Robotica (2025), **43**, pp. 2565–2609
doi:10.1017/S0263574725101914

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

Advancing UAV swarm autonomy with ARCog-NET for task allocation, path planning, and formation control

Gabryel Silva Ramos, Milena Faria Pinto and Diego Barreto Haddad

Federal Center for Technological Education Celso Suckow da Fonseca (CEFET-RJ), Rio de Janeiro, Brazil
Corresponding author: Gabryel Silva Ramos; Email: gabryelsr@gmail.com

Received: 21 March 2025; **Revised:** 19 May 2025; **Accepted:** 5 June 2025; **First published online:** 7 July 2025

Keywords: cognitive architecture; aerial robots; edge-fog-cloud technologies; swarm systems; distributed data processing; autonomy of aerial robots

Abstract

Most research on UAV swarm architectures remains confined to simulation-based studies, with limited real-world implementation and validation. In order to mitigate this issue, this research presents an improved task allocation and formation control system within ARCog-NET (Aerial Robot Cognitive Architecture), aimed at deploying autonomous UAV swarms as a unified and scalable solution. The proposed architecture integrates perception, planning, decision-making, and adaptive learning, enabling UAV swarms to dynamically adjust path planning, task allocation, and formation control in response to evolving mission demands. Inspired by artificial intelligence and cognitive science, ARCog-NET employs an Edge-Fog-Cloud (EFC) computing model, where edge UAVs handle real-time data acquisition and local processing, fog nodes coordinate intermediate control, and cloud servers manage complex computations, storage, and human supervision. This hierarchical structure balances real-time autonomy at the UAV level with high-level optimization and decision-making, creating a collective intelligence system that automatically fine-tunes decision parameters based on configurable triggers. To validate ARCog-NET, a realistic simulation framework was developed using SITL (Software-In-The-Loop) with actual flight controller firmware and ROS-based middleware, enabling high-fidelity emulation. This framework bridges the gap between virtual simulations and real-world deployments, allowing evaluation of performance in environmental monitoring, search and rescue, and emergency communication network deployment. Results demonstrate superior energy efficiency, adaptability, and operational effectiveness compared to conventional robotic swarm methodologies. By dynamically optimizing data processing based on task urgency, resource availability, and network conditions, ARCog-NET bridges the gap between theoretical swarm intelligence models and real-world UAV applications, paving the way for future large-scale deployments.

1. Introduction

Unmanned aerial vehicles (UAVs) are one of the most versatile robotic platforms, offering a broad range of applications [1]. These include surveillance [2], naval operation assistance [3], search and rescue [4], inspection [5], package delivery [6], environmental monitoring [7], topography [8], and many others. Note that the efficiency and capabilities of UAV systems are significantly enhanced when operating in swarm configurations. Coordinated UAVs can share tasks or execute parts of a larger mission individually, achieving greater efficiency or enabling operation beyond the capacity of a single robot [9–11].

Despite their potential, research and development of UAV swarms still face gaps. As reported by Abdelkader et al. [12], most existing aerial swarm systems are validated through simulations, with experimental implementations limited to small groups in controlled indoor settings. Besides, outdoor experiments with autonomous UAV groups are scarce, revealing scalability challenges in real-world environments. These challenges are primarily attributed to uncertainties in individual robot localization, relative positioning within the swarm, and high communication demands required for coordinated and safe operation.

Recent systematic reviews emphasize the potential of UAV swarms for different kinds of applications but highlight persistent obstacles such as scalability, energy management, security, communication reliability, and regulatory compliance [13]. Future research must focus on integrating artificial intelligence (AI) models, advanced communication systems, sustainable energy solutions, and robust ethical and regulatory frameworks. Additionally, the development of generic swarm implementations that are adaptable to various application requirements remains a critical priority [10, 14].

In this context, the present work aims to bridge these gaps by proposing a generalized architecture that addresses the core aspects of UAV swarm development: swarm control, coordination, collaboration, network communication and routing, path planning, collision avoidance, formation control, task assignment, intelligent decision-making, and autonomy [12, 13]. The Aerial Robot Cognitive Network Architecture (ARCog-NET) is a modular, reconfigurable, and programmable framework that utilizes a distributed computation structure and cognitive-based decision-making processes to create a flexible platform for UAV swarm application development. The ARCog-NET aims to replicate current UAV workflows within a single system, providing flexibility for future research while simplifying network infrastructure, decision-making, and simulations for developers and researchers. Additionally, this work distinguishes between three levels of experimentation: (i) pure simulation, performed entirely in synthetic environments such as Gazebo with simplified UAV and environment modeling; (ii) emulation, based on SITL using PX4 flight firmware and ROS middleware, allowing realistic UAV behavior testing under virtualized hardware constraints; and (iii) real-world modeling, where actual hardware constraints (e.g., power consumption, latency, bandwidth) are incorporated into the emulation. While no physical deployment was executed in this study, the use of SITL enables a high-fidelity evaluation of ARCog-NET as a crucial step toward real-world applicability.

1.1. Main contributions

This research introduces significant advancements in the ARCog-NET architecture, extending the capabilities of UAV swarm networks beyond previous implementations [14]. Designed as a cognitive control framework, ARCog-NET enhances autonomous decision-making in swarm robotics by leveraging an Edge-Fog-Cloud (EFC) model. The system follows a decentralized loop-based operation, where multiple UAVs function as intelligent nodes executing parallel tasks at different levels of the network hierarchy. This distributed processing approach improves operational coherence, enabling UAVs to dynamically adapt to mission requirements while maintaining collective efficiency. The architecture integrates key functionalities such as real-time data management, inter-UAV communication, intelligent task allocation, adaptive decision-making, and autonomous mission execution. By incorporating fog and cloud computing, it optimizes mission supervision, data analytics, and robotic coordination while ensuring scalability for large UAV formations. Communication efficiency within the network is the main focus, addressing critical parameters like latency, bandwidth utilization, interference mitigation, and power optimization to enhance overall swarm performance. The main contributions of this work can be summarized as follows:

- Improvements in ARCog-NET, a modular, reconfigurable, and programmable framework for UAV swarm applications, that integrates cognitive-based decision-making processes for flexible and adaptive swarm behavior. The first published study of ARCog-NET [14] only addressed cognitive path planning decision-making affecting navigation. The present work presents a generalization of the previous algorithm, adapting the cognitive distributed structure to implement a dynamic decision-making process. The decisions were categorized into several types, as network, data, and navigation tasks. Then navigation gained two special cognitive subroutines for path planning and formation control, improving the study presented before. This paper addresses the challenges in scalability, communication reliability, and energy efficiency through innovative swarm design and provides a generic framework adaptable to various UAV swarm applications.

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

- Implementation of distributed computing through the EFC computing model in order to optimize data processing and enable real-time decision-making across multiple network layers.
- Incorporation of artificial cognition for intelligent decision-making, supporting autonomy and dynamic adaptability through advanced knowledge representation and decision-making frameworks. This cognitive architecture represents a significant advancement in the field of UAV swarm networks by providing a collective algorithm designed specifically for cognitive UAV swarm networks operating within the EFC architecture. The architecture includes aspects such as data management, information exchange, cognitive decision-making, robot control actions, data processing, storage, and human interaction possibilities.
- ARCog-NET is integrated with Robotic Operating System (ROS), a widely used framework for developing robotic applications. This integration enables communication and interaction between ARCog-NET and ROS-based UAV systems, facilitating the implementation and testing of cognitive algorithms within real-world robotic environments. By simulating the behavior of UAVs and their interactions with ARCog-NET in a virtual environment, software-in-the-loop (SITL) testing allows for comprehensive testing and validation of the cognitive architecture's capabilities before deployment in physical UAV systems.

1.2. organization

The rest of this document is organized as follows. Section 2 describes the related works and the state-of-the-art of cognitive architectures. Section 3 presents the overall proposed architecture and the mathematical foundations for validating the methodology. Section 4 presents the results of the simulations and discusses the adopted algorithms. It shows the ARCOG-net being applied in different scenarios. The last Section 5 concludes the paper and presents ideas for continuing this work.

2. Related works

The study of UAV swarm architectures has gained significant attention, with researchers exploring challenges related to scalability, communication reliability, and energy efficiency [15]. Swarm control models are generally classified into centralized, decentralized, and hybrid approaches. Centralized systems depend on a single control unit, simplifying coordination but creating bottlenecks as the swarm grows. Decentralized models distribute decision-making across multiple UAVs, which enhances scalability and fault tolerance but complicates coordination. Hybrid approaches attempt to integrate both strategies, finding a balance between scalability and efficient swarm management. Tahir et al. [9] provide a comprehensive review of UAV swarm applications and challenges, highlighting the ongoing need for architectures that can adapt to dynamic and unpredictable environments.

Cognitive systems are emerging as a promising approach to improving the autonomy and adaptability of UAV swarms. These systems incorporate artificial intelligence and machine learning techniques to enable real-time decision-making in complex scenarios. Ramos et al. [14] introduced a cognitive framework for UAV swarms that emphasizes knowledge representation and reinforcement learning for optimizing task allocation and path planning. Similarly, research by Khamis et al. [16] examines multitasking in swarm robotics, demonstrating how cognitive-based strategies enhance operational efficiency in demanding missions.

Cognitive architectures play a fundamental role in enabling robots to perceive, interpret, and interact with their surroundings [17]. Dominey et al. [18] discuss how psychology and computer science are converging due to advances in robotics, allowing robots to develop sensory and motor capabilities that facilitate direct interaction with physical environments. Achieving a higher degree of autonomy requires a combination of perception, learning, adaptation, decision-making, and memory [17, 19].

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

Perception is one of the core aspects of cognition, as it allows robots to gather and process information to execute tasks effectively [17]. Haazebroek et al. [20] describe how perception involves more than passive data collection, requiring active interpretation to assess environmental stimuli in context. This process is deeply interconnected with attention, memory, reasoning, judgment, imagination, and language [21].

For autonomous operation in changing environments, robots must continuously learn from experiences and refine their behavior. Machine learning techniques, including supervised, unsupervised, and reinforcement learning, are central to this capability [22]. Adaptation allows robots to improve performance over time based on past actions and environmental feedback. Real-time decision-making and problem-solving are essential for UAVs operating in dynamic conditions, requiring them to analyze possible actions, predict outcomes, and choose the most effective course based on mission objectives. This process integrates information from perception and memory to make informed decisions [17].

Several studies have proposed cognitive architectures to enhance how robotic systems interact with their environment [17, 19, 23–25]. The Intelligent Vehicle Control Architecture (IVCA), introduced by [26], was designed to support cooperation between aerial vehicles. Although IVCA offers flexibility, its primary focus is on collaborative aircraft operation, relying on ontologies for structuring data while lacking detailed descriptions of hardware implementation and learning mechanisms. Alternatively, [27] proposed a UAV framework centered on autonomous operation driven by sensor data. However, its reliance on sensor-based decision-making limits adaptability, and its hardware requirements vary depending on application needs, which is a problem well handled by ARCog-NET due to its flexibility in associating hardware with the network robots when compared to [27]. Aerostack [28] provides a structured approach to managing autonomous UAV teams, prioritizing navigation and obstacle avoidance. While Aerostack has advantages, it presents limitations in human-in-the-loop (HITL) integration, team collaboration, and autonomous decision-making.

In robotic systems, decision-making refers to the ability to select actions based on environmental inputs, operational goals, and internal states. Embedding decision-making capabilities allows UAVs to function effectively in dynamic environments, even without centralized control. This adaptability enables UAVs to respond to changes in their surroundings. Traditional autonomous decision-making strategies often rely on extensive dataset training, limiting their generalization across diverse operational scenarios [29].

In high-stakes domains such as aerospace, robotic systems must ensure predictable and stable decision-making. Several computational techniques address this requirement, including neural networks, expert systems (ES), evolutionary computation (EC), data mining (DM), and case-based reasoning (CBR) [30]. DM is crucial in decision support systems, leveraging pattern recognition to extract valuable insights. EC effectively handles complex multi-criteria problems, making it suitable for real-world applications. CBR provides efficient knowledge representation while requiring minimal computational power but struggles with decision-making in novel situations. Neural networks deliver strong results but demand extensive data for training.

While these techniques provide effective decision-making strategies, they often rely on predefined models or require extensive training data, limiting adaptability in highly dynamic environments. To overcome these limitations, reinforcement learning (RL) has emerged as a promising approach, enabling UAVs to learn and refine their decision-making policies through direct interaction with their surroundings. RL enables an agent to determine optimal decisions through iterative interaction with its environment. The learning process focuses on achieving specific objectives without predefined instructions on how to reach them. Initially introduced by [31], RL has gained traction in machine learning research, addressing diverse artificial intelligence challenges. It operates on a reward-based system, where actions are evaluated based on environmental feedback. Positive outcomes reinforce specific behaviors, while negative ones discourage ineffective strategies. Over time, the agent refines its decision-making to maximize cumulative rewards. According to [32], real-time execution and planning involve managing models that may not always be reliable. Beyond recognizing and responding to failures,

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

an intelligent agent must learn from past errors to reduce the likelihood of repeating them. RL supports this continuous learning process, making it the decision-making framework employed in this study.

In the ARCog-NET framework, reinforcement learning (RL) is integrated to enhance the cognitive decision-making process of UAVs by enabling them to learn from past experiences. Specifically, RL mechanisms are used to update decision weights based on the effectiveness of previously executed actions—such as trajectory adjustments or task allocations—in response to environmental events. When a UAV faces a decision point, it consults its historical knowledge base, which includes matrices of coverage results and corresponding weights. If a past decision yielded better performance (e.g., higher coverage or reduced cost), the associated weight is reinforced. Otherwise, it is penalized. This reward-based update process allows UAVs and coordination layers to progressively favor more effective behaviors over time, thereby improving autonomy, adaptability, and operational efficiency without relying solely on predefined heuristics or human input.

Several researchers have explored multi-UAV simulations and network integration. In [33], an area-sensing application using a novel path planning algorithm was tested in both simulations and real-world field experiments. However, the study did not account for network constraints such as latency, which could impact mission execution. Antonio et al. [34] introduced a swarm coordination strategy inspired by insect group behavior. The study offers a strong theoretical foundation for swarm control, supported by a detailed literature review. However, it lacks insight into practical system constraints, such as hardware limitations and network communication architecture, relying solely on mathematical simulations.

Many recent studies that influenced this work use ROS [35] alongside the Gazebo [36] physics simulator for UAV swarm research. ROS provides a modular framework for robotics applications, offering capabilities such as hardware abstraction, device control, message passing, and package management. It structures robotic computations into a graph-based architecture, where various nodes handle functions like sensor multiplexing, control, planning, and actuation. In this study, ROS was used to develop multi-UAV control packages, facilitating inter-robot communication. Additionally, an autoencoder module, which will be described later, was integrated into the simulation for data compression.

ROS includes MAVROS [37], a communication interface that enables interaction between ROS-based applications, Flight Control Units (FCU), sensors, and Ground Control Stations. MAVROS supports MAVLINK [38], a lightweight protocol for drone communication, enabling integration with PX4 Autopilot firmware [39]. This combination allows the system to simulate realistic flight conditions in Gazebo while generating deployment-ready flight routines for real UAVs. Gazebo is an open-source 3D robotics simulation platform that integrates physics engines such as the Open Dynamics Engine (ODE) and Bullet [40]. It provides high-quality visual rendering and sensor modeling capabilities, allowing realistic simulation of UAV behavior. Sensors such as laser range finders, depth cameras, and Kinect-style devices can be incorporated into simulations. In this study, Gazebo was used to model the environment, simulate RGB cameras, and incorporate realistic physical interactions with UAVs.

The study of [41] proposed a framework that combines ROS with network simulators like Artery/OMNET++ [42], creating a platform for cooperative autonomous vehicle research. While the work contributes to network integration, it lacks concrete application examples and doesn't evaluate communication performance in different scenarios. Similarly, Mohini et al. [43] developed a framework integrating ROS and the NS-3 network simulator [44]. The authors employed PX4 Autopilot packages [45, 46] to build a multi-UAV simulation capable of modeling various communication networks, including Wi-Fi and GSM (Global System for Mobile Communications). The study examined the effects of communication disruptions and delays but did not explore data compression techniques, which is an important aspect covered in this work.

Other studies, such as those by Acharya et al. [47] and [48], also integrated ROS with network simulators for multi-robot research. These works addressed time synchronization challenges, as ROS/Gazebo simulations operate on continuous updates while network simulators rely on discrete event-driven processes. Lastly, Pinto et al. [49] introduced a fog-to-cloud computing framework for UAV data processing.

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

The research focused on the mathematical modeling of communication pipelines in multi-robot networks but did not incorporate real-world robotic simulations.

Despite significant progress in UAV swarm architectures, existing models still face challenges in balancing scalability, real-world applicability, and robust autonomous decision-making. For instance, task allocation in UAV swarms has been extensively studied to enhance autonomous decision-making and optimize mission execution. Various approaches, including market-based mechanisms [50], auction-based models [51], and bio-inspired strategies [52], have been proposed to address dynamic and cooperative task assignments. Additionally, heuristic methods such as task clustering and auction-based assignments have shown effectiveness in distributing workloads while minimizing computational complexity. Reinforcement learning has also emerged as a promising solution, allowing UAVs to adapt to environmental changes and resource constraints autonomously. However, these strategies often lack cognitive adaptability and do not explicitly integrate utility-based decision models or hierarchical coordination mechanisms.

Regarding path planning, PSO and A$*$ algorithms remain popular for dynamic route optimization [53], but many implementations focus on static environments or rely on centralized processing, limiting their responsiveness and scalability in swarm systems. Similarly, formation control techniques, including leader-follower and behavior-based models, have been explored for maintaining UAV formations [54], yet often disregard network dynamics, latency handling, and decision decentralization. Few architectures combine these components within an adaptive framework that also accounts for distributed cognition, resource awareness, and mission-level decision-making. Architectures like Aerostack [28] and IVCA [26] provide modular solutions for UAV coordination but do not integrate multi-layer learning or edge-fog-cloud processing. In contrast, this work builds upon these insights by proposing ARCog-NET, an enhanced UAV swarm architecture that leverages cognitive computing, edge-fog-cloud processing, and reinforcement learning for improved autonomy and operational efficiency

The ARCog-NET performance is evaluated and compared with the literature models above for task allocation, path planning, and formation control. Those models are implemented in the ARCog-NET framework and run in parallel with ARCog-NET models presented in Subsections 3.2, 3.3, and 3.4. For Task Allocation, the evaluated parameters are efficiency (number of successfully allocated tasks per total number of assigned tasks in a given time window), energy consumption per task (average energy used in executing a task), communication overhead (reason between transmitted data for network commands or requests and total transmitted data in a given time window), adaptability (average time taken in reallocating tasks) and scalability (the number of connected UAV that don't saturate fog layer in transmitting data). Path Planning is evaluated and compared using convergence time (how fast the models find a solution), trajectory coverage (the reason between the traveled path and the calculated path), computational cost (time taken in each iteration of path algorithms), and adaptability and scalability. Finally, Formation Control is evaluated through formation error (mean absolute error of UAV in formation), energy consumption, adaptability, communication overhead, and robustness to delays (measures the ability of UAV in formation to maintain itself despite communication or control signal delays between it).

It's also important to notice that some models can handle changes in dynamic environments. For the algorithms that are designed for static environments, ARCog-NET automatically deals with changes by "resetting" the routines when an event happens. For example, if a UAV detects an obstacle in a calculated path using the algorithm A$*$, to avoid this obstacle a new path will be calculated using A$*$ from scratch, using only the new knowledge about neighborhood collected by the UAV and the network. The versatility of ARCog-NET is based on including such information in previous calculations, without needing to remake the path from scratch. Using cognitive weighting factors history in calculation also helps the ARCog-NET performance in task allocation, path planning, and formation control. And finally, depending on the UAV status when an event happens, the algorithms will not be even executed as the knowledge base may be able to provide a solution already calculated by this event in the knowledge base.

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

3. Proposed methodology

In the same way proposed in the previous work from Ramos et al. [14], three simulation scenarios were proposed for validating ARCog-NET in task allocation, path planning, and formation control. The performance of the architecture is compared to the state-of-the-art methods in this area. All development of ARCog-NET custom packages was made with Python 3.8.10 language, using mainly the libraries TensorFlow, NumPy, SciPy, Scikit-learn, OpenCV, rospy, and MAVROS [37]. The robotic control framework was implemented using ROS Noetic [35], while simulations were executed within the Gazebo 11 [36] environment with relevant plugins. All programming and simulations were conducted on a Lenovo IdeaPad Gaming 3i, equipped with a 4.4 GHz Intel Core i5 processor, 16 GB DDR4 3200 MHz RAM, NVIDIA GeForce RTX 3050 (4 GB), a 512 GB SSD, and running Ubuntu 20.04.6 LTS.

The ARCog-NET architecture includes an FCU (Flight Control Unit) solely responsible for flight control, peripherals, and avionic data gathering. Computationally demanding tasks such as data processing, decision-making, communication, and complementary operations are handled by embedded boards like Raspberry Pi [55] or Galileo [56]. These offboard computers will command FCU through serial communication while executing the algorithms presented in the present section. One of the advantages of this topology is also the use of ROS packages that may run on the offboard controllers with their simulation already tested on Gazebo. The information may be exchanged between nodes and layers through free-to-air modulated signals using radio transmitter/receiver pairs. The cloud layer must have improved communication power, as it doesn't have hardware component constraints (since it is expected that it is a ground-based control unit). It may be further from the cloud and edge UAV devices. Figure 1 illustrates the key simulated components in each layer, while Figure 2 provides a simplified overview of the assumed hardware setup for the robots in this study.

The simulated UAVs used in this study are Yuneec Typhoon H480 hexacopters, each equipped with a PixHawk PX4 FCU, a Raspberry Pi 5 as the offboard embedded computer, and 4G telemetry and communication modules. The specifications of these components are detailed in Table II, while Table I presents the simplified power consumption values considered in the simulations. The performance of each mission was assessed in conjunction with the network parameters (latency, energy consumption, throughput, and packet loss). Latency is obtained by the delay between when a packet is sent and when it is received. Packet loss is calculated as the ratio of sent and received packets. The size of the packets is used for throughput estimation, and energy consumption is modeled based on the power of the UAV and the hardware it carries.

In the simulation, Agents (edge) represent individual UAVs equipped with sensors and embedded computers. They handle local processing tasks like navigation and sensor data analysis. The processed data can include images, flight data, and sensor outputs, which are then sent to the Coordinator (fog) for further decision-making or directly to the Server for storage and higher-level analysis. At the core of this architecture, the Coordinator is responsible for trajectory planning and mission supervision. It sends and receives commands to and from the Server (cloud) while also handling requests from Agents. This node processes data such as sensor inputs and navigation commands, making decisions that are then communicated to both the Agents and the Server for cohesive mission execution. The Server, operating within the cloud layer, oversees the entire mission, providing additional processing power for data organization and long-term storage. It also plays a crucial role in decision-making, particularly with HITL commands, where a human operator can intervene and control the swarm's actions, ensuring mission flexibility and adaptability to changing conditions.

3.1. Simulation

The first simulation background represents an offshore oil exploration environment where the UAVs follow programmed paths for environmental surveillance (search for oil spills, for example). The UAV swarm should take off from the oil production facility's helideck and send its camera images to the

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

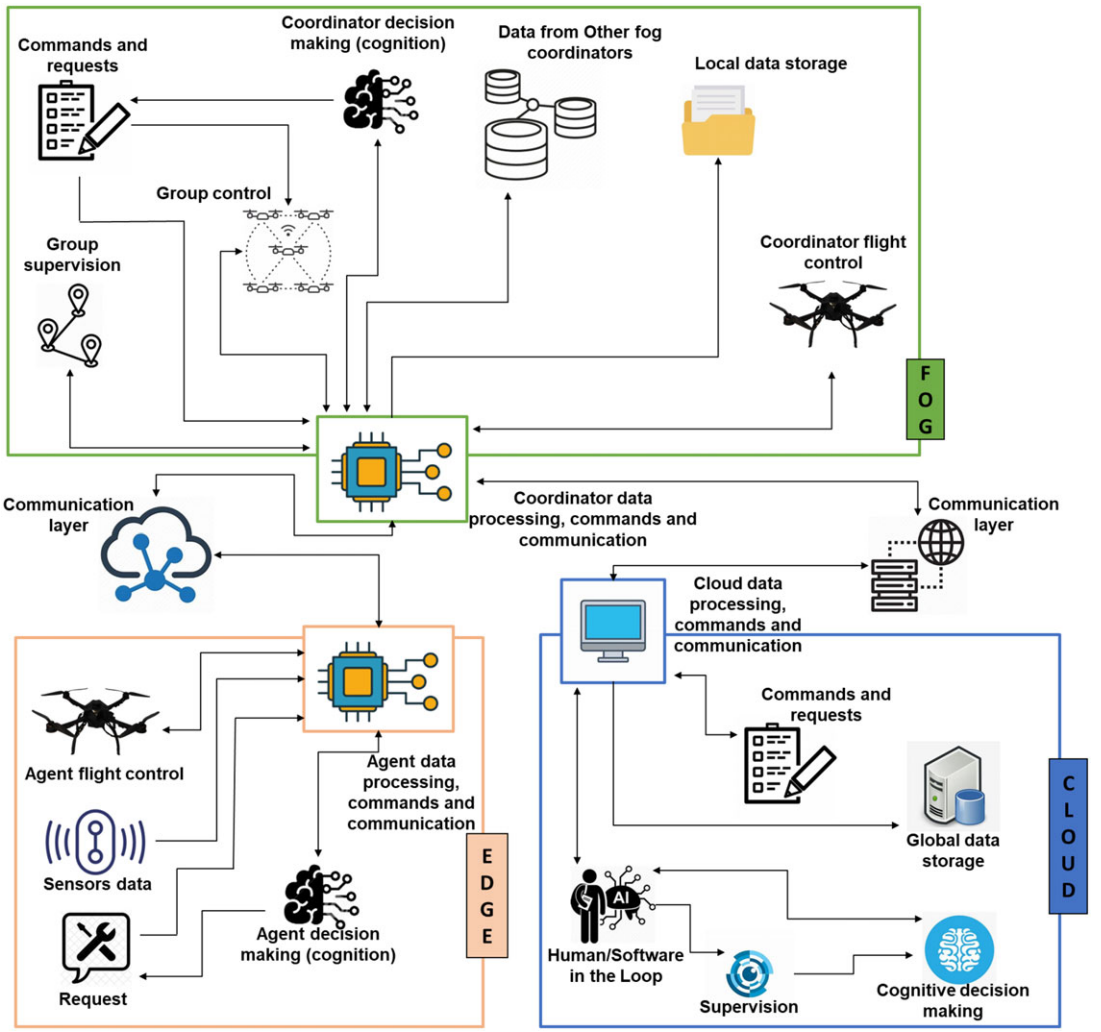

Figure 1. *ARCog-NET basic components.*

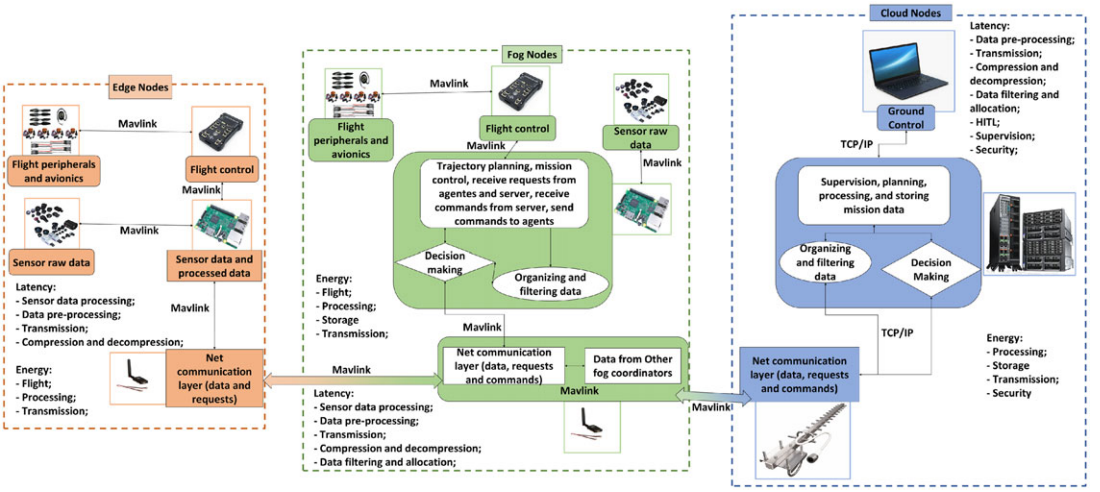

Figure 2. *Standard hardware representation of ARCog-NET.*

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

Table I. *Power consumption of different hardware components.*

Hardware	**Power**
Embedded Computer	27 W
UAV	120 W
Radios	1 W
Mobile Communication Modules and Antenna	100 W
Cloud	65 W

Table II. *Communication and processing power specifications.*

Device	**Pixhawk PX4 FCU**	**Raspberry Pi 5**	**4G Telemetry**
Processor	STM32F427, Cortex M4, 168 MHz	Broadcom BCM2711, Quad-core Cortex-A72 (ARM v8), 1.5GHz	N/A
Memory	2 MB Flash, 256 KB RAM	4 GB, 8 GB LPDDR4-3200 SDRAM	N/A
Comm. Interfaces	UART, SPI, I2C, CAN, USB	Gigabit Ethernet, WiFi 5 (802.11ac), Bluetooth 5.0, BLE	LTE, HSPA+
Data Rates	N/A	Ethernet: up to 1 Gbps, WiFi: up to 866.7 Mbps	LTE: up to 150 Mbps (Download), up to 50 Mbps (Upload)
Frequency Bands	N/A	2.4 GHz, 5.0 GHz bands	Multiple bands depending on carrier and region
Additional Features	Advanced sensor integration for navigation, Multiple safety features including triple redundant IMU	Extensive GPIO and peripheral support, 4K video support	High mobility support, designed for IoT applications

control station. First, to simulate the environment in which the mission will take place, an oil rig and a shuttle tanker vessel were designed with Blender 3D modeling software and integrated as Gazebo models. In the same way, Gazebo packages were used for designing the sky, clouds, sea, and ODE plugins for wind and sea parameters; the world on which the simulation took place was created [1, 57–59]. Figure 3 presents two views of the world created in the Gazebo software.

The second application consists of using ARCog-NET robots as internet routers for a communication failure emergency in a neighborhood scenario. The group should take off from a land base and position itself to provide maximum signal coverage in a city environment, given the random demand generated by an algorithm. Concerns about this mission are mainly the energy of each UAV for keeping its position and transmitting signals. The decision should be mostly based on the robots' positioning and how to switch each robot at the right time for charging, while keeping the internet signals available for the city. Figure 4 presents this second simulation sandbox. Finally, the last simulation represents an industrial plant undergoing an emergency situation (e.g., a gas leakage) to evaluate the indoor behavior of a group of UAVs using ARCog-NET. In this scenario, the robots should take off from inside the building and scan the environment to find human victims to deliver gas masks or send distress signals, emulating a search and rescue mission. This scenario is represented in Figure 5. Note that these experiments were implemented using a hybrid simulation-emulation framework. While the environments were modeled in Gazebo, the UAVs were controlled using SITL with PX4 firmware and MAVROS, allowing for behavior emulation close to real-world hardware constraints.

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

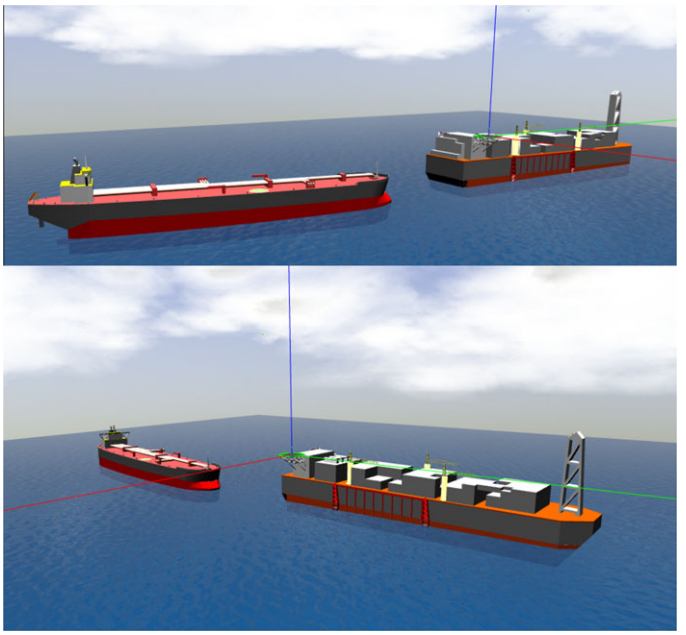

Figure 3. *World created in Gazebo software for simulating the image transmission system on an offshore mission.*

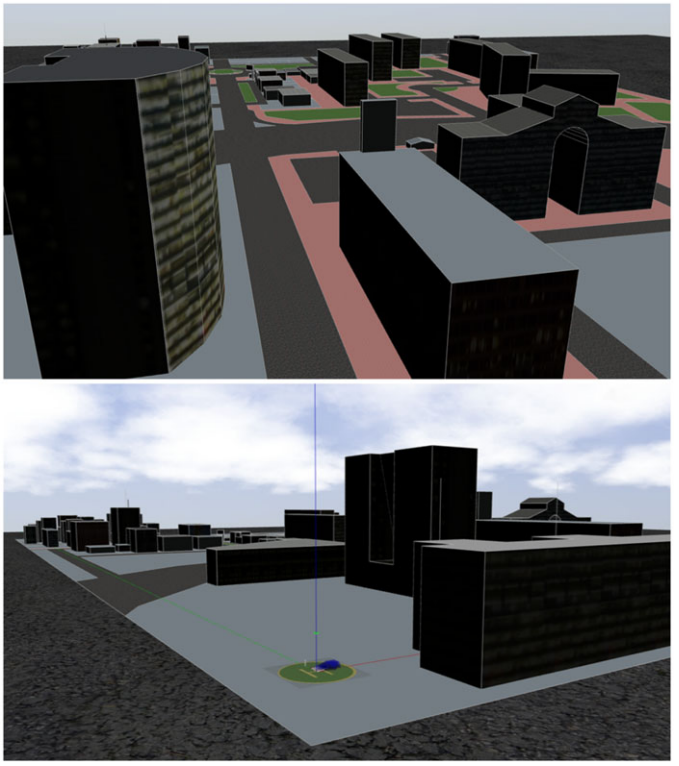

Figure 4. *World created in Gazebo for simulating using the UAV swarm as WiFi routers for communication emergencies.*

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

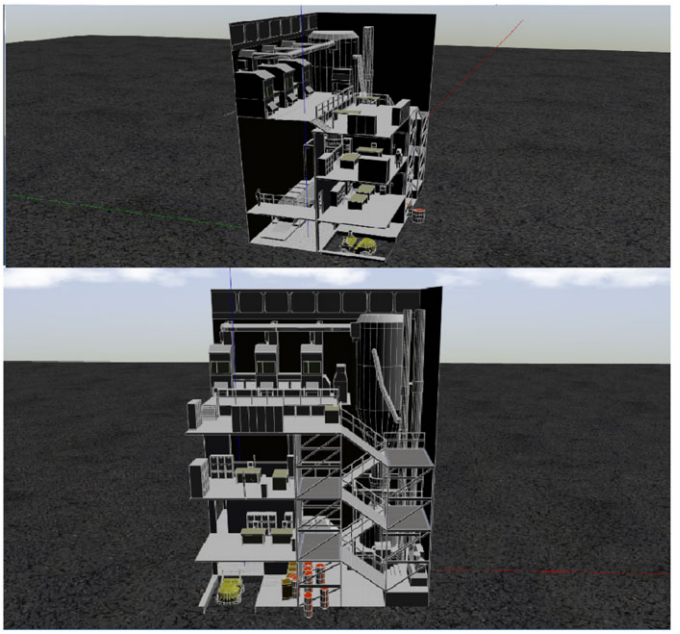

Figure 5. *World created in Gazebo for simulating using the UAV swarm in search and rescue indoor missions.*

A large-scale simulation was performed using a headless simulation environment (graphical interface disabled) to evaluate network scalability and performance and minimize computational overhead systematically. The experiment was initialized with a single UAV, and the swarm size was scaled progressively to 1000 nodes, incrementally testing network performance at each mission iteration. Key parameters—including inter-node latency, energy consumption, and decision-making efficiency—were measured under dynamic mission scenarios. Communication protocols were stress-tested using Poisson-distributed event triggers to emulate real-world operational unpredictability, and statistical robustness was ensured through Monte Carlo simulations (10 trials per swarm size) with randomized obstacle distributions and mission objectives [16]. Data acquisition focused on trajectory optimization success rates, edge-fog-cloud decision latency, and network load balancing, providing a quantitative framework to assess scalability limits and layer-specific bottlenecks. Finally, ARCog-NET has a built-in data compression system [60] that will be evaluated by PSNR (Peak Signal-to-Noise Ratio), SSIM (Structural Similarity), and MS-SSIM (Multi-Scale Structural Similarity).

In the second scenario, the simulation is executed with one to ten vehicles connected in ARCog-NET, 10 times for each connected robot count, that is 100 simulations in total. The server (cloud layer) is initially configured to request the structure to use some of those robots as fog nodes (which can vary from 1 to 10 robots), which are also edge group coordinators. The number of the other edge agents may vary throughout the mission, so a fog coordinator may have up to 9 agents connected to it. While flying to accomplish mission tasks, the UAVs are subjected to possible collisions between each other and other structures, and the ARCog-NET capabilities are used to calculate new trajectories and deviations. The robots may also pre-process sent images (i.e., compression, see [60]) and request superior layers to process decisions or for HITL decisions. The network, energy constraints, external demand, and other factors are also evaluated through simulations to stress the architecture and extract its behavior. Still, due to computer power limitations, this is done without the graphic simulations. In the second scenario case, all evaluations are the same, but 10 graphic simulations ran with 10 UAV each, as the initial tests presented that it did not make sense to simulate the scenario of the robots being used as a communication tower network replacement with fewer than this number of components.

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

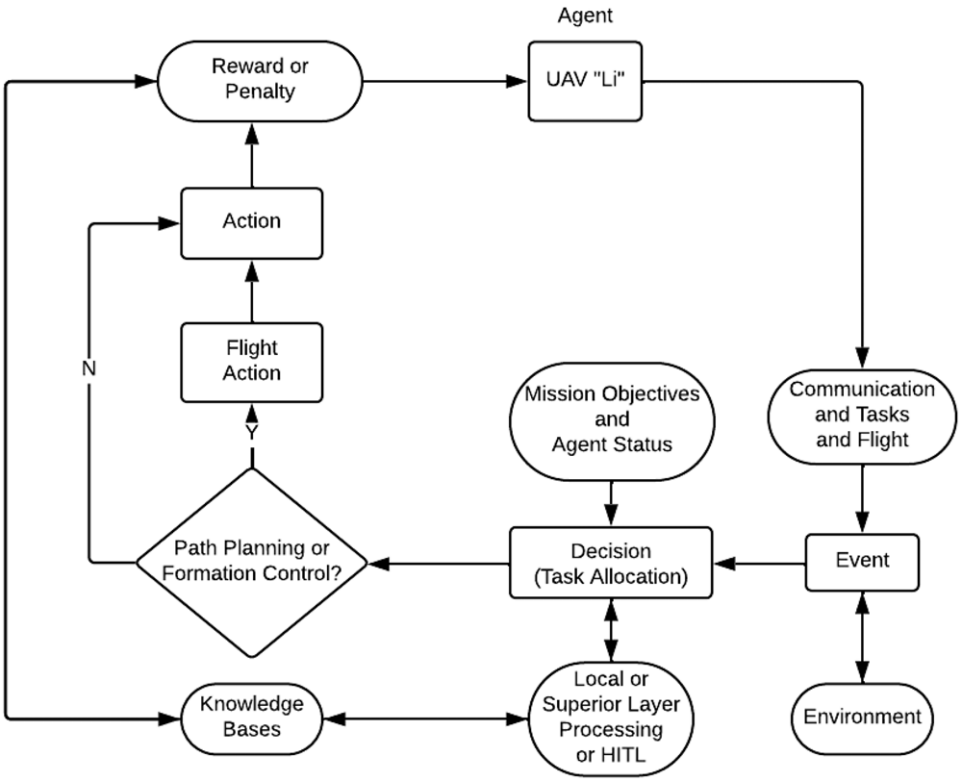

Figure 6. *Cognitive learning process in ARCog-NET.*

3.2. Task allocation

The input for the cognition in ARCog-NET can be modeled as an optimized task allocation problem, in which the utility factor of each task must be maximized. The weights of each decision will help locate the best solution of an operational event in each node search or create a new solution if it doesn't find one. A task is an action that each UAV should perform as a response to an event (avoiding an obstacle or covering a faulty neighbor, for example), and the weight of the task is the importance of that action for accomplishing mission goals. When an event occurs with a UAV, it employs a systematic approach for continuous learning and adaptation throughout the proposed algorithm, incorporating feedback into its respective knowledge bases and adjusting its decision strategies accordingly when executing the mission. If an optimal solution for allocating the task as a response to an event cannot be handled by the UAV, it can send either a network request for a superior layer or a collaboration request to its neighbors from the same layer. This mechanism allows the layers to operate independently and cohesively within the network's cognitive architecture. The processing workflow for cognitively allocating a task is defined as decision-making in the context of this document.

Two specific tasks related to the UAV's navigation run separately in two cognitive subroutines. The first is the path planning task, where a UAV will calculate its navigation route or ask superior layers for a solution, learning the best solutions in the process and gaining access through the network to collective databases. The second is the formation control, which will be triggered every time a group must perform close and similar paths or if it is a formation control mission configured by the HITL. Therefore, ARCog-NET has 3 built-in cascade cognitive algorithms, one for general task allocation and the other two for specific cognitive flight sub-routines that may be triggered in navigation or flight tasks, if allocated. Users may configure, change, and improve those algorithms and triggers, facilitating UAV swarm implementation with the proposed architecture. The basic cognitive reinforcement learning process of ARCog-NET is presented in Figure 6.

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

Tasks may have different natures when accomplishing mission goals. Let $\mathcal{T} = \{T_1, T_2, \ldots, T_R\}$ represent the set of all possible tasks that a UAV can perform, where R is the total number of task types. Each task $T_r \in \mathcal{T}$ is associated with a specific action, such as:

- T_{nav}: Navigate to a new waypoint.
- T_{avoid}: Avoid an obstacle.
- T_{role}: Change role (e.g., edge agent to fog coordinator).
- T_{conn}: Change connection to a fog coordinator.
- T_{compress}: Adjust data compression level.
- $T_{\text{data}_{\text{toggle}}}$: Start or stop data transmission.
- T_{form}: Enter, leave or change formation.

Users may configure new tasks and subroutines within this task-dealing model. At any given time t, a UAV i in layer L may execute a task T_r or remain idle if no task is allocated. To efficiently allocate tasks, ARCog-NET employs a cognitive utility model. Each UAV evaluates the utility of executing a specific task for itself or another node on an inferior layer (if requested) based on the current mission state, network conditions, and task objectives. In the ARCog-NET model, the utility is understood as a quantitative measure that evaluates the benefit or fitness of a UAV (or node) performing a specific task at a given time. The state of UAV i on layer L (edge or fog) at time t is represented as:

$$S_{L_i}(t) = \left(P^{\kappa}_{L_i}(t), E_{L_i}(t), C^{D}_{L'_j,L_i}(t), \mathcal{T}_{L_i}(t)\right), \tag{1}$$

where $P^{\kappa}_{L_i}(t)$ is the current position of the UAV, $E_{L_i}(t)$ is the remaining energy, $C^{D}_{L'_j,L_i}(t)$ is the current task coverage,[1] and $\mathcal{T}_{L_i}(t)$ is the buffer vector of assigned tasks. At each instant t, each UAV (or network node) keeps a weighting factor tensor keeping a multilinear relationship to tree weighting factor matrices that help decide tasks as a response to events, denoted by:

$$W_{L_i}(t) = \begin{bmatrix} W^{TA}_{L_i}(t) \\ W^{PP}_{L_i}(t) \\ W^{FC}_{L_i}(t) \end{bmatrix}, \quad W^{D}_{L_i}(t) = \begin{bmatrix} w^{D}_{L'_1,L_1}(t) & \cdots & w^{D}_{L'_m,L_1}(t) \\ \vdots & \ddots & \vdots \\ w^{D}_{L'_1,L_n}(t) & \cdots & w^{D}_{L'_m,L_n}(t) \end{bmatrix}, \quad D \in \{TA, PP, FC\} \tag{2}$$

in which $w^{D}_{L'_j,L_i}(t)$ is the weighting factors for activities D (TA for task allocation, PP for path planning and FC for formation control) of the task planning decision for the i-th robot at layer L commanded by node j at layer L' [2,3] The same is valid for coverages. The utility for UAV i to perform task T_r at time t is defined as:

$$U_{i,r}(t) = w^{TA}_{L'_j,L_i}(t) \cdot [\Phi^{\text{mission}}_{r}(t) + \Phi^{\text{network}}_{r}(t) + \Phi^{\text{energy}}_{r}(t)] + w^{PP}_{L'_j,L_i}(t) \cdot w^{FC}_{L'_j,L_i}(t) \cdot \Phi^{\text{navigation}}_{r}(t), \tag{3}$$

where:

- $w^{TA}_{L'_j,L_i}(t)$: Weighting factor for task allocation. This factor is randomly assigned using a continuous probability distribution for the first task the UAV receives. This is done for simplicity and as the mission evolves, the weighting factors become more precise due to the continuous learning.
- $w^{PP}_{L'_j,L_i}(t)$: Weighting factor for path planning.

[1] Coverage refers to a quantitative measure that evaluates how effectively a UAV has addressed or completed a set of tasks in ARCog-NET. Path Planning coverage is a specific case which measures the error between a calculated trajectory and the actual flight trajectory and Formation Control coverage measures how well a UAV keeps its position in formation.

[2] Suppose, as an example, a formation with five UAV, being two for coordinators (Fog_0, Fog_1), one with two connected agents ($Edge_0$, $Edge_1$) and another with just one ($Edge_2$). The weighting factor $w^{D}_{L'_j,L_i}(t)$ of an activity D of the edge agent number 1 commanded by its coordinator is represented as $w^{D}_{Fog_0,Edge_1}(t)$.

[3] i may be equal to j and L may be equal to L' as well. If there's no connection between L_i and L'_j at time t, then $w^{D}_{L'_j,L_i}(t) = 0$.

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

- $w^{FC}_{L'_j,L_i}(t)$: Weighting factor for formation control.
- $\Phi^{\text{mission}}_r(t)$: Utility of the task in achieving mission goals.
- $\Phi^{\text{network}}_r(t)$: Utility of the task in improving network performance (e.g., reducing latency or packet loss).
- $\Phi^{\text{energy}}_r(t)$: Utility of the task in conserving energy resources.
- $\Phi^{\text{navigation}}_r(t)$: Utility of the task in UAV flight.

Each $W_{L_i}(t)$ matrix is directly associated with a historical task coverage matrix $H_{L_i}(t)$ of the same size. These historical task coverage matrices store the coverage of a task D assigned by a UAV j in layer L' to a UAV i in layer L at time t. The representation of $H_{L_i}(t)$ is given by:

$$H_{L_i}(t) = \begin{bmatrix} H^{TA}_{L_i}(t) \\ H^{PP}_{L_i}(t) \\ H^{FC}_{L_i}(t) \end{bmatrix}, \quad H^{D}_{L_i}(t) = \begin{bmatrix} C^{D}_{L'_1,L_1}(t) & \cdots & C^{D}_{L'_m,L_1}(t) \\ \vdots & \ddots & \vdots \\ C^{D}_{L'_1,L_n}(t) & \cdots & C^{D}_{L'_m,L_n}(t) \end{bmatrix}, \quad D \in \{TA, PP, FC\} \tag{4}$$

Thus, each element of the matrix $H_{L_i}(t)$ represents the coverage outcome of a task computed by an individual j in layer L' for the corresponding individual i in layer L. In cases where $L_i = L'_j$, the decision-making occurs locally. Each element of $H_{L_i}(t)$ is linked to the corresponding weight matrix $W_{L_i}(t)$, where the elements of $W_{L_i}(t)$ quantify the effectiveness of different trajectory coverage options. These weights serve as reference points for querying the knowledge base. The historical weight tensor records for a UAV are represented as:

$$\omega_{L_i} = \{W_{L_i}(t_0), W_{L_i}(t_1), \cdots, W_{L_i}(t_n)\}, \tag{5}$$

and the knowledge base tensor, which stores the historical coverages, is represented as:

$$\kappa_{L_i} = \{H_{L_i}(t_0), H_{L_i}(t_1), \cdots, H_{L_i}(t_n)\}. \tag{6}$$

These tensors are used by the UAV or its coordinator to search for weighting factors previously applied to a similar task that this robot executed. This makes the UAV "consider" whether a task similar to the current one, allocated to it in a past time τ $(t > \tau)$, has the same utility in the mission and therefore should be executed, or whether it is not relevant or a priority within its scope of current objectives. A UAV selects the task with the highest utility:

$$T^{\text{alloc}}_i(t) = \arg \max_{w_{\text{set}} \in W_{L_i}(t)} U_{i,r}(t), \quad w_{\text{set}} = \left\{w^{TA}_{L'_j,L_i}(t), w^{FC}_{L'_j,L_i}(t), w^{PP}_{L'_j,L_i}(t)\right\} \tag{7}$$

Each w_{set} is associated to a historic task T_r that can be assigned as response to an event, which is also related to $C_{\text{set}} = \{C^{TA}_{L'_j,L_i}(t), C^{FC}_{L'_j,L_i}(t), C^{PP}_{L'_j,L_i}(t)\}$ groups. If multiple tasks have equal utility, the selection is based on predefined tie-breaking rules, such as prioritizing tasks critical to mission continuity or assigning tasks that minimize disruption to the swarm. The Mission Utility component ($\Phi^{\text{mission}}_r(t)$) can be defined by:

$$\Phi^{\text{mission}}_r(t) = \begin{cases} -\|\epsilon_{\text{goal}}(t)\|^2 \cdot C^{TA}_{L'_j,L_i}(t), & \text{if } C^{TA}_{L'_j,L_i}(t) \text{makes the respective task } T_r \text{ contribute to the goal,} \\ 0, & \text{otherwise,} \end{cases} \tag{8}$$

where $\epsilon_{\text{goal}}(t)$ is the error between the UAV's current state coverage and the mission goal, and $C^{TA}_{L'_j,L_i}(t)$ is the task allocation coverage, defined as:

$$C^{TA}_{L'_j,L_i}(t) = \frac{n_{T_{L_i}}}{n_{\text{total}_{L_i}}}, \tag{9}$$

where $n_{T_{L_i}}(t)$ is the number of successfully completed tasks and $n_{\text{total}_{L_i}}(t)$ is the total number of tasks assigned to the UAV until time t, respectively. The Network Utility component ($\Phi^{\text{network}}_r(t)$) can be defined by:

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

$$\Phi_r^{\text{network}}(t) = -C^{TA}_{L'_j,L_i}(t) \cdot \left(\frac{1}{\lambda_{\max}} \cdot \Delta\lambda_{\text{latency}} + \frac{1}{\varrho_{\max}} \cdot \Delta\varrho_{\text{loss}} \right), \tag{10}$$

where $\Delta\lambda_{\text{latency}}$ and $\Delta\varrho_{\text{loss}}$ are the latency and packet loss instant variation, normalized by their respective maximum tolerable values ($\lambda_{\max}$, $\varrho_{\max}$) defined by HITL. Notice that if the instant latency or packet loss variation values overcome the maximum tolerable values, the network utility component has high importance. Finally, the Energy Utility component ($\Phi_r^{\text{energy}}(t)$)is defined by:

$$\Phi_r^{\text{energy}}(t) = -C^{TA}_{L'_j,L_i}(t) \cdot \frac{P_{\text{task}}(t)}{P_{\max}}, \tag{11}$$

where $P_{\text{task}}(t)$ is the power consumption associated with task T_r, and $P_{\max}$ is the UAV's maximum power capacity. The navigation utility $\Phi_r^{\text{navigation}}(t)$ is a specific utility that is applied only for flight tasks, denoted as:

$$\Phi_r^{\text{navigation}}(t) = \begin{cases} -\|\epsilon_{\text{pos}}(t)\|^2 \cdot C^{PP}_{L'_j,L_i}(t) \cdot C^{FC}_{L'_j,L_i}(t), & \text{if } T_r \text{ navigation coverages is both PP and FC,} \\ -\|\epsilon_{\text{pos}}(t)\|^2 \cdot C^{PP}_{L'_j,L_i}(t), & \text{if } T_r \text{ navigation coverages comprehends only PP,} \\ -\|\epsilon_{\text{pos}}(t)\|^2 \cdot C^{FC}_{L'_j,L_i}(t), & \text{if } T_r \text{ navigation coverages comprehends only FC,} \\ 0, & \text{if } T_r \text{ navigation coverages is null.} \end{cases}, \tag{12}$$

being $-\|\epsilon_{\text{pos}}(t)\|$ the position error of the UAV in time t (position of the UAV in relation to the position it should be in trajectory, formation, or both). Notice that all utilities and the general utility for the UAV are calculated as consequences of the UAV's current state.

Tasks may be reallocated during the mission in response to events, such as changes in mission priorities or network conditions. If a UAV fails to execute a task due to resource constraints, the task is reassigned to a neighboring UAV or escalated to a higher network layer (e.g., fog or cloud). The reallocation process minimizes disruption and ensures mission continuity, which is a collaborative process. Collaboration between UAVs is facilitated through shared knowledge bases at the edge, fog, and cloud layers. Each UAV shares its utility values with neighboring nodes to coordinate task allocation. The collaborative utility $\mathcal{U}_r^{\text{collab}}(t)$ for task T_r is computed as:

$$\mathcal{U}_r^{\text{collab}}(t) = \frac{1}{|\mathcal{N}_i|} \sum_{\eta \in \mathcal{N}_i} U_{\eta,r}(t), \tag{13}$$

where $\mathcal{N}_i$ is the set of neighbors for UAV i. The task is allocated to the UAV with the highest collaborative utility. The overall task allocation problem for the swarm can be formulated as an optimization problem to maximize utility:

$$\arg \max_{z[T_{i,r}(t)] \in \mathcal{T}} \sum_{i=1}^{Q} U_{i,r}(t), \tag{14}$$

subject to:

$$\sum_{r=1}^{R} \zeta_{L_i,r} \leq 1, \quad \forall i, \tag{15}$$

where $\zeta_{L_i,r} \in \{0, 1\}$ indicates whether task T_r is assigned to UAV i and $z[T_{i,r}(t)]$ is the function that correlates tasks T_r to the respective w_{set} and C_{set} within tensors in historic tensors.

The task allocation process is re-evaluated when a task-related event occurs, such as introducing a new task or a UAV failure. The edge layer recomputes the suitability scores and assigns tasks locally if possible. If no suitable task is found, the request is escalated to the fog layer, which aggregates task requests from multiple edge agents and optimizes (maximizes the utility of the task) global task allocation. The cloud layer handles critical task reallocation and HITL intervention. After a task execution, the success is evaluated and the weighting factors $w^D_{L'_m,L_n}(t)$ in the weight matrices $W_{L_i}(t)$ are updated by

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

reward or penalty, generating a new entrance on the knowledge bases ω_{L_i} where it is executed. Weight updates are expressed by:

$$w_{L_i}^{\mathrm{D,\,new}}(t) = w_{L_i}^{\mathrm{D}}(t) + \mathcal{L} \cdot \Delta C_{L_j' L_i}^{D}, \tag{16}$$

where $w_{L_i}^{\mathrm{D,\,new}}(t)$ is the new weight, $\mathcal{L}$ is the learning rate and $\Delta C_{L_j' L_i}^{D}$ is the variation of the previous task coverage used as reference for the decision and the actual measured coverage after the decision is taken. The task allocation algorithm is presented in Algorithm 1.

One of the primary challenges in UAV task allocation is balancing efficiency, energy consumption, and communication overhead, particularly in decentralized swarm architectures [52]. Heuristic methods, such as task clustering and auction-based assignments, have demonstrated effectiveness in distributing workloads while minimizing computational complexity [53]. In addition, reinforcement learning has been explored for real-time task assignment, allowing UAVs to adapt to environmental changes and resource constraints autonomously [54]. The ARCog-NET task allocation framework builds upon these advancements by incorporating cognitive reasoning and adaptive utility-based selection mechanisms. It enables UAVs to reconfigure their tasks dynamically while ensuring mission robustness and efficiency. Therefore, it will have task allocation results compared to those in this literature.

This task allocation model aims to allow UAVs to dynamically adapt to changing mission requirements and environmental conditions, leveraging historical data and reinforcement learning to optimize task execution. Integrating task allocation with the existing ARCog-NET framework enhances the network's ability to perform complex missions efficiently and reliably. For navigation (T_{nav}) and formation control (T_{form}) tasks, there are specific cognitive decision-making and controlling models and algorithms. If such a task is prioritized by or for a UAV to execute, the subprocess of flying is triggered and will be presented in the next subsections. The ARCog-NET task allocation is compared with other models as detailed in Section 2.

3.3. Path planning module

Navigation tasks (T_{nav}) are particularly important for ARCog-NET, as it is the most complex action a UAV node can execute. Whenever a navigation task is prioritized and allocated to a UAV, it enters a specific cognitive sub-process to calculate or recalculate its trajectory based on Particle Swarm Optimization (PSO). This approach was chosen for ARCog-NET flight navigation due to its efficiency in exploring search spaces, adaptability to dynamic environments, and suitability for multi-objective optimization in UAV path planning. Integrating cognitive models with PSO makes decision-making more advanced, allowing UAVs to perform complex tasks with minimal human input. This integration enhances mission efficiency and reliability, making it a powerful solution for UAV swarm applications in challenging environments.

As described in [14], it is assumed that UAVs operate within a three-dimensional space $A \in \mathbb{R}^3$ in ARCog-NET. The network is structured into nodes, with edge-layer agents represented as $e = \{e_1, e_2, e_3, \ldots, e_m\}$ and fog coordinators as $f = \{f_1, f_2, f_3, \ldots, f_n\}$, where e_m and f_n denote the maximum numbers of agents and coordinators, respectively. At the initial deployment time t_i, UAV groups, the number of agents per group, and coordinators are defined by the cloud supervisor via HITL intervention.

The human supervisor can also pre-configure the initial trajectory points and possible deviations in flight paths due to operational events. Initially, each UAV follows a predefined trajectory (either computed algorithmically or set manually) in ARCog-NET ($t = 0$), but modifications can occur dynamically at any time $t > 0$ in response to mission requirements. The number of objective trajectory points for the i-th UAV in a given layer $L \in e, f$ at the start of the operation is n^{L_i}. In contrast, the number of objective trajectory points at any moment t is represented as $n_p^{L_i}(t)$. This quantity is determined as follows:

$$n_p^{L_i}(t) = \sum_{\iota=1}^{M} \sum_{\gamma=1}^{N} \sum_{k=1}^{O} P_{L_i}^{\kappa}(x_\iota, y_\gamma, z_k), \tag{17}$$

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

Algorithm 1 Task Allocation Process in ARCog-NET.

Input: $S_{L_i}(t)$: State of UAV i at time t.
$\mathcal{T}$: Set of tasks $\{T_1, T_2, \ldots, T_R\}$.
$W_{L_i}(t)$: Weight matrix for UAV i at time t.
$H_{L_i}(t)$: Historical coverage matrix for UAV i at time t.
κ_{L_i}: Knowledge base for UAV i.
ω_{L_i}: Historical weight tensor for UAV i.
$\mathcal{L}$: Learning rate
Initialize: $T_i^{\text{alloc}}(t) \leftarrow \emptyset$
$\mathcal{U}_r^{\text{collab}}(t) \leftarrow 0$ for all $T_r \in \mathcal{T}$
for each task $T_r \in \mathcal{T}$ **do**
  Compute $C^{TA}_{L'_j,L_i}(t)$
  if $T_r \leftarrow T_{\text{nav}}$ **then**
    Compute $C^{PP}_{L'_j,L_i}(t)$ and $C^{FC}_{L'_j,L_i}(t)$
  end
  else
    $C^{PP}_{L'_j,L_i}(t) \leftarrow 0$ and $C^{FC}_{L'_j,L_i}(t) \leftarrow 0$
  end
  for Each W_{L_i} in ω_{L_i} **do**
    n $\leftarrow$ index of W_{L_i} in ω_{L_i}
    $H_{L_i} \leftarrow \kappa_{L_i}[n]$
    [a, b, c] $\leftarrow$ index of [max($w^{TA}_{L'_j,L_i}$), max($w^{PP}_{L'_j,L_i}$), max($w^{FC}_{L'_j,L_i}$)] in W_{L_i}
    $[C^{TA}, C^{PP}, C^{FC}] \leftarrow [H^{TA}_{L'_j,L_i}[a], H^{PP}_{L'_j,L_i}[b], H^{FC}_{L'_j,L_i}[c]]$
    Switch Case D:
      Case TA and $C^D_{L'_j,L_i}(t) \le C^D$: $C^D_{L'_j,L_i}(t) \leftarrow C^D$
      Case PP and $C^D_{L'_j,L_i}(t) \le C^D$: $C^D_{L'_j,L_i}(t) \leftarrow C^D$
      Case FC and $C^D_{L'_j,L_i}(t) \le C^D$: $C^D_{L'_j,L_i}(t) \leftarrow C^D$
    end
  end
  Compute $\Phi_r^{\text{mission}}(t), \Phi_r^{\text{network}}(t), \Phi_r^{\text{energy}}(t), \Phi_r^{\text{navigation}}(t)$ using $S_{L_i}(t)$ and $H_{L_i}(t)$
  Compute $U_{i,r}(t) = w^{TA}_{L'_j,L_i}(t) \cdot [\Phi_r^{\text{mission}}(t) + \Phi_r^{\text{network}}(t) + \Phi_r^{\text{energy}}(t)] + w^{PP}_{L'_j,L_i}(t) \cdot w^{FC}_{L'_j,L_i}(t) \cdot \Phi_r^{\text{navigation}}(t)$
  Compute $\mathcal{U}_r^{\text{collab}}(t) = \frac{1}{|\mathcal{N}_i|} \sum_{\eta \in \mathcal{N}_i} U_{\eta,r}(t)$
  Update $U_{i,r}(t) \leftarrow \mathcal{U}_r^{\text{collab}}(t)$.
  if $U_{i,r}(t) > U_{i,\text{alloc}}(t)$ **then**
    $T_i^{\text{alloc}}(t) \leftarrow T_r$
    $U_{i,\text{alloc}}(t) \leftarrow U_{i,r}(t)$
  end
end
if $T_i^{\text{alloc}}(t) \neq \emptyset$ **then**
  Execute $T_i^{\text{alloc}}(t)$.
  Update $H_{L_i}(t)$ and $W_{L_i}(t)$ based on task execution: $w^{\text{D, new}}_{L_i}(t) = w^{\text{D}}_{L_i}(t) + \mathcal{L} \cdot \Delta C^D_{L'_j L_i}$
  Append $W_{L_i}(t)$ to ω_{L_i}.
  Append $H_{L_i}(t)$ to κ_{L_i}.
end
else
  Send a collaboration request to neighbors or escalate to a higher layer.
end
Return: $T_i^{\text{alloc}}(t)$
Output: Updated task allocation, UAV state, ω_{L_i} and κ_{L_i}

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

where $t > 0$ represents a moment when trajectory modification is triggered (or a navigation task T_{nav} is allocated) due to an operational event, handled either locally or via commands from higher network layers. Here, $P^{\kappa}_{L_i}$ denotes the κ-th point of a UAV i in layer L following the trajectory ϱ_{L_i}. The parameters M, N, and O represent the maximum number of coordinates in the x, y, and z dimensions, respectively, defining the total number of three-dimensional points in a finite set. The indices ι, γ, and k iterate over the spatial coordinates x, y, and z, and $P^{\kappa}_{L_i}(x_\iota, y_\gamma, z_k)$ indicates the presence of a point at the coordinate (x_ι, y_γ, z_k). Equation (17) thus quantifies the total number of navigational points UAVs must traverse to complete their assigned tasks efficiently. This formulation is crucial as it dictates how UAV trajectories are dynamically optimized in response to changing environmental conditions, ensuring efficient coverage, minimal energy consumption, and reduced operational time—all essential for swarm-based autonomous UAV systems.

It is also important to know an incremental function denoted in Equation (18), which represents how much a point κ changes when it is modified by an event:

$$\Delta P^{\kappa}(t - t_i) = \begin{cases} \delta^{\kappa}_p, & \text{if } ||P^{\kappa}_{L_i}(t)_{\text{new}}|| - ||P^{\kappa}_{L_i}(t)_{\text{old}}|| \geq 0, \\ -\delta^{\kappa}_p, & \text{if } ||P^{\kappa}_{L_i}(t)_{\text{new}}|| - ||P^{\kappa}_{L_i}(t)_{\text{old}}|| < 0, \end{cases} \tag{18}$$

being $\delta^{\kappa}_p = |||P^{\kappa}_{L_i}(t)_{\text{new}}|| - ||P^{\kappa}_{L_i}(t)_{\text{old}}|||$ at time t, $P^{\kappa}_{L_i}(t)_{\text{new}}$ is the new κ trajectory ϱ_{L_i} point and $P^{\kappa}_{L_i}(t)_{\text{new}}$ is the previous one. For a generic i-th flying robot assigned in a layer L (e or f) in the present framework, the position of a node's j-th "current objective" (doesn't matter the layer) at any given time t is $P^{L_i}_{\text{obj}_j}(t) \in \mathbb{R}^3$. For mathematical generalization, each UAV is assumed to navigate toward its objective at a fixed altitude,[4] which may be adjusted dynamically in response to mission events through a PSO algorithm. The objective's position depends on the assigned navigation task, remaining constant (in applications such as photogrammetry) or changing abruptly due to operational events (such as obstacle detection or trajectory modification commands, for example). The resulting "objective motion" can be described as:

$$\begin{cases} \theta^{L_i}_{\text{obj}_j}(t) = \arctan\left(\frac{P^{L_i}_{\text{obj}_j}[y](t_i) - P^{L_i}_{\text{obj}_j}[y](t)}{P^{L_i}_{\text{obj}_j}[x](t_i) - P^{L_i}_{\text{obj}_j}[x](t)}\right) \\ V^{L_i}_{\text{obj}_j}(t) = \frac{||P^{L_i}_{\text{obj}_j}(t) - P^{L_i}_{\text{obj}_j}(t_i)||}{t - t_i} \\ P^{L_i}_{\text{obj}_j}[x](t_{i+1}) = P^{L_i}_{\text{obj}_j}[x](t) + V^{L_i}_{\text{obj}_j}(t) \cdot \cos\theta^{L_i}_{\text{obj}_j}(t) \cdot \Delta t \\ P^{L_i}_{\text{obj}_j}[y](t_{i+1}) = P^{L_i}_{\text{obj}_j}[y](t) + V^{L_i}_{\text{obj}_j}(t) \cdot \sin\theta^{L_i}_{\text{obj}_j}(t) \cdot \Delta t, \end{cases} \tag{19}$$

where $V^{L_i}_{\text{obj}_j}$ denotes the "objective's speed" (or how fast the position of the j-th objective changed considering the UAV as an inertial referential), and $\theta^{L_i}_{\text{obj}_j}$ is the "heading" of this objective (the direction in which this objective changed in relation to the UAV), both at any time where t. The UAVs can operate at varying altitudes relative to ground level. At any time t and for any given layer (except the cloud, which remains fixed as it is not a UAV), the position of a UAV is represented as $P_{L_i}(t) \in \mathbb{R}^3$. To simplify the mathematical model, UAVs are assumed to navigate between objectives using vectorial speed control while maintaining a constant altitude, as described by the following equations:

$$\begin{cases} \theta_{L_i}(t) = \arctan\left(\frac{P_{L_i}[y](t_i) - P_{L_i}[y](t)}{P_{L_i}[x](t_i) - P_{L_i}[x](t)}\right) \\ V_{L_i}(t) = \frac{||P_{L_i}(t) - P_{L_i}(t_i)||}{t - t_i} \\ P_{L_i}[x](t_{i+1}) = P_{L_i}[x](t) + V_{L_i}(t) \cdot \cos\theta_{L_i}(t) \cdot \Delta t \\ P_{L_i}[y](t_{i+1}) = P_{L_i}[y](t) + V_{L_i}(t) \cdot \sin\theta_{L_i}(t) \cdot \Delta t, \end{cases} \tag{20}$$

in which $V_{L_i}(t)$ and $\theta_{L_i}(t)$ denote the instant velocity and the instant heading of any UAV in a given layer, respectively. It is assumed that $C_{L_i}(t)$ is the trajectory coverage at time t of navigated points P_{L_i} of a UAV in relation to its $P^{L_i}_{\text{obj}_j}$ objective trajectory points. The coverage represents the trajectory traveled by a robot i between two consecutive moments t and $t + \tau$, where τ is a fixed time step. As ϱ_{L_i} is the defined

[4]In practice, an altitude range is set in the algorithm startup configuration, and the UAV control just corrects the UAV height if the altitude position is above or below the superior or inferior limits, respectively.

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

trajectory of waypoints performed (or to be performed) by an i-th UAV connected to ARCog-NET at layer L in an observed time t (being t_i the previous time), and $P_\kappa(t)$ is the κ-th point of this trajectory, the total coverage $C_{L_i}^{\text{total}}$ of the trajectory can be described as:

$$\begin{cases} C_{L_i}^{\kappa}(t-t_i) = \Delta P_\kappa(t-t_i) \\ C_{L_i}^{\text{total}} = \frac{1}{n_p^{L_i}} \sum_{k=0}^{n_p^{L_i}} C_{L_i}^{\kappa}(t-t_i) \end{cases}, \tag{21}$$

where $n_p^{L_i}$ is the number of points observed in a time interval. This way, $C_{L_i}^{\text{total}}$ will depend on how efficiently $n_p^{L_i}$ is defined, especially when navigation events happen (which could be a detected obstacle by an edge agent or a command to change trajectory by a superior layer). Thus, $C_{L_i}^{\text{total}}$ can be understood as the reward for a navigation trajectory decision, thus:

$$C_{L_j' L_i}^{PP}(t) = C_{L_i}^{\text{total}}. \tag{22}$$

Selecting a new trajectory in response to an event that results in a navigation task is determined by its expected impact on mission success, as defined by the optimal coverage process in Equation (21). If a processing layer achieves the best coverage at a given time t, its corresponding weight $w_{L_i}^{L_j'}$ is reinforced. Conversely, suboptimal decisions are penalized, leading to adjustments in the weight tensor ω_{L_i}. This reinforcement learning mechanism continuously refines decision-making, prioritizing more effective solutions as the operation progresses.

When an event happens, by default, the method to calculate new trajectory points in the present work is the PSO algorithm [61]. ARCog-NET integrates a cognitively enhanced PSO algorithm for trajectory planning, significantly improving both processing time and trajectory accuracy over conventional PSO. The UAVs store and utilize previously executed trajectories to construct new ones dynamically, reducing computational overhead when recalculating paths for new mission events. By incorporating reinforcement learning into the PSO framework, ARCog-NET achieves a more adaptive and efficient trajectory optimization strategy. The RL-enhanced PSO continuously adjusts its parameters based on real-time navigation feedback, improving convergence speed and solution quality.

In summary, when a navigation task-resulting event occurs for a UAV i in layer L, it attempts to compute a trajectory adjustment to optimize coverage. The UAV first consults its weight tensor ω_{L_i} to identify whether a previously recorded weight matrix $W_{L_i}(t)$ contains a weight $w_{L_i}^{L_j'}(t)$ that surpasses others in effectiveness. If such a weight is found, it references a prior event processed by node L_j' and retrieves relevant historical decisions from the knowledge base tensor κ_{L_i}. This lookup directs the UAV to past coverage matrices $H_{L_i}(t)$ to inform its current trajectory choice. The implemented path planning algorithm for handling navigation events is detailed in Algorithm 2.

If the previously recorded decision yields better coverage than the newly computed result, the UAV will adopt the prior solution, update its weight matrix, store it in the knowledge base, and adjust the weight tensor, historical matrix, and learned policies accordingly. Otherwise, the UAV can either continue searching the knowledge base, use the newly computed result, or escalate the decision request to a higher-layer entity. This hierarchical decision process extends up to the cloud level, where, as a last resort, the human supervisor may intervene. Once the decision is executed, the UAV evaluates its real-world effectiveness by verifying coverage outcomes. If the chosen decision improves coverage, it is reinforced; otherwise, it is penalized, ensuring continuous optimization of the decision-making framework.

At the edge layer, the process begins with identifying an agent's unique tag number, followed by retrieving its current position and objectives. These inputs are used to compute new trajectory objectives aimed at optimizing coverage while fulfilling mission goals. The proposed solution is then evaluated against the existing knowledge base. If a superior solution is found, it is added to the solution tensor and stored in the knowledge base. The agent then executes the action and assesses its effectiveness, adjusting the action weight accordingly. If no improved solution is found, but a previously executed action is deemed suitable, the agent may repeat it, update its weight, and store it as a new entry in the knowledge base. If no suitable solution exists, the edge agent escalates the request to the fog layer coordinator for further computation.

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

Algorithm 2 Particle Swarm Optimization for ARCog-NET navigation event trajectory path planning.

Input: $T_r = T_{\text{nav}}$: A Path Planning (PP) task is allocated
ϱ_{L_i}: Initial trajectory positions of UAV i
$V_{\text{obj}_j}^{L_i}$: Initial velocities of trajectory points
κ_{L_i}: Current knowledge base and ω_{L_i}: Current decision weights history
t_i: Previous time and t: Current time
$C_{\min}$: Minimum coverage threshold, $\max_{it}$: Maximum number of iterations and $\min_{err}$: Minimum error threshold
while Error $>$ $\min_{err}$ and Iterations $\leq$ $\max_{it}$ **do**
  for P in ϱ_{L_i} **do**
    $C_{L_i}(t - t_i) = \Delta P(t - t_i)$
    if $C_{L_i}(t - t_i) \geq C_{\min}$ **then**
      Trajectory **keep** (P)
    end
    else
      Trajectory **drop** (P)
      New_Point = empty
      for a in ω_{L_i} **do**
        for b in $W_{L_i}^{PP}$ (rows) **do**
          for c in $W_{L_i}^{PP}$ (columns) **do**
            Get BestFitting($w_{L'_j L_i}^{PP}$)
          end
        end
      end
      $P \leftarrow \kappa_{L_i}[a]\{H_{L_i}[b, c]\}$
    end
  end
  Update Trajectory points:
  for κ in $range(length(P))$ **do**
    Get best fitting κ-th P:
    $V^{L_i}{}_{\text{obj}_{j\,\kappa}}(t) = \frac{||P_{\text{obj}_{j\,\kappa}}^{L_i}[x](t) - P_{\text{obj}_{j\,\kappa-1}}^{L_i}(t)||}{t - t_i}$
    $\theta_{\text{obj}_{j\,\kappa}}^{L_i}(t) = \arctan\left(\frac{P_{\text{obj}_{j\,\kappa}}^{L_i}[y](t) - P_{\text{obj}_{j\,\kappa-1}}^{L_i}[y](t)}{P_{\text{obj}_{j\,\kappa}}^{L_i}[x](t) - P_{\text{obj}_{j\,\kappa-1}}^{L_i}[x](t)}\right)$
    $P_{\text{obj}_{j\,\kappa}}^{L_i}[x](t) = P_{\text{obj}_{j\,\kappa-1}}^{L_i}[x](t_i) + V_{\text{obj}_{j\,\kappa}}^{L_i}(t) \cdot \cos\theta_{\text{obj}_{j\,\kappa}}^{L_i}(t) \cdot \Delta t$
    $P_{\text{obj}_{j\,\kappa}}^{L_i}[y](t) = P_{\text{obj}_{j\,\kappa-1}}^{L_i}[y](t_i) + V_{\text{obj}_{j\,\kappa}}^{L_i}(t) \cdot \sin\theta_{\text{obj}_{j\,\kappa}}^{L_i}(t) \cdot \Delta t$
    if $|C_{L_i\,\kappa}(t) - C_{L_i\,\kappa-1}(t)| \leq$ Error **then**
      Error $\leftarrow |C_{L_i\,\kappa}(t) - C_{L_i\,\kappa-1}(t)|$
    end
    $C_{L'_j L_i}^{PP}(t) \leftarrow C_{L_i\kappa}(t)$
  end
  Iterations $\leftarrow$ Iterations $+1$
end
$t_i \leftarrow t$
Create new $H_{L_i}(t)$ matrix incorporating the new $C_{L'_j L_i}^{PP}(t)$
Incorporate new $H_{L_i}(t)$ to knowledge base κ_{L_i}
Create new $W_{L_i}(t)$ matrix incorporating modified $w_{L'_j L_i}^{PP}(t)$
Incorporate new $W_{L_i}(t)$ to weights history ω_{L_i}
Output: $\varrho_{L_i} = P_{\text{optimized}}$: Optimized trajectory positions
κ_{L_i}: New knowledge base and ω_{L_i}: New decision weights history

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

In parallel, the fog layer follows a similar process. It checks for incoming requests from edge agents and identifies relevant tags, whether from its own group or other fog coordinators. Based on whether it is handling an edge request or responding to an internal event, it computes new objectives to optimize trajectory coverage. The effectiveness of these objectives is evaluated against their knowledge base, with efficient solutions being reinforced and incorporated into the knowledge base. Upon completing a cognitive processing request or executing an action, the fog layer reassesses the effectiveness of the decision, updating weights and historical records before concluding its cognition process.

Simultaneously, the cloud layer processes incoming requests and commands. When a layer L tag is detected with a path planning solution request, it collects relevant UAV data and objectives to compute new trajectory solutions to optimize flight coverage. These solutions are assessed against the cloud's extensive knowledge base. If a superior solution is identified, it is integrated into the solution tensor, stored in the knowledge base, and executed. Otherwise, a previously stored solution may be reused, as the cloud maintains a more comprehensive knowledge base. The cloud layer finalizes its decision-making process by evaluating the effectiveness of the executed solution, updating weights, saving the decision across all layers, and transmitting the optimized command back to the requester.

As UAVs must navigate dynamic environments while avoiding collisions, ARCog-NET enables rapid path adaptation in response to environmental changes, ensuring uninterrupted operation with minimal human intervention. In swarm applications, each UAV autonomously determines its trajectory while maintaining formation integrity and avoiding both static and dynamic obstacles. If necessary, UAVs can request trajectory recalculations from superior network layers. This functionality is crucial for missions in complex environments such as urban areas and cluttered terrains, where real-time adaptability is essential. Additionally, ARCog-NET compares newly generated PSO trajectories with previously recorded decisions, allowing UAVs to replicate optimal past solutions, further improving efficiency through historical learning. This path planning algorithm is compared with other models from literature as detailed in Section 2.

3.4. Formation control module

The ARCog-NET framework extends its cognitive decision-making processes for navigation to formation control, ensuring coordinated swarm behavior while maintaining compatibility with the path planning models defined earlier. If UAVs are flying close to each other or performing similar paths, ARCog-NET can switch from path-planning flight to formation control, for example. The formation control opportunity can be triggered by any layer, but only the fog or cloud layer can command it.

In other words, if an edge agent detects that its cluster is flying close to each other and in similar trajectories, it sends a formation control request to superior layers, which will decide if this is an optimal solution or not and respond with formation control commands to its agents or not. If a fog coordinator detects a formation control opportunity within its own cluster, it will send formation control commands to each agent. If a fog coordinator detects a formation control opportunity between its own cluster and another coordinator cluster, a request is sent to the cloud to initiate formation control between the two groups commanded by the cloud. If the cloud layer detects a formation control opportunity at any level, it will send commands to initiate it. And finally, if the HITL commands formation control through the cloud, it will send the commands to the respective cluster, group of clusters, or the whole network.

Once again let the position of the i-th UAV in layer L at time t be denoted as $q_{L_i}(t) = [x_{L_i}(t), y_{L_i}(t), z_{L_i}(t)]^T \in \mathbb{R}^3$, consistent with the trajectory definitions in Equation (20). The desired formation is defined through relative positions between UAVs, where $q^d_{L'_j}(t)$ represents the target position of UAV j of layer L' relative to UAV i of layer L. The formation error for UAV i aggregates discrepancies between actual and desired relative positions across its neighbors $\mathcal{N}_i(t)$:

$$e_{L_i}(t) = \sum_{j \in \mathcal{N}_i(t)} \left((q_{L_i}(t) - q_{L'_j}(t)) - (q^d_{L_i}(t) - q^d_{L'_j}(t)) \right), \tag{23}$$

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

where $\mathcal{N}_i(t)$ dynamically updates based on communication range or mission requirements. To minimize this error, the control input $u_{L_i}(t)$ integrates proportional feedback, historical decisions from the knowledge base κ_{L_i}, and innovations from Particle Swarm Optimization (PSO):

$$u_{L'_jL_i}(t) = K_p e_{L_i}(t) + \sum_{k \in L'_j} w^{FC}_{kL_i}(t) u^{\text{hist}}_{kL_i}(t) + \delta_{L_i}(t). \tag{24}$$

Here, K_p is a proportional gain matrix optimized via PSO, $w^{FC}_{L'_jL_i}$ are softmax weights favoring historically effective solutions, and $\delta_{L_i}(t)$ represents PSO-driven adjustments. The weights $w^{FC}_{L'_jL_i}$ are calculated using:

$$w^{FC}_{L'_jL_i} = \frac{e^{-\beta \| e_{L_i}(t) - e^k_{L_i}(t) \|}}{\sum_m e^{-\beta \| e_{L_i}(t) - e^m_{L_i}(t) \|}}, \tag{25}$$

where $e^k_{L_i}(t)$ denotes past formation errors stored in κ_{L_i}. The PSO innovation term $\delta_{L_i}(t)$ minimizes a cognitive cost function balancing current error reduction and historical consistency. Therefore, $\mathcal{J}_p(t)$ can be described as formation stability, which is the necessity of adjusting UAV positions to keep the designed formation:

$$\mathcal{J}_p(t) = \| e_{L_i}(t) + \Delta q^p_{L_i} \|^2 + C^{FC}_{L'_jL_i} \sum_k w^{FC}_{kL_i} \| \Delta q^p_{L_i} - \Delta q^{k,\text{hist}}_{L_i} \|^2, \tag{26}$$

where $\Delta q^p_{L_i}$ represents candidate formation adjustments, and $C^{FC}_{L'_jL_i}$ is the formation control coverage, representing the adherence to historical solutions. The PSO velocity update rule incorporates both local and global best solutions:

$$\Delta q^p_{L_i}(t+1) = \Delta q^p_{L_i}(t) + C^{FC}_{L'_jL_i} \cdot (\Delta q^{p,\text{best}}_{L_i} - \Delta q^p_{L_i}(t)) + C^{FC}_{L'_jL_i} \cdot (\Delta q^{g,\text{best}}_{L_i} - \Delta q^p_{L_i}(t)), \tag{27}$$

with $\Delta q^{g,\text{best}}_{L_i}$ drawn from $H_{L_i}(t)$ in κ_{L_i}. The knowledge base κ_{L_i} is augmented to store formation-specific data, which is the historical formation control coverages $C^{FC}_{L'_jL_i}$ components of the H_{L_i} matrices as described in Equation (4), ensuring compatibility with the trajectory coverage metrics in Equation (21).

During formation-disrupting events (e.g., obstacles, UAV failures), the framework triggers a hierarchical response. At the edge layer, UAVs compute local adjustments via PSO and compare them to κ_{L_i}, rewarding effective solutions by updating $w^{FC}_{L'_jL_i}$. Conflicting proposals escalate to the fog layer, which resolves them by optimizing global formation stability:

$$\max_{\{\Delta q_{L_i}\}} \sum_{i=1}^{N} \mathcal{J}_p(t) \quad \text{subject to} \quad \| q_{L_i}(t) - q_{L'_j}(t) \| \geq d_{\text{safe}}, \tag{27}$$

where d_{safe} enforces collision avoidance. The cloud layer intervenes for critical failures via human-in-the-loop (HITL) commands, updating κ_{L_i} with new formation templates. The existing path planning Algorithm 2 is extended to handle formation events by incorporating $e_{L_i}(t)$ as input and modifying the PSO loop to optimize $\mathcal{J}_p(t)$. For example, after line 37, replacing line 38 of Algorithm 2, the following steps are added.

This approach ensures seamless integration of formation control into ARCog-NET's cognitive architecture, leveraging existing components like the knowledge base κ_{L_i} and reinforcement learning weights ω_{L_i} while preserving scalability across edge, fog, and cloud layers. The performance of ARCog-NET formation control is evaluated against state-of-the-art formation control algorithms as detailed in Section 2.

4. Results and discussion

The ARCog-NET framework has been validated through extensive simulations incorporating real PX4 autopilot firmware via MAVROS. This approach ensures that the simulated UAV behavior closely mirrors actual flight dynamics, communication constraints, and decision-making latency, thereby increasing

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

Algorithm 3 ARCog-NET PSO Extension for Formation Control.

Input: $e_{L_i}(t)$: Formation error from Equation (23)
H_{L_i}: Historical formation adjustments from κ_{L_i}
W_{L_i}: Historical weightings adjustments from ω_{L_i}
while $\|e_{L_i}(t)\| > \min_{err}$ and Iterations $\leq \max_{it}$ **do**
    Compute $\Delta q^{p}_{L_i}$ via PSO velocity update
    Evaluate $\mathcal{J}_p(t)$ using Equation (26)
    Update $w^{FC}_{L'_j L_i}$ using softmax weights
    if $\mathcal{J}_p(t) > \mathcal{J}_p^{\text{best}}$ **then**
        Update $\Delta q^{g,\text{best}}_{L_i}$ and $C^{FC}_{L'_j L_i}$
    end
    Iterations $\leftarrow$ Iterations +1
end

the reliability of the results. Unlike conventional simulations that rely on simplified kinematic models, the setup integrates low-level flight control commands and real-world sensor feedback loops, enabling a high-fidelity representation of UAV swarm operation.

To ensure an easy transition from simulation to real-world deployment, a hardware configuration has been outlined that mirrors the simulated environment (as demonstrated in the previous section). The selected hardware includes PX4-based UAVs with onboard edge computing capabilities, fog coordinators operating via ROS-compatible ground units, and cloud-based processing for reinforcement learning-based decision optimization. This hardware architecture was carefully chosen to align with the computational and communication constraints observed in MAVROS-based simulations, ensuring that the framework remains feasible under real-world conditions.

Despite the high realism of the simulation framework, the next critical step is validating ARCog-NET through field experiments with actual UAVs. While the simulation results demonstrate significant improvements in path planning, formation control, and adaptive task allocation, real-world deployments will allow additional factors to be analyzed, such as sensor noise, network interference, and hardware-induced latency. Future work will focus on implementing these experiments to further confirm the robustness of ARCog-NET in diverse operational scenarios, including offshore missions and urban navigation. By establishing a direct correspondence between simulation and hardware capabilities, this study provides a strong foundation for real-world implementation. The MAVROS-based validation ensures that ARCog-NET is not merely a theoretical framework but a practically deployable solution for UAV swarm coordination and autonomy. *A basic ARCog-NET simulation is available* (https://github.com/gabryelsr/ARCog-NET.git).

4.1. Environment 1: offshore environmental monitoring

This mission aims to evaluate ARCog-NET performance when applied in an outdoor obstacle-free unknown environment by simulating an offshore ocean water surveillance mission scenario. The robots and the network were configured to take off from a helicopter deck placed on an oil production facility and scan the water, searching for oil spills. This time, the strategy programmed in the PSO loop was to deploy one UAV at a time to fly "zigzag" trajectories in order to cover a large area, avoiding colliding with each other. When operating with a single vehicle, it is configured as an edge or fog node (there is no difference in this case) and flies alone, building its knowledge base of trajectories and finding oil spills along the way. If the vehicle doesn't find anything, the loop (local or cloud) will increase the search area. For more than one UAV, as one edge vehicle starts scanning the ocean, it sends the collected data to its

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

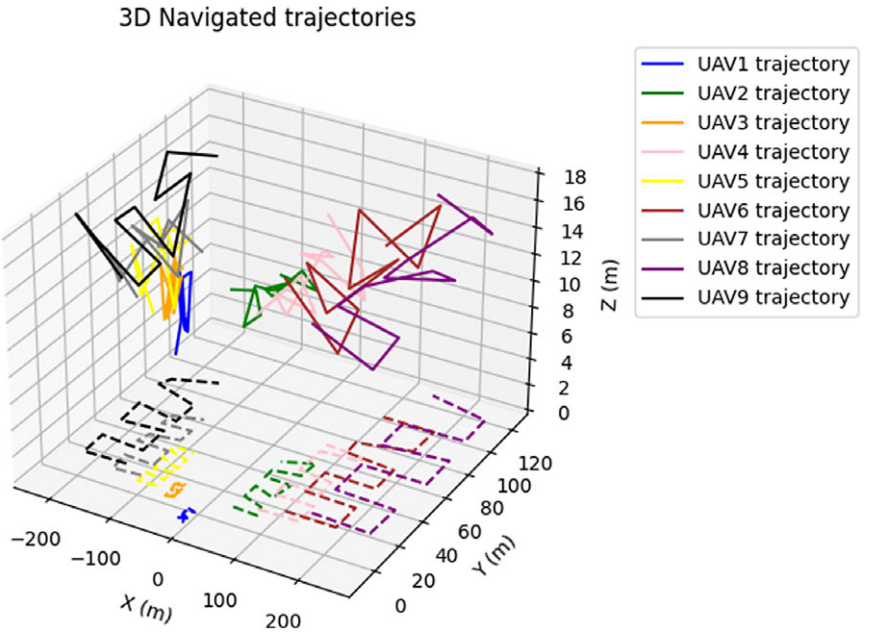

Figure 7. *Group of 10 UAVs flying to search oil spills in offshore simulation.*

Table III. *Path planning comparison in offshore scenario.*

Metric	Classical PSO [62]	A∗ [63]	Dijkstra [64]	RRT [65]	ARCog-NET
Convergence Time (s)	12	8	10	15	9
Trajectory Coverage (%)	85	90	92	88	91
Computational Cost (ms)	150	200	180	120	130
Adaptability (ms)	75	70	65	80	78
Scalability (number of UAVs)	500	300	400	600	1000

fog coordinator, counting spills found and sending other agents to other points, increasing the search area if necessary. Also, the cloud layer will act again as a supervisor layer, commanding the whole system and processing flight decisions. A *video simulation example using 10 UAVs with 3D graphics is presented* (https://youtu.be/vJP7v9KbK0I?si=VMh4cgf5JZRfABSJ), and Figure 7 presents the flight behavior of this video.

In the video, the group of 10 UAVs (8 edge agents separated into two groups, with one fog coordinator each) found all oil spills randomly placed over the water surface (26 spots) in simulation, taking 4 minutes and 32 s. By navigating the routes presented in Figure 7, the swarm covered an area of 48,000 m^2. Compared to state-of-the-art methods, the results of ARCog-NET for this scenario are presented in Table III.

For this scenario, ARCog-NET demonstrates competitive performance in path planning, balancing efficiency, adaptability, and scalability. It achieves a convergence time of 9 s, faster than classical PSO (12 s) and RRT (15 s) but slightly slower than A∗ (8 s). This highlights ARCog-NET's ability to optimize path planning in dynamic environments while maintaining computational efficiency. In trajectory coverage, ARCog-NET achieves 91%, outperforming classical PSO (85%) and Rapidly Exploring Random Trees (RRT) (88%) but slightly trailing Dijkstra (92%). This indicates robust path optimization, though Dijkstra remains optimal for static environments.

ARCog-NET's computational cost is 130 ms, lower than A∗ (200 ms) and Dijkstra (180 ms), but slightly higher than RRT (120 ms). This reflects a trade-off between computational efficiency and

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

Table IV. *Formation control comparison in offshore scenario.*

Metric	DMPC [66]	DRL-Based [67]	Consensus-Based [68]	ARCog-NET
Formation Error (*m*)	0.15	0.20	0.10	0.12
Energy Consumption (J/formation)	50	60	40	45
Adaptability (ms)	80	85	75	78
Communication Overhead (%)	12	15	8	9
Robustness to Delays (%)	85	80	90	88

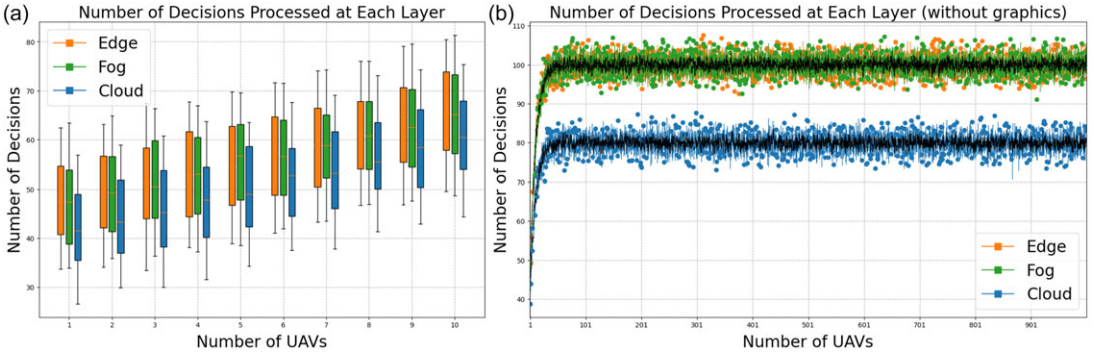

Figure 8. *Number of decisions processed on each ARCog-NET layer in the offshore simulation.*

adaptability. In adaptability, ARCog-NET scores 78 ms, outperforming classical PSO (75 ms) and Dijkstra (65 ms) but slightly below RRT (80 ms). This underscores its ability to handle dynamic scenarios effectively. Finally, ARCog-NET supports 800 UAVs, surpassing classical PSO (500 UAVs), A∗ (300 UAVs), and Dijkstra (400 UAVs), though RRT scales to 600 UAVs. This scalability makes ARCog-NET suitable for large-scale deployments.

Overall, ARCog-NET provides a balanced solution for path planning, excelling in scalability and adaptability while maintaining competitive efficiency and computational performance. It is suitable for this dynamic, large-scale UAV operation in open space, obstacle-free scenario. The formation control asset was triggered very few times, but enough to measure and compare results as presented in Table IV.

The proposed architecture demonstrates strong performance in formation control, balancing precision, energy efficiency, and adaptability. It achieves a formation error of 0.12 m, outperforming Distributed Model Predictive Control (DMPC) (0.15 m) and Deep Reinforcement Learning (DRL)-based methods (0.20 m) but slightly trailing consensus-based approaches (0.10 m). In energy consumption, ARCog-NET uses 45 J/formation, more efficient than DMPC (50 J) and DRL-based methods (60 J) but slightly higher than consensus-based approaches (40 J). ARCog-NET's adaptability is 78 ms, better than consensus-based methods (75 ms) but slightly below DRL-based approaches (85 ms). Its communication overhead is 9%, lower than DMPC (12%) and DRL-based methods (15%), but slightly higher than consensus-based approaches (8%). Finally, ARCog-NET demonstrates 88% robustness to delays, outperforming DMPC (85%) and DRL-based methods (80%) but slightly below consensus-based approaches (90%).

Being an open environment free of obstacles, it is a relatively simple situation. In this case, ARCog-NET structure is not so demanded to process complex flight decisions, the robots are not submitted to high interference rates as the signals have free space to propagate, less data to pack, compress, and stream. The result is a more fluid overall performance of the architecture. To exemplify this observed behavior, the first step is to analyze each layer of the computer load of decisions (task allocation as response to events) processed in Figure 8 based on the presented RL cognitive algorithm.

The average type of decisions is compiled and presented in the Pareto graph of the Figure 9.

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

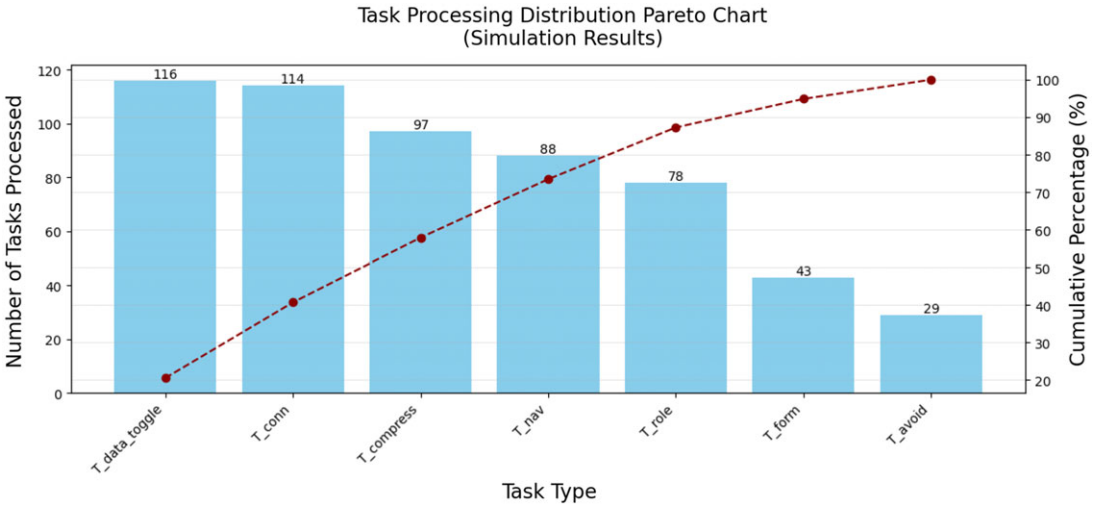

Figure 9. *Pareto graph of the average number of tasks by type in the offshore simulation.*

Table V. *Task allocation comparison in offshore scenario.*

Metric	Market-Based [51]	Heuristic Methods [53]	Bio-Inspired Methods [50]	Reinforcement Learning [54]	ARCog-NET
Efficiency (%)	75	85	88	90	88
Energy Consumption (J/task)	120	100	95	90	95
Communication Overhead (%)	15	10	9	8	7
Adaptability (ms)	70	80	82	85	82
Scalability (number of UAVs)	500	1200	900	800	1000

The architecture behavior for this scenario reveals that less cloud computation was required, as the edge and fog robots could deal with more simple decisions locally or within their layer pairs. An interesting result is the saturation behavior of the network, in which the average number of decisions processed on each layer doesn't increase after a count of approximately 50 nodes connected. This has an impact on network scalability for this scenario, as edge, fog, and cloud layers seem to reach their peak of decisions throughout the whole simulation time after 50 robots are connected, indicating that there is no need for using more UAVs to accomplish mission goals with ARCog-NET in this case. The Pareto graph also demonstrated that the architecture had to deal with fewer flight decisions, with communications and data handling being the most relevant factors (probably because of the large area the UAVs were scanning). Comparing task allocation with other methodologies presented the following results (Table V):

In this scenario, ARCog-NET achieves 88% efficiency, closely matching reinforcement learning (90%) and bio-inspired methods (88%), while outperforming market-based (75%) and heuristic approaches (85%). It minimizes energy consumption to 95 J/task, comparable to bio-inspired methods (95 J/task) and better than market-based methods (120 J/task). ARCog-NET excels in communication overhead at 7%, outperforming all other methods, including bio-inspired (9%) and reinforcement learning (8%). Its adaptability is 82 ms, slightly lower than reinforcement learning (85 ms) but better than market-based (70 ms). ARCog-NET supported 1000 UAVs (although it's not useful to scale so much in this scenario, as demonstrated earlier), surpassing bio-inspired (900 UAVs) and reinforcement learning

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

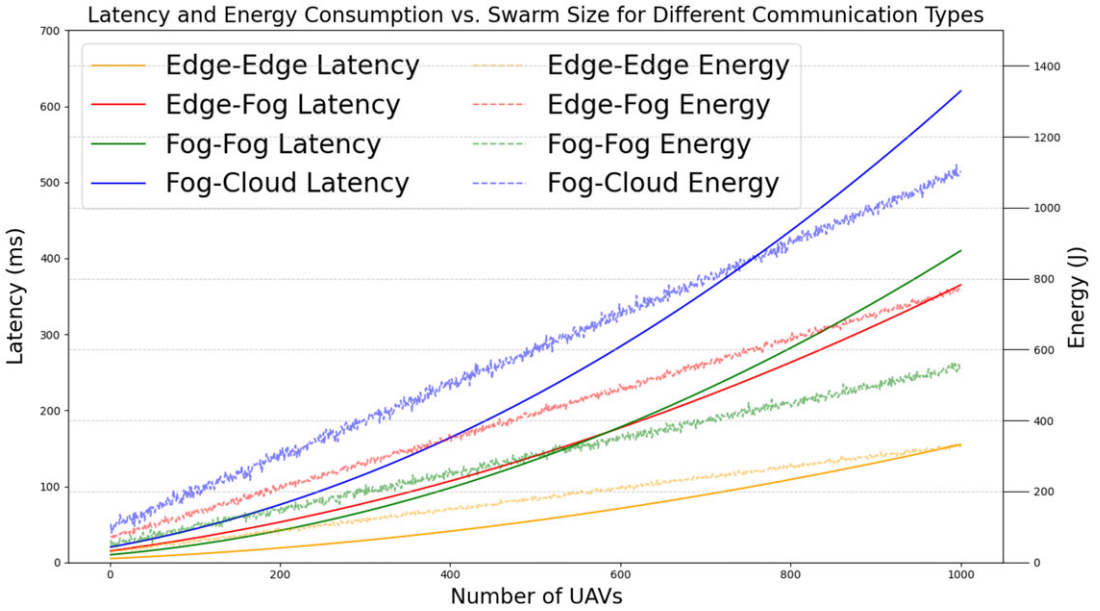

Figure 10. *Average ARCog-NET latency by number of UAVs at each time in the offshore simulation.*

(800 UAVs), making it highly scalable for large deployments. Overall, ARCog-NET balances efficiency, energy use, and scalability, making it a robust solution for UAV swarm task allocation. With the saturation of decision processing, latency and energy become scalability concerns and are represented in Figure 10.

As expected, both energy and latency will increase as more robots are added to the architecture. Edge energy and latency are smaller than other communication pipelines within ARCog-NET, as edge communication is usually less heavy, with agents closer to each other. Latency and energy will also be a result of how the architecture will process the task allocation. With this open scenario, there's not much path planning changing (for example, for avoiding an obstacle) or even formation controls being triggered. ARCog-NET can also reconfigure its structure during the mission (edge agents changing groups, becoming coordinators, cloud demanding HITL decision, size of groups, etc.), which may impact energy and latency. Edge-to-fog and Fog-to-fog communications have similar latency outputs, but edge-fog demands more energy with the latency gain payback.

In other words, for this scenario, investing in edge-to-fog groups will result in less latency than operating only with a fog computing strategy. The cloud layer presents the greater latency and energy due to its distance from UAVs and radio power. Overall, increasing UAV numbers in this scenario doesn't result in better mission goal accomplishment as the decisions processed will saturate after 50 robots, and latency and power consumption keep increasing as more robots are added.

Streamed data, the major task allocator in this scenario, was basically images (the larger messages) and positions, with some requests and commands eventually (that don't take up much bandwidth). When running some tests, different results were observed in spill detection behavior as the images didn't have many edges and textures (it was just plain water texture most of the time). Mathematically, using the ARCog-NET pre-built data compression autoencoder model [60] in this simulation doesn't result in great PSNR, SSIM, and MS-SSIM when comparing raw captured images to compressed images.

In practice, however, the cloud server supervisor HITL will struggle more with the compressed images when using the successive compression method to reduce data, as presented in Figure 11. In other words, as the image has little difference at each frame and poor texture, compressing it is unnecessary and hampers mission objectives so that this compression asset of ARCog-NET can easily be put aside in this kind of mission where the network operates with a good performance.

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

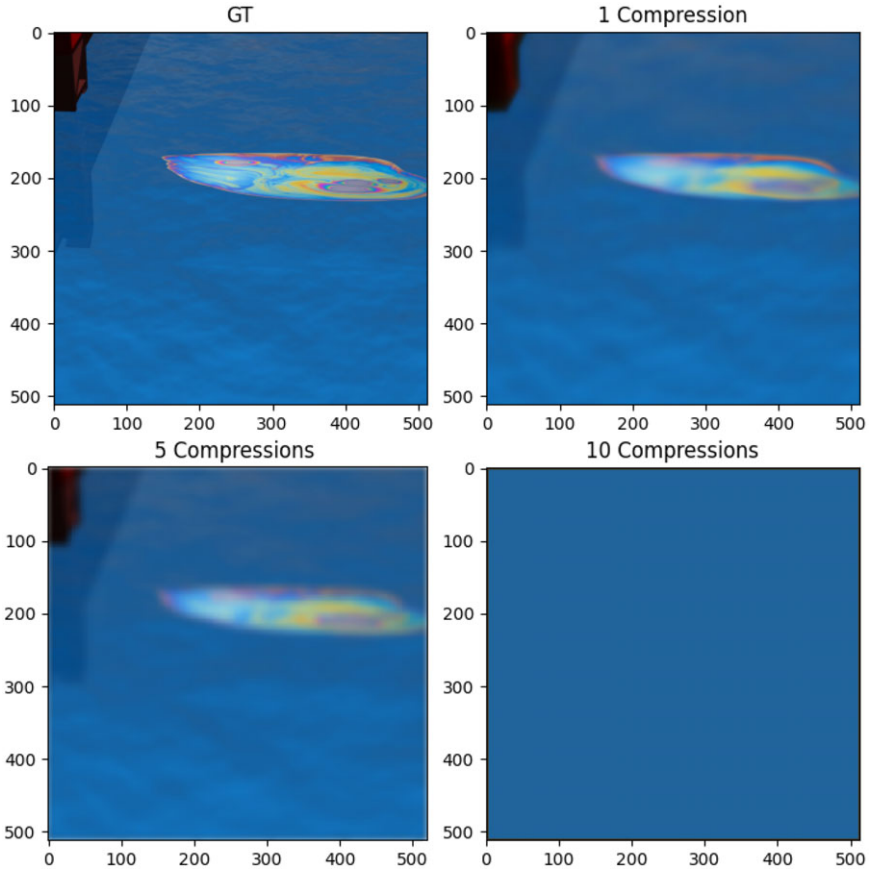

Figure 11. *Compression example of a picture of an oil spill found at the water surface.*

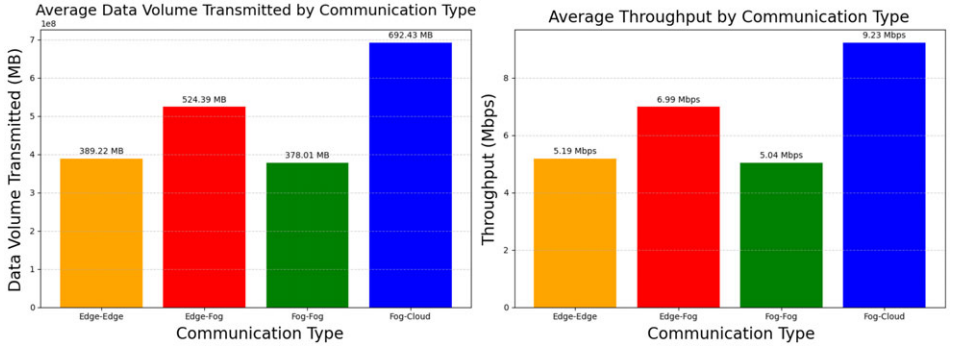

Figure 12. *Average ARCog-NET transmitted data and throughput in offshore simulation.*

Even not using too much compression in this scenario simulations, the volume of data traffic within the network is relatively low, as detailed in Figure 12. The main factor for this is that, in open space, the edge and fog agents don't exchange so much navigation data between themselves or the cloud, and besides, this type of message has smaller sizes. Also, the UAVs (mostly edge agents) demanded fewer requests for superior layers (fog or cloud) to cognitive reinforcement learning navigation decisions, dealing with events locally, which decreased streamed messages for group control, as shown in Figure 8.

And finally, the packet loss and network availability results are presented in Figure 13.

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

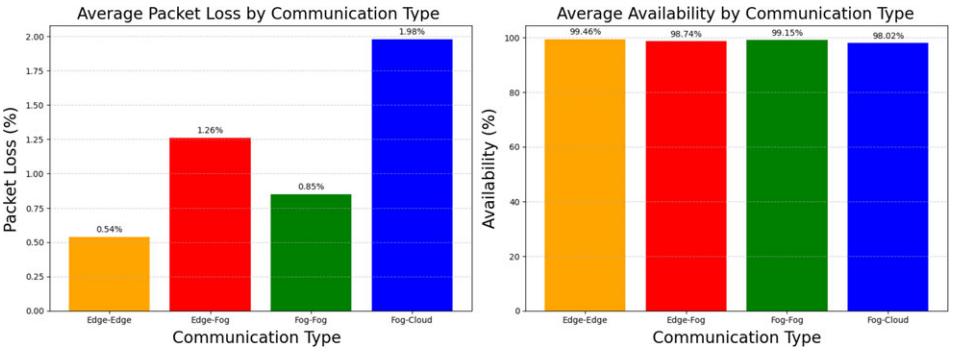

Figure 13. *Average ARCog-NET packet loss and throughput in offshore simulation.*

The results demonstrate a clear trade-off between data volume, throughput, packet loss, and availability across the different communication types in ARCog-NET. Edge-to-edge communication exhibits the lowest data volume and throughput, which aligns with its lower latency and energy consumption, as shown in the latency and energy graphs. This makes Edge-to-edge communication ideal for localized, low-latency tasks such as formation control and obstacle avoidance, where real-time coordination is critical. However, its limited throughput and higher packet loss (compared to Edge-to-fog) suggest it is less suitable for data-intensive tasks. Edge-fog communication shows moderate data volume and throughput, balancing energy efficiency and latency, making it suitable for tasks requiring both computational offloading and real-time responsiveness, such as role changes and data compression.

Fog-to-fog communication demonstrates higher data volume and throughput, with slightly increased latency and energy consumption compared to Edge-to-edge and edge-fog. This makes it effective for tasks requiring significant data exchange between UAVs, such as collaborative path planning or shared mission updates. However, its higher packet loss and lower availability compared to Edge-to-edge highlight potential reliability challenges in dynamic environments. Fog-to-cloud communication exhibits the highest data volume and throughput, supporting large-scale data aggregation and processing, such as mission analytics or global task reallocation. However, its significantly higher latency and energy consumption, as shown in the latency and energy graphs, make it less suitable for time-sensitive or energy-constrained operation.

In this open, obstacle-free scenario covering a large area, ARCog-NET effectively leverages the hierarchical architecture to distribute tasks based on their requirements. Edge-to-edge communication is prioritized for real-time, low-latency tasks, while Fog-to-cloud communication handles data-intensive, non-time-critical tasks. The intermediate edge-fog and Fog-to-fog layers provide a balance, enabling efficient resource utilization and scalability. The results highlight ARCog-NET's adaptability in managing diverse task requirements, ensuring optimal performance across latency, energy, data volume, and reliability metrics. However, the higher packet loss and lower availability in Fog-to-fog and Fog-to-cloud communications suggest the need for robust error correction and redundancy mechanisms to enhance reliability in large-scale deployments. Overall, ARCog-NET demonstrates strong potential for open-environment applications, offering a scalable and efficient framework for UAV swarm coordination and task allocation.

4.2. Environment 2: provide mobile phone coverage to a neighborhood

In this scenario, only graphic simulations with groups of 10 UAVs were executed, being 10 simulations in total, and another 1000 simulations from 1 to 1000 vehicles without graphic interface to test ARCog-NET scalability. The graphic simulation represented a city region with a critical mobile communication problem, and the UAV swarm was deployed to act as communication towers to allow client connection while the main system is restored. This scenario represents an outdoor partially known scenario, as the ARCog-NET needs to position the UAV groups strategically to provide service to clients and not

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

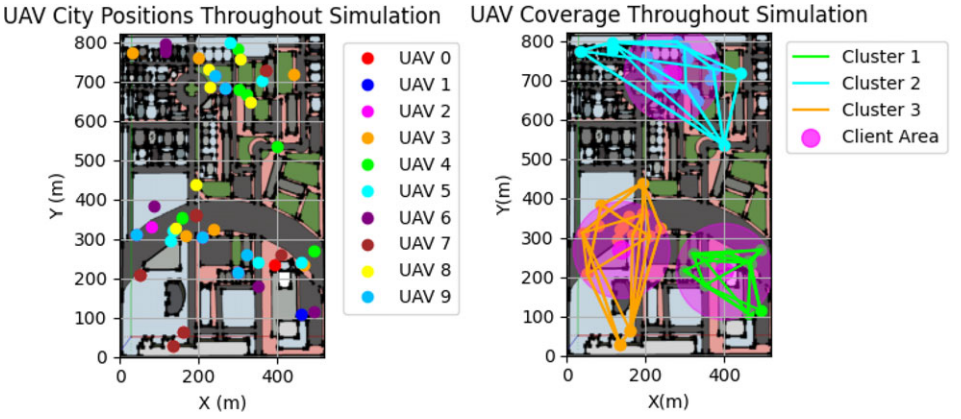

Figure 14. *2D points occupied by each UAV in city simulation and cluster coverage.*

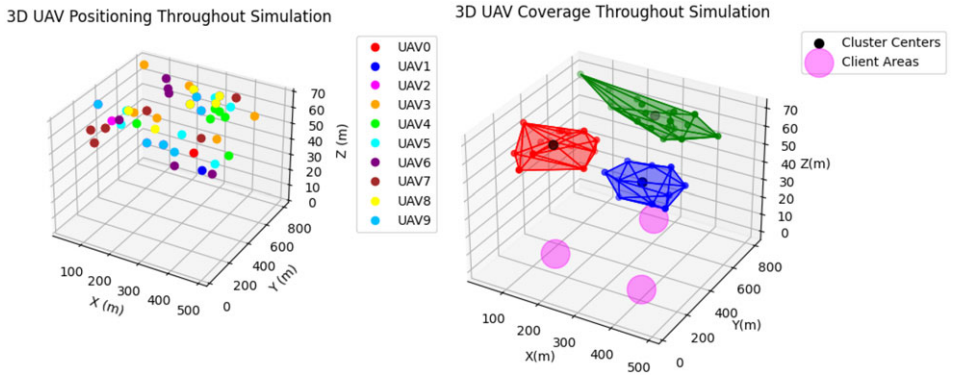

Figure 15. *3D points occupied by each UAV in city simulation and cluster coverage.*

collide between themselves or other city structures like buildings or trees. No connection aspects were studied in this section, as only ARCog-NET behavior is the subject of this research. The focus is how the UAVs cover the client demand areas and how quickly they process events of repositioning and flying to new points. Figure 14 shows the points occupied by each UAV and the cluster coverage of these points throughout the simulation time in 2D.

In the same way, Figure 15 shows the points occupied by each UAV and the cluster coverage of these points throughout the simulation time in 3D. The client concentration hotspots are represented within circular areas that are randomly placed in the neighborhood and change over time throughout the simulation. The UAVs then adapt their positions to guarantee continuous operation. In this simulation, it's supposed that each UAV carries portable radio antennas to provide service to clients, and this equipment has no interference with the ARCog-NET hardware (only powered by the UAVs) and also is not used by the framework). A *video with a simulation example of this scenario is presented* (https://youtu.be/TP5_Ld00ou0?si=V_Bht7oKu7LegAgi).

The UAV group deploys from a fixed base and positions itself, one by one, at a point in the sky. The group's position is coordinated by fog or cloud nodes, but the flight trajectories to change points can be processed by any layer (edge, fog, or cloud). When a client concentration demand shifts within the marked areas, an event will be created by a group serving that area, and the UAVs will change their positions. The client demand variation is perceived by fog or cloud layers through information passed by the edge nodes. Figures 16 and 17 represent variations in X and Y coordinate setpoints, respectively, and how each UAV responds to that change.

A UAV can change its fog coordinator over time, belonging to another cluster for a period and then shifting it. This decision is based on how close the UAV gets to a fog coordinator. An edge agent can

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

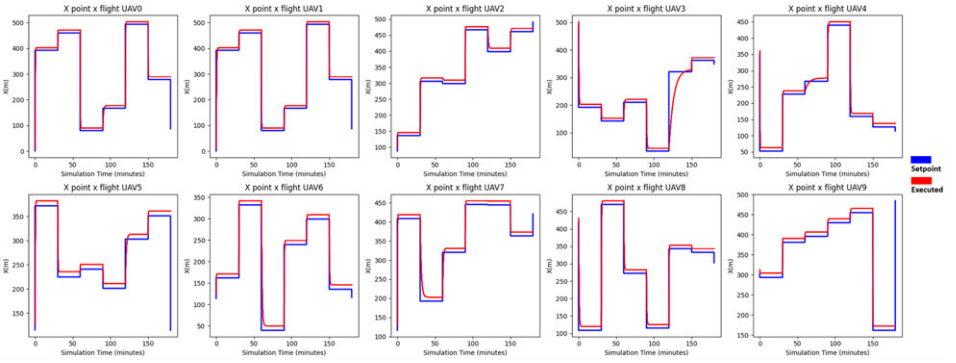

Figure 16. *Response of each UAV to X coordinate setpoint variation in city simulation.*

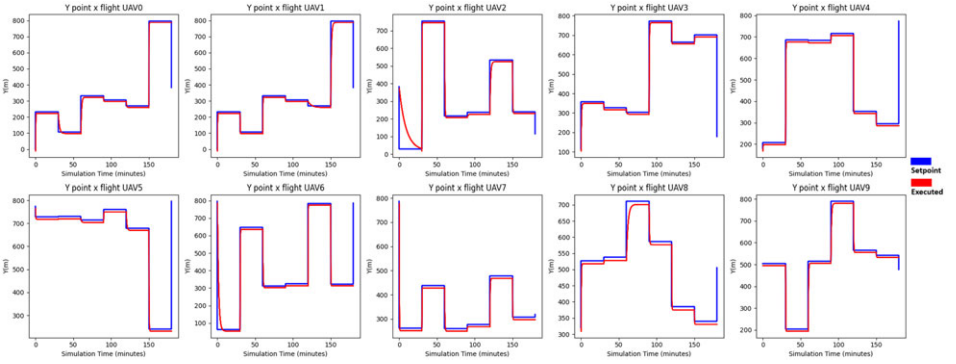

Figure 17. *Response of each UAV to Y coordinate setpoint variation in city simulation.*

Table VI. *Comparison of MAE and RMSE for X and Y coordinates for each UAV.*

UAV TAG	X MAE	X RMSE	Y MAE	Y RMSE
0	0.10	0.15	0.05	0.10
1	0.12	0.16	0.06	0.11
2	0.14	0.18	0.07	0.12
3	0.13	0.17	0.06	0.11
4	0.11	0.16	0.05	0.10
5	0.10	0.15	0.05	0.10
6	0.12	0.16	0.06	0.11
7	0.13	0.17	0.07	0.12
8	0.11	0.16	0.05	0.10
9	0.10	0.15	0.05	0.10

become a fog coordinator and vice-versa, which is a cloud-layer decision to change the network's structure on demand. However, the number of fog clusters is always 3 in all executed simulations with graphics enabled.

Table VI represents the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) for each executed setpoint, respectively. Given the coverage plot, the response to setpoint shifts, and the error table, it's possible to notice that the ARCog-NET could provide a quick response to environment changes and keep clients covered in this simulation scenario.

With small positioning errors, quick response time, and a client area coverage perception, the swarm behavior presented good performance for such kind of application. In terms of network architecture

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

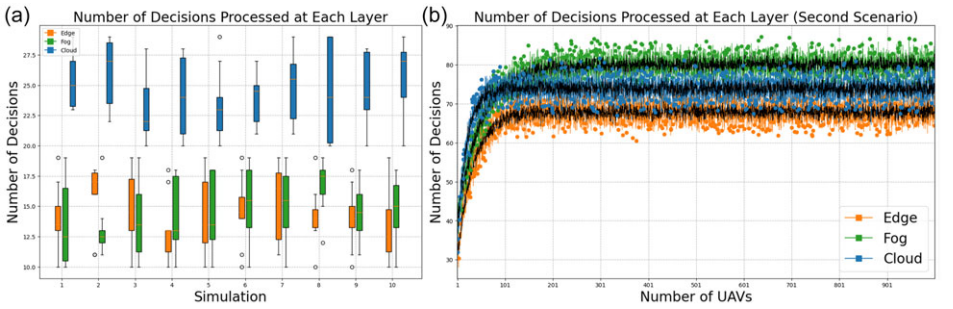

Figure 18. *Number of decisions processed on each ARCog-NET layer in the city simulation.*

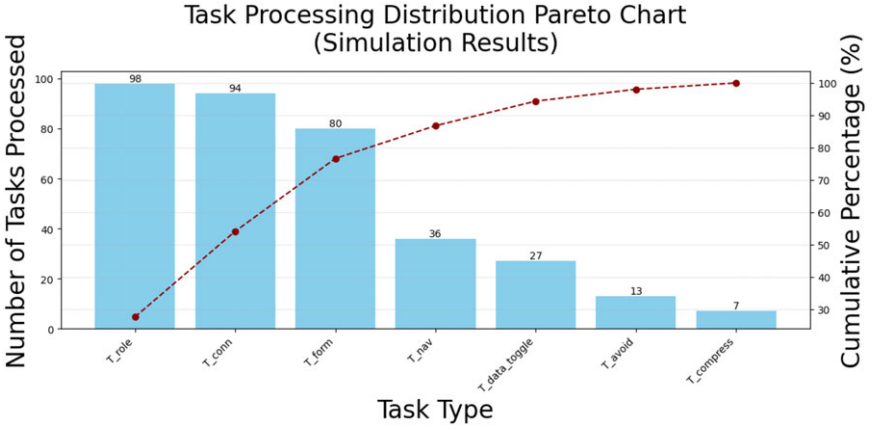

Figure 19. *Pareto graph of the average number of tasks by type in the city simulation.*

behavior, fewer decisions were processed on average, as the UAVs weren't flying continuously but statically positioned at a point and only flying to another point to attend to clients eventually. As the cloud layer has information on all data traffic of clients measured from edge agents through fog coordinators, this layer is responsible for most of the repositioning decisions at the beginning of the mission, and fog cluster control is the main decision-maker as the cognitive knowledge evolves. These results can be noted in Figure 18.

In this scenario, even fewer decisions have to be processed. ARCog-NET presented the saturation behavior one more time as more network nodes were connected to the architecture. As the coverage fault is identified by the cloud server, most of the decisions are initially made there. As the connected nodes increase, the cognitive task allocation makes it easier for fog nodes to perceive the communication fault and reallocate their edge group without messaging the cloud server or HITL. Another interesting asset of Figures 16 and 17 and this decision result is the X and Y coordinates, revealing that probably the formation control cognitive sub-processing was triggered more times, which is indeed confirmed by Figure 19.

In fact, note that formation control is the third most allocated task type. This makes sense for the scenario as, once again, ARCog-NET is dealing with a wide open environment with obstacles apart from each other. As client demand is geographically and randomly set by the algorithm and changes over time, the network will trigger a formation to better answer connectivity calls, and groups will tend to displace themselves as a whole instead of calculating separate routes, making formation control a more logical decision than path planning. This scenario almost doesn't use image compression, as the sent images are probably initial data transmission decisions taken by the network transient while it still learns how to behave to better serve its configured mission purpose of covering a deficient mobile communication network. The task allocation overall efficiency in this scenario was also evaluated and presented in Table VII.

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

Table VII. *Task allocation comparison in city scenario.*

Metric	Market-Based [51]	Heuristic Methods [53]	Bio-Inspired Methods [50]	Reinforcement Learning [54]	ARCog-NET
Efficiency (%)	70	80	85	88	87
Energy Consumption (J/task)	130	110	100	95	159
Communication Overhead (%)	18	12	10	9	8
Adaptability (ms)	65	75	80	82	81
Scalability (number of UAVs)	400	900	850	750	1000

Table VIII. *Path planning comparison in city scenario.*

Metric	Classical PSO [62]	A∗ [63]	Dijkstra [64]	RRT [65]	ARCog-NET
Convergence Time (s)	14	9	11	17	11
Trajectory Coverage (%)	83	88	90	85	89
Computational Cost (ms)	160	210	190	130	140
Adaptability (ms)	72	68	63	78	75
Scalability (number of UAVs)	480	290	390	570	1000

The task allocation comparison table highlights ARCog-NET's competitive performance in urban scenarios. While ARCog-NET achieves 87% efficiency, closely matching reinforcement learning (88%) and bio-inspired methods (85%), it outperforms market-based (70%) and heuristic approaches (80%). However, ARCog-NET's energy consumption (159 J/task) is significantly higher than all other methods, including reinforcement learning (95 J/task) and bio-inspired methods (100 J/task), indicating a trade-off between efficiency and energy use. ARCog-NET excels in communication overhead (8%), outperforming all other methods, which is crucial for scalability in dense urban environments. Its adaptability (81 ms) is slightly below reinforcement learning (82 ms) but better than market-based (65 ms) and heuristic methods (75 ms). Finally, ARCog-NET supports 1000 UAVs, outperforming all other methods, including bio-inspired (850 UAVs) and reinforcement learning (750 UAVs), demonstrating superior scalability for large-scale urban deployments. Overall, ARCog-NET provides a robust solution for task allocation in urban scenarios with mission requirements constantly changing (in this case, the traffic demand), balancing efficiency, communication efficiency, and scalability, though its higher energy consumption warrants further optimization.

The path planning and formation control results are presented in Tables VIII and IX, respectively. Table VIII shows that ARCog-NET maintains a competitive balance between efficiency and adaptability, outperforming traditional methods in scalability (1000 UAVs) while remaining close to Dijkstra in trajectory coverage (89% vs. 90%). However, its convergence time (12 s) is slightly slower than A∗ (10 s) and Dijkstra (12 s), though still faster than RRT (18 s). The computational cost (140 ms) remains moderate, offering better performance than A∗ (210 ms) and Dijkstra (190 ms), but slightly higher than RRT (130 ms). Adaptability (75 ms) is improved over Dijkstra (63 ms) and A∗ (68 ms), though still behind RRT (78 ms), indicating that while ARCog-NET offers robust decision-making, additional refinements may enhance real-time responsiveness.

Table IX indicates notable improvements in ARCog-NET's ability to maintain formations with a low formation error (0.10 m), outperforming DMPC (0.13 m) and DRL-based control (0.18 m), while closely approaching consensus-based methods (0.08 m). However, energy consumption (74 J/formation) is significantly higher than other methods, suggesting that while ARCog-NET ensures high precision in

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

Table IX. *Formation control comparison in city scenario.*

Metric	DMPC [66]	DRL-Based [67]	Consensus-Based [68]	ARCog-NET
Formation Error (*m*)	0.13	0.18	0.08	0.10
Energy Consumption (J/formation)	48	58	38	74
Adaptability (ms)	82	87	77	80
Communication Overhead (%)	10	13	7	8
Robustness to Delays (%)	87	82	91	90

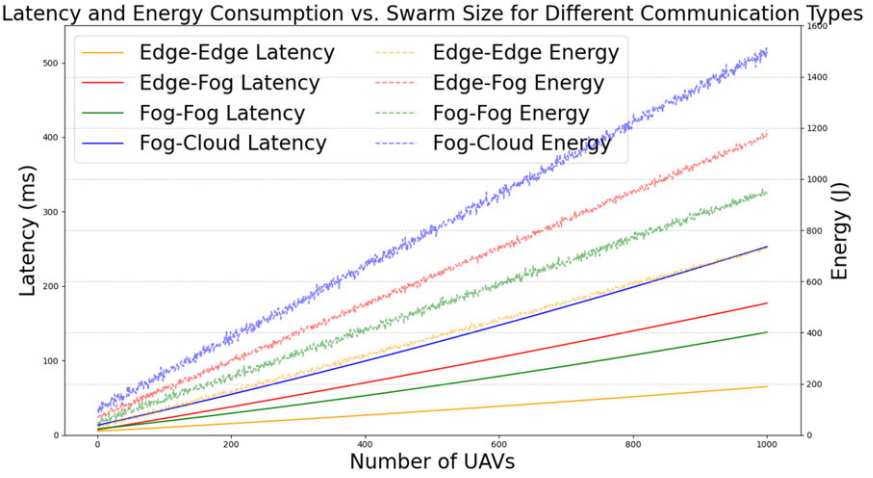

Figure 20. *Average ARCog-NET latency by number of UAVs at each time in the city simulation.*

formation keeping, it does so at a higher energy cost. Despite this, adaptability (80 ms) remains close to DRL-based methods (87 ms) and better than consensus-based control (77 ms), indicating improved flexibility in varying demand environments. The communication overhead (8%) is efficiently minimized, second only to consensus-based methods (7%), while robustness to delays (90%) remains high, comparable to consensus-based methods (91%) and significantly stronger than DRL-based approaches (82%). This suggests that ARCog-NET provides strong formation control capabilities with minor trade-offs in energy efficiency, making it a promising solution for real-world multi-UAV operation with mission objectives constantly changing.

Less decisions (task allocation) and the lack of image data transmission, therefore compression processing time, also resulted in less latency and better connection scalability results. There is an ARCog-NET improvement associated with energy trade-off identified in simulations for this scenario, which is in line with the results presented in Figure 20. It is possible to verify that latency has low values but increasing value rates as more robots are added to simulations are bellow what was seen in offshore scenario, with a more linear latency increase always keeping smaller rates. Energy has higher values, but in this simulation, the energy consumption was also increased by the substitute radios of the transmission towers that the UAVs are carrying.

Data volume was smaller than observed in offshore simulation, while throughput followed very close values compared to the same previous scenario. This is a logical result as less data is transmitted over simulation, and the architecture kept is topology. An interesting result was the Fog-to-cloud throughput decrease, which is in line with mission evolutions as a cognitive process in this simulation, made edge and fog less dependent on the cloud server and HITL. These results can be observed in Figure 21.

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

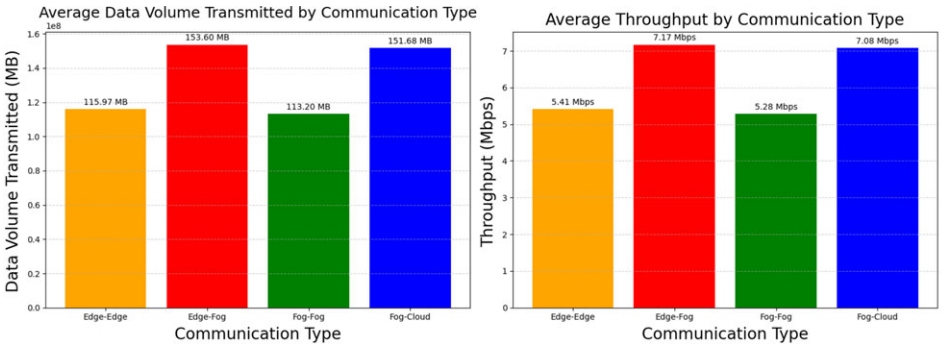

Figure 21. *Average ARCog-NET transmitted data and throughput in city simulation.*

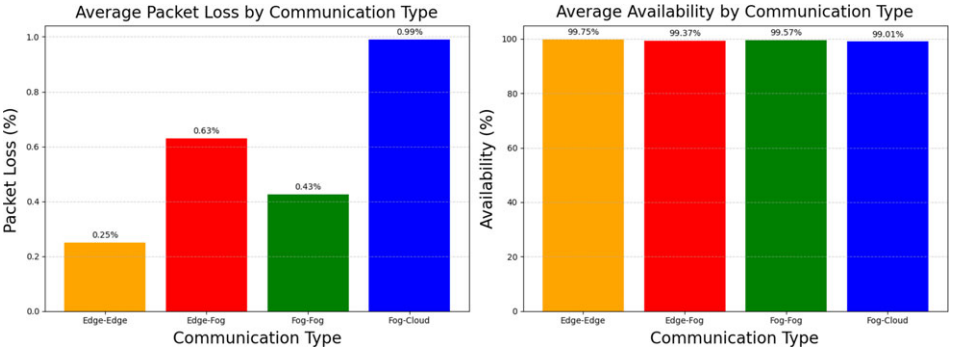

Figure 22. *Average ARCog-NET packet loss and throughput in city simulation.*

The packet loss and availability closely follow what was observed in the offshore simulation with slight improvements, which is a logical result since the architecture topology was not changed. Such results can be observed in Figure 22

In the city simulation, ARCog-NET successfully managed UAV swarms for urban communication restoration. The system effectively adapted to dynamic client demand hotspots, ensuring continuous coverage with minimal positioning errors and quick response times. The fog layer primarily handled the decision-making process, with fewer decisions required overall due to the static positioning of UAVs, which only repositioned as needed. This led to efficient task allocation and formation control, though at a higher energy cost than other methods.

The simulation highlighted ARCog-NET's scalability, supporting up to 1000 UAVs, and its ability to maintain low communication overhead and latency, even as the number of UAVs increased. The system's adaptability and robustness were evident, with formation control and path planning tasks being efficiently managed, though energy consumption remains an area for improvement. Overall, ARCog-NET proved to be a reliable solution for urban scenarios with dynamic mission requirements, balancing efficiency, scalability, and adaptability effectively.

4.3. Environment 3: industrial plant indoor search and rescue

Once again, graphic simulations were carried out with one to ten vehicles, ten times for each count. In this scenario, the proposal is to emulate an indoor search and rescue mission where the UAVs connected to ARCog-NET should fly inside an industrial environment undergoing some emergency situation (a radiation breakout, for example), looking at all rooms and spaces for victims. The group deployed from a fixed safe point and then started flying around using ultrasonic collision sensors to avoid hitting structures or each other while searching for humans with cameras. The *video shows an example of this mission with*

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

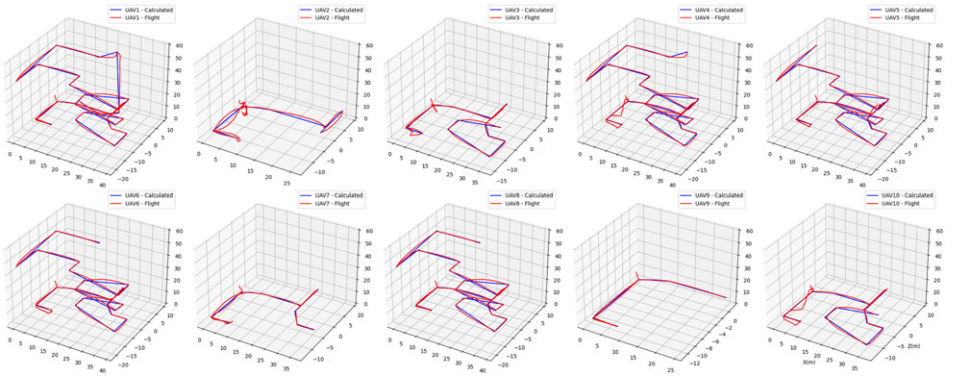

Figure 23. *Comparison between trajectories calculated by ARCog-NET and executed by each UAV for a simulation with* 10 *robots in industrial plant environment.*

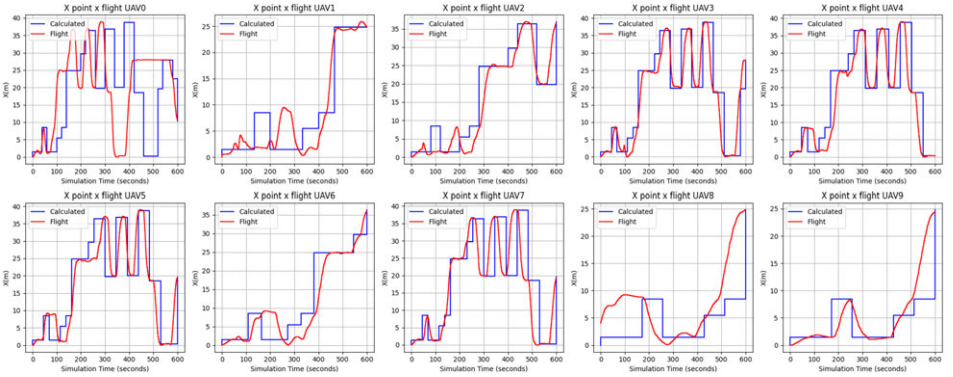

Figure 24. *Response of each UAV to X coordinate setpoint variation in industrial plant simulation.*

10 UAVs (https://youtu.be/o6rse5FPzo8?si=VnK2m15FwfiABtEA), and Figure 23 shows the executed trajectories in this example.

An interesting result of this application is that robots can "learn" about positioning even if they didn't pass by an unknown region, because if another member of the network scanned a determined area and already gained a solution for continuing to fly, these flight decisions will be saved at the knowledge base of ARCog-NET. The initial decision is to avoid obstacles by flying a straight line if an object is detected too close to the respective UAV. The cloud HITL can also send manual commands to any robot, an edge agent, or a fog coordinator. Figures 24 and 25 represent how the robots reacted to variations in X and Y points of the trajectory inside the factory for a simulation with 10 robots.

Note in Figures 24 and 25 that, in some cases, the flight begins before the trajectory calculation changes. This is an example of how a knowledge base is shared between robots, as a robot can make a flight decision by communicating to the network and reacting faster to environmental events. Usually, the UAV behaves as an edge agent in this case and sends a request to the fog coordinator, which delivers a trajectory orientation quicker than the local trajectory PSO loop that the agent runs. Sometimes, the curve is delayed, where the UAV changes its trajectory after the calculations. In this case, the agent is probably passing by a region without mapping in the knowledge base, or maybe there is a communication delay with superior layers so it decreases speed while the results are not delivered. Usually, the robot is a group leader (or a fog coordinator) in this case, and sometimes, requests to the cloud are sent for trajectory orientation. The same flight curve (red) can show a positive or negative time shift when compared to the calculated trajectory (blue) on the same graph, which also represents that the respective UAV probably changed roles during the simulation (it was an edge agent and became a fog coordinator, for example), the cognitive knowledge base increased (making more possible to make even quicker

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

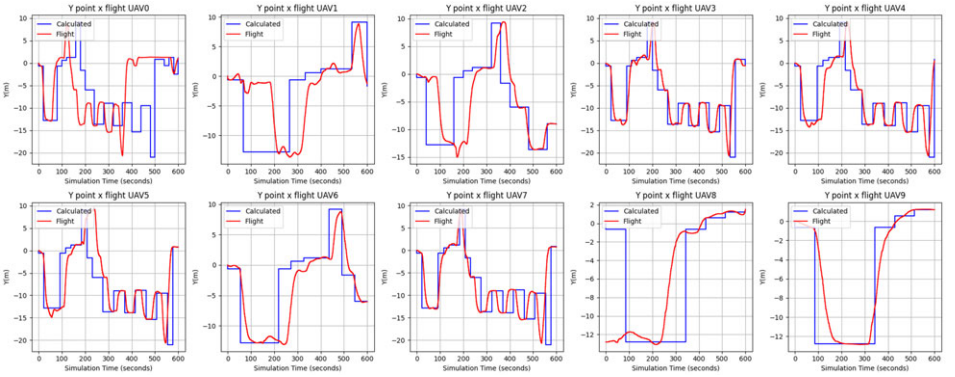

Figure 25. *Response of each UAV to Y coordinate setpoint variation in industrial plant simulation.*

Table X. *MAE and RMSE for X and Y trajectories, and time shift for UAVs.*

UAV	X MAE	X RMSE	Y MAE	Y RMSE	Average Time Shift (s)
1	0.05	0.03	0.08	0.12	−75.72
2	0.04	0.09	0.07	0.11	63.38
3	0.03	0.08	0.06	0.1	44.92
4	0.04	0.09	0.05	0.09	13.21
5	0.03	0.08	0.07	0.1	23.23
6	0.04	0.1	0.08	0.12	11.38
7	0.03	0.09	0.06	0.11	41.11
8	0.05	0.1	0.07	0.11	−17.70
9	0.04	0.09	0.06	0.1	−77.03
10	0.03	0.08	0.08	0.12	41.15

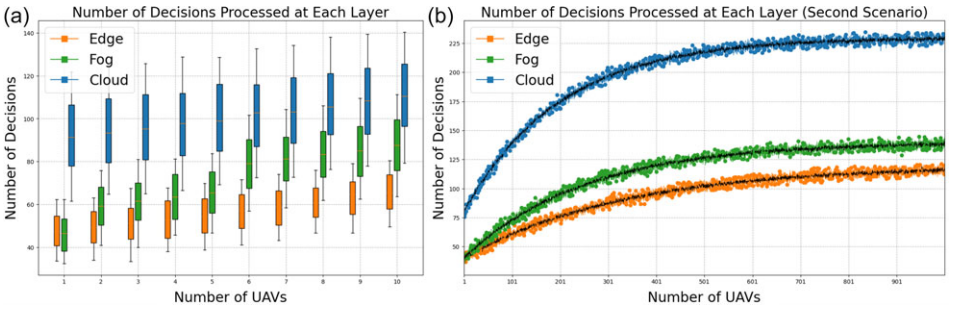

Figure 26. *Number of decisions processed on each ARCog-NET layer in all simulations of the industrial plant environment.*

decisions) or the communication latency in that region, packet loss and throughput are severe. Table X sums up the average *X* and *Y* positioning MAE and RMSE and average time shifts as well.

This results show satisfactorily how the proposed architecture can be used as a effective UAV swarm control mechanism while is also a good solution to manage communication and data transmission in missions. As in the three previous scenarios, the decision-making study was also conducted as presented in Figure 26.

In this indoor search and rescue scenario, the decision-making graphs take more time to saturate because the complexity of the environment and mission demands a higher volume of decisions as more UAVs are added. Unlike the previous urban and offshore scenarios, where UAVs were mostly statically positioned, this mission requires continuous dynamic navigation, obstacle avoidance, and

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

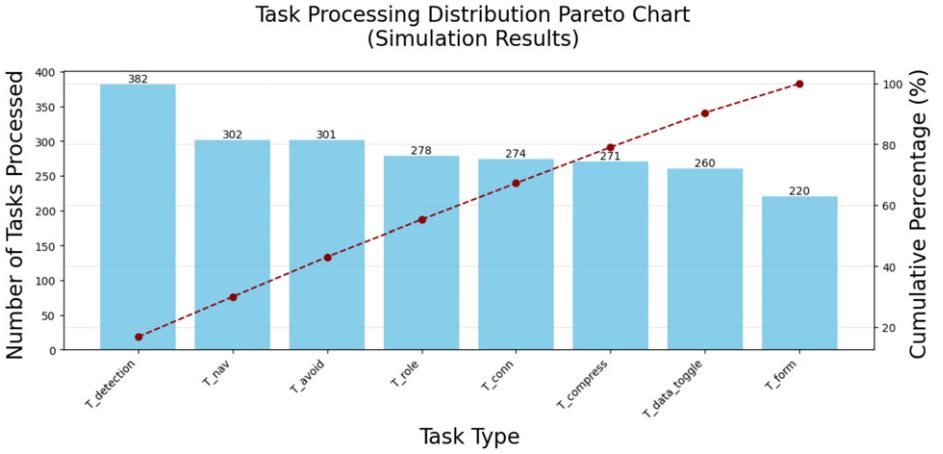

Figure 27. *Pareto graph of the average number of tasks by type in the industrial plant simulation.*

victim detection in a confined, unstructured space. Each additional UAV introduces new sensory data, path planning requirements, and coordination tasks, leading to a logarithmic increase in decisions. The indoor environment's unpredictability, combined with the need for real-time adjustments to avoid collisions and optimize search patterns, ensures that decision-making scales with the number of UAVs. Furthermore, the cognitive allocation of tasks across edge, fog, and cloud layers allows the system to handle this complexity efficiently, delaying saturation and maintaining performance even as more UAVs are deployed. This behavior reflects ARCog-NET's adaptability to mission-specific demands, where increased decision-making is necessary to ensure effective operation in challenging environments.

The graph indicates that the fog layer is critical in task handling within the ARCog-NET architecture for this indoor application. The high median and wide interquartile range for the fog layer suggest it handles a significant portion of the computational load and experiences a wide range of decision-processing activities. This could be due to its position in the network hierarchy, balancing between the high-level, computationally intensive tasks offloaded to the cloud and the low-latency, real-time tasks managed at the edge layer (note that this application is the most active in terms of communication due to the high amount of obstacles and unknown indoor environment). The edge layer's lower median number of decisions and lower variability indicate it primarily handles less complex, real-time tasks that do not require intensive computation, which is in line with agent roles in this mission that are mainly charged for navigation and image transmission to the cloud through fog. This is consistent with the role of edge computing, which aims to reduce latency by processing data closer to the source.

The cloud server, while handling more decisions than the edge agents and fog coordinators, doesn't saturate as quickly as other scenarios. The moderate variability and presence of outliers suggest that the cloud layer processes tasks with varying computational demands, possibly involving more complex analytics and long-term storage tasks, including HITL supervision as well, a result confirmed by the Pareto Graph (see Figure 27). Being a small indoor scenario, this makes sense, as navigation tasks are primarily handled by the robots themselves, while the cloud layer is responsible for most of the image processing and target identification. In summary, fog coordinators are the most active in terms of flight decision processing and allocation, but the cloud layer is the one that most processes decisions. This distribution aligns with the typical roles and responsibilities assigned to each layer in an EFC architecture, reflecting an efficient distribution of computational tasks to optimize performance and resource utilization.

In the industrial search and rescue scenario, navigation is a critical component of the mission, encompassing both path planning and formation control. Path planning ensures that each UAV can dynamically avoid obstacles and efficiently explore the environment, while formation control enables coordinated movement of the swarm to cover larger areas and maintain communication links. These capabilities are essential for navigating the complex, unstructured indoor environment, where obstacles are abundant

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

Table XI. *Path planning comparison in industrial plant scenario.*

Metric	Classical PSO [62]	A∗ [63]	Dijkstra [64]	RRT [65]	ARCog-NET
Convergence Time (s)	18	12	15	20	14
Trajectory Coverage (%)	80	85	88	82	87
Computational Cost (ms)	180	220	200	150	160
Adaptability (ms)	70	65	60	75	72
Scalability (number of UAVs)	450	300	400	500	1000

Table XII. *Formation control comparison in industrial plant scenario.*

Metric	DMPC [66]	DRL-Based [67]	Consensus-Based [68]	ARCog-NET
Formation Error (*m*)	0.15	0.20	0.10	0.12
Energy Consumption (J/formation)	50	60	40	70
Adaptability (ms)	80	85	75	78
Communication Overhead (%)	12	15	8	10
Robustness to Delays (%)	85	80	90	88

and the mission requires precise, real-time adjustments. Integrating these navigation strategies allows the UAVs to operate autonomously, adapt to changing conditions, and ensure effective coverage of the search area.

The path planning comparison in Table XI shows that ARCog-NET maintains a strong balance between convergence time (14 s) and trajectory coverage (87%). While it is slightly slower than A∗ (12 s), it outperforms RRT (20 s) and Classical PSO (18 s). The computational cost of 160 ms is competitive, being lower than A∗ (220 ms) and Dijkstra (200 ms), though slightly higher than RRT (150 ms). ARCog-NET's adaptability (72 ms) is better than Dijkstra (60 ms) and A∗ (65 ms), though slightly below RRT (75 ms). Notably, ARCog-NET supports up to 1000 UAVs, demonstrating superior scalability compared to other methods, making it highly suitable for large-scale indoor search and rescue missions.

Besides not being triggered too much, the formation control comparison in Table XII highlights ARCog-NET's ability to maintain precise formations with a formation error of 0.12 m, outperforming DMPC (0.15 m) and DRL-based methods (0.20 m), and closely approaching consensus-based methods (0.10 m). However, energy consumption (70 J/formation) remains higher than other methods, indicating a trade-off between precision and energy efficiency. ARCog-NET's adaptability (78 ms) is competitive, close to DRL-based methods (85 ms), and better than consensus-based control (75 ms). Communication overhead (10%) is efficiently managed, though it is slightly higher than that of consensus-based methods (8%). Robustness to delays (88%) is high, comparable to consensus-based methods (90%), demonstrating ARCog-NET's reliability in dynamic, obstacle-rich environments despite higher energy costs.

The task allocation comparison in Table XIII demonstrates ARCog-NET's competitive performance in the industrial plant scenario. With an efficiency of 84%, ARCog-NET closely matches reinforcement learning (85%) and outperforms market-based (68%) and heuristic methods (78%). However, its energy consumption (150 J/task) is higher than all other methods, including reinforcement learning (100 J/task) and bio-inspired methods (110 J/task), indicating a trade-off between efficiency and energy use.

ARCog-NET excels in communication overhead (9%), outperforming all other methods, which is critical for scalability in complex indoor environments. Its adaptability (78%) is slightly below reinforcement learning (80%) but better than market-based (60%) and heuristic methods (70%). Notably, ARCog-NET supports up to 1000 UAVs, demonstrating superior scalability compared to other methods, including bio-inspired (780 UAVs) and reinforcement learning (700 UAVs). This makes ARCog-NET

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

Table XIII. *Task allocation comparison in industrial plant scenario.*

Metric	Market-Based [51]	Heuristic Methods [53]	Bio-Inspired Methods [50]	Reinforcement Learning [54]	ARCog-NET
Efficiency (%)	68	78	82	85	84
Energy Consumption (J/task)	140	120	110	100	150
Communication Overhead (%)	20	15	12	10	9
Adaptability (ms)	60	70	75	80	78
Scalability (number of UAVs)	350	800	780	700	1000

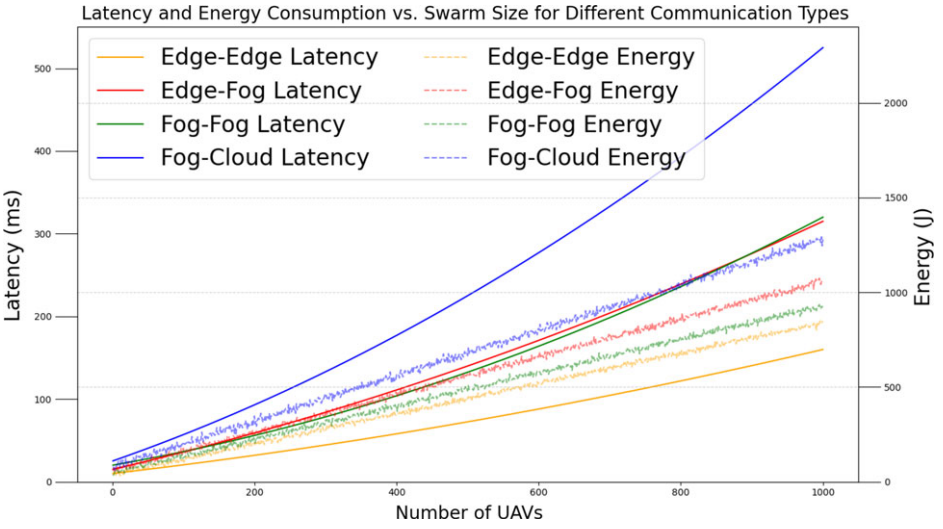

Figure 28. *Average ARCog-NET latency by number of UAVs at each time in the industrial plant simulation.*

a robust solution for task allocation in dynamic, obstacle-rich environments, balancing efficiency, communication efficiency, and scalability, though its higher energy consumption remains an area for further optimization. Figure 28 presents the latency and energy consumption graph.

The indoor environment introduces unique challenges, such as confined spaces, dynamic obstacles, and the need for precise navigation, which significantly impact both latency and energy usage. As the number of UAVs increases, latency grows gradually, reflecting the efficient task allocation and communication management facilitated by ARCog-NET. The fog layer plays a critical role in reducing latency by handling most real-time flight decisions locally, while the cloud layer is primarily used for complex analytics and long-term storage tasks. This distribution ensures that latency remains manageable even for large swarms, though Fog-to-cloud communication exhibits the highest latency due to the additional processing and transmission delays associated with cloud-based tasks. Edge-to-edge communication, on the other hand, offers the lowest latency but is less scalable for large swarms, making it suitable for smaller, localized tasks.

Energy consumption in this indoor scenario is notably higher due to the continuous dynamic navigation, obstacle avoidance, and victim detection tasks required in a confined and obstacle-rich environment. The fog layer's active role in decision-making contributes to increased energy usage, as it processes a significant portion of the computational load. Fog-to-cloud communication consumes the

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

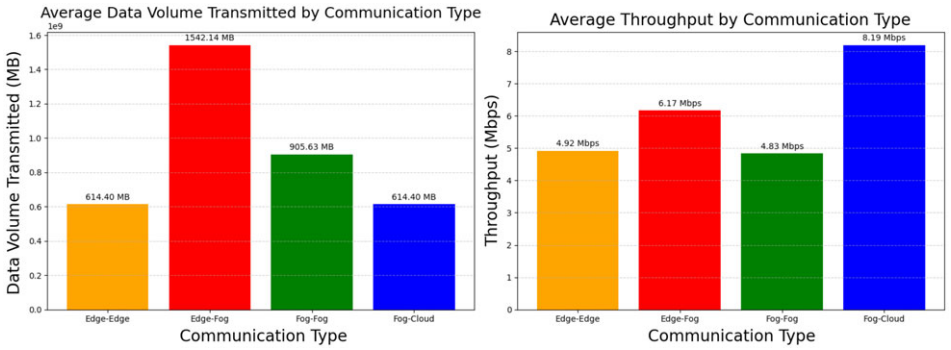

Figure 29. *Average ARCog-NET transmitted data and throughput in industrial plant simulation.*

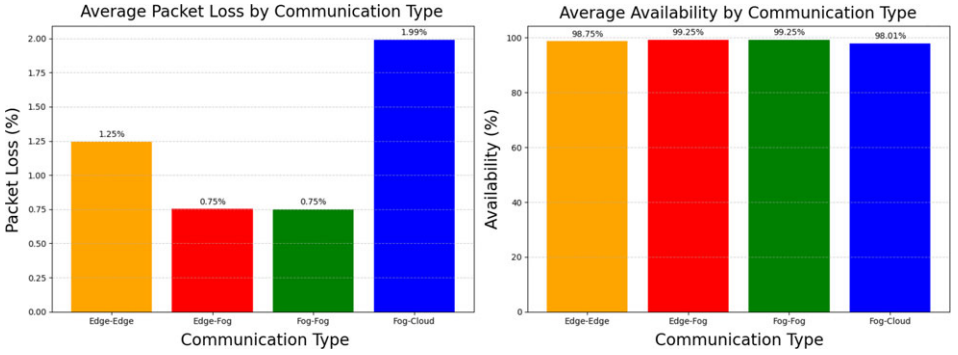

Figure 30. *Average ARCog-NET packet loss and throughput in industrial plant simulation.*

most energy, driven by the computational intensity of cloud-based tasks and long-distance data transmission. Despite this, ARCog-NET's ability to distribute tasks across edge, fog, and cloud layers helps optimize energy usage, ensuring that energy consumption increases steadily and is more manageable as the swarm size grows. The variability in energy consumption reflects the dynamic nature of the indoor environment, where energy usage can fluctuate based on mission requirements and environmental factors.

The trade-off between latency and energy consumption is evident in this scenario. Edge-to-edge communication provides low latency and energy consumption. Still, it is limited in scalability, while Fog-to-cloud communication offers advanced computational capabilities and scalability at the cost of higher latency and energy usage. The fog layer strikes a balance between these extremes, providing a middle ground that ensures efficient task allocation and communication. ARCog-NET's scalability is a key strength, as both latency and energy consumption increase at manageable rates even for large swarms, making it well-suited for indoor missions that require hundreds or thousands of UAVs to cover complex environments effectively. Figure 29 presents the data volume and throughput graph.

The high throughput in Fog-to-cloud communication indicates that, while the cloud layer is used sparingly, it handles critical tasks that require significant data transmission. This ensures that complex analytics and long-term storage tasks are performed efficiently when needed. Edge-to-fog pipelines have high data volume but moderate throughput. This reflects the fog layer's role in handling a large amount of real-time data for decision-making but with a focus on efficient, distributed processing rather than high-speed transmission. Edge-to-edge and fog-to-fog communication have low data volume and throughput, involving localized coordination and limited data transmission. This is consistent with the confined indoor environment, where direct communication between UAVs (Edge-to-edge) and between fog nodes (Fog-to-fog) is limited, but the nodes are usually relatively close to each other.

The packet loss and availability results, presented in Figure 30, highlight the communication challenges in the industrial plant indoor environment. Edge-to-edge communication exhibits the highest

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

packet loss (1.25%) and the lowest availability (98.75%), as direct UAV-to-UAV communication is prone to interference in confined spaces. In contrast, Edge-Fog and Fog-to-fog communication show lower packet loss (0.75% each) and higher availability (99.25% each), reflecting the fog layer's effectiveness in managing real-time coordination. Fog-to-cloud communication has moderate packet loss (1.99%) and availability (98.01%), as long-distance transmission to the cloud layer is less reliable indoors. These results underscore the trade-offs between communication types in a dynamic and obstacle-rich environment.

In this scenario, the high packet loss in edge-to-edge communication limits its suitability for critical tasks. In contrast, the reliability of Edge-Fog and Fog-to-Fog communication demonstrates the fog layer's central role in real-time decision-making. The moderate packet loss in Fog-to-Cloud communication indicates that the cloud layer remains viable for complex analytics, albeit with reduced reliability. ARCog-NET's ability to distribute tasks across edge, fog, and cloud layers ensures robust performance, but further optimizations could focus on reducing packet loss in edge-to-edge communication to enhance swarm coordination in indoor environments.

5. Conclusions and future works

This paper presented a cognitive architecture designed to provide a flexible and adaptable solution for UAV swarm implementation across diverse mission scenarios. The primary goal of ARCog-NET is to address the challenges of scalability, communication reliability, and energy efficiency while offering a modular and reconfigurable framework that can be tailored to various applications. By integrating cognitive-based decision-making processes and the EFC computing model, ARCog-NET enables real-time data processing, intelligent task allocation, and dynamic adaptability, ensuring robust performance in complex and dynamic environments.

The offshore oil rig monitoring scenario demonstrated ARCog-NET's ability to handle large-scale, long-range communication with minimal latency, showcasing its scalability and adaptability to harsh environmental conditions. In the urban communication restoration scenario, the architecture efficiently managed dynamic client demands and maintained low communication overhead, proving its suitability for densely populated and rapidly changing environments. Finally, the industrial plant indoor search and rescue scenario highlighted ARCog-NET's ability to operate in confined, obstacle-rich spaces, with the fog layer playing a critical role in real-time decision-making and coordination. These simulations collectively validated ARCog-NET's versatility and effectiveness across diverse mission contexts, demonstrating its flexibility as a solution for UAV swarm implementation.

ARCog-NET's integration with ROS and its use of actual flight controller firmware in simulations further underscore its adaptability, enabling comprehensive testing and validation in virtual environments before physical deployment. By addressing scalability, communication reliability, and energy efficiency challenges, ARCog-NET provides a generic framework that can be easily adapted to various UAV swarm applications, from environmental monitoring to search and rescue missions. Future work will focus on optimizing energy efficiency, reducing packet loss, and enhancing adaptability to improve performance in complex and dynamic environments further. Another future work focuses on updating the ROS implementation, which is limited and doesn't have continuous support anymore, using a ROS2 build instead of ROS and Gazebo Classic. The ARCog-NET also must be tested in real vehicles and environments as a Field Acceptance Trial. Overall, ARCog-NET represents a significant advancement in UAV swarm technology, offering a flexible and reliable solution for real-world deployment.

Acknowledgements. We thank the following Brazilian Federal Agencies, CAPES, CNPq, FAPERJ, and CEFET/RJ for supporting this research work. This work was carried out with the support of the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior—Brasil (CAPES)—Finance Code 001.

Declarations. To ensure objectivity and transparency in research and that accepted principles of ethical and professional conduct have been followed, all authors provide these declarations statements below.

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

Ethical approval. Not applicable.

Consent to participate. All authors are consent to participate.

Consent to publish. All authors are consent to publish.

Authors contributions. Conceptualization and Methodology, Gabryel Silva Ramos and Milena Faria Pinto; Validation and Investigation, Gabryel Silva Ramos; writing—original draft preparation, Gabryel Silva Ramos and Milena Faria Pinto; writing—review and editing, Gabryel Silva Ramos, Milena Faria Pinto, and Diego Barreto Haddad. All authors have read and agreed to submit the current manuscript version.

Funding. Not applicable.

Competing interests. The authors have no conflicts of interest to declare that they are relevant to the content of this article.

Availability of data and materials. Not applicable.

References

[1] G. S. Ramos, M. F. Pinto, F. O. Coelho, L. M. Honório and D. B. Haddad, "Hybrid methodology based on computational vision and sensor fusion for assisting autonomous uav on offshore messenger cable transfer operation," *Robotica* **40**(8), 2786–2814 (2022).

[2] M. F. Pinto, A. Melo, A. Marcato and C. Urdiales, "Case-based reasoning approach applied to surveillance system using an autonomous unmanned aerial vehicle," **In:** *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)* (2017), Edinburgh, UK, 2017, IEEE Xplore, pp. 1324–1329, doi: 10.1109/ISIE.2017.8001437.

[3] G. S. Ramos, D. B. Haddad, A. L. Barros, L. de Melo Honorio and M. F. Pinto, "EKF-based vision-assisted target tracking and approaching for autonomous UAV in offshore mooring tasks," *IEEE Journal on Miniaturization for Air and Space Systems* **3**(2), 53–66 (2022).

[4] C. D. Rodin, L. N. de Lima, F. A. de Alcantara Andrade, D. B. Haddad, T. A. Johansen and R. Storvold, "Object classification in thermal images using convolutional neural networks for search and rescue missions with unmanned aerial systems," **In:** *2018 International Joint Conference on Neural Networks (IJCNN)* (2018), Rio de Janeiro, Brazil, 2018, IEEE Xplore, pp. 1–8, doi: 10.1109/IJCNN.2018.8489465.

[5] G. S. Berger, M. Teixeira, A. Cantieri, J. Lima, A. I. Pereira, A. Valente, G. G. de Castro and M. F. Pinto, "Cooperative heterogeneous robots for autonomous insects trap monitoring system in a precision agriculture scenario," *Agriculture* **13**(2), 239 (2023).

[6] L. D. P. Pugliese, F. Guerriero and G. Macrina, "Using drones for parcels delivery process," *international Conference on Industry 4.0 and Smart Manufacturing (ISM 2019), Procedia Manufacturing* **42**, 488–497 (2020). [Online]. Available at: https://www.sciencedirect.com/science/article/pii/S2351978920305928.

[7] D. R. Green, J. J. Hagon, C. Gomez and B. J. Gregory, "Chapter 21 - Using Low-Cost Uavs for Environmental Monitoring, Mapping, and Modelling: Examples From the Coastal Zone," **In:** *Coastal Management* (R. Krishnamurthy, M. Jonathan, S. Srinivasalu and B. Glaeser, eds.) (Academic Press, 2019) pp. 465–501. [Online]. Available at: https://www.sciencedirect.com/science/article/pii/B9780128104736000224.

[8] S. Manfreda, P. Dvorak, J. Mullerova, S. Herban, P. Vuono, J. J. A. Justel and M. Perks, "Assessing the accuracy of digital surface models derived from optical imagery acquired with unmanned aerial systems," *Drones* **3**(1), 15 (2019). [Online]. Available at: https://www.mdpi.com/2504-446X/3/1/15.

[9] A. Tahir, J. Boling, M.-H. Haghbayan, H. T. Toivonen and J. Plosila, "Swarms of unmanned aerial vehicles — a survey," *J. Ind. Inf. Integr.* **16**, 100106 (2019). [Online]. Available at: https://www.sciencedirect.com/science/article/pii/S2452414X18300086.

[10] Y. Alqudsi and M. Makaraci, "Uav swarms: Research, challenges, and future directions," *J. Eng. Appl. Sci.* **72**(1), 01 (2025).

[11] M. Luthra, "Aerial robot swarms: A review," *IAES Int. J. Robot. Autom. (IJRA)* **12**(2), 137–145 (2023). [Online]. Available at: https://ijra.iaescore.com/index.php/IJRA/article/view/20532.

[12] M. Abdelkader, S. Güler, H. Jaleel and J. S. Shamma, "Aerial swarms: Recent applications and challenges," *Current Robotics Reports* **2**(3), 309–320 (2021), doi: 10.1007/s43154-021-00063-4.

[13] S. Javed, A. Hassan, R. Ahmad, W. Ahmed, R. Ahmed, A. Saadat and M. Guizani, "State-of-the-art and future research challenges in uav swarms," *IEEE Internet Things J.* **11**(11), 19023–19045 (2024).

[14] G. S. Ramos, F. D. R. Henriques, D. B. Haddad, F. A. A. Andrade and M. F. Pinto, "ARCog-NET: An aerial robot cognitive network architecture for swarm applications development," *IEEE Access* **12**, 129040–129063 (2024).

[15] Y. Zhou, B. Rao and W. Wang, "Uav swarm intelligence: Recent advances and future trends," *IEEE Access* **8**, 183856–183878 (2020).

[16] A. Khamis, A. Hussein and A. Elmogy, "Swarm robotics: A survey from a multi-tasking perspective," *IEEE Access* **9**, 133158–133180 (2021).
[17] M. F. Pinto, L. M. Honório, A. L. Marcato, M. A. Dantas, A. G. Melo, M. Capretz and C. Urdiales, "ARCog: An Aerial Robotics Cognitive Architecture," *Robotica* **39**(3), 483–502 (2021), doi: 10.1017/S0263574720000521.
[18] P. F. Dominey and F. Warneken, "The basis of shared intentions in human and robot cognition," *New Ideas Psychol.* **29**(3), 260–274 (2011).
[19] P. Langley, J. E. Laird and S. Rogers, "Cognitive architectures and general intelligent systems," *AI Mag.* **30**(2), 33 (2009).
[20] P. Haazebroek, S. Van Dantzig and B. Hommel, "A computational model of perception and action for cognitive robotics," *Cogn. Process.* **12**(4), 355–365 (2011).
[21] D. E. Rumelhart, "Schemata: The Building Blocks of Cognition," **In:** *Theoretical Issues in Reading Comprehension*, (Routledge, 2017) pp. 33–58.
[22] T. Ogata, K. Takahashi, T. Yamada, S. Murata and K. Sasaki, "Machine learning for cognitive robotics," *Cognitive Robotics* **1**(1), 167–189, MIT Press, 2022, doi: https://doi.org/10.7551/mitpress/13780.003.0014.
[23] Y. Yang, C. Fermuller and Y. Aloimonos, "A cognitive system for human manipulation action understanding," **In:** Second Annual Conference on Advances in Cognitive Systems (ACS)(2013), Maryland, United States, 2013, Advances in Cognitive Systems, pp. 67–86, Vol. **3**.
[24] J. E. Laird, K. R. Kinkade, S. Mohan and J. Z. Xu, "Cognitive robotics using the soar cognitive architecture," **In:** *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence, Cognitive Robotics* (2012), Ontario, Canada, 2012, Association for the Advancement of Artificial Intelligence, pp. 46-54, Vol. 26. No. 1.
[25] C. C. Insaurralde, "Service-oriented agent architecture for unmanned air vehicles," **In:** *Digital Avionics Systems Conference (DASC), 2014 IEEE/AIAA 33rd*, IEEE (2014) pp. 8B1–1.
[26] S. Emel'yanov, D. Makarov, A. I. Panov and K. Yakovlev, "Multilayer cognitive architecture for uav control," *Cogn. Syst. Res.* **39**, 58–72 (2016).
[27] M. Selecky, M. Rollo, P. Losiewicz, J. Reade and N. Maida, "Framework for incremental development of complex unmanned aircraft systems," **In:** *Integrated Communication, Navigation, and Surveillance Conference (ICNS), 2015*, IEEE (2015) pp. J3–1.
[28] J. L. Sanchez-Lopez, M. Molina, H. Bavle, C. Sampedro, R. A. S. Fernández and P. Campoy, "A multi-layered component-based approach for the development of aerial robotic systems: The aerostack framework," *J. Intell. Robot. Syst.* **88**(2-4), 683–709 (2017).
[29] T. K. Das, "Intelligent techniques in decision making: A survey," *Indian Journal of Science and Technology* **9**(12), 1–6, 2016.
[30] S.-H. Liao, "Expert system methodologies and applications—a decade review from 1995 to 2004," *Expert Syst. Appl.* **28**(1), 93–103 (2005).
[31] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction," **In:** *IEEE Transactions on Neural Networks* **9**(5), 1054–1054 (1998), doi: 10.1109/TNN.1998.712192.
[32] M. Leonetti, L. Iocchi and P. Stone, "A synthesis of automated planning and reinforcement learning for efficient, robust decision-making," *Artificial Intelligence* **241**, 103–130 (2016).
[33] M. A. Luna, M. S. A. Isaac, A. R. Ragab, P. Campoy, P. F. Pena and M. Molina, "Fast multi-uav path planning for optimal area coverage in aerial sensing applications," *Sensors* **22**(6), 2297 (2022). [Online]. Available at: https://www.mdpi.com/1424-8220/22/6/2297.
[34] A. L. Alfeo, M. G. C. A. Cimino, N. D. Francesco, A. Lazzeri, M. Lega and G. Vaglini, "Swarm coordination of mini-uavs for target search using imperfect sensors," *Intelligent Deci-sion Technologies* **12**(2), 149–162 (2018).
[35] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler and A. Y. Ng, "ROS: An open-source robot operating system," **in Proc.** *ICRA workshop on open source software*, Kobe, Japan, 2009, International Conference on Robotics and Automation (ICRA), Vol. **3**, no. 3.2.
[36] Open Robotics, Gazebo Robot Simulator, (2021). Available at: http://gazebosim.org/.
[37] Open Robotics, MAVROS – MAVLink extendable communication node for ROS with proxy for Ground Control Station, (2022). Available at: http://wiki.ros.org/mavros.
[38] MAVLINK, MAVLink Developer Guide, (2022). Available at: https://mavlink.io/en/.
[39] PX4, Open Source Autopilot for Drone Developers, (2022). Available at: https://px4.io/.
[40] T. Erez, Y. Tassa and E. Todorov, "Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx," **In:** *2015 IEEE International Conference on Robotics and Automation (ICRA)*, (2015) pp. 4397–4404.
[41] C. Anagnostopoulos, C. Koulamas, A. Lalos and C. Stylios, "Open-source integrated simulation framework for cooperative autonomous vehicles," **In:** *2022 11th Mediterranean Conference on Embedded Computing (MECO)* (2022) pp. 1–4.
[42] OpenSim Ltd., OMNet++: Discrete Event Simulator, (2022). Available at: https://omnetpp.org/.
[43] A. Mohini, CDSSim - Multi UAV Communication and Control Simulation Framework, Master's thesis, University of Cincinnati. OhioLINK Electronic Theses and Dissertations Center (2019). Available at: http://rave.ohiolink.edu/etdc/view?acc_num=ucin1554373574457271.
[44] NSNAM, NS-3 Network Simulator, (2022). Available at: https://www.nsnam.org/.
[45] Ardupilot, Controller Diagrams, (2021). Available at: https://docs.px4.io/master/en/flight_stack/controller_diagrams.html.
[46] Ardupilot, Multicopter pid tuning guide (manual/advanced), (2021). Available at: https://docs.px4.io/master/en/config_mc/pid_tuning_guide_multicopter.html#rate-controller.

https://doi.org/10.1017/S0263574725101914 Published online by Cambridge University Press

[47] S. Acharya, A. Bharadwaj, Y. Simmhan, A. Gopalan, P. Parag and H. Tyagi, "CORNET: A Co-Simulation Middleware for Robot Networks," **In:** *2020 International Conference on COMmunication Systems & NETworkS (COMSNETS)* (2020), Bengaluru, India, 2020, IEEE Xplore, pp. 245–251, doi: 10.1109/COMSNETS48256.2020.9027459.
[48] M. Calvo-Fullana, D. Mox, A. Pyattaev, J. Fink, V. Kumar and A. Ribeiro, "Ros-netsim: A framework for the integration of robotic and network simulators," *CoRR* **2101**(10113), 1120–1127 (2021). [Online]. Available at: https://arxiv.org/abs/2101.10113
[49] M. F. Pinto, A. L. Marcato, A. G. Melo, L. M. Honório and C. Urdiales, "A framework for analyzing fog-cloud computing cooperation applied to information processing of uavs," *Wireless Communications and Mobile Computing* **2019**, 1–14 (2019).
[50] Z. Yang, H. Zhang and L. Xu, "Multi-uav cooperative task allocation based on improved genetic algorithm," *IEEE Access* **8**, 65432–65442 (2020).
[51] H.-S. Shin and H.-C. Bang, "Market-based task assignment for cooperative timing missions in dynamic environments," *J. Intell. Robot. Syst.* **87**(1), 97–123 (2017).
[52] L. Gupta, R. Jain and G. Vaszkun, "Survey of important issues in uav communication networks," *IEEE Commun. Surv. Tutor.* **18**(2), 1123–1152 (2016).
[53] X. Zhao, Y. Liu and H. Chen, "Autonomous task allocation for uav swarms using heuristic clustering and auction mechanisms," *IEEE Trans. Vehicular Tech.* **70**(8), 7404–7417 (2021).
[54] W. Liu, T. Zhang and J. Huang, "Multi-agent reinforcement learning-based task allocation for uav swarms," *IEEE Trans. Intell. Transp. syst.* **23**(10), 17245–17257 (2022).
[55] Raspberry Pi Foundation, Raspberry Pi, (2022). Available at: https://www.raspberrypi.org.
[56] Intel Corporation, Intel Galileo Board, (2022). Available at: https://www.intel.com/content/www/us/en/products/details/boards-kits/galileo.html.
[57] G. S. Ramos, Multi UAV Environmental Surveyllance Simulation with ROS/Gazebo, (2024). Available at: https://www.youtube.com/watch?v=u_XFkuzfRcs.
[58] G. S. Ramos and M. F. Pinto, Shuttle Tanker, FPSO and FSO Image Dataset for Ship Recognition, IEEE Dataport. (2021), Available at: https://dx.doi.org/10.21227/9xzy-fd09.
[59] G. S. Ramos, M. F. Pinto, E. S. S. de Souza, G. B. Machado and G. G. R. de Castro, "Technical and economic feasibility study for implementing a novel mooring-assisting methodology in offloading operations using autonomous unmanned aerial vehicles," *SPE Production and Operations* **37**(01), 72–87 (2021), doi: 10.2118/205524-PA.
[60] G. S. Ramos, A. A. De Lima, L. F. Almeida, J. Lima and M. F. Pinto, "Simulation and evaluation of deep learning autoencoders for image compression in multi-UAV network systems," **In:** *2023 Latin American Robotics Symposium (LARS), 2023 Brazilian Symposium on Robotics (SBR), and 2023 Workshop on Robotics in Education (WRE)* (2023), Bahia, Brazil, IEEE Xplore, pp. 41–46, Vol. 1, doi: 10.1109/LARS/SBR/WRE59448.2023.10332986.
[61] S. M. H. Kalami, path-planning: A collection of path planning algorithms (2021). https://github.com/smkalami/path-planning. accessed: 2023-03-24.
[62] J. Kennedy and R. Eberhart, "Particle swarm optimization," **In:** *Proceedings of ICNN'95 - International Conference on Neural Networks*, IEEE, vol. **4**, (1995) pp. 1942–1948.
[63] P. E. Hart, N. J. Nilsson and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cyber.* **4**(2), 100–107 (1968).
[64] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik* **1**(1), 269–271 (1959).
[65] S. M. LaValle, "Rapidly-Exploring Random Trees: A New Tool for Path Planning," Technical Report TR 98-11, Department of Computer Science, Iowa State University, 1998, pp. 98–11, Vol. 1.
[66] C. Liu, W.-H. Chen and G. Yang, "Distributed model predictive control for multi-robot formation," *IEEE Trans. Cont. Syst. Tech.* **28**(6), 2456–2468 (2020).
[67] Z. Pu, T. Zhang, X. Ai, T. Qiu and J. Yi, "A deep reinforcement learning approach combined with model-based paradigms for multiagent formation control with collision avoidance," *IEEE Trans. Syst. Man, Cyber: Syst.* **53**(7), 4189–4204 (2023).
[68] X. Dai, K. Liu and Y. Xia, "Event-triggered consensus-based formation control for multi-agent systems," *Automatica* **135**, 109987 (2022).

Cite this article: G. S. Ramos, M. F. Pinto and D. B. Haddad (2025). "Advancing UAV swarm autonomy with ARCog-NET for task allocation, path planning, and formation control", Robotica **43**, 2565–2609. https://doi.org/10.1017/S0263574725101914