accepted to appear in Proceedings of 27th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), June 16-19, 2026, Bologna, Italy.

Distilling LLM Reasoning into Lightweight Neural Policies for Multi-Agent UAV Coordination

Suramya Pokharel and Eyuphan Bulut
Department of Computer Science, Virginia Commonwealth University
401 West Main St. Richmond, VA 23284, USA
{pokharels7, ebulut}@vcu.edu

***Abstract*—Large Language Models (LLMs) are emerging as powerful high-level planners capable of reasoning over complex multi-agent tasks, including collaborative UAV operations. However, their inference cost, latency, and dependence on large external compute make them unsuitable for real-time or onboard deployment. This work presents a framework that leverages LLMs as strategic supervisors to generate high-quality coordination behaviors for multi-UAV coverage tasks and distills these behaviors into lightweight neural policies that execute efficiently. We employ a hybrid spatial-relational architecture to learn from LLM-generated trajectories and action assessments, enabling the distilled model to capture both environment structure and inter-agent dependencies. Unlike LLMs, the resulting policies offer fast inference and can be deployed at scale, while retaining the generality of LLM-derived reasoning across varying team sizes, grid dimensions, and mission configurations. Our approach demonstrates that LLMs can serve as versatile teachers for multi-agent coordination and that compact neural models can approximate their planning capabilities in a form practical for real-time autonomous UAV swarms.**

***Index Terms*—Multi-agent coordination, UAV networks, large language models, autonomous systems; policy distillation.**

I. Introduction

Unmanned aerial vehicles (UAVs) have emerged as a key enabling technology for a wide range of autonomous sensing and monitoring tasks due to their mobility, flexibility, and rapid deployment capabilities [1]. Beyond point-to-point data collection, coordinated UAV swarms are increasingly employed in coverage-centric missions where teams of aerial agents collaboratively explore and monitor large areas [2]. Such missions arise in applications including environmental surveillance, precision agriculture, post-disaster assessment, infrastructure inspection, and public safety operations. Effective coverage requires UAVs to continuously adapt their trajectories based on environmental conditions, mission objectives, and the actions of neighboring agents, making multi-UAV coordination a challenging spatiotemporal decision-making problem. As the scale and complexity of these missions grow, there is a growing need for intelligent, scalable, and real-time coordination strategies that can generalize across varying environments, team sizes, and mission constraints.

In this paper, we explore the problem of intelligent coordination for multi-UAV coverage missions, where a swarm of UAVs must collaboratively plan their movements to efficiently cover a target area while accounting for inter-agent interactions and mission constraints. Current state-of-the-art solutions for such problems are primarily based on rule-based heuristics, optimization methods or, multi-agent reinforcement and deep learning techniques [3]–[6]. While effective in constrained settings, these approaches often require extensive task-specific design, large amounts of training data, or retraining when mission parameters such as team size, environment scale, or operational constraints change. Moreover, many learning-based methods struggle to generalize beyond the scenarios observed during training, limiting their applicability in dynamic or previously unseen environments.

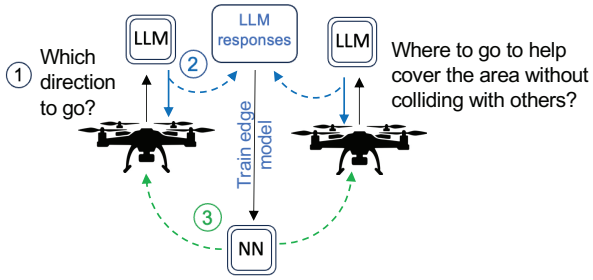


Fig. 1: UAVs use LLM agents to decide their strategy initially, then switch to the model trained from the collected responses.

Recent advances in large language models (LLMs) and their integration into decision-making systems have opened new opportunities for addressing complex planning and coordination problems. LLMs have demonstrated strong reasoning and planning capabilities across a variety of domains, including robotics [7], embodied navigation [8], traffic control [9], and multi-agent collaboration including UAVs [10]. Motivated by these developments, our goal in this paper is twofold. First, we explore the ability of LLMs to act as high-level decision-makers for routing and coordinating a swarm of UAVs in collaborative coverage missions. Second, recognizing the practical limitations of LLM inference for real-time and onboard deployment, we propose to distill the decision-making behavior of LLMs into a lightweight neural network model that enables fast inference while preserving the generality and adaptability of LLM-derived strategies across varying mission configurations.

While there are a few recent studies that explore the usage of LLMs in various UAV related research problems such as UAV path planning [11], optimization of UAV locations [12] and to enhance mission security [13], to the best of our knowledge, there is no study that looks at the multi-UAV coverage problem

through LLM-based planners and the distillation of LLM reasoning into a lightweight neural policy. In this study, we explore this problem in detail, and through extensive results, we demonstrate that the LLMs can be used to optimize the UAV paths through a policy-based design. We also show that their reasoning can be distilled into a deployable edge-friendly neural policy while keeping generalization characteristics.

The rest of the paper is organized as follows. We present an overview of the related work in Section II. In Section III, we describe our system model and problem statement. We also elaborate on how LLM prompts are designed as well as how the neural network is trained from the LLM responses. We then present our simulation results for different scenarios in Section IV. Finally, we conclude and discuss our future work in Section V.

II. Related Work

Multi-UAV coordination and coverage tasks have recently been widely explored through reinforcement learning and deep neural networks [6], [14]–[16]. These methods aim to learn coordinated policies that optimize coverage efficiency or mission objectives through interaction with the environment. However, in dynamic and multi-agent settings, learning-based methods face well-known challenges, including limited generalization to unseen environments and non-stationarity arising from simultaneous policy updates by multiple agents [17]. Thus, many existing approaches require task-specific training and retraining when mission parameters such as team size, environment scale, or operational constraints change. Other model-driven and graph-based extensions to multi-agent reinforcement learning aim to improve scalability and coordination [18]–[20], but they remain sensitive to task-specific training and mission configuration changes.

LLMs have been heavily investigated recently as high-level decision-makers for autonomous systems. Initial works discuss the potential of LLMs for UAV operations [21], while more recent efforts apply LLMs to specific UAV-related problems such as path planning [11], UAV placement optimization [12], and mission security enhancement [13]. These studies show the promise of LLMs for reasoning and planning; however, they typically rely on direct LLM inference, which incurs significant latency and computational overhead and limits scalability in real-time multi-UAV deployments.

In contrast to prior work, this paper aims to distill the decision-making behavior of LLMs for multi-UAV coverage missions into a lightweight neural policy, enabling fast inference while preserving adaptability across diverse mission configurations. This design is conceptually related to studies on model compression and teacher–student learning, which aim to reduce inference cost by transferring knowledge from large models to smaller ones [22]. However, existing teacher–student approaches typically assume that both the teacher and the student operate within the same learning paradigm, whereas in our case, the teacher is an LLM-based planner and the student is a deployable neural policy designed for real-time multi-UAV coordination.

III. System Model

A. Environment Modeling

We assume that there is a set of N UAVs deployed in an environment. The time is slotted and each UAV can move to a new position at each timestep in accordance with its movement constraints (e.g., maximum velocity). Each UAV i is positioned at $\mathbf{p}_i(t) = \big(x_i(t), y_i(t), h\big)$ at time t and has a coverage with radius r_i centered at this position. As many cooperative aerial missions (e.g., coverage missions) are performed at a regulated and nearly constant flight altitude, we assume all UAVs are located at the same altitude h. Consequently, the effective maneuvering freedom during their operation is predominantly horizontal, and the problem is analyzed by projecting movement onto the two-dimensional (x, y) plane.

Without loss of generality, we assume the operational area to be modeled as a two-dimensional grid of size of $W \times H$, where $W, H \in \mathbb{Z}_{>0}$. Each cell in the grid may be free, occupied by an obstacle, or covered by a UAV. A cell is defined as a *free cell* if it is not occupied by a static or dynamic obstacle. If a free cell has lied at any point within the coverage region of at least one UAV i, then it is considered *covered* at timestep t.

The global coverage metric is then defined as

$$\mathrm{C}(t) = \frac{\#\text{covered free cells}}{\#\text{total free cells}}. \tag{1}$$

Dynamic Obstacles. In order to create a dynamic environment, we assume there are also obstacles that appear/disappear or move stochastically across time steps. These obstacles can be thought of as realistic, time-varying disturbances present in the environment as well as physical obstacles that move but are not part of the collaborative mission.

B. Problem Statement

Given an initial configuration $\{\mathbf{p}_1(0), \ldots, \mathbf{p}_N(0)\}$ and a target coverage threshold $\tau \in (0, 1]$, the objective is to find a joint trajectory policy

$$\pi : \mathcal{S} \to \mathcal{A}_1 \times \cdots \times \mathcal{A}_N \tag{2}$$

that minimizes the time T required to achieve coverage $\mathrm{C}(T) \geq \tau$, where $\mathrm{C}(t)$ is defined as in Eq. (1), while satisfying all constraints at each timestep.
Formally, the problem is:

$$\begin{aligned} \min_{\pi} \quad & T \\ \text{s.t.} \quad & \mathrm{C}(T) \geq \tau, \\ & \mathbf{p}_i(t+1) \in \mathcal{A}_i(t), \quad \forall i, t, \\ & \mathbf{p}_i(t) \neq \mathbf{p}_j(t), \quad \forall i \neq j, t, \\ & \mathbf{p}_i(t) \notin \mathcal{O}(t), \quad \forall i, t. \end{aligned} \tag{3}$$

where, $\mathcal{O}(t)$ is the set of obstacle cells at time t and $\mathcal{A}_i(t)$ is the set of actions available for UAV i at time t.

The challenging nature of this problem arises from its combinatorial and limited action space, decentralized execution requirements, and stochastic obstacle dynamics.

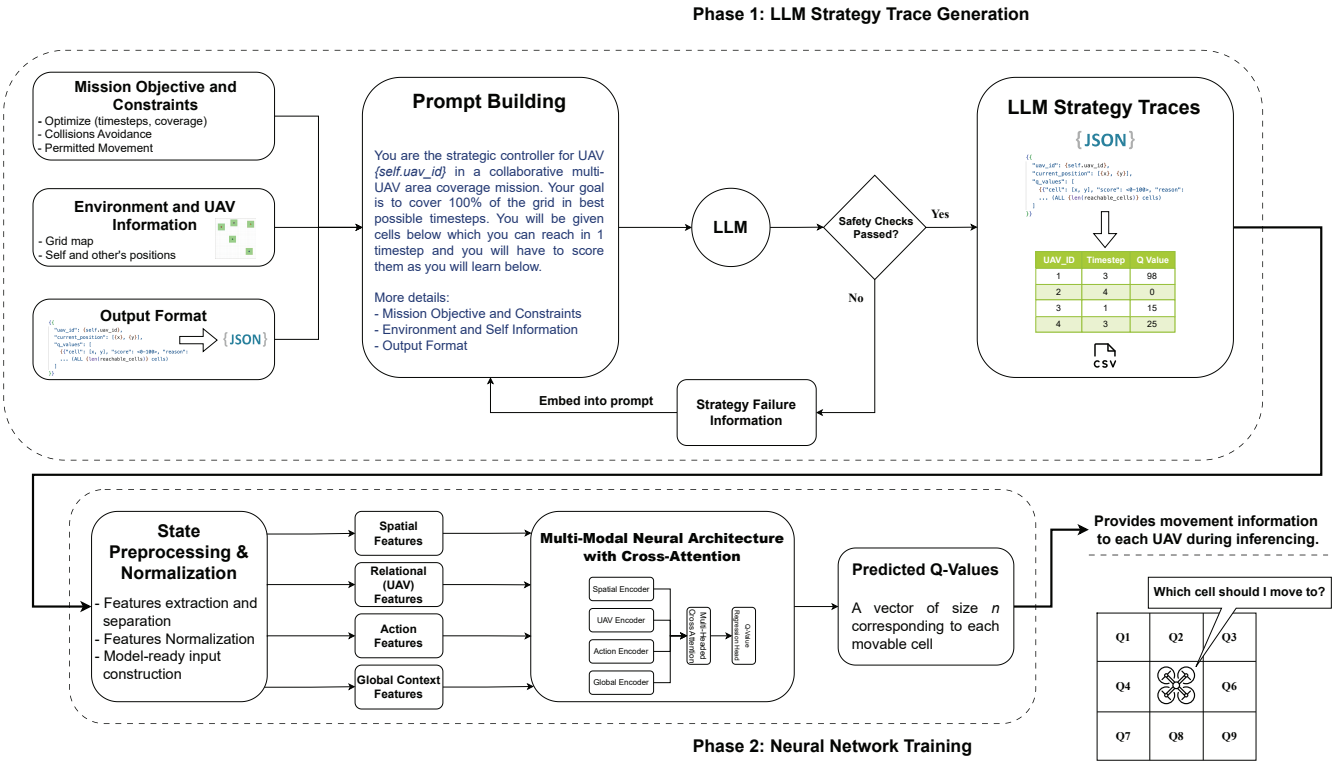


Fig. 2: End-to-end overview of the proposed system, illustrating LLM-based strategy trace generation and model training.

C. LLM for Strategy Trace Generation

To optimize the UAV trajectories under the aforementioned objective, we employ an LLM as a strategic evaluator. We initially attempted to generate the entire path in a single inference step by providing the problem description and the configuration details as input. While the LLM could capture the abstract idea of where each UAV should plan to go, when asked for the exact movement path, it failed to produce a reasonable solution, often producing completely random movement trajectories or trajectories that were impossible to move in. Consequently, we decided to reframe our problem as a sequential decision-making process analogous to a Deep Q-Network (DQN) framework. Instead of holistic generation, the LLM now functions as a reasoning agent for each UAV, assigning Q-values to each candidate action available to it. By modeling the problem through discrete strategy decisions, the LLM could now accurately identify the desirability of all feasible next-step actions, producing dense, structured traces of decision-making behavior.

Thus,

$$Q_{\text{LLM}}(a \mid s_t) \in [0,\ 100], \tag{4}$$

interpreted as an estimated value of action a conditioned on state s_t.

Prompt Construction. At each timestep and for each UAV, a structured prompt is generated containing the following components:

- *Goal and Context:* A high-level description of the problem statement.
- *Environment State:* Grid dimensions, current global coverage percentage, number of UAVs, obstacle positions.
- *Self State:* Ego (or self) UAV position, ego coverage radius, movement history, and the set of reachable cells for the next timestep.
- *Other Agents' States:* Positions and coverage radii of other UAVs, and their projected movements for the next time step, if available[1].
- *Scoring Guidelines:* General principles are provided rather than rigid rules. These principles include: maximizing new coverage, reducing overlap, avoiding obstacles, avoiding collisions, minimizing repeated visitation, and promoting spatial distribution.
- *Safety Information:* When coverage exceeds a particular value or coverage percentage stagnates, the locations of the nearest uncovered cells are included in the input prompt to ensure LLM safely achieves the mission objectives.
- *Output Format:* For consistent output, the LLM is instructed to reply strictly only with a JSON object structured as:

[1]The first UAV queries the LLM without any projected movements of other agents. The second UAV includes the projected movement of the first UAV, and each subsequent UAV includes the projected movements of all previously processed UAVs.

Notation	Description
W, H	Width and height of the grid environment
G	Canonical normalized grid size $G = \max(W, H)$
$\mathbf{p}_i(t)$	Position of UAV i at time t
r_i	Coverage radius of UAV i
N	Number of UAVs in the environment
$\mathcal{A}_i$	Action set of UAV i (candidate reachable cells)
$\lvert\mathcal{A}_i\rvert$	Number of actions available to UAV i
$\lvert\mathcal{A}\rvert_{\max}$	Maximum action count for any UAV
$\mathcal{E}_i(x, y, t)$	Ego-UAV coverage map for UAV i
$\mathcal{F}_i(x, y, t)$	Ego instantaneous footprint mask
$\mathcal{B}(x, y, t)$	Frontier indicator at (x, y)
$\mathcal{G}(x, y, t)$	Global coverage map value at cell (x, y)
$\mathcal{O}(x, y, t)$	Obstacle indicator at (x, y)
$\Delta\mathbf{p}_j(t)$	Projected movement of UAV j
$\Delta\mathbf{p}_{j\to i}(t)$	Relative displacement to ego UAV
$h_{spatial}$	Spatial Embeddings
h_{ego}	Ego UAV Embeddings
h_{global}	Global Context Embeddings
h_{act}	Action Embeddings

TABLE I: Commonly used notations and their descriptions

```
{
  "uav_id": i,
  "current_position": [x_i(t), y_i(t)],
  "q_values": [
    {
      "cell": [x, y],
      "score": S,
      "reason": "..."
    },
    // ... more cell-score pairs
  ]
}
```

Action Selection and Trace Recording. After all UAVs obtain their LLM-evaluated scores, each UAV decides to move to the cell with the highest Q-value.[2] If a collision occurs, only the colliding UAVs are requested to reevaluate their Q values with a corrective prompt stating explicitly, the conflicting movement. An episode terminates when the coverage percentage exceeds the threshold τ.

At each timestep t, for each UAV i, and for every candidate action a, one row of a strategy trace is appended to the dataset. Each row includes details provided to the LLM as an input along with the LLM-generated Q value for that action and the reasoning behind the Q value (for our validation). Thus, a trace encodes the complete mapping:

$$\text{State } s_t,\ \text{Action } a \longrightarrow Q_{\text{LLM}}(a \mid s_t), \tag{5}$$

providing dense supervision for learning value functions and underlying policies.

D. State Preprocessing and Normalization

The collected strategy trace is structured such that each entry corresponds to a specific UAV (i) at a given time step (t) paired with one candidate action. The full state for a UAV at time t is thus represented by multiple rows sharing identical state features but listing all available actions. To convert this heterogeneous, variable-length information into fixed-size tensors suitable for deep learning, we apply the structured preprocessing pipeline described below.

1) Spatial Tensor Construction: Every LLM-simulated episode uses a grid of dimensions $W \times H$, which can vary across episodes. Because neural networks require fixed-size inputs, all spatial maps are resized as

$$G = \max(W, H), \qquad X_{\text{grid}} \in \mathbb{R}^{6\times G\times G}. \tag{6}$$

For each timestep, six spatial channels are constructed:

- *Ego Footprint Mask:* A binary mask representing the instantaneous coverage of ego UAV i (i.e., the UAV querying the LLM) at the current timestep t. It is defined by the coverage radius r_i around the current ego position $\mathbf{p}_i(t)$:

$$\mathcal{F}_i(x, y, t) = \begin{cases} 1, & \text{if } (x, y) \text{ in } r_i \text{ of } \mathbf{p}_i(t), \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

- *Ego UAV Coverage Map:* A binary map indicating all cells covered by the ego UAV from the start of the mission up to the current timestep t, defined as:

$$\mathcal{E}_i(x, y, t) = \max_{\tau\in\{0,\dots,t\}} \mathcal{F}_i(x, y, \tau). \tag{8}$$

- *Combined Footprints of all other UAVs:* A binary map marking all cells currently within the coverage footprint of any UAV $j \in \mathcal{N}$ other than the ego UAV i at the current timestep t.

$$\mathcal{F}_o(x, y, t) = \max_{j\in\mathcal{N}, j\neq i} \mathcal{F}_j(x, y, t) \tag{9}$$

- *Global Coverage Map:* A binary grid indicating which cells of the environment have been covered by any UAV at any point from the start of the mission up to the current timestep t.

$$\mathcal{G}(x, y, t) = \max_{\tau\in 0,\dots,t} \left(\max_{j\in\mathcal{N}} \mathcal{F}_j(x, y, \tau) \right). \tag{10}$$

- *Frontier Mask:* This is a derived binary mask marking the boundaries between covered and uncovered regions at time step t.

$$\mathcal{B}(x, y, t) = \begin{cases} 1, & \mathcal{G}(x, y, t) = 1 \ \wedge\ \exists (x', y') \\ & \in \mathcal{N}(x, y) : \mathcal{G}(x', y', t) = 0, \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

  where $\mathcal{N}(x, y)$ denotes the neighbor cells surrounding the coordinate (x, y).
- *Obstacle Map:* A binary grid indicating locations of obstacles that are present at the current time step t.

$$\mathcal{O}(x, y, t) = 1 \quad \text{if cell } (x, y) \text{ is blocked.} \tag{12}$$

These channels are concatenated into a tensor $X_{\text{grid}} \in \mathbb{R}^{6\times G\times G}$.

[2]By obtaining all Q-values and selecting the best move rather than obtaining only the best move directly from the LLM, we aim to collect sufficient training data to be able to develop a generalizable model.

2) Relational Representation of the UAVs: The Q values obtained from the traces generated by LLM depend not only on the ego UAV's state but also on the positions, projected intents, and footprints of all other UAVs. Thus, predicting the Q-values requires explicit modeling of inter-agent relationships, not just ego features. Hence, we create an 8-dimensional per-UAV descriptor where the descriptors encode:

- *Normalized Position:* It is the UAV's (x, y) location, normalized to the $[0, 1]$ range relative to grid dimensions.
- *Ego vs Non-Ego Binary Flag:* It is a binary flag with value 1 if UAV is the ego UAV and 0 otherwise.
- *UAV Coverage Radius:* It is a scalar value r_j describing the UAV j's coverage radius.
- *Projected movement from previous time step:* It is a vector describing the UAV's intended movement from the last time step. For a UAV j, this is:

$$\Delta \mathbf{p}_j(t) = \mathbf{p}_j(t) - \mathbf{p}_j(t-1). \tag{13}$$

- *Movement Vector Relative to Ego UAV:* It is a vector encoding how each UAV is moving relative to the ego UAV.

$$\Delta \mathbf{p}_{j \to i}(t) = \mathbf{p}_j(t) - \mathbf{p}_i(t). \tag{14}$$

Each descriptor is a feature vector in $\mathbb{R}^8$. Since the number of agents varies between 1 and N, we pad the agent set to a fixed cardinality of N by inserting zero vectors and a corresponding validity mask that distinguishes real agents from padding.

Thus, the relational input becomes $H_{\text{agents}} \in \mathbb{R}^{N \times 8}$, with an associated mask $\mathbf{M}_{\text{agents}} \in \{0, 1\}^N$, which ensures that padded agents do not contribute to downstream attention computations.

3) Action Representation: For the ego UAV, the action space corresponds to all positions reachable within movement distance d_m. For a given d_m, the number of candidate actions is $|\mathcal{A}|$.

Each action is encoded in a 15-dimensional vector comprising of features like position information of the candidate action, collision risk, possible coverage gain from the action, obstacle flag and so on.

As with agents, the per-row action sets are padded to the maximum action count along with an action mask

$$\mathbf{M}_{\text{act}} \in \{0, 1\}^{|\mathcal{A}|_{\max}}. \tag{15}$$

4) Global Context Vector: It is a compact global context vector containing summary of mission-wide information such as current global coverage percentage, grid dimensions, UAV counts, ego UAV radius and obstacle density hence producing:

This produces:

$$\mathbf{z}_{\text{global}} \in \mathbb{R}^6. \tag{16}$$

E. Multi-Modal Neural Architecture

To generate reliable action-value estimates from the data obtained from LLM trace, we have developed a unified multi-modal neural model as illustrated in Fig. 3. The architecture is specifically designed to fuse heterogeneous information sources—spatial field structure, multi-agent interactions, global mission context, and action-level features—into a single coherent Q-value for each candidate motion.

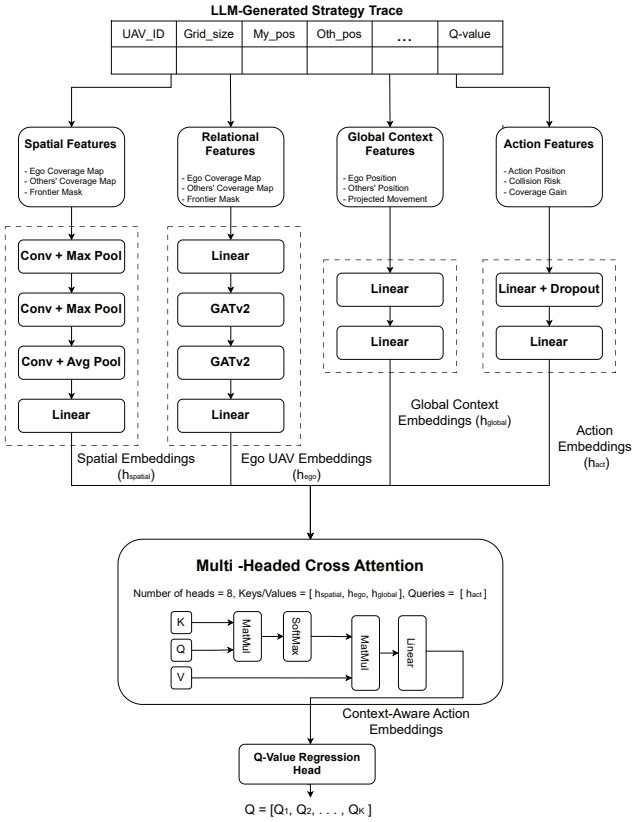


Fig. 3: Overview of the Proposed Model Architecture

1) Spatial Encoder: The spatial tensor X_{grid} is processed by a hierarchical convolutional encoder. The spatial CNN comprises of three convolutional blocks of 64, 128 and 256 channels each followed by batch normalization and max-pooling. The final output is an embedding vector::

$$h_{spatial} = f_{grid}(X_{grid}) \in \mathbb{R}^d. \tag{17}$$

This vector encodes uncovered regions, frontier complexity, UAV foot prints and global spatial layout at the current time step.

2) Multi-Agent Graph Encoder: As already discussed, each UAV is represented by an 8-dimensional descriptor capturing UAV-specific attributes: $H_{agents} \in \mathbb{R}^{N \times 8}$. This is processed by a graph attention module. The underlying graph is fully connected among all active UAV nodes $E = \{(i, j) \mid i \neq j, \quad i, j \in \{1, \ldots, N\}\}$ to capture all pairwise UAV interactions.

Two GAT-v2 layers compute attention-weighted relational embeddings. For each UAV node i:

$$\mathbf{h}'_i = \sum_{j \neq i} \alpha_{ij} W \mathbf{h}_j. \tag{18}$$

where, h_j is the feature vector of node (UAV) j before encoding its current state; W is a learned linear projection that

maps those node features into the attention/message space; α_{ij} is the attention coefficient; and h'_i is the updated feature for node i.

Only the embedding corresponding to the ego UAV (node 0) is then extracted: $\mathbf{h}_{\text{ego}} \in \mathbb{R}^d$. This vector represents the relational context of the ego UAV relative to all other agents.

3) Action Encoder: The ego UAV's action space consists of all possible next cells within its movement distance d_m. Each of these actions are embedded by a two-layer shared MLP:

$$\mathbf{h}_{\text{act}}(k) \in \mathbb{R}^d, \quad k = 1, \ldots, |\mathcal{A}|. \tag{19}$$

This transforms raw action descriptors into latent vectors capturing safety, utility, and local geometric compatibility of each action.

4) Global Context Encoder: The overall condition of the mission is encoded into a 6-dimensional global vector. This includes information like grid size, number of UAVs, coverage percentage, ego coverage radius and obstacle density.

This is the macro-level mission information which impacts the long-horizon strategy. This is embedded through a lightweight MLP:

$$\mathbf{z}_{\text{global}} \in \mathbb{R}^6 \longrightarrow \mathbf{h}_{\text{global}} \in \mathbb{R}^d. \tag{20}$$

5) Multi-Headed Cross-Attention: To evaluate each action in context, the model integrates the three major contextual embeddings:

$$C = \{\mathbf{h}_{\text{spatial}}, \mathbf{h}_{\text{ego}}, \mathbf{h}_{\text{global}}\} \in \mathbb{R}^{3\times d}. \tag{21}$$

Each action embedding serves as a query, while C provides the key–value pairs:

$$\tilde{\mathbf{h}}_{\text{act}}(k) = \text{MHA}(\text{Query} = \mathbf{h}_{\text{act}}(k), \text{Key/Value} = C) \tag{22}$$

This helps model focus on relevant spatial features, attend to high-impact team mates and make reasonable decisions. Cross-attention transforms each action into a context-aware embedding.

6) Q-Value Prediction Head: For each action k, the final evaluation vector concatenates:

$$\mathbf{z}_k = [\tilde{\mathbf{h}}_{\text{act}}(k) \parallel \mathbf{h}_{\text{spatial}} \parallel \mathbf{h}_{\text{ego}} \parallel \mathbf{a}_k] \in \mathbb{R}^{(3d+15)}. \tag{23}$$

A four-layer MLP maps this fused representation to a scalar action-value: $Q_k = f_\theta(\mathbf{z}_k)$. All predicted Q-values form the output vector:

$$\mathbf{Q} = [Q_1, Q_2, \ldots, Q_K] \in \mathbb{R}^K. \tag{24}$$

Hence, the architecture naturally supports variable-sized action spaces and dynamic multi-agent environments.

F. Training Objective and Loss Function

The proposed model is trained to predict action-value functions for a discrete and variable-sized action space using LLM's supervision. Given a state s_i, the network outputs predicted Q-values $\hat{Q}_{i,a}$ for each feasible action $a \in \mathcal{A}_i$. The expert (LLM generated trace) provides corresponding target Q-values $Q_{i,a}$.

The overall training objective is defined as a weighted combination of three complementary loss terms:

$$\mathcal{L} = \lambda_{\text{MSE}}\,\mathcal{L}_{\text{MSE}} + \lambda_{\text{Rank}}\,\mathcal{L}_{\text{Rank}} + \lambda_{\text{Stay}}\,\mathcal{L}_{\text{Stay}}, \tag{25}$$

1) Confidence-Weighted Mean Squared Error Loss: The primary supervision signal is computed across a batch of B training samples using a confidence-weighted mean squared error (MSE) loss between predicted and target Q-values:

$$\mathcal{L}_{\text{MSE}} = \frac{\sum_{i=1}^{B}\sum_{a\in\mathcal{A}_i} m_{i,a}\,(1+\sigma_i)\left(\hat{Q}_{i,a} - Q_{i,a}\right)^2}{\sum_{i=1}^{B}\sum_{a\in\mathcal{A}_i} m_{i,a}}. \tag{26}$$

The action mask $m_{i,a}$ ensures that padded or invalid actions do not contribute to the loss. The weighting factor $1 + \sigma_i$ uses the standard deviation of target Q-values σ_i to increase the contribution of states where the expert exhibits a clear preference among actions, thereby emphasizing confident supervision and reducing overfitting to ambiguous states.

2) Adaptive Pairwise Ranking Loss: While regression ensures numerical accuracy of Q-values, correct decision-making depends primarily on the relative ordering of actions. To explicitly enforce this structure, we introduce an adaptive pairwise ranking loss.

For each state s_i, an adaptive margin is defined as:

$$\delta_i = \delta_0\,(1+\sigma_i), \quad \delta_0 = 0.05. \tag{27}$$

For each action $a \neq a_i^*$, a ranking constraint is enforced only if the expert demonstrates a sufficiently strong preference: $Q_{i,a_i^*} - Q_{i,a} > \delta_i$. The corresponding pairwise ranking loss is then defined as:

$$\ell_{i,a} = w_{i,a} \cdot \max\left(0, \delta_i - \left(\hat{Q}_{i,a_i^*} - \hat{Q}_{i,a}\right)\right), \tag{28}$$

Here, a_i^* denotes the LLM-preferred action in state s_i, defined as

$$a_i^* = \arg\max_{a\in\mathcal{A}_i} Q_{i,a}, \tag{29}$$

where $Q_{i,a}$ represents the target Q-value assigned by the expert policy to action a. The adaptive weight $w_{i,a}$ in (28) is used to reflect the strength of expert preference and is defined as:

$$w_{i,a} = \min\left(\frac{\left|Q_{i,a_i^*} - Q_{i,a}\right|}{0.5},\ 2.0\right). \tag{30}$$

The overall ranking loss is computed as:

$$\mathcal{L}_{\text{Rank}} = \frac{1}{M}\sum_{i=1}^{B}\sum_{\substack{a\in\mathcal{A}_i \\ a\neq a_i^*}} \ell_{i,a}, \tag{31}$$

where M denotes the total number of valid action pairs satisfying the ranking condition.

3) Coverage-Aware Stay Action Penalty: In multi-UAV coverage tasks, remaining stationary during low coverage is generally suboptimal. To encode this domain-specific prior while preserving flexibility near task completion, we introduce a coverage-aware stay action penalty.

A low-coverage indicator is defined as:

$$\mathbf{1}_{\text{low}}(i) = \begin{cases} 1, & \text{if } c_i < 95\%, \\ 0, & \text{otherwise.} \end{cases} \tag{32}$$

The stay penalty loss is given by

$$\mathcal{L}_{\text{Stay}} = \frac{1}{B} \sum_{i=1}^{B} \mathbf{1}(c_i < 95\% \ \wedge \ a_i^* \neq a_{\text{stay}}) \times \max\left(0, \ \hat{Q}_{i,a_{\text{stay}}} - 0.3\right). \tag{33}$$

This term penalizes assigning excessive value to the stay action when the expert favors movement and overall coverage remains low.

IV. Simulation Results

A. Simulation Settings and Training

We used *Gemini 2.5-Flash* as the LLM to gather our strategy traces. We collected 20 episodes (with coverage threshold $\tau = 99\%$) of data, which were partitioned into 13 episodes for training, 4 episodes for validation and, 3 episodes for testing. The simulation environment was initialized stochastically at the start of each episode to cover a wide range of operational scenarios. As detailed in Table II, the grid dimensions were selected from a set of $\{20, 25, 30, 35\}$ units to simulate varying area sizes, while the number of UAVs ranged from 1 to 7. To facilitate discrete spatial reasoning compatible with grid-based logic, the coverage radius of each UAV ($r \in \{1, 2, 3\}$) was defined using the Chebyshev distance metric. Consequently, the candidate actions for each UAV were constrained to the cells within this radius. The data generation process followed a sequential query loop: each UAV queried the LLM to obtain Q-values for all feasible actions, and movement was executed only after all agents had determined their optimal local move. This process iterated until the collective grid coverage exceeded the target threshold.

Model training was performed on an Intel Xeon CPU with two threads and 12.7 GB of system RAM. Training was performed for up to 60 epochs. To prevent overfitting, we used early stopping with a patience of 15 epochs. The network was optimized using the AdamW optimizer with a learning rate of 3×10^{-4} and a weight decay of 10^{-5}. The scheduler reduced the learning rate by a factor of 0.5 upon patience detection (patience = 5). The final model was computationally lightweight, with a size of around 20 MB and an average latency of 26ms per inference.

B. Simulation Results

We start by comparing the coverage percentages achieved by the proposed model and the LLM in the three test episodes. As shown in Fig. 4, our model achieves a similar performance to the LLM in all three episodes. We observe that our model could also achieve larger coverage ratios in some intermediate time steps.

Parameters	Values
LLM	Gemini-2.5-Flash
Grid dimensions W, H	$\{20, 25, 30, 35\}$
UAV count (N)	$[1, 7]$
Coverage radius r_i	Chebyshev distance in $1, 2, 3$
Coverage Threshold τ	Chebyshev distance in $1, 2, 3$
Training episodes	13
Validation episodes	4
Candidate Actions	Cells in coverage radius
Epochs Trained	60 (early stopped, patience = 15)
Batch Size	32
Learning Rate	3×10^{-4}
Optimizer	AdamW, Weight Decay = 10^{-5}
LR Scheduler	ReduceLROnPlateau (factor=0.5, patience=5)

TABLE II: Simulation and training parameters and values.

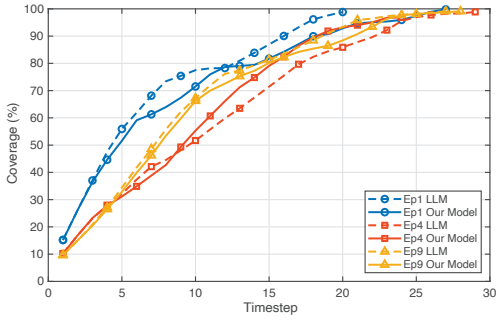


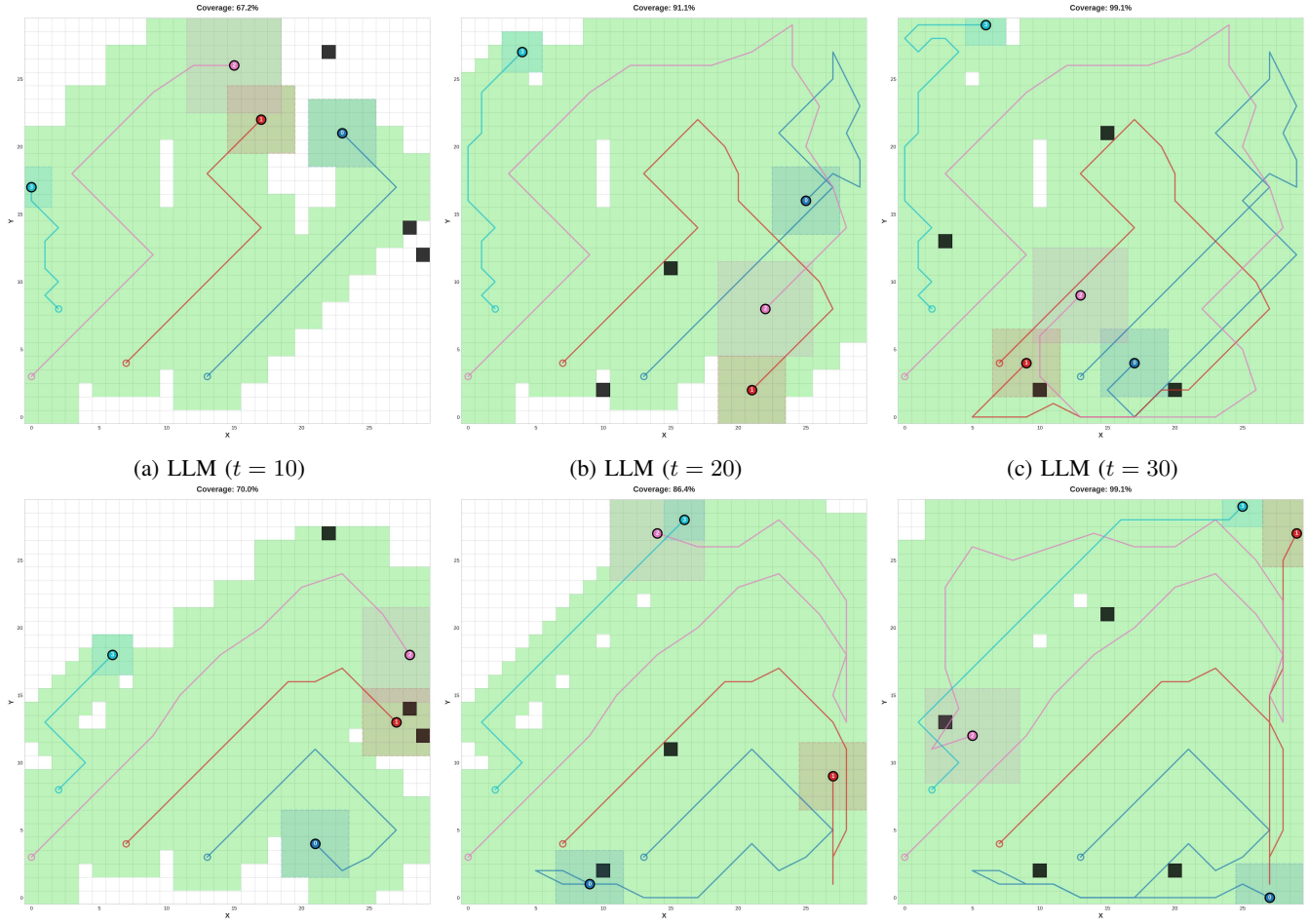
Fig. 4: Comparison of coverage percentages by LLM and the developed model in different episodes.

As an example, we show the coverage maps and UAV paths in one scenario at three different time steps in Fig. 5. The figures illustrate that our model can choose different paths for UAVs but still ends up covering similar amount of area around the same timestep. This shows that our model is able to understand the underlying features of the LLM policies properly.

Finding 1. Our model is able to achieve its goal with performance comparable to LLM, suggesting that our model learns the underlying decision-making policy of the LLM while being much faster, smaller, and computationally efficient.

1) Generalization and Scalability Analysis: To test how our model scales and generalizes to unseen scenarios, we omitted episodes with specific UAV counts from the collected 20 episodes and trained a separate model for each scenario. Table III presents these scenarios alongside other settings such as the grid size and UAV coverage radii. Each of these models was then tested on the omitted scenarios with unseen UAV counts.

The results show that our model was generally able to reach the coverage threshold in timesteps similar to or slightly



(a) LLM ($t = 10$) (b) LLM ($t = 20$) (c) LLM ($t = 30$)

(d) Our model ($t = 10$) (e) Our model ($t = 20$) (f) Our model ($t = 29$)

Fig. 5: UAV paths generated by LLM (a-c) and our model (d-f) for an example scenario at intermediate and final timesteps.

Missing UAVs	**Grid Size**	**LLM Steps**	***Model Steps***	**UAV Radii**
2	25×20	43	**44**	1,2
2	30×20	21	**30**	3,3
4	30×35	28	**18**	3, 3, 2, 2
4	30×25	25	**32**	1, 1, 3, 3
5	30×35	20	**19**	1, 3, 2, 3, 3
5	30×30	38	**41**	1, 2, 3, 1, 1
6	30×25	15	**21**	3, 2, 3, 2, 2, 1

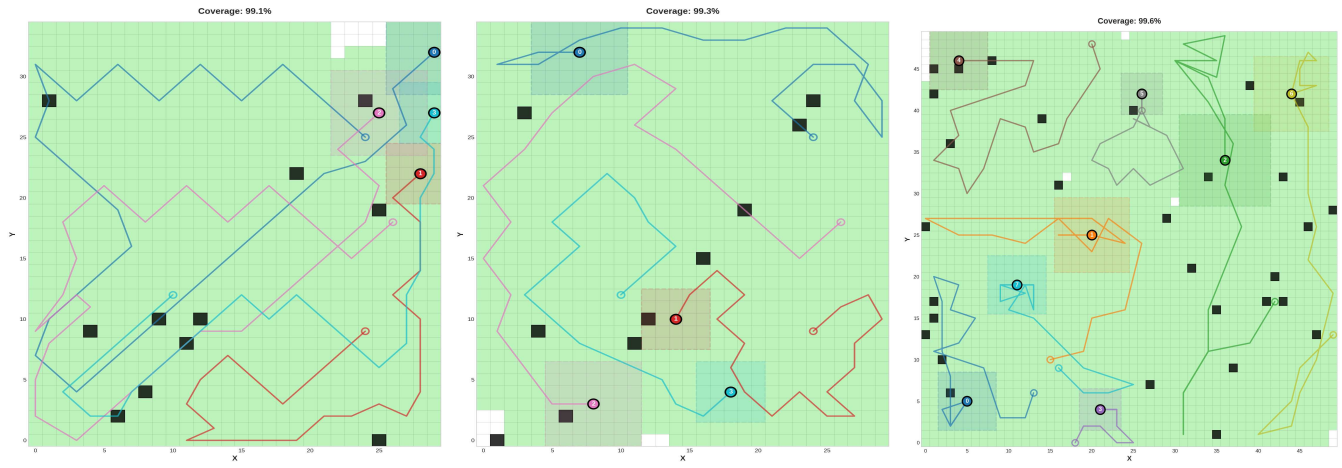
TABLE III: Time steps required to exceed the coverage threshold under varying UAV configurations.

worse than the LLM despite having never been exposed to such samples during training. In certain cases, our model even achieved better performance than the LLM, as evidenced in Fig. 6a-b. These results demonstrate that our model can generalize as well as the LLM while possessing a significantly small model size and having been developed from limited training data.

We also checked how well our approach performs when given scenarios contain attributes highly scaled from the training data. In Fig 6c, we can see a simulation of 8 UAVs operating in a 50x50 grid. Although these settings were out of ranges seen in training data, the model could result in reasonable paths for UAVs, reaching the desired coverage ultimately, showing another evidence of its generalizability. However, we can see that the model struggles a bit in providing straight paths as in previous scenarios, which could be mitigated with more training data.

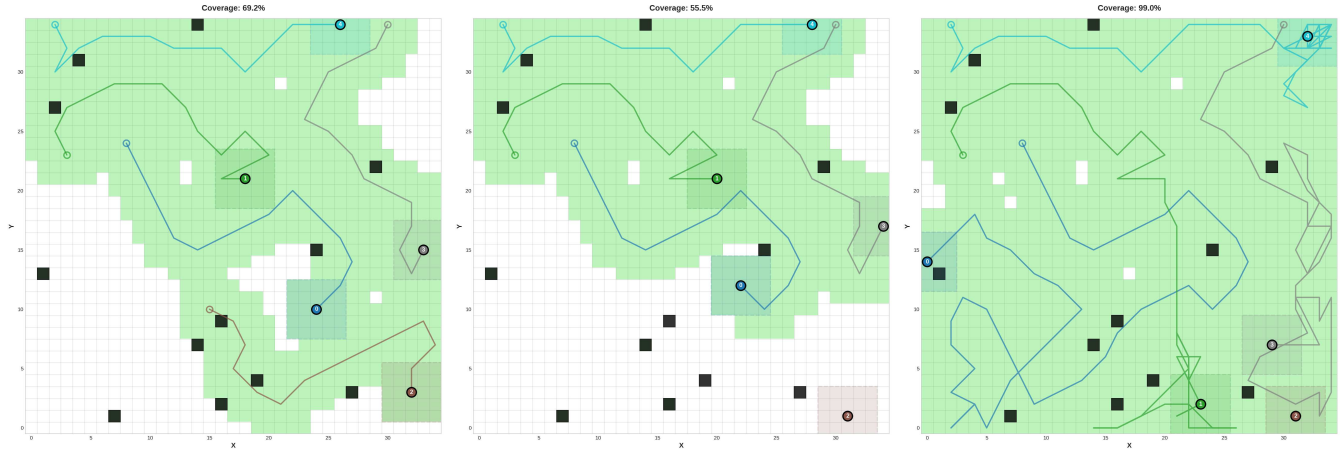
Finding 2. Our model architecture and training methodology demonstrate significant robustness, generalizing effectively to scenarios outside the training distribution and maintaining performance across extreme environmental scales thanks to its sample-efficient nature.

2) Adaptability and Consistency Analysis: In real-world operations, missions may encounter unexpected dynamic events, such as a UAV failure or the deployment of additional agents. In Fig. 7, UAV 2 is simulated to fail at time step 15, resulting in the loss of coverage for its covered cells. As illustrated, the remaining UAVs adapt and coordinate to successfully achieve the coverage goal.



(a) LLM output ($t = 28$) (b) Model output ($t = 18$) (c) 8 UAVs in a 50x50 grid ($t = 21$).

Fig. 6: The proposed model generalizes to scenarios absent from the training data, attaining the coverage threshold faster than the LLM in (a)-(b). Furthermore, as shown in (c), it adapts to scaled grid sizes and UAV counts not within the distribution of training data.



(a) Before UAV 2 breaks ($t = 15$) (b) When UAV 2 breaks down ($t = 15$) (c) Final coverage ($t = 47$)

Fig. 7: Results of our model when a UAV breaks down in the middle of a mission.

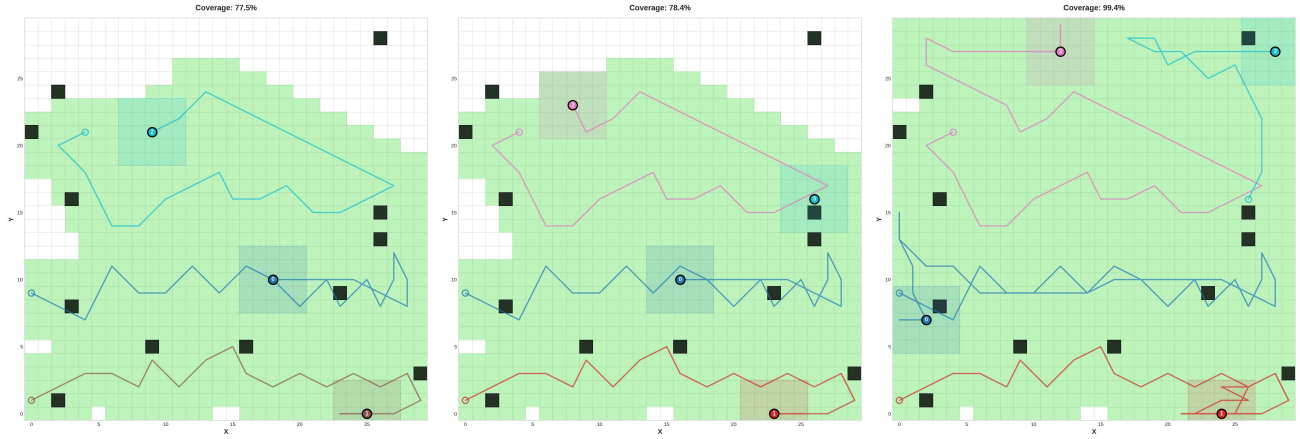
Similarly, in Fig. 8, a new UAV with a radius of 2 joins an existing fleet of three UAVs while the mission has not been completed yet. The existing UAVs adapt to this change and all four of them effectively collaborate to satisfy the coverage requirement, as demonstrated in the figure.

To evaluate the consistency of our model, we selected five random configurations and executed 10 independent trials with different initializations for each. The box plot in Fig. 9 shows the distribution of time steps required for the model to reach the coverage threshold in each configuration. As shown, the model generally reaches the coverage threshold within a consistent time range for any configuration. However, in the second configuration, while the model reached high coverage levels (97.9% and 98.7%), it failed to converge to 99% and its coverage percentage stagnated until it reached the maximum limit of 80 steps.

Finding 3. The model demonstrates the ability to adapt to dynamic scenario changes without prior preparation, showcasing resilience even in adverse conditions. Likewise, it is also robust as it consistently converges to similar solutions when tested across multiple trials of comparable scenarios.

V. Conclusion

In this paper, we have studied the LLM-guided optimization of UAV trajectories in a multi-UAV coverage mission problem. By modeling the problem through discrete strategies for each time step of each UAV, we could obtain results through the LLM reasoning. We then trained a multi-modal customized model using the strategy traces obtained from LLMs and have demonstrated that the produced model can achieve not only



(a) Before UAV 3 is introduced ($t = 24$) (b) When UAV 3 is introduced ($t = 25$) (c) Final coverage ($t = 40$)

Fig. 8: Results of our model when a new UAV joins in the middle of a mission.

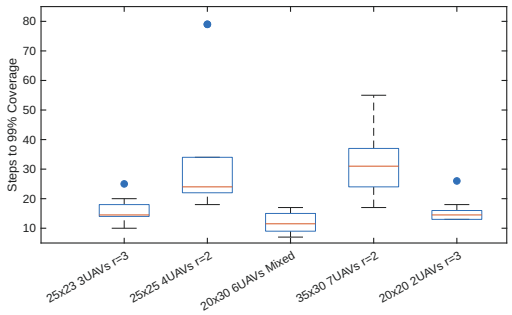


Fig. 9: Consistency of time steps for the model to solve random initializations (10) in five different configurations.

similar performance with LLM but it can also generalize and scale well to unknown scenarios.

In our future work, we will consider more realistic communication and sensing models for UAVs together with three dimensional movement patterns. We will also extend our training data and evaluate the generalization capabilities of the proposed model further.

REFERENCES

[1] H. Wang, H. Zhao, J. Zhang, D. Ma, J. Li, and J. Wei, "Survey on unmanned aerial vehicle networks: A cyber physical system perspective," *IEEE Comm. Surveys & Tutorials*, vol. 22, no. 2, pp. 1027–1070, 2019.

[2] Z. Mou, Y. Zhang, F. Gao, H. Wang, T. Zhang, and Z. Han, "Deep reinforcement learning based three-dimensional area coverage with UAV swarm," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 10, pp. 3160–3176, 2021.

[3] P. Shukla, S. Shukla, and A. K. Singh, "Trajectory-prediction techniques for unmanned aerial vehicles (uavs): A comprehensive survey," *IEEE Communications Surveys & Tutorials*, 2024.

[4] A. Chapnevis and E. Bulut, "Time-efficient approximate trajectory planning for AoI-centered multi-UAV IoT networks," *Internet of Things*, vol. 29, p. 101461, 2025.

[5] A. Chapnevis and E. Bulut, "UAV mesh network trajectory planning for age optimal data collection in infrastructureless areas," in *IEEE International Conf. on Communications (ICC)*, 2024, pp. 1563–1568.

[6] P. Kumar, K. Pal, and M. C. Govil, "Comprehensive review of path planning techniques for unmanned aerial vehicles (uavs)," *ACM Computing Surveys*, vol. 58, no. 3, pp. 1–44, 2025.

[7] E. Merlo, M. Lagomarsino, and A. Ajoudani, "A human-in-the-loop approach to robot action replanning through llm common-sense reasoning," *IEEE Robotics and Automation Letters*, 2025.

[8] B. Lin, Y. Nie, K. Loun Zai, Z. Wei, M. Han, R. Xu, M. Niu, J. Han, L. Lin, C. Lu *et al.*, "Evolvenav: Self-improving embodied reasoning for llm-based vision-language navigation," pp. arXiv–2506, 2025.

[9] M. Movahedi and J. Choi, "The crossroads of llm and traffic control: A study on large language models in adaptive traffic signal control," *IEEE Transactions on Intelligent Transportation Systems*, 2024.

[10] Y. Tian, F. Lin, Y. Li, T. Zhang, Q. Zhang, X. Fu, J. Huang, X. Dai, Y. Wang, C. Tian *et al.*, "UAVs meet LLMs: Overviews and perspectives towards agentic low-altitude mobility," *Info. Fusion*, vol. 122, 2025.

[11] W. Xiao, C. Shi, M. Chen, A. V. Vasilakos, M. Chen, and A. Farouk, "LLM-based UAV Path Planning for Autonomous and Adaptive Industry Systems," *ACM Trans. on Autonomous and Adaptive Systems*, 2025.

[12] Y. Wang, J. Farooq, H. Ghazzai, and G. Setti, "Multi-uav placement for integrated access and backhauling using llm-driven optimization," in *IEEE Wireless Comm.and Networking Conference*, 2025, pp. 1–6.

[13] B. Dharmalingam, R. Mukherjee, B. Piggott, G. Feng, and A. Liu, "Aero-llm: A distributed framework for secure uav communication and intelligent decision-making," *arXiv preprint arXiv:2502.05220*, 2025.

[14] Y. Bai, H. Zhao, X. Zhang, Z. Chang, R. Jäntti, and K. Yang, "Toward autonomous multi-uav wireless network: A survey of reinforcement learning-based approaches," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 4, pp. 3038–3067, 2023.

[15] M. Jones, S. Djahel, and K. Welsh, "Path-planning for unmanned aerial vehicles with environment complexity considerations: A survey," *ACM Computing Surveys*, vol. 55, no. 11, pp. 1–39, 2023.

[16] J. Farley, A. Chapnevis, and E. Bulut, "Generalized path planning for collaborative UAVs using reinforcement and imitation learning," in *Proc. of MobiHoc- REUNS Workshop*, October 2023, pp. 457–462.

[17] A. Wong, T. Bäck, A. V. Kononova, and A. Plaat, "Deep multiagent reinforcement learning: Challenges and directions," *Artificial Intelligence Review*, vol. 56, no. 6, pp. 5023–5056, 2023.

[18] A. Goeckner, Y. Sui, N. Martinet, X. Li, and Q. Zhu, "Graph neural network-based multi-agent reinforcement learning for resilient distributed coordination of multi-robot systems," in *IEEE/RSJ International Conf. on Intelligent Robots and Systems (IROS)*, 2024, pp. 5732–5739.

[19] M. Elrod, N. Mehrabi, R. Amin, M. Kaur, L. Cheng, J. Martin, and A. Razi, "Graph based deep reinforcement learning aided by transformers for multi-agent cooperation," *arXiv preprint arXiv:2504.08195*, 2025.

[20] Z. Xiao, P. Li, C. Liu, H. Gao, and X. Wang, "MACNS: A generic graph neural network integrated deep reinforcement learning based multi-agent collaborative navigation system for dynamic trajectory planning," *Information Fusion*, vol. 105, p. 102250, 2024.

[21] S. Javaid, H. Fahim, B. He, and N. Saeed, "Large language models for uavs: Current state and pathways to the future," *IEEE Open Journal of Vehicular Technology*, 2024.

[22] C. Hu, X. Li, D. Liu, H. Wu, X. Chen, J. Wang, and X. Liu, "Teacher-student architecture for knowledge distillation: A survey," *arXiv preprint arXiv:2308.04268*, 2023.