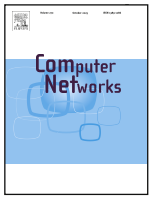
Computer Networks 277 (2026) 112064

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



Energy-aware holistic optimization in UAV-assisted fog computing: Attitude, trajectory, and task assignment ⋆

Shuaijun Liu [a,b], Jinqiu Du [c], Yaxin Zheng [d], Jiaying Yin [e], Yuhui Deng [a], Jingjin Wu [a,*]

[a] Guangdong Provincial Key Laboratory of IRADS, Beijing Normal-Hong Kong Baptist University, Guangdong, China
[b] Department of Computer Science, Boston University, Boston, MA, USA
[c] Department of Biostatistics, University of Washington, Seattle, WA, USA
[d] Department of Statistics, Columbia University, New York, NY, USA
[e] Institute of Precision Medicine, The First Affiliated Hospital, Sun Yat-Sen University, Guangdong, China

ARTICLE INFO

Keywords:
Fog computing
Unmanned aerial vehicles (UAV)
Attitude control
Trajectory planning
Ant colony algorithm

ABSTRACT

Unmanned Aerial Vehicles (UAVs) have significantly enhanced fog computing by acting as both flexible computation platforms and communication mobile relays. In this paper, we consider four important and interdependent modules: attitude control, trajectory planning, resource allocation, and task assignment, and propose a holistic framework that jointly optimizes the total latency and energy consumption for UAV-assisted fog computing in a three-dimensional spatial domain with varying terrain elevations and dynamic task generations. We first establish a fuzzy-enhanced adaptive reinforcement proportional-integral-derivative control model to control the attitude. Then, we propose an enhanced Ant Colony System (ACS) based algorithm, that includes a safety value and a decoupling mechanism to overcome the convergence issue in classical ACS, to compute the optimal UAV trajectory. Finally, we design an algorithm based on the Particle Swarm Optimization technique, to determine where each offloaded task should be executed. Under our proposed framework, the outcome of one module would affect the decision-making in another, providing a holistic perspective of the system and thus leading to improved solutions. We demonstrate by extensive simulation results that our proposed framework can significantly improve the overall performance, measured by latency and energy consumption, compared to existing mainstream approaches.

1. Introduction

Driven by the development of the Internet of Things (IoT) [1], mobile terminal devices such as smartphones and tablets are now capable of generating and collecting massive amounts of data. However, the ability of processing these data, such as performing computational tasks, in the IoT devices are still limited [2,3]. On the other hand, Unmanned Aerial Vehicles (UAVs) have been recently identified as a versatile platform that connects IoT devices and servers or data centers via the network edge [4]. In addition, some UAVs are equipped with computational capabilities and thus can be regarded as “moving fog nodes" for offloading certain computational tasks. To fully utilize the versatility and flexibility of UAVs in fog computing, key considerations include the management of each UAV’s attitude, the strategic planning of their respective trajectories, and the efficient task assignment policy to determine the appropriate computing device for each task.

We consider a fog computing environment with a single UAV deployed at the network edge, aiming at maximizing the energy efficiency by collaboratively controlling the attitudes, planning the trajectories, allocating the transmission and computation resources, and assigning computing tasks to appropriate devices for execution. Specifically, **attitude control** (by adjusting pitch, roll, and yaw) ensures a stable and precise orientation of the UAV during operation, which is a necessary condition to maintain a high quality of communication with IoT devices and other fog nodes [5]. In addition, a stable attitude facilitates the UAV to effectively perform computation and storage tasks as a fog

⋆ A preliminary version of this paper was presented at IEEE HPCC 2022 in December 2022, and was included in its proceedings (DOI: 10.1109/hpcc-dss-smartcity-dependsys57074.2022.00217). Project website: https://shuaijun-liu.github.io/HATTO-UFog.

* Corresponding author.
E-mail addresses: shuaijun@bu.edu (S. Liu), turbodddu@gmail.com (J. Du), kristenzyx@163.com (Y. Zheng), yinjy35@mail.sysu.edu.cn (J. Yin), ivandeng@bnbu.edu.cn (Y. Deng), jj.wu@ieee.org (J. Wu).

https://doi.org/10.1016/j.comnet.2026.112064
Received 15 July 2025; Received in revised form 22 January 2026; Accepted 26 January 2026
Available online 30 January 2026

Table 1
A summary and comparison of related studies.

Research	Year	Process(es) focused	Method(s)	Objective(s)
[41]	2022	TA	Heuristic Harris Hawks Optimization	Min. energy consumption
[18]	2022	TP	Clustered-PSO	Min. energy efficiency
[32]	2022	TP	Enhanced ACS	Avoid path deadlock
[33]	2022	TP	Segmented ACS with Self-healing Routing	Avoid path deadlock
[27]	2021	3D-TP	Double-layer ACS	Optimal trajectory
[12]	2024	3D-TP	Genetic Algorithm (GA) & SCA	Optimal trajectory
[20]	2024	3D-TP	Chaotic-Polarized-Simulated scheme ACO	Min. trajectory loss rate
[13]	2019	ATC	Traditional PID controller	Max. control accuracy
[15]	2024	ATC	Fuzzy PID Controller	Min. control bias
[17]	2024	ATC	RL (Deep Q-Network) PID Controller	Max. control accuracy
[30]	2023	TA & RA	MARL & Stochastic Game Model	Min. energy consumption and latency
[14]	2016	3D-TP & ATC	Fuzzy PID controller	Control attitude and track trajectory
[16]	2024	3D-TP & ATC	RL PID Controller	Avoidance of obstacles
[35]	2023	2D-TP & TA	RL (Deep Q-Network)	Min. energy consumption and latency
[28]	2020	3D-TP & TA	MARL	Max. long-term resource efficiency
[9]	2022	3D-TP & TA	Successive convex approximation	Min. latency
[37]	2023	3D-TP & TA	DRL	Min. latency
[42]	2019	2D-TP & TA & RA	GA & Stepwise Approximation	Min. energy consumption and latency
[29]	2020	3D-TP & TA & RA	SCA & ADMM (Alternating Direction Method of Multipliers)	Max. UAV energy efficiency
[34]	2023	3D-TP & TA & RA	RL (MADDPG)	Min. energy consumption and latency
[11]	2024	3D-TP & TA & RA	Differential Evolutionary (DE) & RL (TD3)	Optimal trajectory and convergence
Our Work	2025	3D-TP & TA & RA & ATC	ACS-DS & PSO & FEAR-PID	Min. energy consumption and latency

Note: *TP = Trajectory Planning, TA = Task Assignment, RA = Resource Allocation, ATC = Attitude Control*

node. On the other hand, **trajectory planning** of the UAV can reduce its power consumption by identifying the most efficient path to collect data and tasks based on the locations of IoT devices [6]. Finally, **task assignment** refers to the process of deciding whether a specific task should be handled locally by the IoT device, processed in the fog layer (including the UAV), or offloaded to the central cloud based on real-time application-specific scenarios [7]. Specific objectives, including minimizing latency, maximizing throughput, or optimizing energy efficiency, can be achieved by assigning tasks to appropriate devices. Task assignment is often jointly optimized with **resource allocation**, where transmission resources such as power and bandwidth are distributed among different transmission pairs in the network, to facilitate the transmission process and improve the overall efficiency.

The four processes that we consider are inherently linked in UAV-assisted fog computing. For example, a UAV's attitude control would ensure that it maintains optimal orientations while following a planned trajectory or processing a task. Also, when deciding whether to offload a certain task to the fog or the cloud and how to allocate relevant transmission resources, the energy consumption and latency for a certain UAV to reach the proximity of the IoT initiating the task along a planned trajectory should also be taken into account.

Existing studies have considered two or three processes for joint optimization. For example, Cheng *et al.* [8] proposed three decision-making algorithms to solve the joint optimization problem involving energy consumption and mean delay. Zhou *et al.* [9] proposed a two-time-scale optimization framework that jointly determines caching placement and task offloading decisions while adaptively adjusting the UAV trajectory. However, few considered all these aspects together in an interconnected manner. A summary of relevant studies is provided in Table 1, and we will discuss them in more detail in Section 2.

This work substantially extends our earlier conference paper [10], to better capture the practical dynamics of UAV-assisted fog computing networks. The major improvements include: 1) While the previous study focused on a two-dimensional setting with simplified flight and communication assumptions, the present work develops a three-dimensional, terrain-aware network model that incorporates altitude variation and realistic environmental constraints; 2) We introduce an additional module (attitude control) for quadrotor UAVs, enabling the system to maintain stable flight, enhance link reliability under varying orientations, and mitigate latency during altitude transitions; and 3) We refine the algorithm for trajectory planning to better avoid deadlock by introducing decoupling and safety values mechanisms. These extensions allow the UAV to adapt its physical configuration and communication behavior in a coordinated manner, thereby improving overall energy efficiency, communication stability, and task execution performance beyond what was achieved in [10]. Compared to similar existing studies (e.g.,[11,12]), which only considered the trajectory planning and flight attitude at independent static points, we jointly consider planning the optimal trajectory and determining the attitudes, taking into account the extra consumption required for changing attitude along the trajectory. Our research is expected to provide new useful insights in UAV-assisted fog computing applications for improving the overall performance and efficiency.

Although some of the individual strategies employed in this work, such as fuzzy, adaptive, and PID algorithms, are well-established, the novelty of our approach lies in how these algorithms are integrated and jointly optimized within a unified UAV-assisted fog computing framework. In contrast to most existing studies that treat control and networking processes separately, our framework co-designs attitude control, trajectory planning, task assignment, and resource allocation in a mutually dependent manner. This cross-layer integration enables the UAV to adapt its physical dynamics and communication decisions in real time according to network conditions, task demands, and environmental factors. As a result, the proposed approach improves system-level performance in terms of energy efficiency, communication reliability, and latency, thereby providing a new perspective on the joint optimization of control and communication functions in UAV-enabled fog computing networks.

The contributions of this paper are summarized as follows.

- From a computer and communication network perspective, this work proposes a unified optimization framework for UAV-assisted fog computing networks that jointly integrates attitude control, trajectory planning, task assignment, and resource allocation within a three-dimensional network topology. By coupling the UAV's physical-layer control (attitude control) with network-layer decision making (resource allocation and task assignment), our cross-layer joint optimization framework captures the mutual dependencies among communication stability, mobility dynamics, and computational load distribution, thereby achieving adaptive connectivity, reduced latency, and improved overall network performance in realistic, terrain-aware IoT environments.

- We develop a fuzzy-enhanced adaptive reinforcement proportional-integral-derivative (FEAR-PID) model for attitude management of quadrator UAVs. Compared with classical PID and conventional fuzzy-PID control commonly used in existing studies, FEAR-PID captures interdependencies among parameters and adapts dynamically to environmental changes. Our results demonstrate that FEAR-PID significantly enhances stability during takeoff, cruising, and landing phases, thereby effectively reducing latency and energy consumption in UAV-assisted fog computing systems.
- We propose a computationally efficient algorithm called ACS-DS (Ant Colony System with Decoupling and Safety values) for UAV trajectory planning. The ACS-DS algorithm integrates decoupling and safety value mechanisms to address typical limitations of classical ACS, such as slow convergence rates and susceptibility to local optima. Numerical experiments also confirm ACS-DS's superior convergence performance relative to mainstream heuristic and reinforcement learning-based methods.
- We propose a heuristic algorithm based on PSO principles to effectively resolve the resource allocation and task assignment problems, given that an initial trajectory of the UAV has been determined. The heuristic algorithm efficiently overcomes the inherent complexities of the underlying non-convex optimization problems, providing quasi-optimal solutions for task assignment and resource allocation decisions.
- We demonstrate, through extensive numerical experiments, that our proposed holistic framework can reduce overall operational efficiency cost by more than 67% reduction in overall operational costs compared to existing heuristic and reinforcement learning-based methodologies. Our analysis and results underscore the importance of considering interdependencies among attitude control, trajectory planning, resource allocation, and task assignment in UAV-assisted fog computing. Consequently, the holistic approach significantly outperforms methods optimizing individual components separately, highlighting the cumulative benefits of joint optimization in such environment.

The rest of this paper is organized as follows. Section 2 reviews recent advancements on attitude control, trajectory planning, and task assignment in UAV-assisted fog computing. Section 3 provides descriptions on the UAV architecture as well as key metrics at the system level. Section 4 explains the formulation of the joint optimization problem. Section 5 describes the proposed computationally efficient algorithms to solve the problem in detail. Section 6 demonstrates the improvements of the proposed algorithm by extensive numerical results. Section 7 concludes the paper.

2. Related work

2.1. Attitude control

The proportional-integral-derivative (PID) control system is a fundamental approach widely used in the attitude control and trajectory planning of quadrotor UAVs [13,14]. Building on this foundation, fuzzy PID control introduces adaptive capabilities by incorporating fuzzy logic to dynamically adjust the proportional-integral (PI) and proportional-derivative (PD) components, resulting in improved stability and reduced trajectory tracking errors [15]. Recent studies further enhance these methods by leveraging reinforcement learning (RL). For instance, RL has been employed to regulate linear velocity while fuzzy logic manages angular velocity, achieving a complementary control strategy [16]. Another approach is the participation of RL in the construction of the PID controllers, which optimizes the gain parameters in real-time to achieve more accurate adaptive control under dynamic and complex environmental conditions [17].

2.2. Trajectory planning

For trajectory planning of UAVs, heuristic algorithms such as PSO [18], ACS [19,20], and genetic algorithm (GA) [12], have been adopted to overcome the space and computation complexities in such problems. Compared with another branch of approaches that use neural networks and deep learning as the key techniques (e.g., [11,21]), heuristic algorithms are more interpretable and less data dependent, and thus more appropriate for UAV-assisted fog computing scenarios where the operational environments are usually highly diverse and dynamic [22], and transparent heuristics are preferred for regulators to validate and verify the operation. Among the heuristic algorithms, the PSO [23] is one of the most commonly used techniques in such problems. However, one major concern of applying PSO in complex systems is that PSO may be converged prematurely to local optima [24].

ACS, based on the study of ants searching for food, is another commonly adopted technique in discrete and continuous optimization problems [25,26]. While ACS-based approaches are well-known for their robustness, they also suffer the disadvantage of being prone to local optima due to premature convergence like PSO. To overcome this issue, Wang *et al.* [27] presented an adaptive double-layer ant colony optimization algorithm (DL-ACS) based on an elitist strategy (ADAS) and an improved moving average algorithm (IMA) to solve a three-dimensional UAV trajectory planning problem. Recently, Yan *et al.* [20] proposed a chaotic-polarized-simulated ant colony optimization (CPS-ACO) algorithm, by incorporating chaotic mapping for initial pheromone distribution, a polarizing pheromone recording rule, and a simulated annealing mechanism, CPS-ACO demonstrated enhanced convergence speed and robustness against local optima in trajectory planning.

2.3. Task assignment

Resource allocation or task assignment problems have been extensively studied in existing studies. For example, Cui *et al.* [28] proposed a multi-agent Q-learning-based reinforcement learning (MARL) framework, where each agent independently executes the allocation algorithm to optimize the overall energy efficiency. Li *et al.* [29] both considered a joint optimization problem involving trajectory optimization and task allocation, with the goal of minimizing UAV energy consumption and optimizing computation offloading and using successive convex approximation (SCA) technique to solve it. Wu *et al.* [30] proposed a cooperative multi-agent deep reinforcement learning framework, which combines task assignment and allocation of limited communication resources to minimize the overall energy consumption and delay.

2.4. Joint optimization

One notable issue of applying ACS, GA or PSO in joint optimization involving trajectory planning and task assignment is that the algorithm may enter the deadlock state where one or more tasks wait endlessly for resources [31]. Hou *et al.* [32] proposed ACS-based algorithms with enhanced communication mechanisms to avoid deadlocks. While the proposed methods managed to reduce the likelihood of deadlocks, they did not eliminate the possibilities of such undesirable events. Another effort to overcome the deadlock problem is segmented planning and reintegration [33], which introduces self-healing mechanisms like In-Road Repairing and Intersection Repairing. These methods dynamically recalculate paths and penalize deadlocked routes using a negative reinforcement strategy. While this approach effectively avoids deadlocks and improves reliability, it incurs computational overhead due to frequent evaluations and path adjustments.

Another emerging approach to solving UAV-related planning and optimization problems is deep reinforcement learning (DRL) (e.g., [34–36]). Recently, Tan *et al.* [11] built a two-layer optimization framework by combining differential evolutionary algorithms and deep

reinforcement learning to optimize the UAV deployment, tasks assignment and resource allocation efficiency when participating in multi-access edge computing. Proximal policy optimization-based deep reinforcement learning algorithms were used in [37] to learn the optimal offloading strategy for dependent tasks in a joint optimization problem that also involves trajectory planning. DRL-based approaches have been also jointly applied with other optimization modules in edge computing, such as blockchain consensus leader election and waiting time window decision [38], and estimation of resource requirements through digital twins [39].

Compared with DRL-based approaches, the heuristic algorithms (such as ACS and PSO) are known to be more scalable, more adaptive and less computationally intensive [40]. These advantages make them particularly well-suited for the dynamic and large-scale nature of UAV-assisted fog computing scenarios. We will also compare their performance and computational efficiency in Section 6 of this paper.

Our work in this paper combines the techniques mentioned above that have been demonstrated to be effective in respective processes, including FEAR-PID attitude control, ACS-based trajectory planning, and PSO-based task allocation and resource allocation. Following this foundation, we further improve several aspects of the entire framework, such as 1) the hardware of the quadrotor is designed to match the efficient performance of the FEAR-PID control system, 2) two enhanced anti-lockout mechanisms (decoupling and safety values) in the classical ACS for trajectory planning, and 3) a modified PSO approach with improved efficiency that is more appropriate for task assignment and resource allocation in large-scale UAV-assisted fog computing systems.

3. System model

3.1. Network structure and components

We consider a UAV-assisted fog computing system in a three-dimensional Euclidean space. The system consists of a quadrotor UAV, a remote data center (DC) in the cloud, and K mobile IoT devices (MDs). A demonstration of the key structures in the system is shown in Fig. 1.

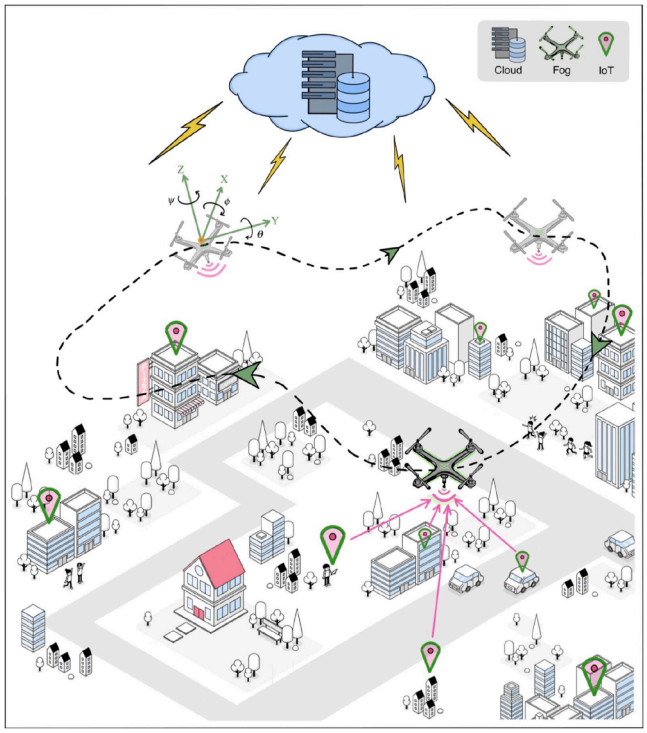


Fig. 1. The structure of UAV-assisted fog computing network.

The positions of K MDs are randomly distributed according to a Poisson Point Process (PPP), with the coordinate of the jth MD denoted by $\boldsymbol{M}_j = [x_j, y_j, z_j]$. For convenience, we discretize a continuous time horizon with length T uniformly into N timeslots, and thus each timeslot has a length of $L = \frac{T}{N}$. We assume that N is sufficiently large, or equivalently, the timeslots are sufficiently short, such that the position of the UAV can be considered to be fixed within each timeslot. Note that this timeslot refers to the high-level system decision interval for task allocation, trajectory updates, and communication resource allocation, rather than the low-level attitude control loop of the UAV. The attitude control loop typically operates at 100–500 Hz and is handled automatically by the flight controller, whereas the system-level decision timeslot in UAV or general mobile edge computing (MEC) systems commonly ranges from 0.1 s to several seconds and has no direct correspondence with the flight control loop frequency. We use $[x(t), y(t), z(t)]$, where $t \in \{1, 2, \cdots, N\}$, to denote the UAV position at the tth timeslot. The UAV is powered by a battery with a maximum capacity of B_c. The height and speed are restricted to not exceed $z^{\max}$ and $v^{\max}$, respectively, at all times.

We consider that task arrivals from each MD conform to a Poisson process, with an arrival rate λ_j from the jth MD. We use $s_{ij}(t)$, which is assumed to be exponentially distributed, to denote the data size of the ith task from the jth MD to be processed at the tth timeslot, and c_{ij} to denote the number of CPU cycles per bit required to process the task. As we introduced in [10], the proportion of channels allocated to the jth MD would be based on the Gamma distribution,

$$P_j(t) = \frac{\beta^\alpha}{(\alpha-1)!} s_{ij}(t)^{\alpha-1} e^{-\beta s_{ij}(t)}, \tag{1}$$

where α and β are the shape and rate parameters in the Gamma distribution. For demonstration purposes, we set $\alpha = \beta = 2$ throughout the paper.

Given the total number of available channels N_c, the number of channels allocated to the jth MD at the tth timeslot is $C_j(t) = \lfloor N_c \cdot P_j(t) \rceil$ where $[x]$ rounds x to the nearest integer. The rationale of this arrangement is to prevent the channel from being monopolized by extremely large tasks, and thus to ensure a certain level of transmission efficiency for all tasks.

For the ith task from the jth MD, we denote the energy consumption and delay as E_{ij} and D_{ij}, respectively. For each task, the quadrotor UAV needs to move sufficiently close to the MD that initiates the task through a planned trajectory, in order to receive and further process the task. As mentioned earlier, a task may be executed locally, in the fog layer by the UAV, or further offloaded to the data centers in the central cloud. The computing results of any offloaded task need to be transmitted back to the initiating MD. In this context, we define $\mathbf{o}_{ij}(t) = (o_{ij}^{\mathrm{MD}}(t), o_{ij}^{\mathrm{UAV}}(t), o_{ij}^{\mathrm{DC}}(t))$ as an array consisting of binary variables that indicate the specific execution location of the ith task from the jth MD during the tth timeslot.

For ease of reference, Table 2 summarizes the core notations used throughout the system model in Section 3; experiment-specific parameter values will be provided separately in Tables 5 and 6 in the results section.

3.2. The controllable structure of a quadrotor UAV

Quadrotor UAVs are capable of achieving six degrees of freedom in pitch, yaw, roll, vertical, fore and aft, and lateral motion by translating and rotating along the x, y and z axes. The process is accomplished by controlling four kinetic input quantities of the motor, denoted by $\mathbf{U} = (U_1, U_2, U_3, U_4)^T$, and six output control variables that include the changes of movements $\mathbf{v} = (\Delta x, \Delta y, \Delta z)$ on the x, y, and z directions, and the changes of attitude angles $\mathbf{A} = (\Delta\phi, \Delta\theta, \Delta\psi)$ resulting from rotation on the three axes.

Specifically, $\mathbf{v}$ stands for linear velocity, which is the speed at which the UAV moves in real space and determines the location and speed

Table 2
Key notations in the system model.

Notation	Definition	Notation	Definition
Network, Task Assignment, and Resource Allocation-Related Parameters			
K	The total number of MDs	L	The length of every timeslot
T	The total number of timeslots required to complete all tasks	(x_j, y_j, z_j)	The location of the jth MD
$(x(t), y(t), z(t))$	The location of the UAV at the tth timeslot	$\mathbf{v}(t)$	The UAV's velocity vector at the tth timeslot
$d_j(t)$	The distance between the UAV and jth MD at the tth timeslot	B_c	The battery capacity of the UAV
N_j	The total number of tasks from the jth MD	N_c	The total number of wireless channels
$C_j(t)$	The number of channels assigned to the jth MD at the tth timeslot	N_b	The number of UAV propeller blades
λ_j	Task arrival rate of the jth MD	$s_{ij}(t)$	Input data size of the ith task from the jth MD at timeslot t
c_{ij}	CPU cycles per bit required for the ith task from the jth MD	$P_j(t)$	Channel allocation proportion for the jth MD (Gamma-based)
α, β	Shape and rate parameters in the Gamma-based channel allocation	$\mathbf{o}_{ij}(t)$	Task execution decision vector (MD/UAV/DC) for task (i, j) at timeslot t
UAV Dynamics and Propulsion Parameters			
m	UAV mass	g	Gravitational acceleration
I_x, I_y, I_z	Rotational inertia around x, y, z axes	l	Distance from UAV center to rotor
$\mathbf{U}$	Dynamics input vector $(U_1, U_2, U_3, U_4)^T$	$\omega_i(t)$	Angular velocity of rotor i at timeslot t
R	The radius of UAV propeller blades	γ	The mounting angle of UAV propeller blades
P_w, P_t	Width and thickness of UAV propeller blades	$p_u(t), p_d(t), p_j(t)$	The transmission power of the UAV, the DC, or the jth MD at the tth timeslot
C_T, C_M	Thrust and torque coefficients of the propellers	$Q_\gamma(t)$	Yaw-/mounting-angle-dependent coefficient in the propeller model
Computation and Queuing Parameters			
$f(t), f_j(t)$	The processing frequency of equipment for UAV, or the jth MD at the tth timeslot	δ_u, δ_j	Computation-energy coefficients for UAV and MD
$\bar{Q}$	UAV computation queue-capacity parameter (bit-equivalent)	$\mathbb{S}(t)$	The weighted sum of network energy consumption and delay at the tth timeslot

of the UAV in the map. **A** determines angular velocities, the attitude change and the turning speed of the UAV.

According to the Newton-Euler theorem, the nonlinear dynamics equations of a quadrotor UAV are [43],

$$\begin{cases} \ddot{x}(t) = \frac{1}{m}U_1(t)(\sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi) \\ \ddot{y}(t) = \frac{1}{m}U_1(t)(-\sin\phi\sin\psi + \cos\phi\sin\theta\sin\psi) \\ \ddot{z}(t) = \frac{1}{m}U_1(t)(\cos\phi\cos\theta) - g \\ \ddot{\phi}(t) = \frac{1}{I_x}\left(lU_2(t) - \dot{\theta}\dot{\psi}\left(I_z - I_y\right)\right) \\ \ddot{\theta}(t) = \frac{1}{I_y}\left(lU_3(t) - \dot{\phi}\dot{\psi}\left(I_x - I_z\right)\right) \\ \ddot{\psi}(t) = \frac{1}{I_z}\left(U_4(t) - \dot{\phi}\dot{\theta}\left(I_y - I_x\right)\right) \end{cases} \tag{2}$$

where the left hand side of each equation represents the second order derivative of the relevant variable with respect to t. I_x, I_y, and I_z are the rotational inertia of the UAV body around its own x, y, and z axes, respectively. I_x and I_y are approximated based on the assumption of structural symmetry of the quadrotor UAV. Meanwhile, l is the distance from the center of mass of the body to the center of the rotor, which is also equal to half of the airframe wheelbase.

Fig. 2 shows the schematic diagram of a quadrotor UAV in motion. Two pairs of motors $(M1, M4)$ and $(M2, M3)$ rotating in opposite directions are used to eliminate counter-torque. Moreover, m is the mass of the UAV. g is the acceleration of gravity, I_x, I_y, and I_z are the rotational inertia of the UAV body around x, y, and z axes, respectively. In Fig. 2, subscripts e and b denote the earth (inertial) frame and body frame, respectively; the corresponding coordinate axes are denoted as (x_e, y_e, z_e) and (x_b, y_b, z_b). These axes correspond to the translational directions associated with the variables x, y, z in (2).

The input and output quantities interact with each other to change the UAV motion by adjusting the rotational speed of the four rotors (propellers). The thrusts $F_i(t), (i = 1, 2, 3, 4)$ provided by the motors are directly proportional to the square of the angular velocity of the rotors (propellers) of the corresponding motors $\omega_i(t), (i = 1, 2, 3, 4)$, namely,

$$F_i(t) = C_T(t)\omega_i(t)^2, \tag{3}$$

where $C_T(t)$ is the thrust coefficient of the propellers. The torque representing the magnitude of the torque to overcome the air resistance is given by

$$M_p(t) = C_M(t)\omega_i(t)^2, \tag{4}$$

with $C_M(t)$ as the torque coefficient.

For more details on principles related to attitude control and the calculation of the dynamics input **U**, see A.

3.3. Rotor propeller design of quadrotor UAV

The design and performance modeling of multirotor UAVs, including propeller configurations and power systems, have been extensively studied [44,45]. We now introduce the factors that would affect $C_T(t)$ and $C_M(t)$. The propeller is the power source of the UAV. Fig. 3 shows the structure of the quadrotor UAV propeller, where the propeller radius R refers to the sum of the blade length r and the radius of the mounting interface r_0. The blade mounting angle is the angle between the propeller blades and the plane of the UAV airframe, denoted by γ. Along with the blade width P_w, thickness P_t, we define the torque

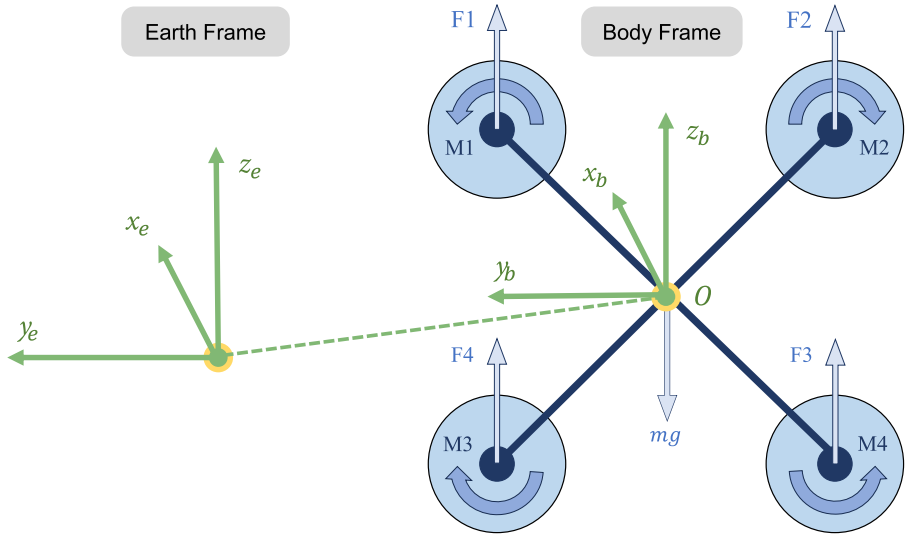


Fig. 2. Schematic diagram of a quadrotor UAV in motion, where subscripts e and b denote the earth and body frames, respectively, and (x_e, y_e, z_e) and (x_b, y_b, z_b) denote their coordinate axes associated with the translational directions x, y, z.

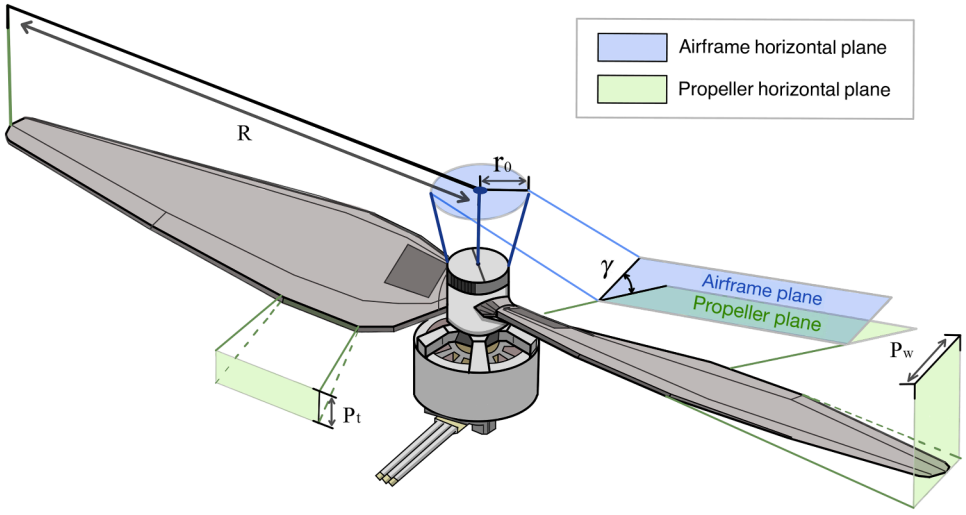


Fig. 3. Schematic of quadrotor UAV propeller hardware.

coefficient as,

$$C_M(t) = 2R \cdot C_T(t) = \frac{\left(\frac{1}{2\pi}\right)^2 \rho_a(t)(2R)^5}{N_B \int_{r_0}^{R} P_t^4 P_w Q_\gamma(t) dr}, \tag{5}$$

where N_B is the number of propeller blades, and $Q_\gamma(t)$ is determined by the propeller blade mounting angle and yaw angle in the current state. The power loading and disk loading characteristics of propellers significantly influence the overall energy efficiency [46]. $\rho_a(t)$ is the air density determined by the atmospheric pressure at the location of the UAV and the absolute temperature, specifically,

$$\rho_a(t) = \frac{P_g \cdot M_g}{R_g \cdot \Theta(t)}, \tag{6}$$

where P_g is the standard atmosphere, M_g is the molecular weight of gas, R_g is the gas constant, $\Theta(t)$ is the outside ambient temperature, $z(t)$ is the height of flight at the location of the UAV, and M_g and R_g are constants in the ideal gas state.

The values of propeller radius R and airframe wheelbase $2l$ need to be set properly, in order to avoid collisions between neighboring propellers and underpowered situations. We illustrate the structural relationship in Fig. 4. In a symmetrically structured quadrotor configuration, the diagonal separation distance between neighboring propellers is $\sqrt{2}l$, and the total width of the two propellers $(2R)$ must be less than this distance to avoid collision, namely $2R < \sqrt{2}l$. On the other hand, a too small R (less than $l/3$) will not provide enough lift for the UAV according to (3). Thus, there is a constrained relationship between R and l, namely $\frac{\sqrt{2}}{2}l \geqslant R \geqslant \frac{1}{3}l$.

3.4. Transmission model

We define a tuple (m, n) to represent a transmission from m to n, where m and n can be $j \in \{1, 2, \cdots, K\}$ (representing the jth MD), u (the UAV), or D (the DC).

We consider ζ_j to be a parameter determined by the carrier frequency f_c and obstacle density, corresponding to the proportion of the Line of Sight (LoS) path obstructed between the jth MD and the UAV, while the elevation angle of the link is θ_j.

In our deployment, the communication antenna of the UAV is bottom-mounted, and the MDs are always located on the ground surface. Therefore, the UAV always maintains a higher altitude than all MDs, ensuring that no geometric airframe shadowing occurs. The aggre-

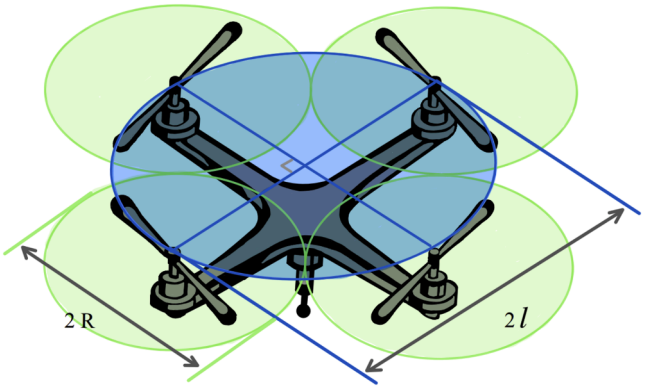


Fig. 4. Propeller radius in relationship to wheelbase for quadrotor UAV.

gated blockage effects–including terrain and man-made obstacles–are modeled through a parameter ζ_j, as defined in the following equation.

The probability to establish a LoS connection between the jth MD and the UAV is,

$$P_j^{\mathrm{LoS}}(t) = \frac{1}{1+\zeta_j \cdot \exp\left(-\left(\theta_j - \zeta_j\right)\right)}. \tag{7}$$

The Signal-to-Interference-plus-Noise Ratio (SINR) for (m,n) at the tth timeslot is,

$$\xi_{(m,n)}(t) = \frac{p_m(t)}{\mathcal{L}_{(m,n)}(t)\cdot\left(I_{(m,n)}(t)+\sigma_T^2\right)}, \tag{8}$$

where $p_m(t)$ is the transmit power of node m at the tth timeslot. We denote by $P_{(m,n)}^{\mathrm{LoS}}(t)$ the LoS probability of link (m,n), which equals $P_j^{\mathrm{LoS}}(t)$ for MD–UAV links and is set to 1 for the UAV–DC backhaul link for simplicity. $\mathcal{L}_{(m,n)}(t) = P_{(m,n)}^{\mathrm{LoS}}(t)\cdot\left(\frac{4\pi f_c}{c} d_{(m,n)}(t)\right)$ is the path loss, c and $d_{(m,n)}(t)$ stand for the speed of light and the distance between m and n, respectively. Here, $I_{(m,n)}(t)$ denotes the aggregate external interference from concurrent transmissions outside our UAV-assisted fog network, which we model as additive noise for simplicity. σ_T^2 is the thermal noise. Our transmission model assumes orthogonal channel allocation within the UAV-assisted network (via the channel allocation mechanism in Section 3.1), which ensures negligible co-channel interference among the K MDs communicating with the UAV. This simplified interference modeling approach is standard in MEC and fog computing literature, where detailed multi-cell interference coordination would require additional coordination mechanisms and is considered as future work.

We can then obtain the effective transmission rate for (m,n) at tth timeslot as,

$$r_{(m,n)}(t) = W_{(m,n)}\log_2\left(1+\xi_{(m,n)}(t)\right), \tag{9}$$

where $W_{(m,n)}$ is the bandwidth of the wireless link.

3.5. Energy consumption model

We consider that the total energy consumption includes the consumption for the transmission and computation during the task offloading process, as well as those for the movement of the UAV. Hereafter, we use the binary variables $o_{ij}^{\mathrm{MD}}(t)$, $o_{ij}^{\mathrm{UAV}}(t)$, and $o_{ij}^{\mathrm{DC}}(t)$ to specify the assignment of the ith task from the jth MD at the tth timeslot. A value of 1 indicates that the task is assigned to the corresponding location.

3.5.1. UAV movement

We consider that the motion state of the UAV is directly related to the rotational speed of the four DC motors. The energy consumption for UAV movement is highly dependent on the payload and flight characteristics [45,47,48]. Thus, the energy consumption due to UAV movement at the tth timeslot is,

$$E^{\mathrm{MOV}}(t) = \sum_{i=1}^{4}\left(I_i(t)\cdot L\right) = \sum_{i=1}^{4}\left(\frac{V_m - \omega_i(t)P_m}{R_m}\cdot L\right), \tag{10}$$

where $I_i(t)$ is the motor current, which can be calculated by (A.2) and (2) given that U and $\omega_i(t)(i=1,2,3,4)$ (the linear speed of ith motor rotation (r/min) at the tth timeslot) are obtained by our proposed attitude control method. V_m is the motor rated voltage, R_m is the rated resistance of the UAV motor circuit, and P_m is the electric potential constant determined by the motor structure.

3.5.2. Energy consumption for transmission

If a task is assigned to the UAV, two segments of transmission will be incurred, namely from the MD to the UAV and the UAV back to the MD. On the other hand, if a task is assigned to the DC, two additional segments, namely from the UAV to the DC and from the DC back to the UAV, will be incurred. Therefore, energy consumption at the tth timeslot for transmitting the ith task from the jth MD is,

$$\begin{aligned} e_{ij}^{\mathrm{TR}}(t) =& \left(o_{ij}^{\mathrm{UAV}}(t)+o_{ij}^{\mathrm{DC}}(t)\right)\left(\frac{p_j(t)s_{ij}(t)}{C_j(t)r_{(j,u)}(t)}+\frac{p_u(t)s_{ij}(t)}{C_j(t)r_{(u,j)}(t)}\right) \\ &+ o_{ij}^{\mathrm{DC}}(t)\left(\frac{p_u(t)s_{ij}(t)}{C_j(t)r_{(u,D)}(t)}+\frac{p_D(t)s_{ij}(t)}{C_j(t)r_{(D,u)}(t)}\right). \end{aligned} \tag{11}$$

where $C_j(t)$ is the number of channels assigned to the jth MD.

3.5.3. Energy consumption for computation

We consider that the additional energy consumption for computing a finite number of tasks in the DC is negligible as the DC is assumed to be always active in processing tasks from different sources. Therefore, the energy consumption for computing ith task of the jth MD can be obtained by

$$e_{ij}^{\mathrm{COMP}}(t) = o_{ij}^{\mathrm{UAV}}(t)\delta_u s_{ij}(t)c_{ij}(f(t))^2 + o_{ij}^{\mathrm{MD}}(t)\delta_j s_{ij}(t)c_{ij}\left(f_j(t)\right)^2, \tag{12}$$

where δ_u and δ_j are the computation-energy coefficients (unit: J/(cycle^2·bit)) for the UAV and the jth MD, respectively. Specifically, $\delta_u = 1.2\times10^{-28}$ J/(cycle^2·bit) for the UAV and $\delta_j = 3.0\times10^{-28}$ J/(cycle^2·bit) for MDs, where the higher value for MDs reflects their weaker hardware efficiency.

3.5.4. Overall energy consumption

Based on the discussions above, the overall energy consumption at the tth timeslot can be expressed as,

$$\mathbb{E}(t) = \sum_{j=1}^{K}\sum_{i=1}^{N_j}\left[e_{ij}^{\mathrm{TR}}(t)+e_{ij}^{\mathrm{COM}}(t)\right] + E^{\mathrm{MOV}}(t), \tag{13}$$

where N_j is the total number of tasks the jth MD.

3.6. Delay model

We consider three delay components: transmission, computation, and queuing.

3.6.1. Transmission delay

The transmission delay for the ith task from the jth MD at the tth timeslot is,

$$\begin{aligned} d_{ij}^{\mathrm{TR}}(t) =& o_{ij}^{\mathrm{DC}}(t)\left(\frac{s_{ij}(t)}{C_j(t)r_{(u,D)}(t)}+\frac{s_{ij}(t)}{C_j(t)r_{(D,u)}(t)}\right) \\ &+ \left(o_{ij}^{\mathrm{UAV}}(t)+o_{ij}^{\mathrm{DC}}(t)\right)\left(\frac{s_{ij}(t)}{C_j(t)r_{(j,u)}(t)}+\frac{s_{ij}(t)}{C_j(t)r_{(u,j)}(t)}\right). \end{aligned} \tag{14}$$

3.6.2. Computation delay

The delay for computing the ith task from the jth MD can be expressed as,

$$d_{ij}^{\mathrm{COMP}}(t) = o_{ij}^{\mathrm{UAV}}(t)\frac{s_{ij}(t)c_{ij}}{f(t)} + o_{ij}^{\mathrm{MD}}(t)\frac{s_{ij}(t)c_{ij}}{f_j(t)}. \tag{15}$$

where $f(t)$ (unit: GHz) denotes the UAV's processing frequency, bounded by $f^{\min} = 1.0$ GHz and $f^{\max} = 3.0$ GHz, representing a high-performance onboard processor. Similarly, $f_j(t)$ (unit: GHz) denotes the jth MD's processing frequency, bounded by $f_j^{\min} = 0.5$ GHz and $f_j^{\max} = 2.0$ GHz. The UAV distributes its processing frequency $f(t)$ among all assigned tasks according to the joint optimization algorithm, which minimizes the overall weighted energy–delay cost. Thus, resource allocation is implicitly determined by the solution of the optimization problem rather than by a fixed scheduling rule.

Note that as the computing resources at the DC are sufficient for all practical purposes, we assume that the computation delay at the DC is negligible.

3.6.3. Queuing delay

We consider two types of queuing delays: transmission queuing and computation queuing. Transmission queuing occurs when the arrival rate exceeds the communication service rate, while computation queuing occurs when the aggregated task workload exceeds the UAV's effective computing rate.

Our queuing model adopts an M/M/1-type abstraction, which is a standard first-order modeling approach in MEC literature for capturing queuing behavior under stochastic task arrivals [49,50]. This abstraction enables tractable analysis while providing reasonable approximations of system performance under moderate to high load conditions. While more detailed queueing network models (e.g., multi-server queues or priority-based scheduling) could offer finer granularity, they would significantly complicate the optimization problem formulation. Our simplified model remains compatible with such extensions, which could be incorporated as future work when detailed scheduling policies are required.

Consider that the arrival rate for transmission from the jth MD to UAV is λ_j, and the processing rate at the UAV for receiving transmissions is $[L \cdot C_j(t) \cdot r_{j,u}(t)]/s_j(t)$, where $s_j(t) = \sum_{i=1}^{N_j} s_{ij}(t)$ is the sum of tasks data size from that MD. Let π_j denote the probability that an arriving task from the jth MD does not enter the transmission/computation queue (e.g., executed locally), so the effective arrival rate is $\lambda_j(1-\pi_j)$. If the total arrival rate is larger than the processing rate, a queuing delay for transmission will be incurred [42], that is,

$$d_{ij}^{\mathrm{Q}}(t) = \frac{\lambda_j\left(1-\pi_j\right)s_j(t)^2}{L \cdot C_j(t) r_{j,u}(t) \cdot \left(L \cdot C_j(t) r_{j,u}(t) - \lambda_j\left(1-\pi_j\right)s_j(t)\right)}. \tag{16}$$

Similarly, if the total sizes of arriving tasks exceed the computation capacity of the component, queuing delay for computation would be incurred for a proportion of the tasks. The expected queuing delay of the ith task from the jth MD executed at UAV at tth timeslot is,

$$d_{ij}^{\mathrm{UAVQ}}(t) = \frac{\bar{Q}}{\sum_{j=1}^{K}\lambda_j\left(1-\pi_j\right)} - \frac{\sum_{j=1}^{K}\lambda_j\left(1-\pi_j\right)s_j(t)c_j}{\tau f(t)\sum_{j=1}^{K}\lambda_j\left(1-\pi_j\right)}, \tag{17}$$

where $\bar{Q} = 5 \times 10^7$ (unit: bit-equivalent) is a queue-capacity parameter representing a queue threshold corresponding to the UAV's onboard computing throughput, and $\tau = 1$ s is a normalization constant that normalizes the aggregated computational workload into an effective service time per task.

3.6.4. Total delay

The total delay is obtained by summing all delay components mentioned earlier, that is,

$$\mathbb{D}(t) = \sum_{j=1}^{K}\sum_{i=1}^{N_j}\left(d_{ij}^{\mathrm{TR}}(t) + d_{ij}^{\mathrm{COMP}}(t) + d_{ij}^{\mathrm{UAVQ}}(t) + d_{ij}^{\mathrm{Q}}(t)\right). \tag{18}$$

Congestion in the network manifests through the growth of queue lengths, which directly contributes to the queuing delay components $d_{ij}^{\mathrm{Q}}(t)$ and $d_{ij}^{\mathrm{UAVQ}}(t)$ in the total delay expression. Since our objective function minimizes the weighted sum of delay and energy consumption, congested conditions (characterized by increased queuing delays) are implicitly penalized. This approach eliminates the need for an explicit congestion metric, as the optimization naturally steers the system away from congestion-prone configurations to minimize the overall operational cost.

4. Joint optimization problem

We consider that the operational efficiency cost of the network is determined by the weighted sum of the energy and delay components. Therefore, our objective function consists of the energy consumption defined in (13) and the time required to complete all tasks defined in (18), during the entire period of T, as follows.

$$\begin{aligned}
&\min_{\{\mathbf{o}_{ij}(t),\mathbf{v}(t),\omega_i(t)|t\in\{0,\cdots,T\}\}} && \mathbb{S} = \sum_{t=0}^{T}[\mathbb{D}(t) + \epsilon\cdot\mathbb{E}(t)] \\
&\text{s.t.} && (\mathtt{C}_1)\quad \omega^{\min} \le \omega_i(t) \le \omega^{\max}, \quad \forall i \\
& && (\mathtt{C}_2)\quad 0 \le |\mathbf{v}(t)| \le v^{\max} \\
& && (\mathtt{C}_3)\quad 0 \le z(t) \le z^{\max} \\
& && (\mathtt{C}_4)\quad \textstyle\sum_{j=1}^{K} C_j(t) \le N_c \\
& && (\mathtt{C}_5)\quad f^{\min} \le f(t) \le f^{\max} \\
& && (\mathtt{C}_6)\quad f_j^{\min} \le f_j(t) \le f_j^{\max} \\
& && (\mathtt{C}_7)\quad p^{\min} \le p_u(t) \le p^{\max} \\
& && (\mathtt{C}_8)\quad p_j^{\min} \le p_j(t) \le p_j^{\max} \\
& && (\mathtt{C}_9)\quad o_{ij}^{\mathrm{MD}}(t) + o_{ij}^{\mathrm{UAV}}(t) + o_{ij}^{\mathrm{DC}}(t) = 1 \\
& && (\mathtt{C}_{10})\quad o_{ij}^{\mathrm{MD}}(t), o_{ij}^{\mathrm{UAV}}(t), o_{ij}^{\mathrm{DC}}(t) \in \{0,1\} \\
& && (\mathtt{C}_{11})\quad \textstyle\sum_{t=0}^{T}\left(E^{\mathrm{UAV}}(t)\right) \le B_c \\
& && i \in \{1,\ldots,N_j\}, j \in \{1,\ldots,K\}
\end{aligned} \tag{19}$$

The decision variables include the angular velocities $\omega_i(t)$ related to attitude control, the velocity vector $\mathbf{v}(t)$ for trajectory planning, and task assignment decisions $\mathbf{o}_{ij}(t)$. The parameter ϵ is a weighting factor in the objective function, which flexibly caters to different magnitudes and ranges of the two measurements, as well as reflects the preference in terms of the trade-off between the two metrics in various fog computing scenarios. We will demonstrate in the result section that our proposed approach can achieve high performances for a wide range of ϵ values. In terms of the constraints, $\mathtt{C}_1$ to $\mathtt{C}_4$ are constraints imposed by the hardware limitations of the respective components. $\mathtt{C}_5$ to $\mathtt{C}_8$ ensure that the processing frequency and transmission power of UAV and MDs are always within the effective range. $\mathtt{C}_9$ and $\mathtt{C}_{10}$ guarantee that every task is executed at exactly one place at any moment (no duplicate computing). $\mathtt{C}_{11}$ restricts that the total energy consumption of the UAV during T does not exceed its battery capacity, as $E^{\mathrm{UAV}}(t)$ denotes the energy consumption of the UAV at the tth timeslot,

$$\begin{aligned}
E^{\mathrm{UAV}}(t) = E^{\mathrm{MOV}}(t) + \sum_{j=1}^{K}\sum_{i=1}^{N_j}\Bigg[o_{ij}^{\mathrm{DC}}(t)\left(\frac{p_u(t)s_{ij}(t)}{C_j(t)r_{(u,D)}(t)}\right) \\
+ o_{ij}^{\mathrm{UAV}}(t)\left(\frac{p_u(t)s_{ij}(t)}{C_j(t)r_{(u,j)}(t)} + \delta_u s_{ij}(t)c_{ij}(f(t))^2\right)\Bigg].
\end{aligned} \tag{20}$$

Note that the processing frequencies $f(t)$, $f_j(t)$, and transmission powers $p_u(t)$, $p_j(t)$ are dependent on the decision variables $\mathbf{o}_{ij}(t)$. Numerous scholarly works (e.g., [42]) provided comprehensive analyses of these intricate interactions.

5. Algorithms

The problem (19) possesses an intrinsic nonconvex nature. We divide the problem into three subproblems, one focusing on controlling the attitude, the second on generating an initial trajectory and altitude, and the other on task assignment and resource allocation, as well as further tuning the trajectory, respectively.

Table 3
Initial fuzzy PID control rules.

NB: Negative Big (−3) NM: Negative Middle (−2) NS: Negative Small (−1) ZO (0): zero PB: positive big (3) PM: positive middle (2) PS: positive small (1)							
	EC						
E	**NB**	**NM**	**NS**	**ZO**	**PS**	**PM**	**PB**
	$\Delta k_p/\Delta k_i/\Delta k_d$	$\Delta k_p/\Delta k_i/\Delta k_d$	$\Delta k_p/\Delta k_i/\Delta k_d$	$\Delta k_p/\Delta k_i/\Delta k_d$	$\Delta k_p/\Delta k_i/\Delta k_d$	$\Delta k_p/\Delta k_i/\Delta k_d$	$\Delta k_p/\Delta k_i/\Delta k_d$
NB	PB/NB/PS	PB/NB/NS	PM/NM/NB	PM/NM/NB	PS/NS/NB	ZO/ZO/NM	ZO/ZO/PS
NM	PB/NB/PS	PB/NB/NS	PM/NM/NB	PS/NS/NM	PS/NS/NM	ZO/ZO/NS	NS/PS/ZO
NS	PM/NB/ZO	PM/NM/NS	PM/NS/NM	PS/NS/NM	ZO/ZO/NS	NS/PM/NS	NS/PM/ZO
ZO	PM/NM/ZO	PM/NM/NS	PS/NS/NS	ZO/ZO/NS	NS/PS/NS	NM/PM/NS	NM/PM/ZO
PS	PS/NM/ZO	PS/NS/ZO	ZO/ZO/ZO	NS/PS/ZO	NS/PS/ZO	NM/PM/ZO	NM/PB/ZO
PM	PS/ZO/PB	ZO/ZO/PM	NS/PS/PM	NM/PS/PM	NM/PM/PS	NM/PB/PS	NB/PB/PB
PB	ZO/ZO/PB	ZO/ZO/PM	NM/PS/PM	NM/PM/PM	NM/PM/PS	NB/PB/PS	NB/PB/PB

5.1. Attitude control

5.1.1. Classical PID control

The classical PID control algorithm is a widely used algorithm for UAV attitude control [13]. It considers the following three control parts:

- Proportional control (P): the error of the controller input is amplified by a certain proportion. This constitutes the basic control part;
- Integral control (I): an integral term is introduced for correction of the proportional control to maintain the system stability. Its coefficient will approach 0 as time goes by;
- Differential control (D): a differential term based on the rate of change of the input error is added. The purpose is to improve the response accuracy by predicting the future trend of the input error.

The classical PID control is described by the following equation,

$$U(t) = k_p e(t) + k_i \int e(t)dt + k_d \frac{d}{e}(t)dt, \tag{21}$$

where k_p, k_i and k_d are the proportional, integral and differential coefficients, respectively. The output of the PID control system, $U(t)$, will affect $\omega, \theta, \phi, \psi$, which will in turns have an impact on the power consumption of the UAV (see Section 3.2 and A for details).

5.1.2. Fuzzy PID control

The fuzzy PID control algorithm improves on the classical PID control by adding nonlinear adaptive capability through Fuzzy Set Theory [14,15]. By inputting fuzzification and a set of fuzzy rules, the PID gain is dynamically adjusted according to the current error E and the error rate EC. In addition, the coefficients k_p, k_i, and k_d are no longer constants, enabling them to adapt to environmental changes. However, since the fuzzy rules are preset based on expert knowledge, the controller lacks flexibility in adapting to unforeseen complex environmental changes in real time [17].

5.1.3. FEAR-PID control

To solve the above problems, we propose the Fuzzy-Enhanced Adaptive Reinforcement PID (FEAR-PID) controller. FEAR-PID integrates RL in fuzzy PID to provide an additional adaptive layer. The control process of FEAR-PID is demonstrated in Fig. 5. Specifically, the initial control strategy is first provided by fuzzy logic based on expert knowledge. The specific rules are shown in Table 3.

Then, the PID gain coefficients in the fuzzy controller are obtained by a Double Deep Q-Network (DDQN). The RL module continuously optimizes the PID gain coefficients based on real-time feedback, thus effectively adapting to the dynamic environment and improving the control effect. The RL agent in FEAR-PID is designed as follows:

- **State:** The state S_t at timeslot t consists of the E, EC, and the PID coefficients Δk_p, Δk_i, and Δk_d. Represented as $S_t = [E, EC, \Delta k_p, \Delta k_i, \Delta k_d]$.
- **Action:** The action $A_t = [\Delta k_p, \Delta k_i, \Delta k_d]$ represents the adjustments made to the PID coefficients at each step, resulting in updated PID gains,

$$\begin{cases} k_{p-FEAR} = k_p + \Delta k_p \\ k_{i-FEAR} = k_i + \Delta k_i \\ k_{d-FEAR} = k_d + \Delta k_d. \end{cases} \tag{22}$$

- **Reward:** The reward function R_t is designed to guide the controller toward minimizing the control error and achieving stable responses.

$$R_t = -\left(\vartheta_1 |E| + \vartheta_2 \left|\frac{dE}{dt}\right| + \vartheta_3 \cdot \Delta U^2 + \vartheta_4 \cdot \Upsilon\right) \tag{23}$$

where the absolute error $|E|$ aims to minimize the error. The term $\left|\frac{dE}{dt}\right|$ controls the rate of error change for stability, ΔU^2 captures control output variation, incentivizing smoother adjustments, AND Υ denotes the overshoot penalizing excessive responses and aids in maintaining stability. The weights $\vartheta_1, \vartheta_2, \vartheta_3$, and ϑ_4 are provided to balance accuracy, convergence speed, stability, and control smoothness. The specific values and ranges of relevant parameters summarized in Table 4.

5.2. Task assignment and resource allocation

An overview of the steps in our proposed framework to address the task assignment, resource allocation and trajectory planning modules in the joint optimization problem is shown in Fig. 6. We will apply an Ant Colony System based algorithm (Algorithm 2, to be described in detail later) to determine an initial trajectory based on the terrain of the area and the position of MDs. Then, the attitude control mechanism described in Section 5.1 will be invoked to determine the as well as tune the trajectory. Finally, a Particle Swarm Optimization based algorithm (Algorithm 1, which was initially proposed in the conference version [10]) will determine the optimal task assignments and resource allocations. We will also analyze the convergence performance of the proposed algorithms at the end of the section.

We first introduce an algorithm based on the idea of Particle Swarm Optimization (PSO), for assigning tasks and allocating resources given that the trajectory is already known. The steps of our proposed algorithm are demonstrated in Algorithm 1, which we refer the readers to the conference version of this paper [10] for more detailed explanations. We categorize the job assignment decision, transmission power, and processing frequency as a collective entity $s_h = (\mathbf{o}_{ij}, p_j(t), f_j(t))$ within the particle swarm. Here, $h \in \{1, \ldots, H\}$. Next, the determination of the particle group's position is carried out by uniformly sampling a set of particle groups denoted by $H \in \mathbb{N}_+$. In order to search for the optimal solution, the velocity of the particle groups is initialized and subsequently updated using the method in [51]. The algorithm terminates when the discrepancy between the outcomes of 20 successive rounds is less than

Table 4
FEAR-PID parameter settings.

Parameters	Values	Parameters	Values	Parameters	Values
E, EC	-6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6	Δk_p, Δk_d	-3, -2, -1, 0, 1, 2, 3	Δk_i	-1.5, -1, 0, 1, 1.5
ϑ_1	3	ϑ_2	0.5	ϑ_3, ϑ_4	1.5

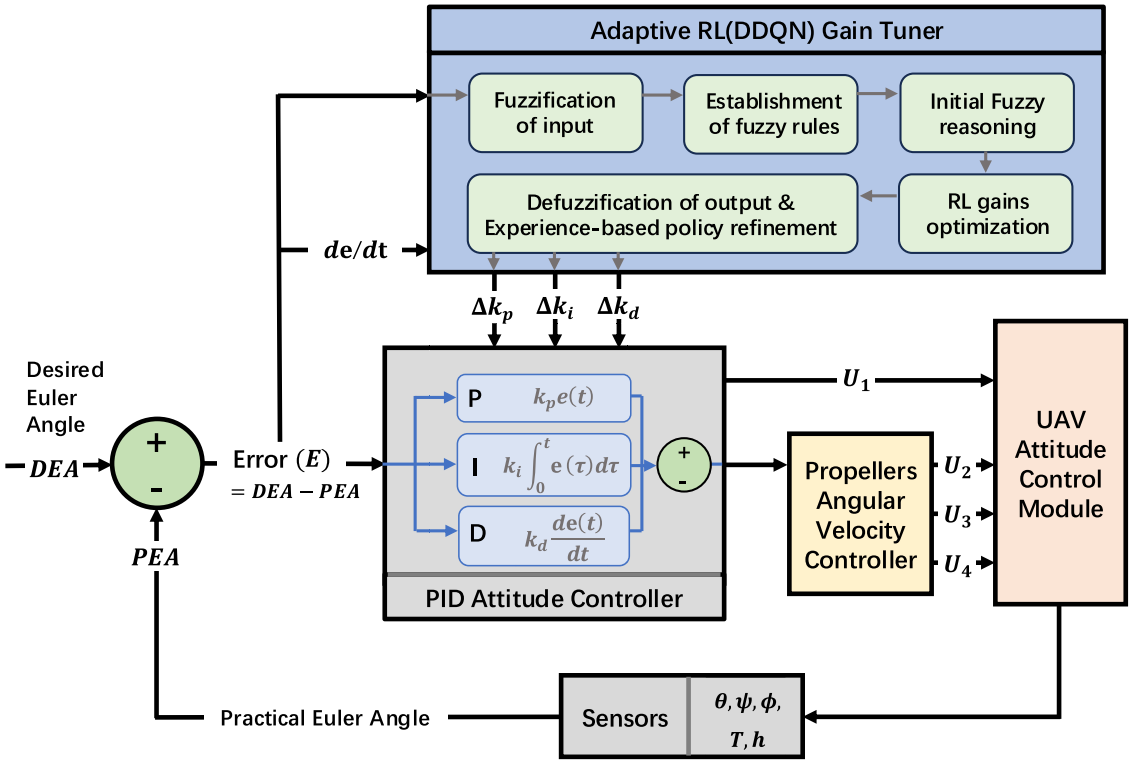


Fig. 5. Schematic structure of FEAR-PID control system for UAV.

a specified threshold ð, and then outputs the minimal value $\mathbb{S}^*$, the optimal decision for job assignment $\mathbf{o}_{ij}^*$, the transmission power $p_j^*(t)$, and the processing frequency $f_j^*(t)$.

Regarding fairness considerations, we note that fairness is not explicitly included as an optimization target in our objective function, which focuses on minimizing overall energy consumption and delay. However, our system design incorporates natural fairness mechanisms through orthogonal channel allocation (preventing channel monopolization) and the feasibility constraints that ensure all tasks are assigned to appropriate execution locations. The Gamma-based channel allocation strategy further promotes equitable resource distribution by preventing extremely large tasks from dominating available channels. A comprehensive quantitative fairness analysis, including metrics such as Jain's fairness index or per-user delay variance, would require additional modeling extensions and is considered as future work for scenarios where explicit fairness guarantees are required.

5.3. ACS-DS for trajectory planning: motivation and concept

We now present the concept and outlines of the ACS-DS (Ant Colony System with **Decoupling** and **Safety values**) algorithm to generate the optimal UAV trajectory to minimize E^{MOV}. Then, we will prove that ACS-DS always converges to the optimal solution in polynomial time in subsequent subsections.

We first divide the 3D space into discrete grids, with the center of each unit of the grid serving as a *waypoint*, that is, the position that the UAV will pass. The 3D terrain is generated randomly, and we denote the area at and below terrain surfaces, referred as the *no-fly zone*, by $\boldsymbol{O} = \{(x_n^o, y_n^o, z_n^o)\}$, where $n \in \{1, 2, \ldots, N_o\}$, and N_o is the total number of grids in the no-fly zone. Note that the classical ACS is not applicable to our three-dimensional trajectory planning problem as it struggles to converge in path searching in large three-dimensional spaces, and is likely to be trapped in local optima.

To overcome these issues and obtain the optimal trajectory with minimum cost in an efficient manner, we propose the ACS-DS, by incorporate the two special mechanisms to the classical ACS. We now elaborate these two mechanisms.

5.3.1. Safety values

We enhance the heuristic function of the ACS-DS by adding a security value at each step based on the number of currently known the numbers of feasible and infeasible waypoints. Specifically, we determine the safety value of a waypoint based on the proportion of known feasible waypoints in the next available position for that waypoint. The safety value from waypoint μ to ν is calculated as $\kappa_{\mu\nu} = \frac{N_f - N_{\mu\nu}}{N_f}$, where N_f denotes the total number of waypoints in a preset constant range $[0, R_f]$, and $N_{\mu\nu}$ denotes the number of infeasible waypoints in the range of N_f over direction from μ to ν. Safety values are updated using a similar rule to pheromones, and when choosing its next move, the ant would add the safety values to the pheromone levels for all possible actions. We will show in the results section that this method can reduce the running time and improve the operation efficiency compare to the classical ACS algorithm.

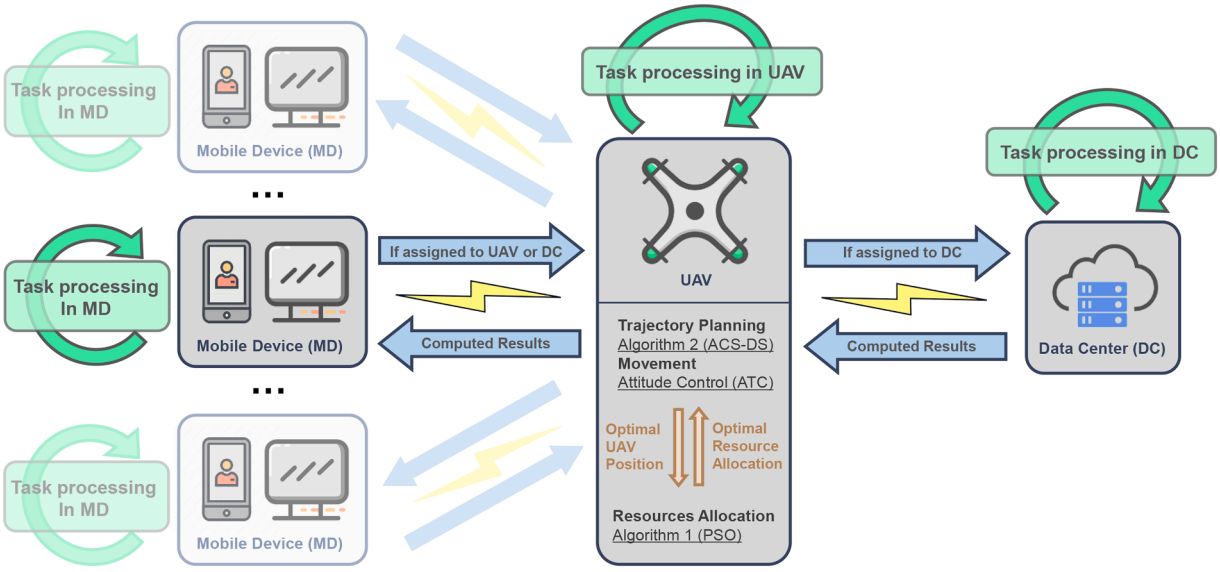


Fig. 6. A flowchart demonstration of the components in our proposed framework.

Algorithm 1 Task assignment, power and frequency allocations based on particle swarm optimization.

Input: The length of an interval t, $p_j^{\max}(t)$; $p_j^{\min}(t)$; $f_j^{\max}(t)$; $f_j^{\min}(t)$; F_{A1}; F_{A2}; F_I

Output: $\mathbb{S}^*(t)$; $\mathbf{o}_{ij}^*$; $p_j^*(t)$; $f_j^*(t)$

for $h = 1$ to H **do**
  Initialize particles' positions G_h
  Initialize particles' velocities V_h
  Initialize $pBest_h \longleftarrow G_h$
end for
Initialize $gBest(0) \longleftarrow \text{argmin} fit(pBest_h)$, where fit represent the equation to compute $\mathbb{S}^*(t)$; $k = 1$
for $h = 1$ to H **do**
  Update V_h and G_h by acceleration factors F_{A1}, F_{A2} and F_I
  if $fit(G_h) < fit(pBest_h)$ **then**
    $pBest_h \longleftarrow G_h$
    if $fit(pBest_h) < fit(gBest)$ **then**
      $gBest(k) \longleftarrow pBest_h$
    end if
  end if
  $k \longleftarrow (k+1)$
end for
while $|gBest(k+1) - gBest(k)| < \eth$ **do**
  $\mathbb{S}^*(t) \longleftarrow fit(gBest)$
  $(\mathbf{o}_{ij}^*; p_j^*(t); f_j^*(t)) \longleftarrow gBest$
end while
Output $\mathbb{S}^*(t)$; $(\mathbf{o}_{ij}^*; p_j^*(t); f_j^*(t))$

5.3.2. Decoupling

In ACS, when an ant reaches a point with no further viable options, it becomes trapped in a deadlock. To address this issue, we introduce a mechanism that allows the ant to escape the deadlock. Specifically, when any of following rule is satisfied during the ant's movement, the Decoupling mechanism will be triggered to perform a backtracking behavior with depth (step size) D_b to find alternative directions:

- An ant repeats a closed cycle consisting of two or more waypoints over multiple consecutive timeslots.
- A sudden drop in the amount of pheromone and safety values occurs, as indicated by the fact that the κ of the currently selected waypoint is less than half that of the previous waypoint.
- An ant falls into a local optimum and undetected in the early stage, that is, the ant has not triggered backtracking behavior for more than 25 consecutive waypoints in the first one-third of all iterations.

It is worth noting that the third rule is only considered in the early stages of the algorithm (first one-third of all iterations) to avoid excessive backtracking that reduces the convergence speed of the algorithm. This mechanism, by dynamically adjusting the pheromone levels and allowing the ant to backtrack and explore new possibilities, ensures that the ACS-DS remains robust and capable of finding optimal trajectories even in complex and challenging scenarios.

5.4. ACS-DS: Detailed steps

We let m be the total number of ants in the colony, and $\sigma = V_P + \kappa$ be the guidance factor. We further denote $\sigma_{\mu\nu}(h)$ as the sum of pheromone values and safety values between the neighboring waypoints μ and ν in the hth iteration. The initial pheromones on each edge are equal, namely $\sigma_{\mu\nu}(0) = C$ for all μ and ν. We denote by s^* the global best path obtained by the algorithm, corresponding to the minimal-cost trajectory in our UAV optimization problem. Next, for each ant $k \in \{1, 2 \dots, m\}$ in the colony, we initialize the pheromone V_{P0}, the safety value κ_0, and the heuristic value η. We also define the evaporation rate $\rho \in (0, 1)$ representing the degree to which the guidance factor $\sigma_{\mu\nu}(h)$ decays with iterations. Finally, we define $\mathcal{S}_\iota$ as the set of waypoints that ant ι can pass next, and $p_{\mu\nu}^\iota(h)$ as the probability that ant ι moves from position μ to ν in the hth iteration. The exact value of $p_{\mu\nu}^\iota(h)$ is jointly determined by the guidance factor and heuristic value at the waypoint, as in the following equation,

$$p_{\mu\nu}^\iota(h) = \begin{cases} \dfrac{\sigma_{\mu\nu}^\alpha(h)\eta_{\mu\nu}{}^\beta(h)}{\sum_{r\in\mathcal{S}_\iota}\sigma_{\mu r}^\alpha(h)\eta_{\mu r}{}^\beta(h)} & \nu \in \mathcal{S}_\iota, \\ 0 & \text{otherwise.} \end{cases} \tag{24}$$

Detailed steps of ACS-DS are presented in Algorithm 2.

5.5. ACS-DS: Proof of convergence

We now prove that Algorithm 2 will eventually converge, starting by the following proposition.

Proposition 1. *In Algorithm 2, for the guidance factor $\sigma_{\mu\nu}$ on any edge (μ, ν) generated by the ants during the searching process, there exists a maximum value $g(s^*)$ as $h \to \infty$.*

Algorithm 2 ACS-DS: ACS-based trajectory planning with decoupling and safety values mechanisms.

Input: Positions of all MDs (x_j, y_j, z_j);ρ; η; V_{P0}; κ_0; terrain surface with no-fly zone $(x_n^o, y_n^o, z_n^o) \in \boldsymbol{O}$
Output: $E^{\mathrm{MOV}*}(t)$; Trajectory of the UAV $(x_\iota(t), y_\iota(t), z_\iota(t))$

for $h = 1$ to H **do**
  Randomly set the initial coordinate of the UAV, $(x(0), y(0), z(0))$
  while $(x(0), y(0), z(0)) \notin \boldsymbol{O}$ **do**
    for each edge **do**
      Set initial pheromone, and calculate the initial safety values
    end for
    for each ant ι **do**
      $(x_\iota(h), y_\iota(h), z_\iota(h)) = (x(0), y(0), z(0))$
      for each edge (μ, ν) **do**
        Choose the next coordinate with probability $p^\iota_{\mu\nu}(h)$ by σ and η
        while $(x_\iota(h+1), y_\iota(h+1), z_\iota(h+1)) \notin \boldsymbol{O}$ **do**
          Output $(x_\iota(h+1), y_\iota(h+1), z_\iota(h+1))$
        end while
      end for
      Compute and output the length $\sum_{t=1}^{T} d_u(t)$ of the path by the ιth ant and E_u
      for each edge (μ, ν) **do**
        Update $V_{P_{\mu\nu}}$ and $\kappa_{\mu\nu}$ by ρ
        Update $\sigma_{\mu\nu}(h)$ by $V_{P_{\mu\nu}}$ and $\kappa_{\mu\nu}$
      end for
    end for
  end while
end for
Compute and output $E^{\mathrm{MOV}*}(t)$ by (10).

Proof. For ACS-DS, the local update of the guidance factor for edge (μ, ν) after the completion of each round of iteration can be represented as $\sigma_{\mu\nu}(h+1) = (1-\varphi)\cdot\sigma_{\mu\nu}(h) + \varphi\cdot\Delta\sigma_{\mu\nu}(h)$, while the global guidance factor is updated in a similar way, that is, $\sigma(h+1) = (1-\varphi)\cdot\sigma(h) + \varphi\cdot\Delta\sigma(h)$. Here, $\Delta\sigma_{\mu\nu}(h) = \sigma_{\mu\nu}(h) - \sigma_{\mu\nu}(h-1)$ is the increment of guidance factor on edge (μ, ν) at the hth iteration.

$$
\begin{aligned}
&\sigma_{\mu\nu}(1) = (1-\varphi)\cdot\sigma_{\mu\nu}(0) + \varphi\cdot\Delta\sigma_{\mu\nu}(0)\\
&\cdots\\
&\sigma_{\mu\nu}(h) = (1-\varphi)^h\cdot\sigma_{\mu\nu}(0) + (1-\varphi)^{h-1}\cdot\varphi\cdot\Delta\sigma_{\mu\nu}(1) + \cdots\\
&\qquad + (1-\varphi)\cdot\varphi\cdot\Delta\sigma_{\mu\nu}(h-1) + \varphi\cdot\Delta\sigma_{\mu\nu}(h)\\
&\qquad = \sum_{q=1}^{h}\left((1-\varphi)^{h-q}\cdot\varphi\cdot\Delta\sigma_{\mu\nu}(q) + (1-\varphi)^h\cdot\sigma_{\mu\nu}(0)\right)\\
&\lim_{h\to\infty}\sigma^{\max}_{\mu\nu}(h) = \lim_{h\to\infty}\left(\sum_{q=1}^{h}\left((1-\varphi)^{h-q}\cdot\varphi\cdot\Delta\sigma_{\mu\nu}(q) + \cdots\right)\right)\\
&\qquad = g(s^*) < \infty.
\end{aligned}
\tag{25}
$$

Thus, the guidance factor on each edge is bounded from the above by $g(s^*)$. □

After the first optimal solution is found, the guidance factor on the elements belonging to an optimal solution is guaranteed to be no less than that on other elements as we have a sufficient number of subsequent generations. That is, the guidance factor on any element not belonging to an optimal solution will keep decreasing until it is no larger than the guidance factor on the elements belonging to an optimal solution. In mathematical terms, we have the following corollary.

Corollary 1. *As $h \to \infty$, it is always true that $\sigma_{\mu\nu}(h) \geq \sigma_{\mu'\nu'}(h)$, if $(\mu, \nu) \in s^*$ and $(\mu', \nu') \notin s^*$.*

Proof. Suppose it takes h^* generations to find the first optimal solution. We assume that for a certain $(\mu, \nu) \in s^*$, there exists a $(\mu', \nu') \notin s^*$, $\sigma_{\mu\nu}(h^*) < \sigma_{\mu'\nu'}(h^*)$. Then, based on the guidance factor update rule in ACS-DS, by the $h^* + h'$ generation, the guidance factor on (μ', ν') can be derived as

$$
\sigma_{\mu'\nu'}(h^* + h') = \max\left\{\sigma_{\mu\nu}(h^*), (1-\rho)^{h'}\cdot\sigma_{\mu'\nu'}(h^*)\right\} \tag{26}
$$

Therefore,

$$
\begin{aligned}
\lim_{h\to\infty}\sigma_{\mu'\nu'}(h^* + h') &= \max\lim_{h\to\infty}\left\{\sigma_{\mu\nu}(h^*), \left[(1-\rho)^{h'}\cdot\sigma_{\mu'\nu'}(h^*)\right]\right\}\\
&= \max\left\{\sigma_{\mu\nu}(h^*), 0\right\}\\
&\leq \sigma_{\mu\nu}(h^*)
\end{aligned}
\tag{27}
$$

□

Proposition 1 and Corollary 1 jointly prove that, after a sufficient number of iterations, the guidance factor on an optimal path is bounded and no less than those on other paths. Noticeably, the backtracking behavior caused by the decoupling mechanism in ACS-DS will only increase the number of iterations for the current waypoint, but does not affect the updating of the guidance factor and the convergence to the optimal solution as in the original ACS. Therefore, the ACS-DS algorithm always converges to the global optimal solution.

5.6. ACS-DS: Complexity analysis

We now analyze the complexity of Algorithm 2 when a total of n users initiate computation requests in the current timeslot. Assume that in the worst case, Algorithm 2 triggers backtracking at every waypoint for each ant upfront, the path construction time complexity for m ants is $O(m\cdot n^2)$. The complexity of each update of the guidance factor, and each calculation of the transfer probability is $O(n^2)$. If the algorithm is finished after H iterations, the complexity of the entire algorithm is $O(H\cdot m\cdot n^2 + H\cdot n^2)$. Since the magnitude of m is similar to n, the complexity of the ACS-DS algorithm in completing all the tasks of the network is approximately $O(H\cdot n^3)$, which means that Algorithm 2 has a polynomial time complexity.

6. Performance evaluation

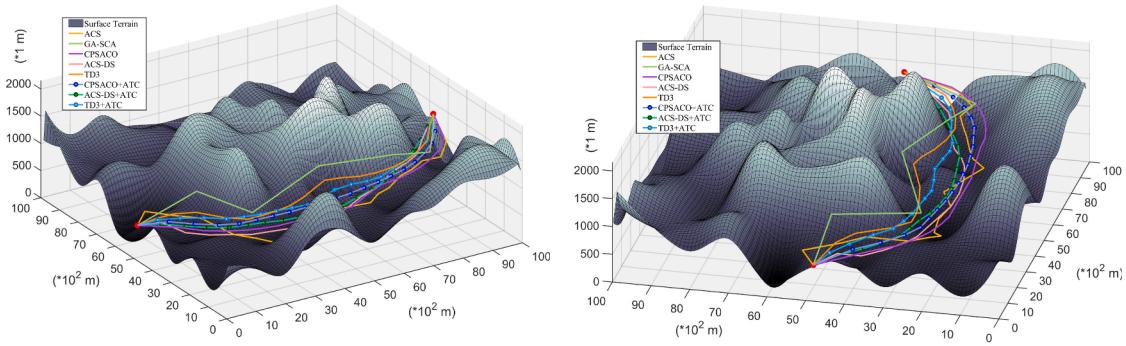
We now present numerical results to assess the efficacy and versatility of our proposed solutions.

6.1. Experiment setup

We consider a three-dimensional spatial domain with dimensions of $S\times S\times Z$. We randomly generate a continuous surface (no-fly zone) within a three-dimensional spatial domain as shown in Figs. 7a and 7b, and consider that K MDs are placed on the ground conforming to the terrain height, with their positions uniformly and randomly distributed in the space. All MDs are placed at the terrain surface height, and the UAV always operates at an altitude higher than the MD elevations. The values of key system parameters are listed in Table 5.

The UAV flight time T is obtained by cumulatively summing the propulsion energy consumed at each timeslot, where the energy usage is determined from the rotor-speed-dependent motor model already defined in Section 3.2. The endurance is reached once the accumulated consumption equals the battery capacity B_c, giving the maximum feasible T under a given trajectory and control strategy. Specifically, T is calculated as:

$$
T = \max\left\{t\in\mathbb{N} : \sum_{\tau=0}^{t} E^{\mathrm{UAV}}(\tau) \leq B_c\right\}, \tag{28}
$$



(a) Side view (b) Top view

Fig. 7. Trajectories planned by different mechanisms on a 3D spatial domain. The terminal point shown in Fig. 7 represents the UAV's final hovering location for sustained communication rather than the physical position of the MD.

Table 5
Values of key system parameters.

Parameter	Value	Parameter	Value
S	50,000 m	Z	3000 m
L	0.1 s	ϵ	[0.05, 1.00]
m	50 kg	a	5.28 m/s^2
g	9.81 m/s^2	l	1150 mm
K	50	R	512 mm
N_c	40	P_t	7.5 mm
N_B	2	P_w	250 mm
B_c	5000 Wh	I_x	5.5
$v^{\max}$	15 m/s	I_y	5.5
$z^{\max}$	2500 m	I_z	10.0
$\omega^{\min}$	200 rpm	$\omega^{\max}$	5000 rpm
C_T	1.483 $(\mathrm{N}\cdot\mathrm{m}\cdot\mathrm{min}^2)/\mathrm{r}^2$	C_M	2.925 $(\mathrm{N}\cdot\mathrm{m}\cdot\mathrm{min}^2)/\mathrm{r}^2$
M_g	29 g/mol	R_g	8.314 J/(mol $*$ K)
T	20-35 min	N_j	1-5
$f^{\min}$	1.0 GHz	$f^{\max}$	3.0 GHz
$f_j^{\min}$	0.5 GHz	$f_j^{\max}$	2.0 GHz
δ_u	1.2×10^{-28} J/(cycle^2·bit)	δ_j	3.0×10^{-28} J/(cycle^2·bit)
$\bar{Q}$	5×10^7 (bit-equivalent)	τ	1.0 s

Table 6
Values of key algorithmic parameters.

Parameter	Value	Parameter	Value	Parameter	Value
ρ	0.25	V_{P0}	3.8	η	2.5
F_{A1}, F_{A2}	2.0	F_I	0.65	R_f, D_b	200 m

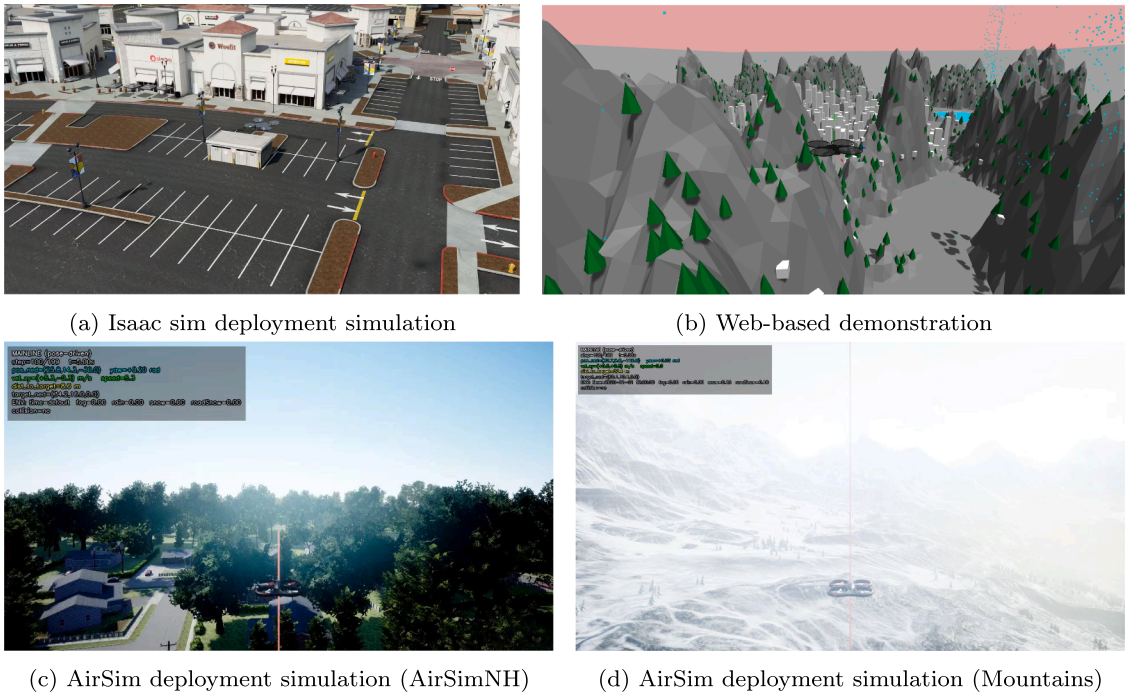
where $E^{\mathrm{UAV}}(\tau)$ is defined in (20) and includes both movement energy $E^{\mathrm{MOV}}(\tau)$ defined in (10)) and other energy consumption components. Values of key parameters used in the algorithms described earlier in this paper are listed in Table 6. All experimental results presented in this section are based on the average of the 50 independent runs with the weighting factor ϵ randomly generated within its domain for each run.

For the task assignment and resource allocation part, we adopt PSO (Algorithm 1), which has been demonstrated effective and robust in our earlier conference paper [10], for all experiments in this section. For the attitude control and trajectory planning modules, we focus on evaluating the performance of our proposed ACS-DS (Algorithm 2) and ATC (Section 5.1). Note that our work addresses a joint UAV control–communication–computation optimization problem, which fundamentally differs from traditional networking-only scheduling tasks. Conventional networking baselines such as Round Robin, Proportional Fair, or queue-based schedulers cannot operate in our setting, as they assume fixed infrastructure and do not interact with UAV flight dynamics, propulsion energy, or real-time orientation-dependent channel variations. These classical methods also do not model UAV physical constraints, motor-level energy consumption, computation frequency control, or end-to-end latency decomposition, all of which are essential components of our optimization problem. Therefore, consistent with the standard practice in the UAV-assisted MEC and cross-layer optimization literature, we adopt representative algorithmic families that can jointly optimize continuous control variables and discrete resource allocation decisions. Specifically, we examine the following ten implementations, including our proposed approach, recently proposed heuristic (e.g., [12,20]), and RL-based methods [11], and compare their performances in terms of overall operational efficiency cost.

- **ACS**: The trajectory of the UAV is planned using the classical Ant Colony System (ACS). Task assignment, processing frequencies, transmission powers, and channel allocation follow the optimization procedures described in Section 3. No special mechanisms are applied for attitude control.
- **GA-SCA**: The UAV trajectory is optimized using a hybrid method combining Genetic Algorithm (GA) and Successive Convex Approximation (SCA), as proposed in [12]. The remaining parameters follow the same strategy as ACS.
- **CPS-ACO**: The UAV trajectory is planned using the Chaotic-Polarized-Simulated Ant Colony Optimization (CPS-ACO) method, proposed in [20]. Other system parameters are optimized as in ACS.
- **TD3**: The UAV trajectory is planned using the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm, as described in [11,36]. Other configurations are consistent with the baseline setup in ACS.
- **ACS-DS**: This is an enhanced version of ACS, where UAV trajectory planning incorporates our proposed decoupling and safety value mechanisms, detailed in Algorithm 2. Other parameter settings remain unchanged.
- **CPS-ACO + ATC**: Based on CPS-ACO, this method integrates the attitude control mechanism described in Section 5.1 to enable more stable UAV dynamics.
- **TD3 + ATC**: Based on TD3, this variant incorporates the attitude control mechanism described in Section 5.1 to improve stability and responsiveness.
- **ACS-DS + ATC**: This is the full version of our proposed method, which combines the decoupled safe trajectory planning in ACS-DS with the attitude control mechanism in Section 5.1.

6.2. Simulation environments

The experimental evaluation of the proposed framework is conducted using three complementary simulation environments, each designed to address different aspects of system validation and serving distinct roles in the evaluation pipeline.



(a) Isaac sim deployment simulation (b) Web-based demonstration

(c) AirSim deployment simulation (AirSimNH) (d) AirSim deployment simulation (Mountains)

Fig. 8. Simulation environments used for framework evaluation.

- **Isaac Sim**: This Omniverse/PhysX-based robotics simulation platform provides high-fidelity rigid-body UAV dynamics that accurately model inertia, thrust, and torque characteristics. As shown in Fig. 8a, the environment incorporates realistic environmental disturbances including wind fields, actuator latency, and platform vibration, enabling high-fidelity physical verification of control algorithms. We utilize Isaac Sim to validate the FEAR-PID controller's stability under realistic dynamics and to verify that the UAV parameters specified in Table 5 exhibit physically correct behavior. We additionally leverage Isaac Sim's built-in/official USD environment assets as realistic scene data for representative outdoor scenarios, so that the controller is also inspected under non-trivial geometry and terrain context beyond synthetic setups. This physics-based validation ensures that our control strategies and parameter selections are feasible for real-world deployment.
- **Microsoft AirSim**: This platform offers photorealistic rendering powered by the Unreal Engine, featuring realistic suburban and mountainous environments that represent the large-scale outdoor deployment scenarios described in our system model (e.g., AirSimNH and LandscapeMountains scenes). As shown in Fig. 8c and Fig. 8d, the environment provides high-fidelity visual representation of complex terrain with realistic lighting and atmospheric effects. The platform provides comprehensive sensor emulation including RGB cameras, depth sensors, semantic segmentation, IMU, GPS, and realistic noise models, and also exposes environment controls such as time-of-day and weather. We employ AirSim to support interactive mission execution with the built-in multirotor dynamics under deployment-scale scenes and configurable conditions, and to generate synchronized FPV/chase-view visualizations for representative runs via ExternalPhysicsEngine (pose-driven) replay, which facilitates consistent presentation and debugging across different methods.
- **Lightweight web-based demonstration**[1]: This accessible interactive visualization tool is inspired by AirSim's visual style. As illustrated in Fig. 8b, the simplified environment employs streamlined UAV dynamics without full rigid-body physics, yet incorporates dynamic wind effects, simplified control loops, and procedural terrain generation. The web demo serves to qualitatively visualize UAV behavior, trajectory evolution, and controller stability, offering practitioners and researchers an intuitive understanding of the system's operational characteristics without requiring specialized simulation software.

These three environments complement each other: Isaac Sim validates physical feasibility and control robustness with high-fidelity dynamics and asset-based scenes, AirSim evaluates deployment-scale performance in interactive Unreal Engine environments, and the web demonstration provides intuitive qualitative insights. Together, they form a comprehensive evaluation framework that spans from low-level control validation to high-level mission assessment.

6.3. Module-wise ablation study

To evaluate the individual contribution of each proposed component in our holistic framework, we conduct an ablation study by selectively removing one module at a time while keeping the others unchanged. This experimental design allows us to quantify the performance degradation caused by the absence of specific functionalities and to demonstrate the necessity of integrating all components for optimal system performance. We consider the following four key modules:

- **FEAR-PID-based Attitude Control (ATC)**: Enhances UAV stability and reduces energy overhead during flight maneuvers;
- **ACS-DS Trajectory Planner**: Provides obstacle-aware and energy-efficient path planning with fast convergence. Determining the operational efficiency costs of the movement;
- **PSO-based Task and Resource Assignment**: Allocates task execution and communication resources effectively. Determining the operational efficiency costs of tasks computation and unloading.;
- **Gamma-based Channel Allocation Strategy**: Prevents bandwidth monopolization by larger tasks. Determining the operational efficiency costs of transmission.

[1] https://shuaijun-liu.github.io/UAV-Assisted-Fog-Computing-Simulation-Demo

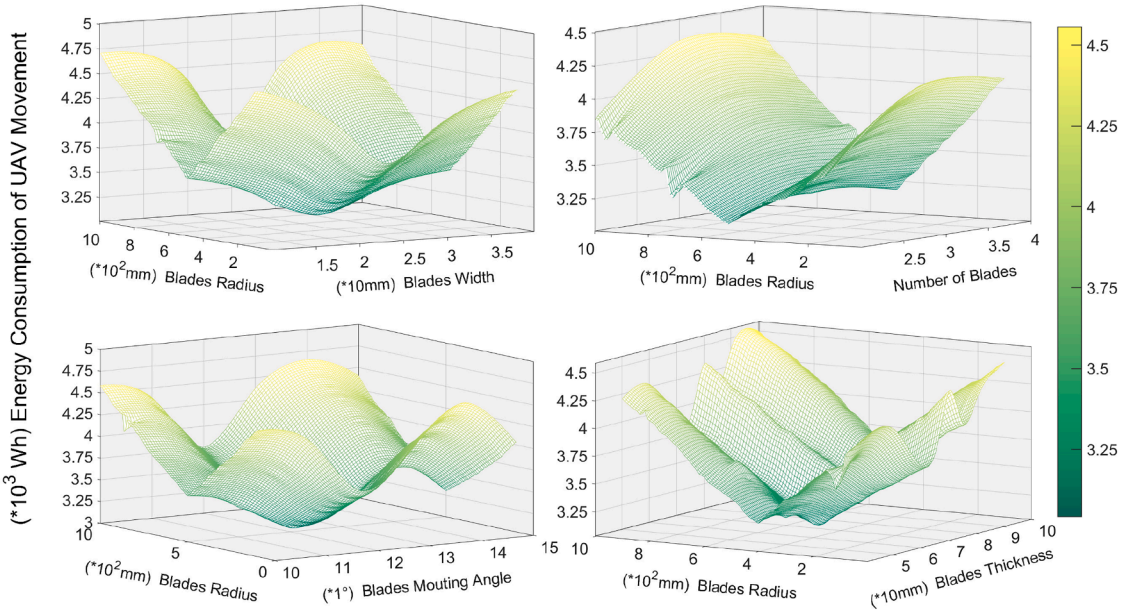


Fig. 9. Overall energy consumption vs. propeller blades parameters.

Table 7
Ablation variants with specific module removals.

Variant Name	Description
Full	All modules enabled (ATC + ACS-DS + PSO + Gamma)
w/o ATC	Replaces FEAR-PID with classical PID control
w/o ACS-DS	Replaces ACS-DS with TD3 trajectory planner
w/o PSO	Uses random task assignment and uniform power allocation
w/o Gamma	Uses uniform channel assignment ($P_j(t) = 1/K$)
Baseline	Classical PID + TD3 trajectory + uniform task/power/channel assignment

Each of these modules is removed individually to create a controlled ablation variant of the full framework. The tested ablation variants are summarized in Table 7.

For each variant, we use the same network environment and task configuration as in the full model experiments. All experiments are repeated 50 times with randomized task generation to ensure statistical robustness. The evaluation metric is the **overall operational efficiency cost**, defined as the sum of total task delay and energy consumption, normalized across scenarios.

6.4. Numerical results

6.4.1. Optimal propeller parameters

Fig. 9 shows the relationship between the overall energy consumption and propeller blade parameters including the number of blades, and the radius, width, mounting angle, and thickness of each blade, given that all other parameters are fixed. From the results, we can infer that, for a symmetric quadrotor UAV with a total mass of 80 kg, four rotors and a maximum acceleration up to 5.28 m/s^2, it is optimal to equip 2 propeller blades with a radius of 500 mm, a width of 250 mm and a thickness of 7.5 mm. The average energy consumption under configurations with optimal values of parameters can be reduced by more than 30%, compared to the average amount of 50 sets of random parameters generated uniformly within the respective allowable range of each parameter.

6.4.2. Convergence performance

Fig. 10 demonstrates the average convergence speeds of GA-SCA, CPS-ACO, and ACS-DS over 60 iterations, and TD3 algorithm over 3000 episodes for 200 independent runs. The results show that both ACS-DS and TD3 can optimize the energy efficiency through efficient exploring optimal trajectories. However, the average training time per episode for the RL-based TD3 are significantly higher than the average running time per iteration of the other three heuristic algorithms. Therefore, due to its faster convergence and lower computational overhead, ACS-DS is more suitable for rapid decision-making and trajectory planning in unknown environments.

6.4.3. Trajectory planning with attitude control

From the trajectories shown in Figs. 7a and 7b, we observe that, the CPS-ACO, TD3, and ACS-DS algorithms are more capable of avoiding no-fly zones. In addition, our proposed Attitude Control (ATC) method can improve the trajectories planned by all algorithms, by effectively correcting unwanted directional changes. Specifically, the FEAR-PID controller dynamically optimizes flight attitude by adjusting the UAV rotor speed in real-time based on sensor feedback, including error and environmental data, resulting in more stable and precise flight control.

6.4.4. Operational efficiency cost

We present the results of the operational efficiency cost (considering both delay and energy consumption) achieved by different implementations in Fig. 11. Note that the horizontal axis in this figure (and similar plots throughout this section) represents discrete control time slots, each of duration $L = 0.1$ s as defined in Section 3, rather than the UAV's actual physical flight time. The overall mission duration T spans 20 to 35 minutes in our experiments, calculated as described in Section 6.1 below Table 5), and the figures display representative segments of the trajectory for clarity of visualization. One important observation is that, the decoupling mechanism and safety values in ACS-DS can significantly reduce the overall consumption compared to the classical ACS, and have a slight advantage over SOTA (state-of-the-art) reinforcement learning methods such as TD3.

Moreover, the ATC mechanism further enhances the performance of both RL-based (TD3) and heuristic-based (ACS-DS and CPS-ACO) methods. Overall, ACS-DS + ATC achieves the best performance in terms of operational efficiency, closely followed by TD3 + ATC. Specifically, compared with conventional ACS, TD3 + ATC reduces the operational efficiency cost by 43.5%, while ACS-DS + ATC reduces it by 48.1%.

These results validate our initial claim in this paper, that jointly optimizing attitude control, trajectory planning, and task assignment

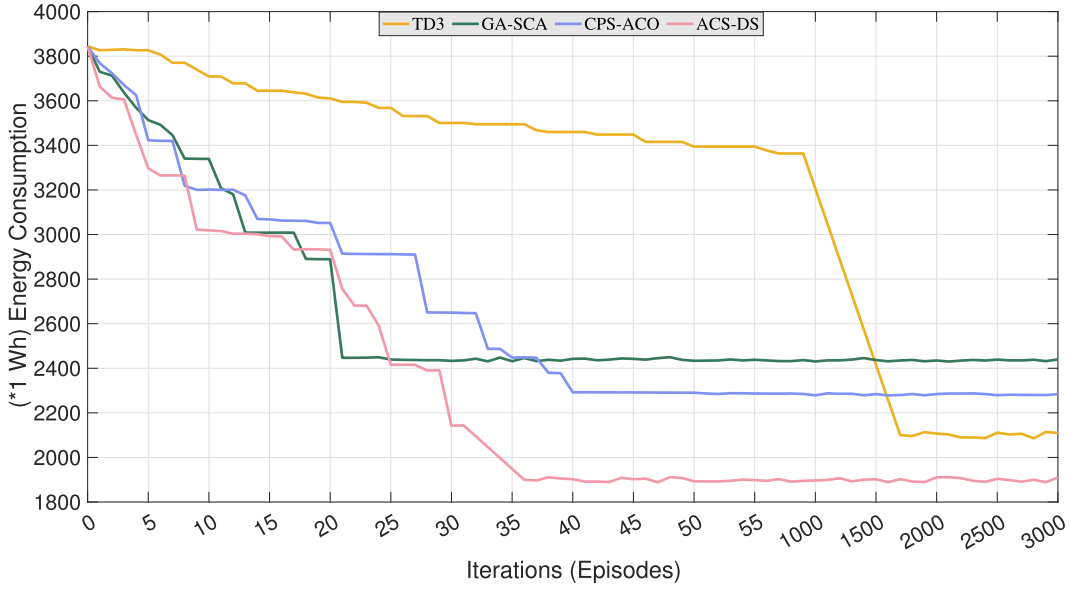


Fig. 10. Convergence performances of trajectory planning algorithms.

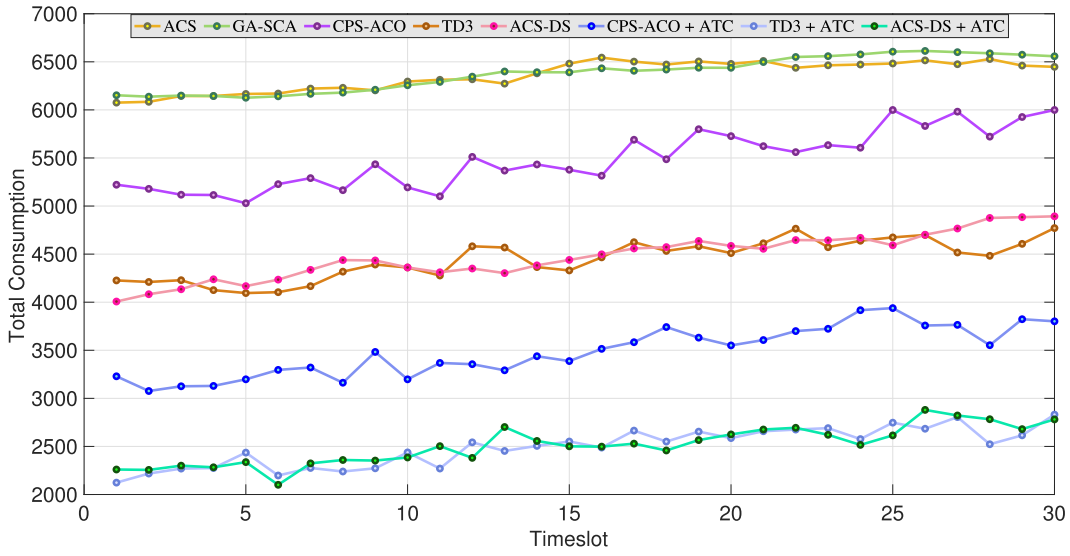


Fig. 11. Comparison of operational efficiency cost.

in UAV-assisted fog computing systems effectively captures the interdependencies among these modules. Such integrated approaches (e.g., TD3 + ATC and our proposed ACS-DS + ATC) lead to significant improvements in system performance compared to optimizing each aspect individually.

6.4.5. *Trajectory stability and robustness evaluation*

To evaluate the stability and robustness of different control and planning algorithms in complex 3D environments, we conducted 50 independent runs for each method under identical initial conditions and environmental settings. The trajectories were generated by our Framework, validated in Isaac Sim and AirSim, and subsequently rendered in the WebGL visualization environment to provide a clearer and lightweight presentation. In all figures, each polyline corresponds to one complete trajectory obtained from a single run.

Fig. 12 compares the tracking performance of the classical PID controller (red) and the proposed FEAR-PID controller (blue) along the same reference path. The trajectories generated by PID exhibit noticeable drift and oscillation, especially in densely cluttered urban regions. In contrast, FEAR-PID produces a significantly more compact trajectory cluster with reduced lateral deviation, demonstrating superior attitude stability and higher run-to-run consistency.

Fig. 13 presents the trajectory distributions of three planning algorithms: ACS (yellow), TD3 (green), and ACS-DS (blue). ACS exhibits the largest dispersion, indicating higher sensitivity to environmental variations. TD3 shows moderate improvement in stability, while ACS-DS produces the most concentrated trajectories among the three, confirming that its decoupling mechanism and dynamic safety factors substantially enhance planning robustness across repeated trials.

Overall, these results show that,

1. FEAR-PID effectively reduces disturbance-induced drift and improves attitude control precision;
2. ACS-DS generates more stable and reliable flight paths under repeated execution;
3. Jointly optimizing control and planning is essential for ensuring dependable UAV operation in realistic 3D environments.

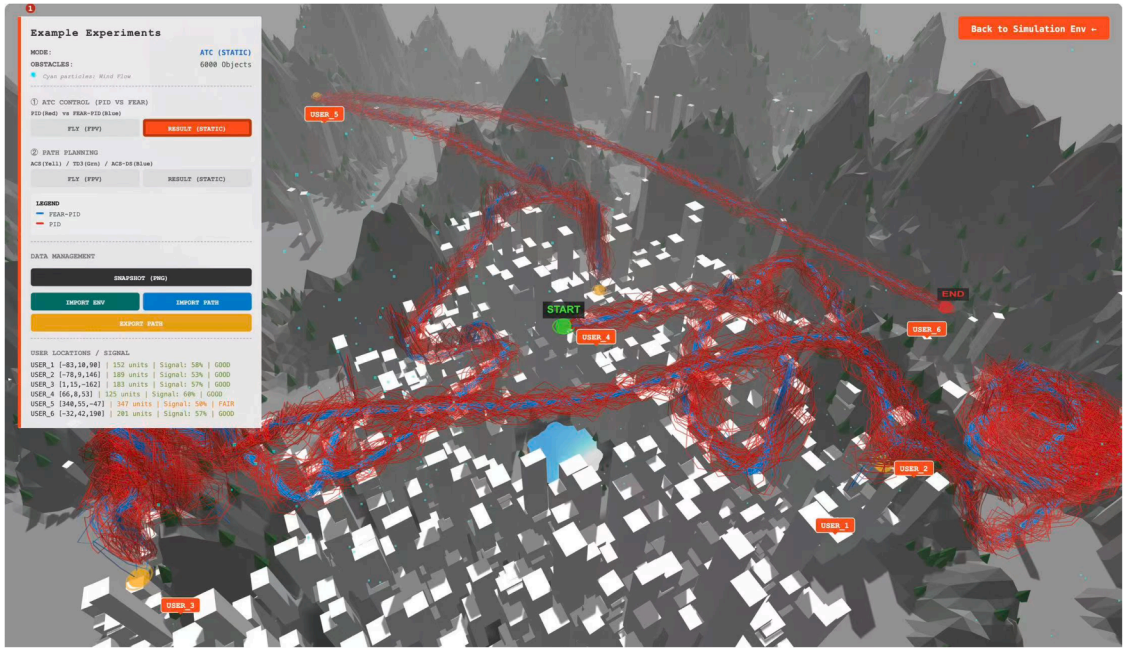


Fig. 12. Trajectory comparison between PID (red) and FEAR-PID (blue). Visualization of 50 independent runs for evaluating trajectory stability and robustness.

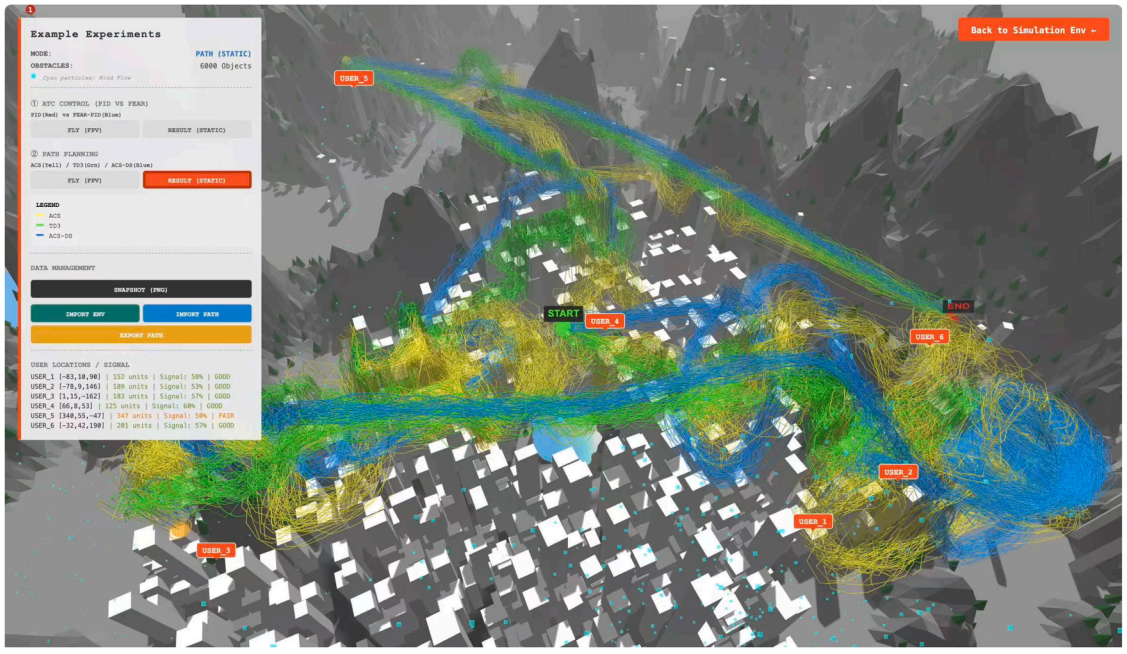


Fig. 13. Trajectory comparison of ACS (yellow), TD3 (green), and ACS-DS (blue). Visualization of 50 independent runs for evaluating trajectory stability and robustness.

6.4.6. Ablation study results

We summarize the ablation results in Fig. 14, which report the average operational efficiency cost and its variance across configurations. The full model consistently achieves the lowest cost. Removing the ACS-DS trajectory planner causes the sharpest degradation, reflecting its importance in balancing energy and delay. The absence of PSO-based task assignment also leads to significant overhead due to poor task-resource coordination.

Interestingly, even the removal of the FEAR-PID attitude controller, while seemingly less impactful than trajectory and assignment modules, causes noticeable inefficiencies, particularly in maneuvering-intensive segments such as ascent, descent, and turns. This confirms that flight stability indirectly influences task execution and communication quality. The gamma-based channel allocation strategy, though relatively lightweight, also contributes to system-wide efficiency by avoiding bandwidth contention and ensuring smoother transmission.

Overall, the results confirm that all modules are complementary. The holistic approach offers not only the best mean performance but also the most stable behavior, reinforcing the need for joint optimization in UAV-assisted fog computing.

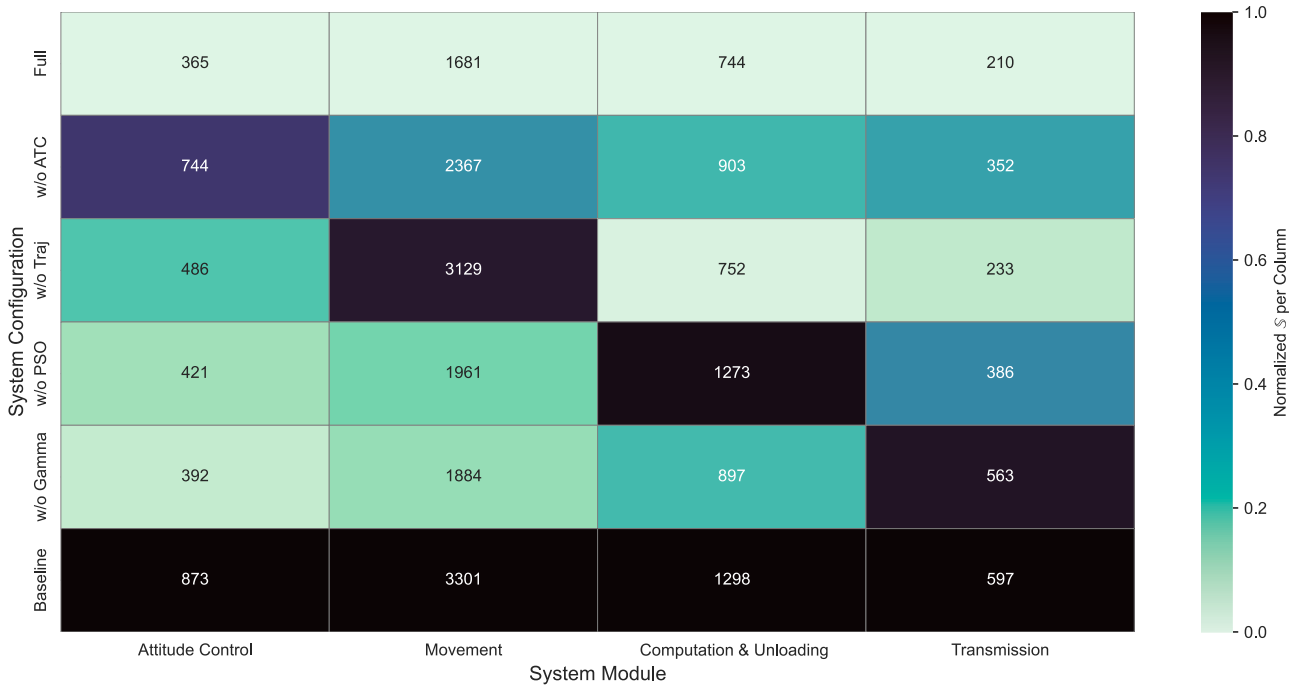


Fig. 14. Average operational efficiency cost across ablation variants.

7. Conclusions

In this paper, we have proposed a joint optimization framework to reduce the operational efficiency cost in UAV-assisted fog computing systems. Our proposed framework involves multiple modules, including quadrotor UAV attitude control, trajectory planning in a three-dimensional spatial domain with continuously varying terrain heights, and energy-efficient assignment of computing tasks to different components in the network. We have designed appropriate mechanisms or algorithms for each module in the framework, and integrated them together to obtain a holistic solution to improve the overall efficiency in UAV-assisted fog computing. Specifically, we have proposed a novel FEAR-PID control mechanism for effective attitude control, designed the ACS-DS algorithm that overcomes the convergence issue in conventional approaches for trajectory planning in three-dimensional domains, and a modified PSO algorithm to determine the optimal task assignment. Numerical results from a wide range of experiments have shown that our proposed framework can reduce the operational efficiency cost significantly compared to existing approaches.

CRediT authorship contribution statement

Shuaijun Liu: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization; **Jinqiu Du:** Writing – review & editing, Visualization, Investigation, Formal analysis, Data curation; **Yaxin Zheng:** Writing – review & editing, Visualization, Methodology, Data curation; **Jiaying Yin:** Writing – review & editing, Software, Formal analysis, Conceptualization; **Yuhui Deng:** Supervision; **Jingjin Wu:** Writing – review & editing, Resources, Project administration, Methodology, Funding acquisition, Conceptualization.

Data availability

No data was used for the research described in the article.

Declaration of competing interest

The authors declare no financial interests/personal relationships which may be considered as potential competing interests.

Funding

This work was partly supported the Guangdong Provincial Key Laboratory of Interdisciplinary Research and Application for Data Science, BNU-HKBU United International College (Project code 2022B1212010006), and Guangdong Higher Education Upgrading Plan (2021–2025) UIC R0400001-22 and R0400024-22.

Appendix A. The Dynamics of a quadrotor UAV

We consider two key components in the structure of a quadrotor UAV, namely the body frame and the earth frame. The body frame is commonly used for attitude control of the UAV, where the positive direction is the direction of ascent corresponding to the centers of the four motors.

We assume that the airframe has the following characteristics and constraints,

- The structure of the quadrotor UAV is symmetric;
- Friction between propeller and motor spindle is negligible;
- The stator magnetic field speed and rotor speed of the motors are infinitely close to each other, such that the slip rate is 0;
- The quadrotor and propeller structures are rigid, with a uniform mass distribution and the geometric center is the mass center;
- The Euler angles are bounded, i.e., $-\pi/2 < \phi < \pi/2, -\pi/2 < \theta < \pi/2, -\pi < \psi < \pi$.

Based on the above assumptions, we combine the dynamics principles to establish a rigid body model for attitude control of quadrotor UAV. It should be noted that all UAV coordinate system transformations are obtained by rotation synthesis with respect to the fixed coordinate system, the rotation matrix for the conversion of the body frame to the

earth frame with the following equation,

$$R_b^e = \left(R_e^b\right)^{-1} = \left(R_e^b\right)^T = \begin{bmatrix} \cos\theta\cos\psi & \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi \\ \cos\theta\sin\psi & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \tag{A.1}$$

Four inputs, including total thrust U_1, roll torque U_2, pitch torque U_3, and yaw torque U_4, can be controlled based on the rotation speeds of the four motors of the UAV, that is,

$$\begin{cases} U_1(t) = C_T(t)\left(\omega_1(t)^2 + \omega_2(t)^2 + \omega_3(t)^2 + \omega_4(t)^2\right) \\ U_2(t) = C_T(t)\left(-\omega_2(t)^2 + \omega_4(t)^2\right) \\ U_3(t) = C_T(t)\left(-\omega_1(t)^2 + \omega_3(t)^2\right) \\ U_4(t) = C_M(t)\left(-\omega_1(t)^2 + \omega_2(t)^2 - \omega_3(t)^2 + \omega_4(t)^2\right) \end{cases} \tag{A.2}$$

References

[1] K. Ashton, That 'internet of things' thing, RFiD J. 22 (7) (1999) 97–114.
[2] N. Hu, Z. Tian, X. Du, N. Guizani, Z. Zhu, Deep-Green: a dispersed energy-efficiency computing paradigm for green industrial IoT, IEEE Trans. Green Commun. Netw. 5 (2) (2021) 750–764. https://doi.org/10.1109/TGCN.2021.3064683
[3] J. Pei, H. Chen, L. Shu, UAV-assisted connectivity enhancement algorithms for multiple isolated sensor networks in agricultural internet of things, Comp. Netw. 207 (2022) 108854.
[4] O. Ghdiri, W. Jaafar, S. Alfattani, J.B. Abderrazak, H. Yanikomeroglu, Offline and online UAV-enabled data collection in time-constrained IoT networks, IEEE Trans. Green Commun. Netw. 5 (4) (2021) 1918–1933. https://doi.org/10.1109/TGCN.2021.3104801
[5] P. Poksawat, L. Wang, A. Mohamed, Gain scheduled attitude control of fixed-wing UAV with automatic controller tuning, IEEE Trans. Control Syst. Technol. 26 (4) (2018) 1192–1203. https://doi.org/10.1109/TCST.2017.2709274
[6] E. Besada-Portas, L. de la Torre, J.M. de la Cruz, B. de Andrés-Toro, Evolutionary trajectory planner for multiple UAVs in realistic scenarios, IEEE Trans. Rob. 26 (4) (2010) 619–634. https://doi.org/10.1109/TRO.2010.2048610
[7] C. Rottondi, F. Malandrino, A. Bianco, C.F. Chiasserini, I. Stavrakakis, Scheduling of emergency tasks for multiservice UAVs in post-disaster scenarios, Comp. Netw. 184 (2021) 107644.
[8] Z. Cheng, M. Liwang, N. Chen, L. Huang, X. Du, M. Guizani, Deep reinforcement learning-based joint task and energy offloading in UAV-aided 6G intelligent edge networks, Comp. Commun. 192 (2022) 234–244.
[9] R. Zhou, X. Wu, H. Tan, R. Zhang, Two time-scale joint service caching and task offloading for UAV-assisted mobile edge computing, in: IEEE INFOCOM 2022 - IEEE Conference on Computer Communications, 2022, pp. 1189–1198.
[10] S. Liu, J. Yin, Z. Zeng, J. Wu, Optimal trajectory planning and task assignment for UAV-assisted fog computing, in: 2022 IEEE 24th International Conference on High Performance Computing & Communications (HPCC), 2022, pp. 1400–1407. https://doi.org/10.1109/HPCC-DSS-SmartCity-DependSys57074.2022.00217
[11] L. Tan, S. Guo, P. Zhou, Z. Kuang, S. Long, Z. Li, Multi-UAV-enabled collaborative edge computing: deployment, offloading and resource optimization, IEEE Trans. Intell. Transp. Syst. 25 (11) (2024) 18305–18320. https://doi.org/10.1109/TITS.2024.3432818
[12] J. Zheng, K. Liu, 3D UAV trajectory planning with obstacle avoidance for UAV-enabled time-constrained data collection systems, IEEE Trans. Veh. Technol. (2024) 1–14. https://doi.org/10.1109/TVT.2024.3419842
[13] K.C.U. Obias, M.F.Q. Say, E.A.V. Fernandez, A.Y. Chua, E. Sybingco, A study of the interaction of proportional-integral- derivative (PID) control in a quadcopter unmanned aerial vehicle (UAV) using design of experiment, in: 2019 IEEE 11th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM), 2019, pp. 1–4. https://doi.org/10.1109/HNICEM48295.2019.9072806
[14] B.E. Demir, R. Bayir, F. Duran, Real-time trajectory tracking of an unmanned aerial vehicle using a self-tuning fuzzy proportional integral derivative controller, Int. J. Micro Air Veh. 8 (4) (2016) 252–268. https://doi.org/10.1177/1756829316675882
[15] Z. Yu, Y. Li, M. Lv, B. Pei, A. Fu, Event-triggered adaptive fuzzy fault-tolerant attitude control for tailless flying-wing UAV with fixed-time convergence, IEEE Trans. Veh. Technol. 73 (4) (2024) 4858–4869. https://doi.org/10.1109/TVT.2023.3329470
[16] B. Xia, I. Mantegh, W.-F. Xie, Hybrid framework for UAV motion planning and obstacle avoidance: integrating deep reinforcement learning with fuzzy logic, in: 2024 10th International Conference on Control, Decision and Information Technologies (CoDIT), 2024, pp. 2662–2669. https://doi.org/10.1109/CoDIT62066.2024.10708494
[17] B.G. Shri Varu, N. Jayarajan, T. Ganesan, Optimizing quadcopter trajectory tracking with deep-Q reinforcement PID controller in uncertain environments, in: 2024 IEEE Third International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES), 2024, pp. 341–346. https://doi.org/10.1109/ICPEICES62430.2024.10719095
[18] L. Shen, N. Wang, D. Zhang, J. Chen, X. Mu, K.M. Wong, Energy-aware dynamic trajectory planning for UAV-enabled data collection in mmtc networks, IEEE Trans. Green Commun. Netw. 6 (4) (2022) 1957–1971. https://doi.org/10.1109/TGCN.2022.3186841
[19] M. Diallo, A. Quintero, S. Pierre, An efficient approach based on ant colony optimization and tabu search for a resource embedding across multiple cloud providers, IEEE Trans. Cloud Comp. 9 (3) (2019) 896–909.
[20] M. Yan, C.A. Chan, A.F. Gygax, C. Li, A. Nirmalathas, C.-L. I, Efficient generation of optimal UAV trajectories with uncertain obstacle avoidance in MEC networks, IEEE Int. Things J. (2024) 1–1. https://doi.org/10.1109/JIOT.2024.3446664
[21] Y. Tao, J. Qiu, S. Lai, A hybrid cloud and edge control strategy for demand responses using deep reinforcement learning and transfer learning, IEEE Trans. Cloud Comp. 10 (1) (2021) 56–71.
[22] S. Liu, J. Yin, J. Du, Y. Zheng, Y. Deng, J. Wu, Meteorological and topographical big data-driven UAV trajectory planning, in: 2024 34th International Telecommunication Networks and Applications Conference (ITNAC), 2024, pp. 1–6. https://doi.org/10.1109/ITNAC62915.2024.10815424
[23] M. Jain, V. Saihjpal, N. Singh, S.B. Singh, An overview of variants and advancements of PSO algorithm, Appl. Sci. 12 (17) (2022) 8392.
[24] J.-J. Shin, H. Bang, UAV path planning under dynamic threats using an improved PSO algorithm, Int. J. Aerosp. Eng. 2020 (2020) 1–17.
[25] M. Dorigo, G. Di Caro, L.M. Gambardella, Ant algorithms for discrete optimization, Artif. Life 5 (2) (1999) 137–172.
[26] K. Socha, ACO for continuous and mixed-variable optimization, in: International Workshop on Ant Colony Optimization and Swarm Intelligence, Springer, 2004, pp. 25–36.
[27] C. Wang, Y. Nan, S. Zhang, L. Jiang, Application of the adaptive double-layer ant colony algorithm in UAV trajectory planning, in: 2021 4th International Conference on Intelligent Autonomous Systems (ICoIAS), 2021, pp. 371–377. https://doi.org/10.1109/ICoIAS53694.2021.00073
[28] J. Cui, Y. Liu, A. Nallanathan, Multi-agent reinforcement learning-based resource allocation for UAV networks, IEEE Trans. Wireless Commun. 19 (2) (2020) 729–743. https://doi.org/10.1109/TWC.2019.2935201
[29] M. Li, N. Cheng, J. Gao, Y. Wang, L. Zhao, X. Shen, Energy-efficient UAV-assisted mobile edge computing: resource allocation and trajectory optimization, IEEE Trans. Veh. Technol. 69 (3) (2020) 3424–3438.
[30] G. Wu, H. Wang, H. Zhang, Y. Zhao, S. Yu, S. Shen, Computation offloading method using stochastic games for software-defined-network-based multiagent mobile edge computing, IEEE Int. Things J. 10 (20) (2023) 17620–17634. https://doi.org/10.1109/JIOT.2023.3277541
[31] G. Francesca, A. Santone, G. Vaglini, M.L. Villani, Ant colony optimization for deadlock detection in concurrent systems, in: 2011 IEEE 35th Annual Computer Software and Applications Conference, 2011, pp. 108–117. https://doi.org/10.1109/COMPSAC.2011.22
[32] W. Hou, Z. Xiong, C. Wang, H. Chen, Enhanced ant colony algorithm with communication mechanism for mobile robot path planning, Rob. Auton. Syst. 148 (2022) 103949. https://doi.org/10.1016/j.robot.2021.103949
[33] J. Liu, H. Weng, Y. Ge, S. Li, X. Cui, A self-healing routing strategy based on ant colony optimization for vehicular ad hoc networks, IEEE Int. Things J. 9 (22) (2022) 22695–22708. https://doi.org/10.1109/JIOT.2022.3181857
[34] P. Qin, Y. Fu, Y. Xie, K. Wu, X. Zhang, X. Zhao, Multi-agent learning-based optimal task offloading and UAV trajectory planning for AGIN-Power IoT, IEEE Trans. Commun. 71 (7) (2023) 4005–4017. https://doi.org/10.1109/TCOMM.2023.3274165
[35] X. Wei, L. Cai, N. Wei, P. Zou, J. Zhang, S. Subramaniam, Joint UAV trajectory planning, DAG task scheduling, and service function deployment based on DRL in UAV-Empowered edge computing, IEEE Int. Things J. 10 (14) (2023) 12826–12838. https://doi.org/10.1109/JIOT.2023.3257291
[36] C. Zheng, K. Pan, J. Dong, L. Chen, Q. Guo, S. Wu, H. Luo, X. Zhang, Multi-agent collaborative optimization of UAV trajectory and latency-aware DAG task offloading in UAV-assisted MEC, IEEE Access 12 (2024) 42521–42534. https://doi.org/10.1109/ACCESS.2024.3378512
[37] Y. Huang, R. Zhou, An online framework for joint UAV trajectory planning and intelligent dependent task offloading, in: 2023 20th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), 2023, pp. 258–266.
[38] C. Xu, P. Zhang, H. Yu, Y. Li, Dynamic blockchain-empowered trustworthy end-edge collaborative computing via rotating multi-agent DRL, IEEE Trans. Wireless Commun. 24 (6) (2025) 4864–4878.
[39] C. Xu, Z. Tang, H. Yu, P. Zeng, L. Kong, Digital twin-driven collaborative scheduling for heterogeneous task and edge-end resource via multi-agent deep reinforcement learning, IEEE J. Sel. Areas Commun. 41 (10) (2023) 3056–3069.
[40] W.H. Yew, C. Fat Chau, A.W. Mahmood Zuhdi, W. Syakirah Wan Abdullah, W.K. Yew, N. Amin, Investigating the performance of deep reinforcement learning-based MPPT algorithm under partial shading condition, in: 2023 IEEE Regional Symposium on Micro and Nanoelectronics (RSM), 2023, pp. 9–12. https://doi.org/10.1109/RSM59033.2023.10326748
[41] K. Dev, P.K.R. Maddikunta, T.R. Gadekallu, S. Bhattacharya, P. Hegde, S. Singh, Energy optimization for green communication in IoT using Harris hawks optimization, IEEE Trans. Green Commun. Netw. 6 (2) (2022) 685–694. https://doi.org/10.1109/TGCN.2022.3143991
[42] X. Wei, C. Tang, J. Fan, S. Subramaniam, Joint optimization of energy consumption and delay in cloud-to-thing continuum, IEEE Int. Things J. 6 (2) (2019) 2325–2337. https://doi.org/10.1109/JIOT.2019.2906287
[43] X. Huang, W. Luo, J. Liu, Attitude control of fixed-wing UAV based on DDQN, in: 2019 Chinese Automation Congress (CAC), 2019, pp. 4722–4726. https://doi.org/10.1109/CCDC49329.2020.9164051
[44] P. Biczyski, et al., Multirotor UAV design and performance modeling–a survey, Drones (2020).
[45] K. Zhu, R. Chen, L. Zhang, A survey on heavy-lift multirotor UAV systems: payload, propulsion, and endurance, Unmanned Syst. 9 (4) (2021) 185–198.

[46] J. Pollet, et al., Power loading and disk loading analysis for multirotors, Aerospace Sci. (2020).
[47] A. Karam, et al., Performance study of agricultural quadrotors under variable payload, Appl. Sci. (2024).
[48] M. Silva, et al., Heavy-lift quadrotor flight characterization and endurance modeling, J. Field Rob. (2024).
[49] M. Zukerman, Introduction to queueing theory and stochastic teletraffic models,(2013). arXiv preprint arXiv:1307.2968
[50] J. Li, J. Tang, Z. Liu, On the data freshness for industrial internet of things with mobile-edge computing, IEEE Int. Things J. 9 (15) (2022) 13542–13554.
[51] T.A. Khan, S.H. Ling, A.S. Mohan, Advanced particle swarm optimization algorithm with improved velocity update strategy, in: 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2018, pp. 3944–3949. https://doi.org/10.1109/SMC.2018.00669