Graphical Abstract

DRL-based Task Offloading for Networked UAVs with Random Mobility and Collision Avoidance[⋆]

Xixin Zhang, Dongge Jia, Junfei Xie

Highlights

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

Xixin Zhang, Dongge Jia, Junfei Xie

- Research highlight 1
- Research highlight 2

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

Xixin Zhang[a,b,1], Dongge Jia[b,c,2], Junfei Xie[b,3,*]

[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 is often criticized for hindering their ability to execute complex tasks in real-time. To address this challenge, Networked Airborne Computing (NAC) has emerged, which leverages the collective computing power of multiple UAVs to enable efficient handling of 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 resources while moving randomly within an area. Without prior knowledge of the system models, we aim to optimize task allocation among UAVs with uncertain mobility. To achieve this, we propose a Deep Reinforcement Learning algorithm based on the Twin Delayed Deep Deterministic Policy Gradient (TD3). Simulation results demonstrate that our

⋆This document is the results of the research project funded by the National Science Foundation.

*Corresponding author

Email addresses: xiz166@ucsd.edu (Xixin Zhang), doj14@pitt.edu (Dongge Jia), jxie4@sdsu.edu (Junfei Xie)

[1]This is the first author footnote.

[2]Another author footnote, this is a very long footnote and it should be a long footnote. But this footnote is not yet sufficiently long enough to make two lines of footnote text.

[3]Yet another author footnote.

approach significantly speeds up task execution compared to existing methods.

Keywords: Computation Offloading, Deep Reinforcement Learning, 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]. 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*. [3] 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*. [4] 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 viewed as relays that bring edge servers or remote clouds closer, rather than computing nodes. They get sufficient computing resources at the cost of a high data transmission delay, which may not be acceptable for time-sensitive UAV applications not to mention real-time tasks. Moreover, mobile edge servers require a reliable local network infrastructure, which is difficult to deploy and scale, especially in underdeveloped or post-disaster areas [5].

With technological advancements, the emergence of small, lightweight yet powerful micro-computers has significantly accelerated the onboard computing capacity of UAVs. This has spurred researchers to explore UAVs' potential in acting as edge servers. In [6], 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 offloading ratios, user scheduling, and UAV trajectory in a UAV-aided mobile edge computing

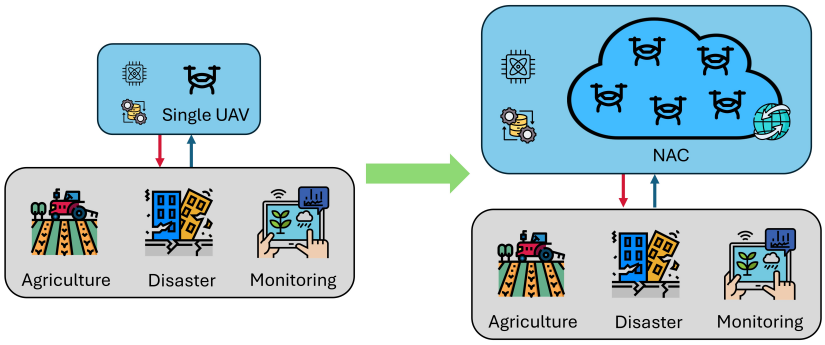

Figure 1: Networked Airborne Computing (NAC)

system. Miao *et al.* [7] 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 studies treat them as separate servers.

To harness the full computational potential of multi-UAV systems, a new paradigm called Networked Airborne Computing (NAC) is proposed, where multiple aerial vehicles share resources among each other[8] as in Fig.**??**. The fast deployment, infrastructure-free, and low-cost characteristics make the UAV-based NAC a promising technique. Nevertheless, research in NAC 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 [9]. In [10], we introduced a coded distributed computing scheme based on deep reinforcement learning (DRL) for optimally partitioning and allocating tasks to multiple networked UAVs. This scheme addresses two typical NAC scenarios. The first scenario involves uncontrollable UAV mobility, which can happen when they are operated by different owners. In the second scenario, UAVs are controlled to assist in task computation. Simulation results demonstrate the effectiveness of the proposed scheme. However, in the first scenario, we assumed UAVs maintain a consistent movement pattern throughout the execution of a particular task and did not account for motion interference between UAVs due to collision avoidance. Moreover, the simple matrix multiplication tasks were considered.

Orthogonal Frequency Division Multiple Access (OFDMA) is a sophis-

ticated wireless communication technology that divides the available bandwidth into multiple orthogonal subcarriers, dynamically allocating them to users based on their channel conditions and service requirements [11]. This dynamic resource allocation is particularly advantageous in networked Unmanned Aerial Vehicle (UAV) systems, where the mobility and dynamic topology of UAVs demand robust and flexible communication solutions. By leveraging OFDMA, networked UAV systems can achieve high spectral efficiency, reduced latency, and reliable performance in diverse environments, supporting applications such as real-time surveillance, package delivery, and disaster response [12]. Energy efficiency is another critical concern in UAV systems due to the limited onboard battery capacity, which constrains mission duration and operational effectiveness. Task offloading, while reducing onboard computational load, introduces additional energy costs for data transmission and reception. This makes it essential to adopt optimization strategies that minimize total energy consumption—an especially pressing challenge for energy-limited systems such as the NAC system. In addressing these challenges, [13] proposes an optimization framework that simultaneously allocates subcarriers and adjusts power levels to balance spectral efficiency and energy consumption. This approach is particularly effective in dynamic multi-UAV communication networks, where varying channel conditions and interference necessitate adaptive and efficient resource management.

In this paper, we investigate a more common yet challenging NAC scenario where all UAVs, including both offloaders and offloadees, move randomly during task execution while actively avoiding collisions. None of the UAVs have prior knowledge of the environment or system models, and their movement patterns or future trajectories are not shared among each other. Additionally, we generalize computation tasks as any functions or operations that can be partitioned into arbitrary subtasks for parallel computation. To model UAV movement, we extend the traditional Random Direction model [14], originally designed for individual entities, to capture collision avoidance interactions among multiple UAVs. Furthermore, we formulate a nonlinear optimization to optimize task allocation and develop a DRL algorithm based on the Twin Delayed Deep Deterministic Policy Gradient (TD3)[15] to solve it. We evaluate the performance of the proposed method through extensive comparative simulation studies, which demonstrate its promising performance.

In the rest of this paper, Sec. 3 details the system models and formulates

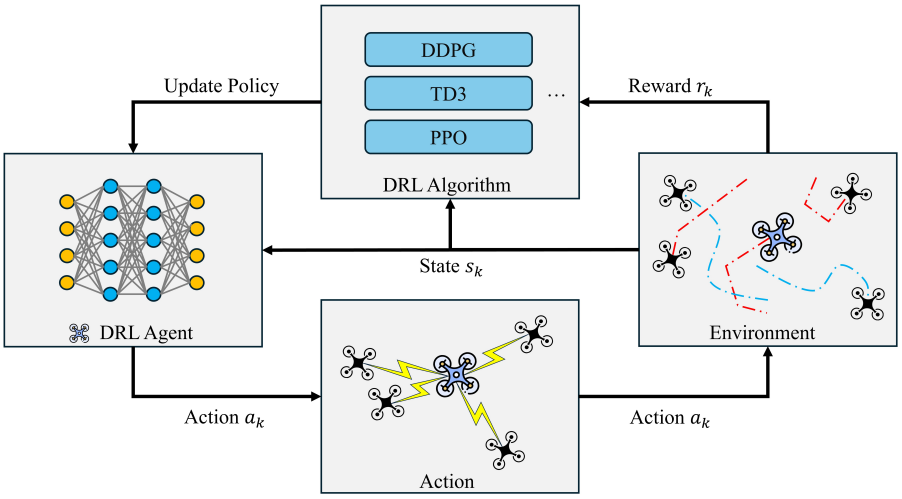

Figure 2: Caption

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

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 can be 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 computing tasks. For a general computing task k, its input data size is S_k (bits), and its required computation intensity is ξ_k

(cycles/bit)[16]. 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 computing can be 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 [17].

3.2. Communication Model

At time t, let the distance between UAV i and UAV j be denoted as $d_{ij}(t)$. The UAV-to-UAV links are typically 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 [18] 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 specific channel between UAV i and j, d_r is constant reference distance (meter), G is the unitless constant equal to the path gain of the 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), ν_{ij} is the threshold to suppress the data rates that are too low, which can be regarded as the constraint on QoS.

Unlike our previous work [19] which assumed the channel bandwidth and transmitted power as constants, here we treat both B_{ij} and ψ_{ij} as variables to be determined for the 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 is also determined follows Eq.3. Then the overall transmission time is as follows

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

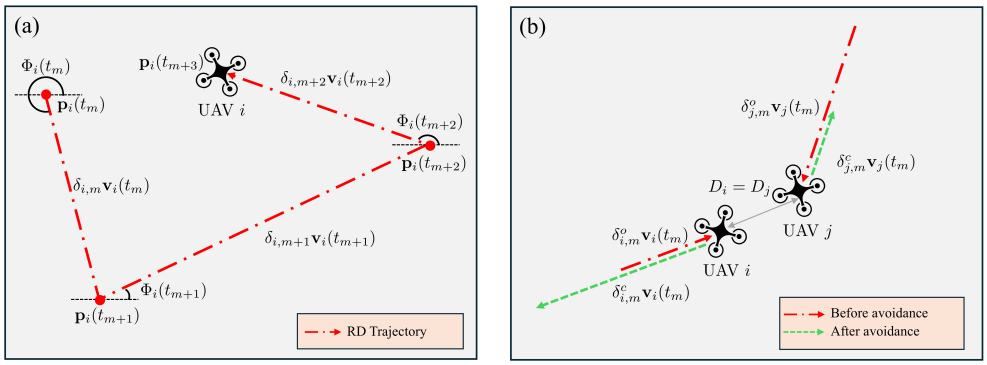

Figure 3: Caption

The reception power of UAV j is denoted as $\tilde{\psi}_j$ (mW) for completeness as in [20], then the energy (Joule) consumed for offloading task by the system is a combination of transmission energy and reception energy as

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

3.3. Mobility Model

We assume the UAVs fly at the same altitude. Therefore, the position of each UAV i at time t can be depicted as $\mathbf{p}_i(t) = (x_i(t), y_i(t)) \in \mathbb{R}^2$ with constraints $0 \leq x_i(t) \leq W$, $0 \leq y_i(t) \leq W$, such that the position is bounded within an area of $E \times E(m^2)$. Given initial position $\mathbf{p}_i(0) = (x_i(0), y_i(0))$, we adapt the Random Direction (RD) model [14] and Smooth Turn (ST) model[21] to model its movement, which have been widely used for describing UAVs, particularly multirotor drones [22]. At time t, let the velocity of UAV i be $\mathbf{v}_i(t) = (v_{i,x}(t), v_{i,y}(t)) \in \mathbb{R}^2$ and the heading direction be $\Phi_i(t) \in [0, 2\pi]$, where $v_{i,x}(t)$, $v_{i,y}(t)$ are component of velocity in x and y direction.

3.3.1. Random Direction

If UAV i moves in the mode of Random Direction (RD), it randomly picks a velocity and moves along a straight line at this velocity for a duration randomly picked as well until another set of velocity and duration is selected and repeats the process (Fig 3a).

Suppose the start time of m-th duration is denoted as $t_m = \sum_{l=0}^{m-1} \delta_{i,l}$ which is also the m-th instance when UAV i changes its velocity, where each $\delta_{i,l}$ is randomly sampled from 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 $v_{max} \in \mathbb{R}$, *i.e.* $0 < |\mathbf{v}_i(t_m)| < v_{max}$ and the heading direction $\Phi_i(t_m)$ in radian uniformly across 2π, leading to the new velocity $\mathbf{v}_i(t_m)$ for the m-th duration such that

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

As velocity remains unchanged within each duration, the UAV i's position at time t, where $t_m \leq t < t_{m+1}$, can be represented as follows

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

The traditional RD model [14] is originally designed to describe the mobility of independent single entities, which ignores the spatiotemporal correlations of trajectory across entities. However, in multi-UAV systems, the mobility of UAVs can change to avoid collisions. This necessitates the incorporation of collision avoidance mechanisms into the RD model. Here, we assume that each UAV i will turn around and move in the opposite direction without changing speed until the current duration is completed when its distance to any other UAV j falls below a threshold D_i. Note that if both UAVs have the same threshold $D_i = D_j$, UAV j will also reserve its direction (Fig. 3b). 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 $\tilde{\delta}_{i,m}^c = (t - t_m) - \tilde{\delta}_{i,m}^o$.

3.3.2. Smooth Turn

In the smooth turn (ST) mobility model, UAV i randomly picks a turning center located at a point along the line perpendicular to its current heading direction and moves at a constant forward speed following the circular

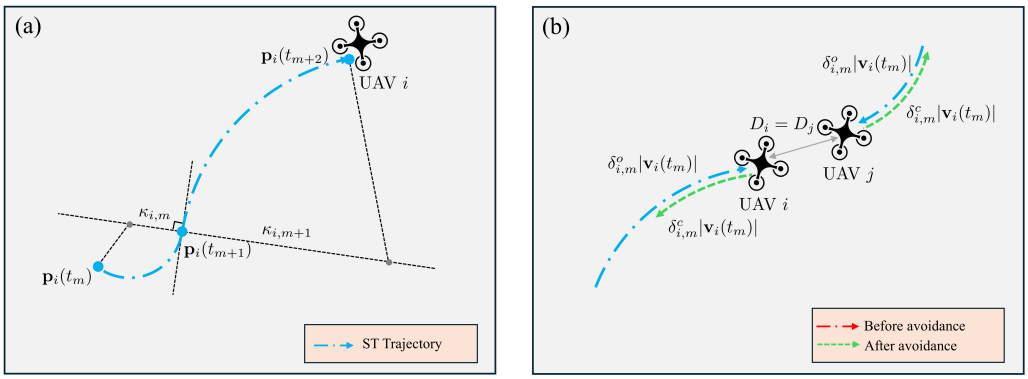

Figure 4: Caption

trajectory around the turning center for a duration randomly selected until another turning center picked and repeat the process.

Given the velocity magnitude $|\mathbf{v}_i(t)|$ constant for all 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 exponential distribution with parameter λ_{st} and t_0 is set to be 0. At time t_m, UAV i samples a variable $\kappa_{i,m} \in \mathbb{R}$ from a Gaussian distribution as $\frac{1}{\kappa_{i,m}} \sim \mathcal{N}(0, \sigma^2)$. We define $|\kappa_{i,m}|$ as the radius of the circular trajectory, and the center of the circle is thereafter uniquely determined along the line perpendicular to the heading angle $\Phi_i(t_m)$, assuming counterclockwise rotation when $\kappa_{i,m} < 0$ and clockwise rotation when $\kappa_{i,m} < 0$. As $\kappa_{i,m} \in \mathbb{R}$ remains unchanged for the m-th duration, the heading angle $\Phi_i(t)$ of UAV i at time t, where $t_m \leq t < t_{m+1}$, gives as

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

and the velocity follows as

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

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

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

The same issue appears in the original ST mobility model as the RD mobility model that it fails to model the interactions between nodes since it did not include collision avoidance mechanisms either. Here, we propose a collision avoidance strategy similar to the one introduced for the RD model in the previous section. In this approach, UAVs retrace their prior circular trajectory in the reverse direction, restoring the safe inter-UAV spacing and spatial relationship (Fig. 4b). The mobility in Eq. 10 of UAV i will become

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

where $\delta_{i,l}^o$ represents the duration that UAC i moves along the designated arc trajectory in its original direction the (determined by the sign of $\kappa_{i,l}$), and $\delta_{i,l}^c$ denotes the duration that UAV i moves in the opposite direction along the circular trajectory due to the collision avoidance maneuver within the l-th duration, such that $\delta_{i,l}^o + \delta_{i,m}^c = \delta_{i,m}$. Let $\tilde{\delta}_{i,m}$ be the duration in m-th instance until t, the splits of duration for the original direction and reversed direction are $\tilde{\delta}_{i,m}^o$, $\tilde{\delta}_{i,m}^c$ respectively so 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. Suppose a sequence of computing 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 of size $\ell_k = \frac{S_k}{L_k}$, which can be computed in parallel. Consider a UAV network utilizing OFDMA, where the total available communication bandwidth is represented by B. This bandwidth is evenly divided into W orthogonal subcarriers, with each subcarrier having a bandwidth of $b = \frac{B}{W}$. The master UAV, tasked with managing communication and data transmission, operates within a total power budget denoted by $\Psi \in \mathbb{R}$.

4.1. Joint Optimization

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 $0 \leq c_j^k \leq L_k$. Therefore, $\sum_{j=0}^{N} c_j^k = L_k$. Of note, when there is no workload offloaded to UAV j, then $c_j^k = 0$. With respect to the network resources, let the number of subcarriers allocated to the channel between master and UAV $j \in \mathcal{N} \backslash \{0\}$ be non-negative integer $0 \leq w_j^k \leq W$, the channel bandwidth for will be $B_{0j}^k = w_j^k b$. On the other hand, the corresponding transmission power master UAV allocate to B_{0j}^k will be $\psi_{0j}^k \in \mathbb{R}$ such that$\sum_{j=1}^{N} \psi_{0j}^k = \Psi$.

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 the result size is small and the latency of sending it back is negligible, as often assumed in existing works [23]. Therefore, $T_{k,i}$ 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 $T_{k,i}^{comp} = \frac{\xi_k c_i^k \ell_k}{f_i}$ and $T_{k,j}^{trans}$ satisfies

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

Concerning the energy consumption E_k of offloading task k, we focus on the energy consumption of the entire NAC system, which consists of energy consumed for computation and transmission among all the computing nodes including both the master and workers, that is

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

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 defined to minimize the weighted sum of the system's total energy consumption and overall task completion latency, which can be mathematically formulated as follows

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

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

$$c_j^k \in \mathbb{Z}, \ 0 \leq c_j^k \leq L_k, \tag{15c}$$

$$w_j^k \in \mathbb{Z}, \ 0 \leq w_j^k \leq W, \tag{15d}$$

$$\psi_j^k \in \mathbb{R}, \ 0 \leq \psi_j^k \leq \Psi, \tag{15e}$$

$$\forall j \in \mathcal{N}, k \in \mathcal{K}. \tag{15f}$$

where $\eta \in [0, 1]$ is the weight parameter of time latency and energy consumption that control the trade-off between energy consumption and time latency.

4.2. Markov Decision Process

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.

The complexity of the optimization problem lies in the nonlinear and unknown relationship between the objective components T_k and E_k and the decision variables c_j^k, w_j^k, and ψ_j^k. Furthermore, the stochastic nature of UAV mobility and the lack of explicit system models make traditional optimization approaches unsuitable for solving this problem effectively. To address these

challenges, the optimization problem defined in Eq. 15 is reformulated as a Markov Decision Process (MDP), represented by the tuple $(\mathcal{S}, \mathcal{A}, r, P)$. This conversion provides a framework for modeling the sequential decision-making process in a dynamic environment. By employing reinforcement learning 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, enabling an effective solution for minimizing the weighted objective.

4.2.1. State

We assume 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 discrete time step 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 serve as 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 deci-

sions 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.

4.2.2. Action

Every time a task k arrives, the master UAV partitions it into sub-tasks and offloads them among all UAVs while allocating communication resources such as subcarriers and transmission power. The fraction of task k assigned to UAV $j \in \mathcal{N}$ is denoted by $\zeta_j^{k,d} \geq 0$, where the superscript d indicates data allocation. Similarly, the fraction of subcarriers allocated to UAV j for task k is represented as $\zeta_j^{k,s} \geq 0$, and the fraction of transmission power allocated to UAV j is denoted by $\zeta_j^{k,p} \geq 0$, where the superscripts s and p represent subcarrier and 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 subcarrier allocations expressed as the number of subcarriers are computed by rounding the fractional values. The workload offloaded to UAV j is given as:

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

where S_k is the size of task k. The number of subcarriers allocated to UAV j is computed as

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

where W is the total number of available subcarriers. The transmission power allocated to UAV j is retained as a fractional value:

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

where Ψ is the total transmission power budget. The action taken by the master UAV at the start time t^k is 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 formed the action space $\mathcal{A}$.

4.2.3. 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 time latency and energy efficiency for each task k. The reward function is expressed 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 weight parameter that controls the trade-off between energy consumption and time latency. A smaller η emphasizes minimizing task completion time, while a larger η prioritizes reducing energy consumption.

In this formulation, T_k represents the actual task completion time achieved through offloading, and $\tilde{T}_k = \frac{\xi_k S_k}{f_0}$ 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.

4.2.4. Transition

As discussed in Sec. 3, all UAVs move randomly according to predefined mobility models 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 random mobility model. 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 generally unknown and depends on the selected mobility model. For the random direction (RD) model, $v_{\max}$ controls the maximum velocity of the UAVs, while λ_{rd} governs the frequency of direction changes. In the smooth turn (ST) model, σ determines the radius of the circular trajectory, and λ_{st} specifies the frequency of direction changes, thereby controlling 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 decision-making in task allocation and resource management across dynamic, networked UAV systems.

5. Deep Reinforcement Learning Solution

To address the task offloading and resource allocation challenges in dynamic UAV networks, we adapt three deep reinforcement learning (DRL) algorithms: Deep Deterministic Policy Gradient (DDPG) [24], Twin Delayed Deep Deterministic Policy Gradient (TD3) [15], and Proximal Policy Optimization (PPO) [25]. Each algorithm is tailored to handle the continuous action spaces and simplex constraints inherent in UAV task offloading while maintaining the integrity of their unique design principles.

5.1. Actor-Critic DRL Methods

Actor-critic architectures represent a sophisticated class of deep reinforcement learning (DRL) algorithms optimized for continuous action spaces, making them exceptionally well-suited for dynamic resource allocation challenges. There are two essential components built with neural networks: an actor network that learns optimal policy mappings from states to actions, and a critic network that performs value assessment of the chosen actions. Building upon this foundation, our research extends two prominent actor-critic variants—DDPG and TD3, to address the complex multi-objective optimization of NAC.

5.1.1. Actor

The actor in both DDPG and TD3 is is implemented as a neural network parameterized by ϕ, denoted as $\pi_\phi : \mathcal{S} \to \mathcal{A}$, where $\mathcal{S}$ represents the state space and $\mathcal{A}$ the continuous action space. 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, where the input layer encodes the state features, intermediate layers apply non-linear activations (e.g., ReLU) extract latent features that capture the movement patterns of individual UAVs and their interactions, such as collision avoidance. The output layer produces 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 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 β 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.

5.1.2. Critic

The critic in actor-critic frameworks serves the crucial role of evaluating the expected return for a given state-action pair, guiding the actor to optimize its policy. In our resource allocation problem, the critic evaluates the quality of data, 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:

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

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 as 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 $\zeta_j^{k,\chi}$ 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 resource allocation problem.

5.1.3. Training

The training procedure for the actor-critic framework 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, 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 9-11) and actor (Lines 12-13). Particularly, the parameters θ_i, $i \in \{1, 2\}$, of each primary critic is updated by minimizing the following loss that an approximation of the expectation in Eq. 22 with

Algorithm 1 Training for task offloading (DDPG, TD3)

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 $\theta'_1 \leftarrow \theta_1, \theta'_2 \leftarrow \theta_2$
Initialize the actor network π_ϕ and the target actor $\pi_{\phi'}$ by randomly assigning values to ϕ and setting $\phi' \leftarrow \phi$
Initialize the replay buffer by $\mathcal{D} \leftarrow \emptyset$
for $h = 1$ to U **do**
    Select action $a \sim (1-\beta)\pi_\phi(s) + \beta\epsilon$, with exploration noise $\epsilon \sim \mathrm{Dir}(\alpha_{explore})$, observe reward r and new state s'
    Store the transition tuple (s, a, r, s') in $\mathcal{D}$
    if $h > H$ **then**
        Sample a batch of $|\mathcal{B}|$ transitions (s, a, r, s') from buffer $\mathcal{D}$
        Update the critics by $\theta_i \leftarrow \arg\min_{\theta_i} \frac{1}{|\mathcal{B}|}\sum(y - Q_{\theta_i}(s,a))^2$, $i = 1, 2$
        if $h \mod u = 0$ **then**
            Update the actor by $\phi \leftarrow \arg\max_\phi \frac{1}{|\mathcal{B}|}\sum Q_{\theta_1}(s, \pi_\phi(s))$
            Update the target network by $\theta'_i \leftarrow \theta_i + (1-\tau)(\theta'_i - \theta_i), i = 1, 2$ and $\phi' \leftarrow \phi + (1-\tau)(\phi' - \phi)$
        end if
    end if
end for

average over the sampled batch

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

where y is derived from Eq.23 or Eq. 24

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.

6. Experiments

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

6.1. Environment Setting

We evaluate our method in simulated scenarios with one master UAV and $N = 3$, 6, 9, or 12 offloadee UAVs respectively that moves universally following the Multi-RD model or Multi-ST model. For the multi-UAV RD mobility model, we set its parameters as $v_{max} = 20$ m/s, $\lambda_{rd} = \frac{1}{15}$, and $D_j = 5$, $\forall j \in \mathcal{N}$. 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 sizes, $E = 200$m and $W = 300$m. The initial idle computing power $f_j(0)$ of each UAV j is randomly configured by selecting values from the range of $[1, 1.6]$ Ghz, and it vaires at the start time of task uniformly between $0.9f_j(0)$ and $1.1f_j(0)$. The task list consists of $K = 25$ tasks that can be locally completed in $\tilde{T}_k \in [20, 60]$ seconds by the master UAV. All tasks can be divided into $L_k = 1000$ atomic tasks. The computation intensity is set to be the same $\xi_k = 10^6$ cycles/kB for all tasks, and the size of each task is determined by $S_k = \frac{f_0(t^k)\tilde{T}_k}{\xi_k}$. As a case of a networked UAV swarm utilizing Mmwave Air-to-Air communication for data transmission [26], we set the total available bandwidth $B = 2$ GH and divided it into subcarriers each has a bandwidth of 15kHz, 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 requirement of the Quality of service for the transmission data rate is set to $\nu_{min} = 30$ kb/s. For simplicity, the power consumed by receiver j is set to a constant $\tilde{\psi}_j = 100$ mW.

6.2. Training Performance

We employ a 3-layer Multilayer Perceptrons (MLP)[27] architecture for both the actor and critic networks. The actor network takes observations as input, whose dimension is based on 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 the concatenation of observations and actions as input and outputs a scalar representing the state action value. The width of the hidden layer in both networks is set to twice the dimension of the input. All parameters are initialized using Kaiming initialization[28].

The learning rates for the actor and critic networks are set to 0.0001 and 0.0002, respectively. The threshold H is set to 1000 and the batch size is $|\mathcal{B}| = 512$. The gradient norm is clipped between 0 and 0.2. The exploration and policy noises are sampled from a Dirichlet distribution with parameters $\alpha_{explore} = \alpha_{policy} = 1$. The corresponding noise weight is $\beta_{explore} = 0.1$ and $\beta_{policy} = 0.01$. The discount factor γ is 0.99 and the soft update weight τ for target networks is 0.005. Specially, the actor network in TD3 is updated every $u = 25$ iterations while in DDPG $u = 1$.

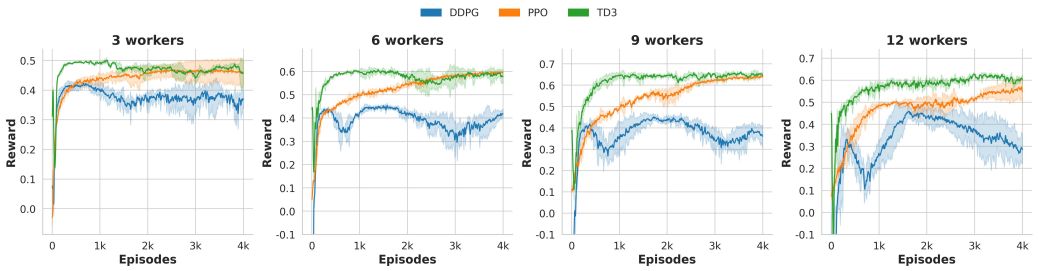

Figure 5: Learning Curve of RD ($200 \times 200m^2$)

In each scenario, we train the agent for 4000 episodes, i.e., $U = 4000K = 10^5$ iterations, with three different random seeds. 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

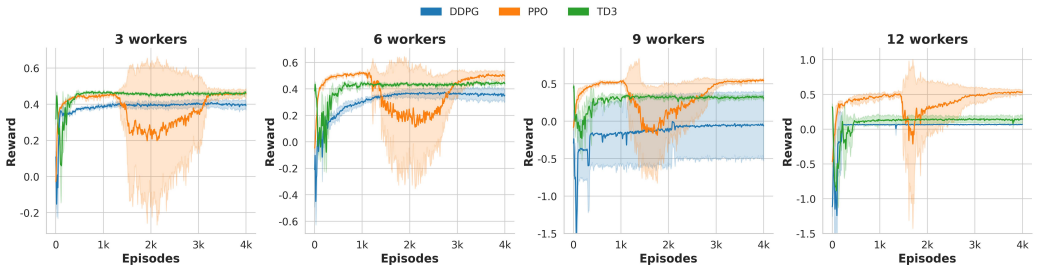

Figure 6: Learning Curve of ST ($200 \times 200m^2$)

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.

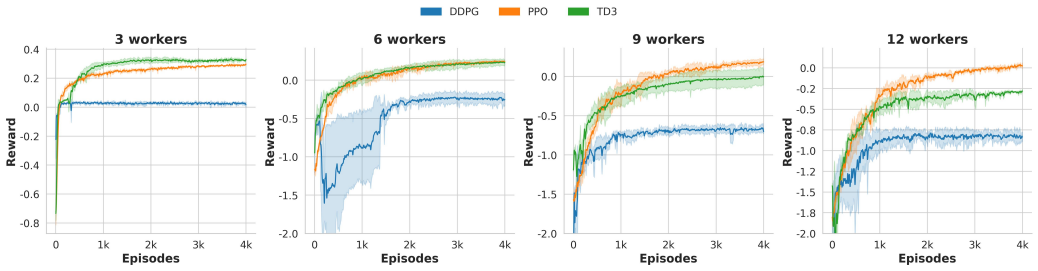

Figure 7: Learning Curve of RD ($300 \times 300m^2$)

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

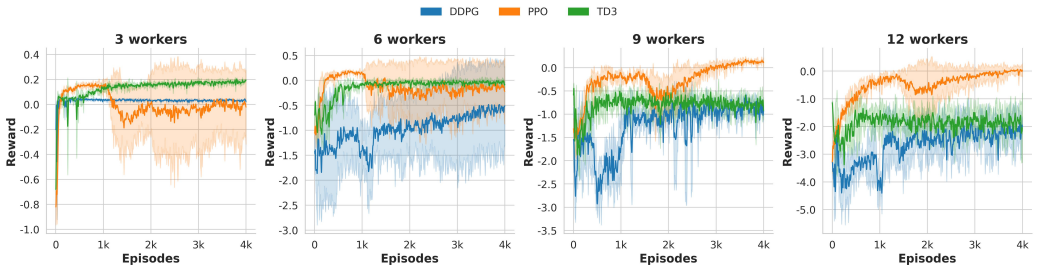

Figure 8: Learning Curve of ST ($300 \times 300m^2$)

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 computing tasks and allocating network resources.

6.3. Comparison Studies

To evaluate the performance of the proposed method, we compare it with five benchmarks, including (1) **Equal (all)** that equally divides and distributes tasks to all UAVs; (2) **Equal (close)** that equally partitions and distributes tasks to UAVs that are close enough with distance to the master satisfying $d_i < 100$m; (3) **Naive Prediction** that selects offloadees based on predicted future trajectories and assigns sub-tasks 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_i of predicted UAV positions that remain within 200m of the master ($d_i < 200$m). UAVs with $q_i \geq 40\%$ are selected as offloadees; (4) **Reliable** that allocates tasks based on a reliability score defined as reliability $= q_i \psi_{ij} f_i$. It picks the top $\lceil \frac{N+1}{2} \rceil$

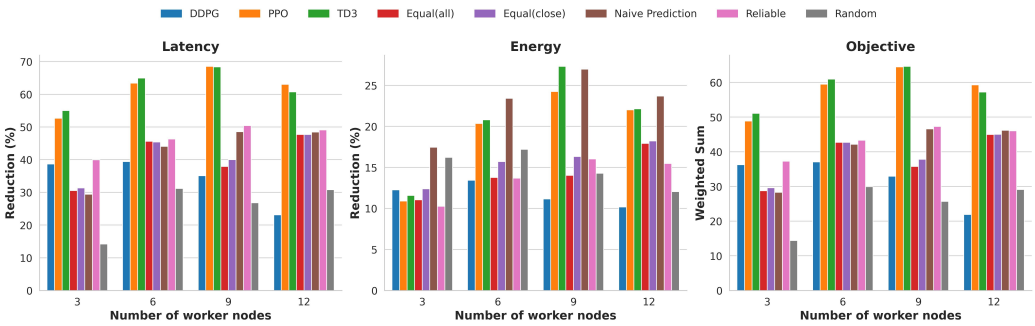

Figure 9: Performance Comparison ($200 \times 200m^2$)

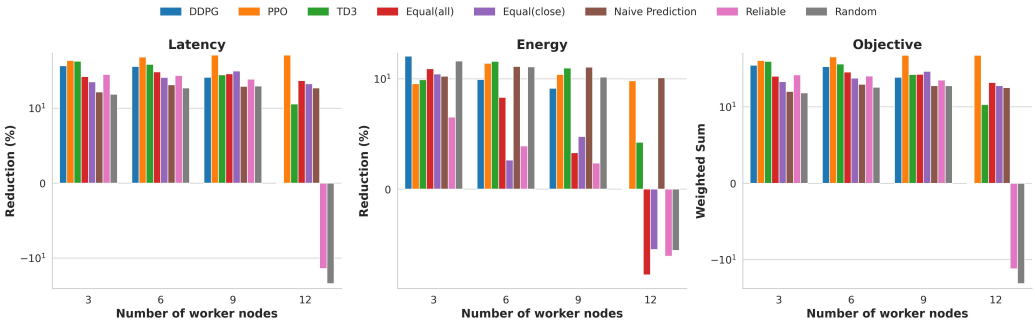

Figure 10: Performance Comparison ($200 \times 200m^2$)

UAVs with the highest reliability scores and assigns tasks to these UAVs proportionally to their reliability scores; (5) **Random** that randomly sample actions from the action space.

One phenomenon should be mentioned that, given the restricted work zone, the increase in the number of workers can not guarantee the policy learned converge to a better performance. The performance degradation happenes across the size of the working zone and the mobility taken by the swarm, as showed when $N > 9$ in Fig. 5, may be due to increased collision avoidance maneuvers, which make UAVs' mobility more uncertain and harder to learn and predict. Intuitively, the master agent tends to share workload with UAVs that are more likely to remain nearby throughout task execution, as indicated by a higher reliability score z_i^k. Therefore, when UAV mobility becomes more uncertain, fewer workers are selected as offloadees due to reduced reliability. This is confirmed by the results shown in Fig. **??**,

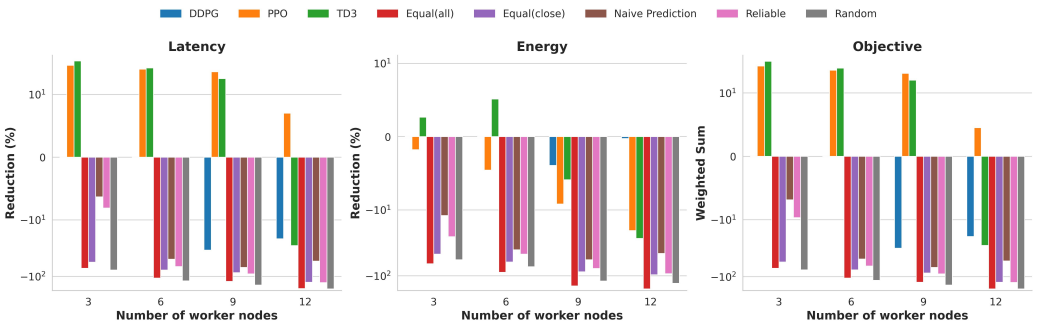

Figure 11: Performance Comparison ($300 \times 300m^2$)

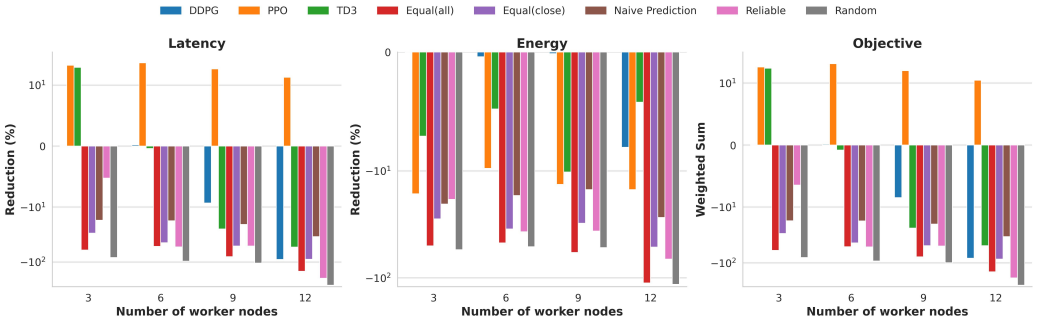

Figure 12: Performance Comparison ($300 \times 300m^2$)

where performance improves when collision avoidance mechanisms are not in place. It also indicates that the task completion time cannot be infinitely reduced by continuously adding more UAVs to the region.

The results in Fig. **??** and Fig. **??** demonstrate the promising performance of our method, which achieves the highest latency reduction across all scenarios. When the area is $300 \times 300m^2$, the **Naive Prediction** ranks second in scenarios with 3, 6, and 9 workers, while the **Equal (close)** ranks second in scenarios with 12 workers. When the area expands to $400 \times 400m^2$ (Fig. **??**), the **Equal (all)** and the **Random** provide no benefit in reducing latency; instead, they significantly delay task completion. Comparing Fig. **??** and Fig. **??**, we can observe that as the area increases for a fixed N, the performance of all methods degrades. This is due to the sparser airspace, which causes UAVs to be farther apart from each other, thereby increasing transmission latency and task completion time.

7. Conclusion

In this paper, we addressed the optimal task offloading problem 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 and uncertainties in UAV mobility. To address these challenges, we developed a DRL algorithm based on TD3. To capture the uncertain mobility of the UAVs, we proposed a multi-UAV random mobility model, which extends the traditional RD 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. 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 UAVs have more complex movement patterns and where tasks are generated at multiple UAVs.

Appendix A. Example Appendix Section

Appendix text.

Example citation, See [4].

References

[1] 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).

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

[3] 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.

[4] 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).

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

[6] 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.

[7] 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).

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

[9] 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).

[10] 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).

[11] 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).

[12] 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.

[13] X. Guan, Y. Huang, Q. Shi, Joint subcarrier and power allocation for multi-uav systems, China Communications 16 (1) (2019) 47–56.

[14] 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.

[15] S. Fujimoto, H. van Hoof, D. Meger, Addressing Function Approximation Error in Actor-Critic Methods (Oct. 2018). arXiv:1802.09477.

[16] 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.

[17] 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).

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

[19] 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.

[20] 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.

[21] 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.

[22] D. S. Lakew, U. Sa'ad, N.-N. Dao, W. Na, S. Cho, Routing in flying ad hoc networks: A comprehensive survey, IEEE Communications Surveys & Tutorials 22 (2) (2020) 1071–1120.

[23] 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).

[24] 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:1509.02971.
URL `http://arxiv.org/abs/1509.02971`

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

[26] 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.

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

[28] 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.