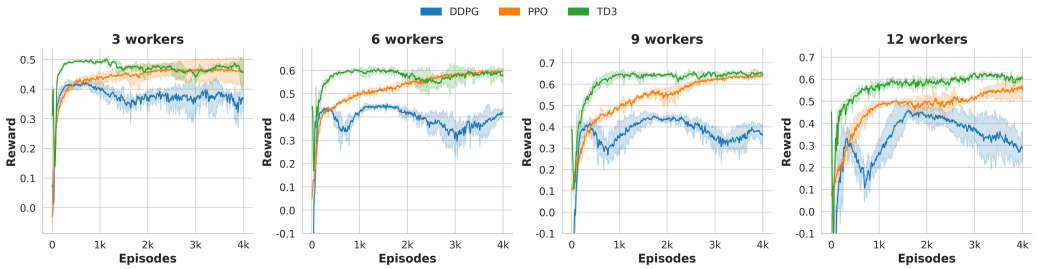

Figure 5: Learning Curve of RD ($200 \times 200m^2$)

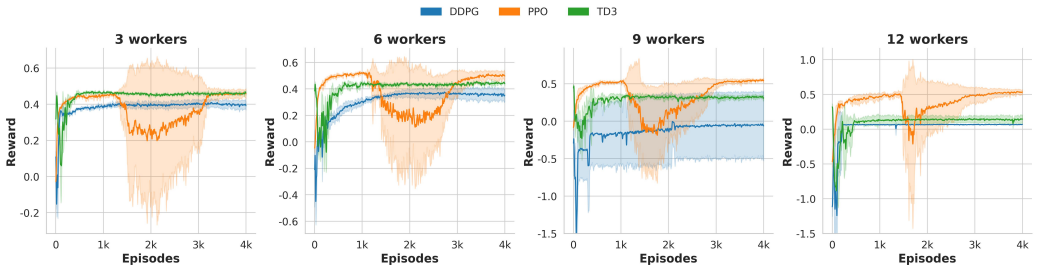

Figure 6: Learning Curve of ST ($200 \times 200m^2$)

$4000K = 10^5$ transitions, using three different random seeds.

6.2. Experimental Results

This subsection presents the results of the experiments.

6.2.1. Training Performance

Multi-RD.

Multi-ST. We first show the training performance of the three DRL algorithms. The results are shown in Fig. 5-Fig. 8. The reward is the averaged reward Eq. (17) across the episode. In Fig. 5, the learning curves of all three DRL algorithms in 4 variant scenarios are displayed, when the UAV swarm moves following RD mobility within a restricted zone of area 200×200. DDPG converges only when there are 3 workers, and as the number of workers increases, it fails to learn a stable policy within U iteration since reward fluctuates largely. TD3 and PPO converge in all 4 scenarios with different numbers of workers, where TD3 has a slight edge over PPO regarding the reward of convergence during training. In the same area of 200×200, if UAVs

move following ST mobility instead, the training performance is shown in Fig. 6. All three DRL agents eventually converge after training although there's a clear drop in the performance of the PPO agent in the middle with much greater variance among experiments of different random seeds, which may be due to the trap and escape from the local minimum during training. In contrast, the agents trained by DDPG and TD3 both have a drop in performance at the very beginning when there are 9 or 12 workers in the environment and converge to a local minimum at a relatively lower level, which means they are trapped in the local minimum and fail to escape. In addition, when there are 9 workers, the training process of DDPG also displays an extreme instability where the variance across experiments is very high.

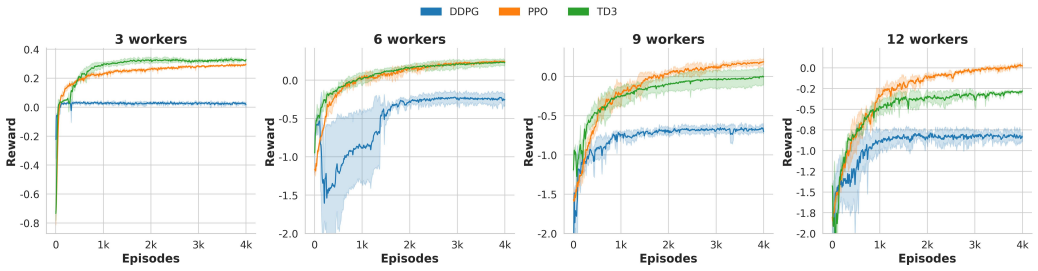

Figure 7: Learning Curve of RD ($300 \times 300m^2$)

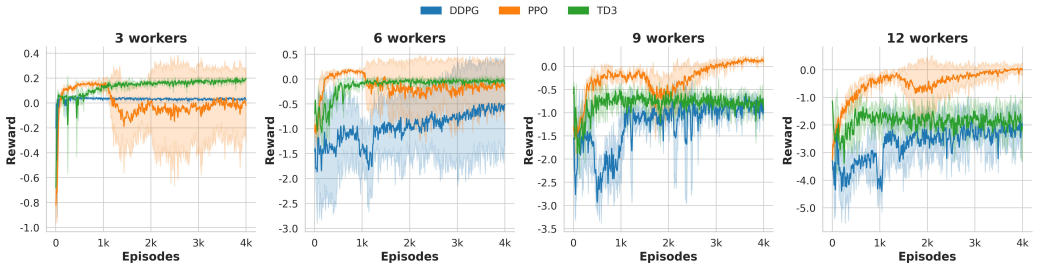

Figure 8: Learning Curve of ST ($300 \times 300m^2$)

If we expand the area to 300×300 m^2 when UAV swarm follows the multi-RD mobility model, the learning curves are displayed in Fig. 7. We can see all three DRL algorithms converge after training, although the averaged reward achieved by the DDPG agent in all 4 variant scenarios is much lower than the

agent trained with TD3 and PPO. When the number of workers increases, PPO gradually achieves an advantage concerning the convergence level over TD3, as the TD3 agent converges to the highest performance when only 3 workers are deployed while PPO takes the lead when there are 12 workers available. If we switch the mobility pattern of the swarm to multi-ST mobility, then the learning processes of all three DRL agents become unstable. Although PPO agents can achieve the highest performance in all 4 variant scenarios, great variance appears when there are 3 workers or 6 workers, demonstrating the PPO is expressive enough to solve the task while its instability during training is also exposed. The learning curves demonstrate the effectiveness of our DRL-based algorithm on the problem that jointly optimizes the time latency and energy consumption by appropriately offloading computation tasks and allocating network resources.

6.2.2. Comparison Studies

6.2.3. Benchmarks

To evaluate the performance of the proposed methods, we compare them with five benchmarks, including

1. **Equal (all)** that equally divides and distributes tasks and network resources to all UAVs;
2. **Equal (close)** that equally partitions and distributes tasks and network resources to UAVs that are close enough with distance to the master satisfying $d_{0j} < 100$m;
3. **Naive Prediction** that selects offloadees based on predicted future trajectories and assigns sub-tasks and allocates network resources of equal size to these offloadees. Specifically, it predicts each UAV's trajectory for the next 20 steps, assuming the UAVs maintain their current velocity and ignoring potential collisions. It then calculates the percentage q_j of predicted UAV positions that remain within 200m of the master ($d_{0j} < 200$m). UAVs with $q_j \geq 40\%$ are selected as offloadees;
4. **Reliable** that allocates tasks based on a reliability score defined as reliability $= q_j f_i$. It picks the top $\lceil \frac{N+1}{2} \rceil$ UAVs with the highest reliability scores and assigns tasks and network resources to these UAVs proportionally to their reliability scores;
5. **Random** that randomly sample actions from the action space.

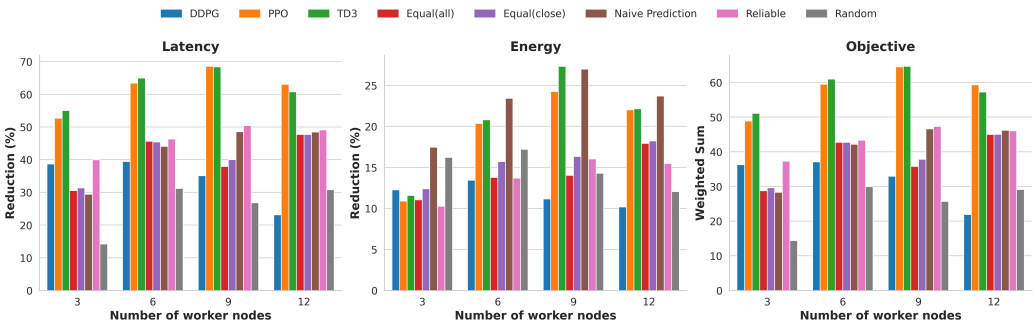

Figure 9: Performance Comparison under Multi-RD mobility in area of $200 \times 200m^2$

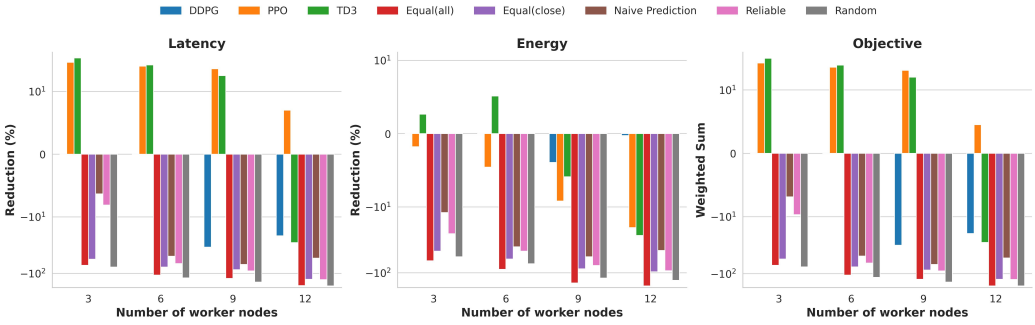

Figure 10: Performance Comparison under Multi-RD mobility in area of $300 \times 300m^2$

6.2.4. Performance Under Multi-RD Mobility

The results in Fig.9 and Fig.10 highlight the superior performance of our DRL-based methods under Multi-RD mobility, particularly in optimizing latency reduction and energy consumption.

When the working zone is limited to $200 \times 200\text{m}^2$, both TD3 and PPO demonstrate significant advantages over baseline methods across all four variant scenarios, achieving the highest latency reduction and superior overall performance considering energy consumption, as shown in Fig. 9. The DDPG agent performs acceptably only in the 3-worker scenario, but its overall performance is slightly worse than the **Reliable** baseline, despite outperforming the remaining five heuristic baselines. This suboptimal result likely stems from DDPG's sensitivity to hyperparameters, particularly as the state dimensionality increases with the addition of more workers in the region. Furthermore, its limited ability to mitigate overestimation bias in

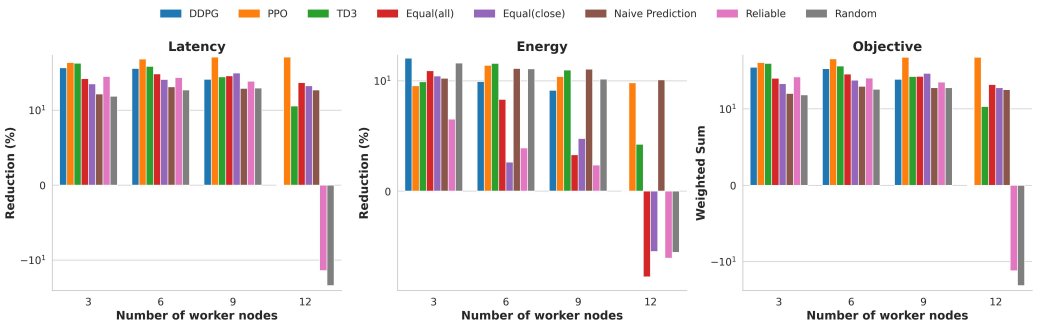

Figure 11: Performance Comparison under multi-ST mobility in area of $200 \times 200m^2$

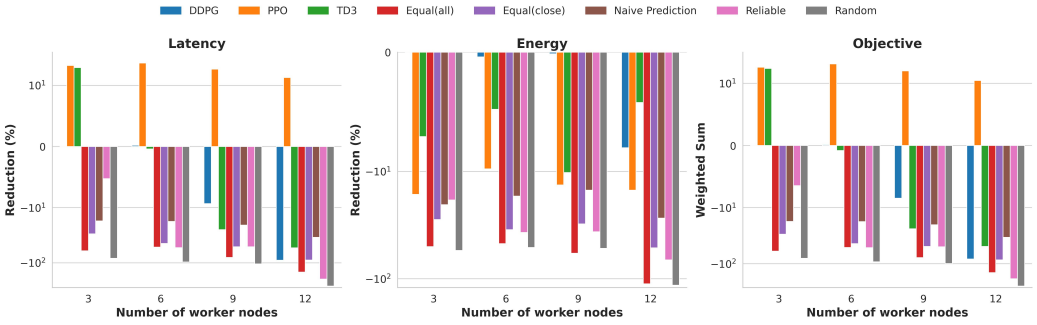

Figure 12: Performance Comparison under multi-ST mobility in area of $300 \times 300m^2$

high-dimensional state spaces contributes to its weaker performance. Among the baseline methods, **Naive Prediction** achieves substantial energy savings by allocating tasks to workers predicted to remain nearby in the future. However, it overlooks differences in UAV computing capabilities, leading to less competitive latency reductions and an overall performance gap compared to TD3 and PPO.

In the expanded $300 \times 300, \mathrm{m}^2$ area, the UAV swarm struggles to achieve meaningful benefits in latency reduction or energy efficiency with policies trained by DDPG or any of the six baseline heuristic methods. However, TD3 outperforms PPO in scenarios with 3 or 6 workers, achieving the best trade-off between time latency and energy consumption. This success is attributed to TD3's ability to leverage the relatively stable conditions of fewer workers to optimize resource allocation precisely. On the other hand, PPO consistently provides overall benefits across all scenarios by prioritizing latency reduction,

albeit with increased energy consumption. In contrast, TD3 fails to deliver any meaningful benefits when the number of workers increases to 12, likely due to convergence to suboptimal policies during training.

6.2.5. Performance Under Multi-RD Mobility

The performance of our DRL-based methods under Multi-ST mobility, as shown in Fig.11 and Fig.12, demonstrates their adaptability in environments with smooth-turn trajectories.

In the $200 \times 200\text{m}^2$ working zone, all three DRL methods—DDPG, TD3, and PPO—outperform baseline methods in scenarios with 3 or 6 workers, showcasing better overall performance (Fig.11). However, as the number of workers increases, the performance of DDPG and TD3 degrades, falling below even the heuristic baselines. This degradation arises from these methods converging to local optima during training, as evidenced in Fig.6. In the 12-worker scenario, baseline methods such as **Reliable** and **Random** also experience significant performance declines, unable to effectively utilize task offloading and resource allocation. Notably, PPO consistently achieves the greatest overall benefits across all scenarios, demonstrating its robustness even as mobility complexity increases.

In the larger $300 \times 300, \text{m}^2$ working zone, only PPO demonstrates consistent benefits across all scenarios under Multi-ST mobility (Fig. 12). It achieves latency reduction at the expense of increased energy consumption, effectively balancing these competing objectives. TD3 provides comparable benefits to PPO in the 3-worker scenario but fails to deliver substantial advantages as the number of workers increases to 6 or 12. DDPG, along with the baseline methods, fails to achieve meaningful performance improvements in this setting, highlighting the challenges posed by higher uncertainty and increased mobility complexity in larger regions.

6.2.6. General Observations and Implications

One notable phenomenon observed across all scenarios is that increasing the number of workers in a restricted working zone does not guarantee improved performance. As shown in Fig.9-Fig.12, performance degradation occurs consistently for all methods, irrespective of the size of the working zone or the mobility model. This degradation is primarily attributed to increased collision avoidance maneuvers [18], which introduce greater uncertainty in UAV mobility, making it harder for the learning algorithm to predict mobility patterns effectively.

Intuitively, the master agent tends to prioritize offloading tasks to UAVs with higher reliability scores, z_i^k, as these are indicative of UAVs likely to remain nearby throughout task execution. However, when UAV mobility becomes more uncertain due to collision avoidance or other dynamics, fewer UAVs can be reliably selected as offloadees. Selecting unreliable offloadees leads to worse outcomes in both latency and energy consumption, sometimes making local computing a preferable alternative.

Furthermore, this observation highlights an inherent limitation: the task completion time cannot be infinitely reduced by simply increasing the number of UAVs in the region. Instead, the optimal number of UAVs that can be deployed to maximize performance benefits depends on the working zone size and the mobility model. These insights underscore the importance of balancing UAV density with the specific constraints of the operational environment to achieve effective resource allocation and task offloading.

7. Conclusion

In this paper, we addressed task offloading problem by jointly optimizing the time latency and energy consumption in a typical NAC scenario, where multiple UAVs with random mobility and collision avoidance capabilities coordinate to perform computational tasks by sharing resources. This problem is challenged by unknown system models uncertainties in UAV mobility and the limited budget of resources for computing and communication. To address these challenges, we developed DRL algorithms based on DDPG, TD3, and PPO. To capture the uncertain mobility of the UAVs, we proposed a multi-UAV random mobility model, which extends the traditional RD model and ST model designed for individual entities by incorporating a collision avoidance mechanism. The simulation results demonstrate that our approach significantly reduces task completion time compared to existing methods with optimal energy consumption. Additionally, they show a bounded improvement in performance when more UAVs are engaged in computing, highlighting that performance gains are not infinite with additional nodes. The results also demonstrate the negative impact of collision avoidance maneuvers on system performance, which increases uncertainty in UAV mobility. Our future work will extend to scenarios where tasks are generated at multiple UAVs.

References

[1] P. K. R. Maddikunta, S. Hakak, M. Alazab, S. Bhattacharya, T. R. Gadekallu, W. Z. Khan, Q.-V. Pham, Unmanned aerial vehicles in smart agriculture: Applications, requirements, and challenges, IEEE Sensors Journal 21 (16) (2021) 17608–17619.

[2] H. Kurunathan, H. Huang, K. Li, W. Ni, E. Hossain, Machine learning-aided operations and communications of unmanned aerial vehicles: A contemporary survey, IEEE Communications Surveys & Tutorials (2023).

[3] Y. Zeng, R. Zhang, T. J. Lim, Wireless communications with unmanned aerial vehicles: Opportunities and challenges, IEEE Communications magazine 54 (5) (2016) 36–42.

[4] M. Erdelj, E. Natalizio, Uav-assisted disaster management: Applications and open issues, in: 2016 international conference on computing, networking and communications (ICNC), IEEE, 2016, pp. 1–5.

[5] B. Liu, W. Zhang, W. Chen, H. Huang, S. Guo, Online computation offloading and traffic routing for uav swarms in edge-cloud computing, IEEE Transactions on Vehicular Technology 69 (8) (2020) 8777–8791.

[6] Z. Bai, Y. Lin, Y. Cao, W. Wang, Delay-aware cooperative task offloading for multi-uav enabled edge-cloud computing, IEEE Transactions on Mobile Computing (2022).

[7] M. Satyanarayanan, The emergence of edge computing, Computer 50 (1) (2017) 30–39.

[8] Q. Hu, Y. Cai, G. Yu, Z. Qin, M. Zhao, G. Y. Li, Joint offloading and trajectory design for uav-enabled mobile edge computing systems, IEEE Internet of Things Journal 6 (2) (2018) 1879–1892.

[9] Y. Miao, K. Hwang, D. Wu, Y. Hao, M. Chen, Drone swarm path planning for mobile edge computing in industrial internet of things, IEEE Transactions on Industrial Informatics (2022).

[10] K. Lu, J. Xie, Y. Wan, S. Fu, Toward uav-based airborne computing, IEEE Wireless Communications 26 (6) (2019) 172–179.

[11] B. Wang, J. Xie, K. Lu, Y. Wan, S. Fu, Learning and batch-processing based coded computation with mobility awareness for networked airborne computing, IEEE Transactions on Vehicular Technology (2022).

[12] H. Zhang, B. Wang, R. Wu, J. Xie, Y. Wan, S. Fu, K. Lu, Exploring networked airborne computing: A comprehensive approach with advanced simulator and hardware testbed, Unmanned Systems (2023).

[13] E. M. Royer, P. M. Melliar-Smith, L. E. Moser, An analysis of the optimum node density for ad hoc mobile networks, in: ICC 2001. IEEE International Conference on Communications. Conference Record (Cat. No. 01CH37240), Vol. 3, IEEE, 2001, pp. 857–861.

[14] Y. Wan, K. Namuduri, Y. Zhou, D. He, S. Fu, A smooth-turn mobility model for airborne networks, in: Proceedings of the first ACM MobiHoc workshop on Airborne Networks and Communications, 2012, pp. 25–30.

[15] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, arXiv preprint arXiv:1509.02971 (2015).

[16] S. Fujimoto, H. Hoof, D. Meger, Addressing function approximation error in actor-critic methods, in: International conference on machine learning, PMLR, 2018, pp. 1587–1596.

[17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, arXiv preprint arXiv:1707.06347 (2017).

[18] X. Zhang, J. Xie, Drl-based task offloading for networked uavs with random mobility and collision avoidance, in: 2024 20th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), IEEE, 2024, pp. 514–519.

[19] H. Nashaat, W. Hashem, R. Rizk, R. Attia, Drl-based distributed task offloading framework in edge-cloud environment, IEEE Access 12 (2024) 33580–33594. doi:10.1109/ACCESS.2024.3371993.

[20] G. Nieto, I. de la Iglesia, U. Lopez-Novoa, C. Perfecto, Deep reinforcement learning techniques for dynamic task offloading in the 5g edge-cloud continuum, Journal of Cloud Computing 13 (1) (2024) 94.

[21] D. Sha, R. Zhao, Drl-based task offloading and resource allocation in multi-uav-mec network with sdn, in: 2021 IEEE/CIC International Conference on Communications in China (ICCC), IEEE, 2021, pp. 595–600.

[22] R. Zhu, M. Huang, K. Sun, Y. Hou, Y. Wan, H. He, Deep reinforcement learning based task offloading for uav-assisted edge computing, in: 2023 IEEE International Conference on Unmanned Systems (ICUS), IEEE, 2023, pp. 1104–1111.

[23] A. Khanna, S. Kaur, Internet of things (iot), applications and challenges: a comprehensive review, Wireless Personal Communications 114 (2020) 1687–1762.

[24] A. Alwarafy, K. A. Al-Thelaya, M. Abdallah, J. Schneider, M. Hamdi, A survey on security and privacy issues in edge-computing-assisted internet of things, IEEE Internet of Things Journal 8 (6) (2020) 4004–4022.

[25] Z. Hu, R. Zhong, C. Fang, Y. Liu, Caching-at-stars: The next generation edge caching, IEEE Transactions on Wireless Communications (2024).

[26] Y. Hou, C. Wang, M. Zhu, X. Xu, X. Tao, X. Wu, Joint allocation of wireless resource and computing capability in mec-enabled vehicular network, China Communications 18 (6) (2021) 64–76.

[27] L. Nadeem, M. A. Azam, Y. Amin, M. A. Al-Ghamdi, K. K. Chai, M. F. N. Khan, M. A. Khan, Integration of d2d, network slicing, and mec in 5g cellular networks: Survey and challenges, IEEE Access 9 (2021) 37590–37612.

[28] Y. Xu, T. Zhang, Y. Liu, D. Yang, L. Xiao, M. Tao, Uav-assisted mec networks with aerial and ground cooperation, IEEE Transactions on Wireless Communications 20 (12) (2021) 7712–7727.

[29] Z. Xiao, Y. Chen, H. Jiang, Z. Hu, J. C. Lui, G. Min, S. Dustdar, Resource management in uav-assisted mec: state-of-the-art and open challenges, Wireless Networks 28 (7) (2022) 3305–3322.

[30] B. Wang, J. Xie, K. Lu, Y. Wan, S. Fu, Coding for heterogeneous uav-based networked airborne computing, in: 2019 IEEE Globecom Workshops (GC Wkshps), IEEE, 2019, pp. 1–6.

[31] B. Wang, J. Xie, K. Lu, Y. Wan, S. Fu, On batch-processing based coded computing for heterogeneous distributed computing systems, IEEE Transactions on Network Science and Engineering 8 (3) (2021) 2438–2454.

[32] B. Wang, J. Xie, S. Li, Y. Wan, Y. Gu, S. Fu, K. Lu, Computing in the air: An open airborne computing platform, IET Communications 14 (15) (2020) 2410–2419.

[33] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, et al., Ros: an open-source robot operating system, in: ICRA workshop on open source software, Vol. 3, Kobe, Japan, 2009, p. 5.

[34] N. Koenig, A. Howard, Design and use paradigms for gazebo, an open-source multi-robot simulator, in: 2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566), Vol. 3, Ieee, 2004, pp. 2149–2154.

[35] K. Cheng, Y. Teng, W. Sun, A. Liu, X. Wang, Energy-efficient joint offloading and wireless resource allocation strategy in multi-mec server systems, in: 2018 IEEE international conference on communications (ICC), IEEE, 2018, pp. 1–6.

[36] F. Pervez, A. Sultana, C. Yang, L. Zhao, Energy and latency efficient joint communication and computation optimization in a multi-uav assisted mec network, IEEE Transactions on Wireless Communications (2023).

[37] A. Goldsmith, Wireless communications, Cambridge university press, 2005.

[38] D. H. Choi, S. H. Kim, D. K. Sung, Energy-efficient maneuvering and communication of a single uav-based relay, IEEE Transactions on Aerospace and Electronic Systems 50 (3) (2014) 2320–2327.

[39] W. Lu, P. Si, Y. Gao, H. Han, Z. Liu, Y. Wu, Y. Gong, Trajectory and resource optimization in ofdm-based uav-powered iot network, IEEE Transactions on Green Communications and Networking 5 (3) (2021) 1259–1270.

[40] E. Shtaiwi, A. Abdelhadi, H. Li, Z. Han, H. V. Poor, Orthogonal time frequency space for integrated sensing and communication: A survey, arXiv preprint arXiv:2402.09637 (2024).

[41] N. T. Hoa, N. C. Luong, D. Van Le, D. Niyato, et al., Deep reinforcement learning for multi-hop offloading in uav-assisted edge computing, IEEE Transactions on Vehicular Technology (2023).

[42] M. Andrychowicz, A. Raichuk, P. Stańczyk, M. Orsini, S. Girgin, R. Marinier, L. Hussenot, M. Geist, O. Pietquin, M. Michalski, et al., What matters in on-policy reinforcement learning? a large-scale empirical study, arXiv preprint arXiv:2006.05990 (2020).

[43] J. Schulman, P. Moritz, S. Levine, M. Jordan, P. Abbeel, High-dimensional continuous control using generalized advantage estimation, arXiv preprint arXiv:1506.02438 (2015).

[44] H. Zhang, J. Xie, Y. Wan, S. Fu, K. Lu, Advancing networked airborne computing with mmwave for air-to-air communications, in: International Symposium on Intelligent Computing and Networking, Springer, 2024, pp. 34–50.

[45] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors, nature 323 (6088) (1986) 533–536.

[46] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 1026–1034.