Proceedings of Machine Learning Research vol vvv:1–21, 2026

Online Motion Planning for Connected Multi-Robot Systems using Vision Language Models as High-level Planners

Kunal Garg KGARG24@ASU.EDU
School for Engineering of Matter, Transport, and Energy at ASU

Devika Shaj Kumar Nair DSHAJKUM@ASU.COM
School of Electrical, Computing and Energy Engineering at ASU

Songyuan Zhang SZHANG21@MIT.EDU

Jacob Arkin JARKIN@MIT.EDU

Chuchu Fan CHUCHU@MIT.EDU
Department of Aeronautics and Astronautics, MIT

Abstract

Connected multi-agent robotic systems (MRS) are prone to deadlocks in an obstacle environment where the robots can get stuck away from their desired locations under a smooth low-level control policy. Without an external intervention, often in terms of a high-level command, a low-level control policy cannot resolve such deadlocks, and grid-based planners are very slow for real-time online intervention. Utilizing the generalizability and low data requirements of foundation models, this paper explores the possibility of using vision-language models (VLMs) as high-level planners for deadlock resolution. We propose a hierarchical control framework where a foundation model-based high-level planner helps to resolve deadlocks by assigning a leader for the MRS as well as a set of waypoints for the MRS leader. Then, a low-level distributed control policy is executed to safely follow these waypoints, thereby evading the deadlock. We conduct extensive experiments on various MRS environments using the best available pre-trained VLMs. We compare their performance with a graph-based planner in terms of computational time and effectiveness in helping the MRS reach their target locations. Our results illustrate that, compared to grid-based planners, the foundation models perform better and can assist MRS operating in complex obstacle-cluttered environments to resolve deadlocks efficiently. Project website: https://mit-realm.github.io/VLM-gcbfplus-website/.

Keywords: Multi-robot systems; Vision Language Models; Deadlock Resolution

1. Introduction

Multi-agent robotic systems (MRS) are widely used in various applications today, such as warehouse operations (Li and Ma, 2023; Wurman et al., 2008), self-driving cars (Dinneweth et al., 2022), and coordinated drone navigation in a dense forest for search-and-rescue missions (Tian et al., 2020), among others. In various MRS applications for navigating in unknown environments, such as coverage (Cortes et al., 2004) and formation control (Mehdifar et al., 2020), robot agents must remain connected so that they can actively communicate with each other to share information and build the unknown or partially known environment collectively. Additionally, ensuring safety in terms of collision avoidance and scalability to large-scale multi-agent problems are also crucial requirements of the control design of MRS. When the requirements of connectivity and safety come together, existing methods for multi-agent coordination and motion planning (Ma et al., 2017) often

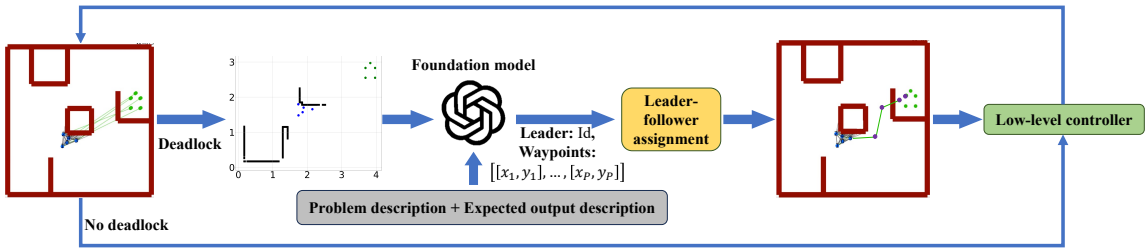


Figure 1: **Overview of the hierarchical control framework**: The robots are shown in blue, their goals in green, and obstacles in red. Based on the currently available environmental information, a foundation model acts as a high-level planner and assigns a leader, as well as a set of waypoints (shown in purple), for the leader of the MRS, resulting in a leader-follower formation. Then, a low-level controller provides a distributed control policy for safety and connectivity.

result in deadlocks, where agents get stuck away from their desired goal locations. Particularly, with the additional requirement of connectivity, even one robot getting stuck in a deadlock results in all the agents getting stuck, making it crucial to resolve the deadlocks for connected MRS. Without external intervention, a smooth low-level control policy cannot resolve such deadlocks. A high-level planner, on the other hand, can intervene in such situations and suggest a set of intermediate waypoints to move the MRS away from the deadlock situation. Since the environment (in terms of locations, sizes, and shapes of obstacles) is a priori unknown, such planning must be done online based on the currently available information. However, traditional path planners, such as grid-based planners, cannot be used for real-time path planning due to their computational demands.

This work proposes a hierarchical control architecture in which a high-level planner can intervene and provide a mechanism to resolve deadlocks. Our proposed planner first configures the MRS in a leader-follower formation and takes the environment information available to the MRS so far. Then, the planner proposes a high-level command in terms of assigning a set of waypoints for the MRS leader to navigate safely in obstacle environments. Motivated by the generalizability and low data requirements of foundation models (Kojima et al., 2022) as well as the recent success of foundation models in assisting a control framework for complex robotics problems (Ren et al., 2024; Huang et al., 2024), we explore the possibility of using vision language models (VLMs) as high-level planners to resolve deadlocks in MRS. However, unlike other works that use foundation models *proactively*, whether directly for planning, translation, or reward design, we are instead interested in using them *reactively* to resolve a class of failure modes in low-level controllers for MRS, namely deadlocks. This helps to ensure that the VLM does not lead to a violation of safety as it is taken care of by the provably safe low-level control policy. When a deadlock is detected, a VLM is prompted to generate a set of deadlock-resolving navigation waypoints for the leader, conditioned on a top-view image-based description of the so far observed environment. This temporary high-level assignment aims to move the MRS out of the deadlock so that the low-level controller can continue progressing toward the goal.

Contributions The contributions of the paper are as follows. 1) We propose a novel hierarchical control architecture using foundation models to resolve deadlocks in an obstacle-cluttered environment for MRS (see Figure 1). For large-scale MRS, where a single leader assignment can lead to a deterioration in performance, we propose a hierarchy of leaders for efficient deadlock resolution. 2) We compare the performance of various VLMs in efficient deadlock resolution for MRS. To illustrate the utility of vision-based models over text-based models, we use large language mod-

els (LLMs) as baseline planners. We first evaluate the spatial reasoning capability of various vision- and text-based foundation models by evaluating their ability in locating the obstacles correctly and generating viable waypoints in the free space. Then, we conduct extensive experiments on a variety of MRS environments, varying the number of agents from 5 to 50 with various VLMs. Our results demonstrate that VLM-based high-level planners are effective at resolving deadlocks in MRS and more efficient in terms of MRS goal-reaching rate in a given time budget as compared to LLMs as well as graph-based planners such as A^* (Hart et al., 1968). We provide a detailed discussion and possible future directions to improve the performance of foundation models as high-level planners for assisting low-level controllers in complex MRS problems.

2. Related work

In recent years, learning-based methods have shown promising results in computing a low-level control policy for complex robotic systems (Dawson et al., 2023; Zhang et al., 2023; Garg et al., 2024). Considering deadlocks, many works have been proposed for detecting and moving out of deadlocks under safety constraints (Grover et al., 2021, 2023; Chen et al., 2024b; Zhang et al., 2025a). However, these works do not consider connectivity constraints. Most similar to our work, (Sinha et al., 2024) uses an LLM to intervene on high-level planning; however, this work addresses only a single agent and chooses among a small set of control strategies. More recently, VLMs have become popular in robotic applications due to their strong semantic reasoning capabilities (Ren et al., 2024). VLMs have shown promising results in reasoning about future actions of robotic systems with partial environment information (Kwon et al., 2023; Ma et al., 2023; Wen et al., 2023; Chen et al., 2024a; Gao et al., 2023; Jiang et al., 2022). In particular, VLMs have been successfully employed in robot navigation tasks (Shah et al., 2023a; Dorbala et al., 2024; Shah et al., 2023b), even when long-horizon reasoning is required (Huang et al., 2024). In addition, VLMs have also been used to convert images to text descriptions for prompting LLMs with state information (Sinha et al., 2024; Xie et al., 2024; Kwon et al., 2023; Shah et al., 2023a; Dorbala et al., 2024); in contrast, our work directly uses the output of a VLM for planning.

3. Problem formulation

The MRS consists of N robots navigating in an obstacle-cluttered environment to reach their goal locations $\{p_i^{\text{goal}}\}_{i=1}^N$. The environment $\mathcal{X} \subset \mathbb{R}^2$ consists of stationary obstacles $\mathcal{O}_l \subset \mathbb{R}^2$ for $l \in \{1, 2, \ldots, M\}$, which denote walls, blockades, and other obstacles in the path of moving agents. Each agent has a safety distance $r > 0$ and a limited sensing and communication radius $R > r$ such that the agents can only sense and communicate with other agents or obstacles if they lie within this radius. The agents use LiDAR to sense the obstacles, and the observation data for each agent i consists of n_{rays} evenly-spaced LiDAR rays $y_j^{(i)}$ originating from each robot and measures the relative location of obstacles. The time-varying connectivity graph $\mathcal{G}(t) = (\mathcal{V}(t), \mathcal{E}(t))$ dictates the network among the agents and obstacles. Here, $\mathcal{V}(t) = \mathcal{V}^a \cup \mathcal{V}^o(t)$ denotes the set of nodes, where $\mathcal{V}^a = \{1, 2, \ldots, N\}$ denotes the set of agents, $\mathcal{V}^o(t)$ is the collection of all LiDAR hitting points at time $t \geq 0$, and $\mathcal{E}(t) \subset \mathcal{V}^a(t) \times \mathcal{V}(t)$ denotes the set of edges, where $(i, j) \in \mathcal{E}(t)$ means the flow of information from node j to agent i. In addition to safety, the resulting underlying graph topology for the MRS is required to remain connected (see (Zavlanos and Pappas, 2008) for MRS connectivity) so that robots can build team knowledge and share information, where given a

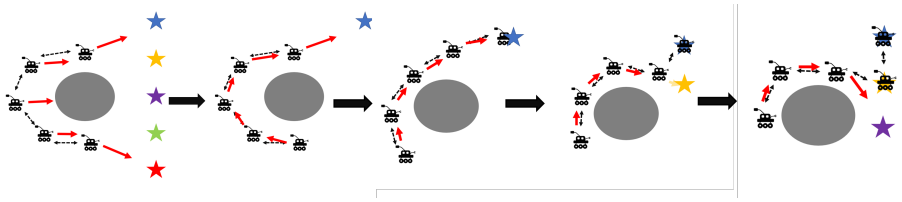


Figure 2: Illustration of deadlock resolution using leader-follower formation. The gray circle in the center is an obstacle, and the goals are denoted by the stars.

communication radius $R > 0$, two agents i, j are connected if $\|p_i - p_j\| \leq R$. The formal problem statement studied in this paper is described next (more details in Appendix A of the supplementary material).

Problem 1 *Consider the multi-agent system with connected initial topology $\mathcal{G}(0)$, safety parameters $r > 0$, a communication radius $R > 0$, a set of stationary obstacles $\{\mathcal{O}_j\}_{j=1}^{M}$, goal locations $\{p_i^{\text{goal}}\}_{i=1}^{N}$, and a terminal time $T_F > 0$. Design a distributed control architecture such that*

- ***Safety**: Each agent maintains a safe distance from other agents and obstacles at all times, i.e., $\|p_i(t) - p_j(t)\| \geq 2r, \forall j \neq i$ and $\|y_j^{(i)}(t)\| > r, \ \forall j$ for all $t \geq 0$;*
- ***Connectivity**: Graph $\mathcal{G}(t)$ remains connected $\forall t \geq 0$;*
- ***Performance**: Agents reach their respective goals, i.e., $\inf_{t \leq T_F} \|p_i(t) - p_i^{\text{goal}}\| = 0$.*

4. Hierarchical control architecture

In an MRS problem with connectivity requirements, the presence of obstacles can lead to deadlock situations for the entire MRS, as illustrated in Figure 1. We propose a hierarchical control architecture consisting of a low-level control policy that accounts for safety and connectivity constraints and a high-level planner that assists the low-level controller with the goal-reaching requirement upon detection of a deadlock. We first describe how we detect the deadlocks.

Deadlock detection In the proposed hierarchical architecture, the high-level planner is triggered upon detection of a deadlock, which we define as a situation when the average speed of the MRS falls below a minimum threshold $\delta_v > 0$, i.e., $\frac{\sum_i \|\dot{p}_i\|}{N} < \delta_v$ and the average distance of the agents from their goals is at least $\delta_d > 0$, i.e., $\frac{\sum_i \|p_i - p_i^{\text{goal}}\|}{N} > \delta_d$. Since the graph topology is connected, the average MRS speed can be computed through consensus updates.

Leader-follower formation When a deadlock is detected, the high-level planner assigns a leader among the N agents along with a set of intermediate waypoints for the leader, so that the leader does not get stuck in a deadlock due to obstacles on its path to its goal. The MRS remains in the leader-follower mode for a fixed time $T_{LF} > 0$, which is a user-defined hyper-parameter. We provide the details of the VLM-based high-level planner in Section 4.1. Once a leader and its waypoints are obtained, the MRS reconfigures into a leader-follower formation. Figure 2 illustrates a relatively simple scenario consisting of 5 agents and one obstacle where a goal-reaching control policy with the ability to maintain connectivity and safety leads to a deadlock. Through the leader-follower formation, the MRS is able to evade this situation and complete its task. The leader-follower assignment is done by sequentially assigning the closest unassigned follower to its closest assigned agent as its leader. Let $\mathcal{V}_{\text{lead}}(t, k)$ be the set of agents that have been assigned as a leader

at time t, iteration k, initiated as $\mathcal{V}_{\text{lead}}(t,0) = \{i_{\text{lead},0}\}$, where $i_{\text{lead},0} \in \mathcal{V}$ is the leader agent. Then, the $k-$th follower with $k \geq 1$ is chosen as $i_{\text{follow},k} = \arg\min_{j \in \mathcal{V} \setminus \mathcal{V}_{\text{lead}}(t,k-1)} \min_{i \in \mathcal{V}_{\text{lead}}(t,k-1)} \|p_i - p_j\|$, and this follower is added to the set of the leaders, i.e., $\mathcal{V}_{\text{lead}}(t,k) = \mathcal{V}_{\text{lead}}(t,k-1) \cup \{i_{\text{follow},k}\}$. The leader for the $k-$th follower is given as $i_{\text{lead},k} = \underset{i \in \mathcal{V}_{\text{lead}}(t,k-1)}{\arg\min} \|p_{i_{\text{follow},k}} - p_i\|$. The process is repeated till each agent i is assigned an agent i_{lead} to follow. Next, for each agent i that is at a given minimum distance away from its goal, i.e., if $\|p_i - p_i^{\text{goal}}\| \geq d_{\min}$ for some $d_{\min} > 0$, their temporary goal is chosen as the location of their leaders, i.e., $\bar{p}_i^{\text{goal}} = p_{i_{\text{lead}}}$. The complete leader-follower assignment algorithm is described in Appendix B of the supplementary material.

Multi-leader assignment using k-means clustering In the cases of large-scale MRS, e.g., $N \geq 10$, assigning one leader to the MRS might lead to a sub-optimal performance. To this end, we decompose the MRS into sub-teams and assign a sub-leader to each of the sub-teams along with a *main* leader for the MRS. The decomposition of the MRS agents into $K \geq 1$ disjoint clusters $\mathcal{V}_k^a, k = 1, ..., K$ such that $\mathcal{V}^a = \cup_{k=1}^K \mathcal{V}_k^a$ and $\mathcal{V}_i^a \cap \mathcal{V}_j^a = \emptyset$ for $i \neq j$, is performed based on the inter-agent distances using k-means clustering (MacQueen et al., 1967). The main leader $l_M \in \mathcal{V}^a$ is chosen as the agent with minimum distance to its goal. Once the clusters of agents $\{\mathcal{V}_k^a\}$ are obtained, a sub-leader l_k is chosen based on the vicinity of the agents in $\mathcal{V}_k^a$ to the cluster of the main leader $\mathcal{V}_M^a$. Note that for large-scale MRS, the high-level planner is utilized only for the waypoint assignment, as the leader is assigned heuristically based on the distance to the goal locations.

4.1. Foundation model-based high-level planner

Based on the success of foundation models in a variety of robotic tasks that require spatial understanding, we explore their utility as the high-level planner for the leader and waypoint assignment. To use a pre-trained foundation model for waypoint assignment, we provide the model with task-relevant context that is expected to be helpful when generating a decision. In particular, the prompt to the model consists of three main components: (1) the *Task description*, (2) an *Environment state*, and (3) the *Desired output*. Next, we explain each of the prompt components in more detail. The exact prompts used in the experiments are provided in Appendix C in the supplementary material.

Task description The initial part of the prompt consists of a description of the deadlock resolution problem for a multi-robot system. This includes the system requirements of maintaining safety, connectivity, and each agent reaching its assigned goal. Further, we include a description of the planner's role in providing high-level commands when the MRS is stuck in a deadlock. The description also includes the number of waypoints $P > 0$ that the planner is supposed to suggest. This component of the prompt is created offline and is fixed for all calls.

Environment state The environment state of the MRS is a necessary context to make a good leader assignment decision, so we encode it in a textual description that is included as a component of the prompt. Since the obstacle information depends on the roll-out of the system, we construct this part of the prompt online after a deadlock has been detected. At any given time instant $t_q \geq 0$ when the VLM is queried, the environment state is constructed via a `base64` encoded `JPEG` image that includes the location of the agents, their goals, and the obstacles seen by the MRS so far for all $t \leq t_q$ whose information comes from the LiDAR data. An example input image to the VLM is given in Figure 1. To assist the VLM with the precise locations of the agent and the goals, we provide their text description as well.

Desired output Finally, we describe the desired output, both in terms of content and format. The high-level planner is responsible for choosing a leader and a set of waypoints for the

leader. To help constrain the model's output and enable consistent output parsing, we request the generated response to be formatted as a `JSON` object with an expected output of the form {"Leader" : Id, "Waypoints" : $[[x_1, y_1], \ldots, [x_P, y_P]]$}.

4.2. Distributed low-level policy

The high-level planner provides the leader and the waypoint information to the low-level controller. One desirable property of the low-level controller is scalability and generalizability to new environments (i.e., changing the number of agents and obstacles) while keeping the MRS safe and connected. Given the leader and goal information in terms of the immediate waypoint to follow, the low-level controller synthesizes an input u_i to maintain connectivity, keep the system collision-free, and drive the system trajectories toward its goals (and in the case of the leader robot, toward its waypoint). While any low-level controller that can satisfy the requirements from Problem 1 can be used, we extend the distributed graph-based policy from (Zhang et al., 2025b). Since the main focus of this paper is on the high-level planner, the details of the low-level controller are provided in Appendix D of the supplementary material.

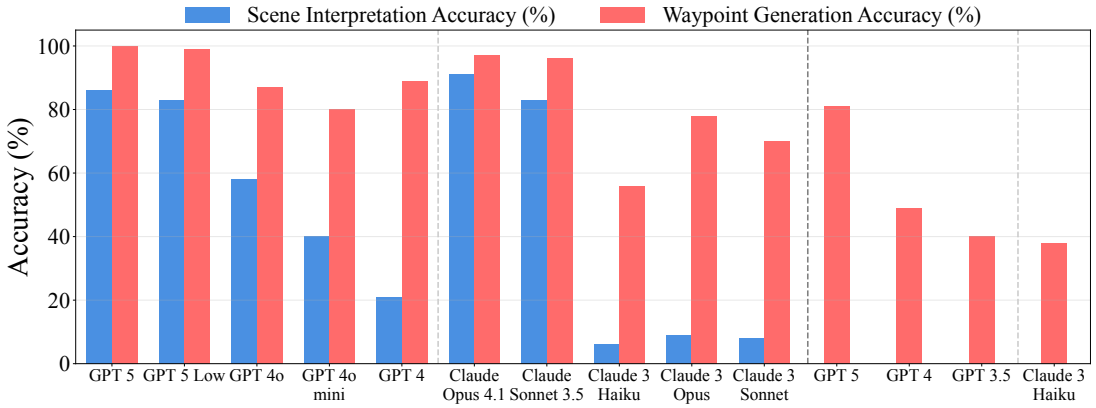


Figure 3: Quantitative accuracy of different VLMs in scene understanding and feasible waypoint-generation capability across 100 environments.

5. Spatial reasoning capability of VLMs

The performance of our proposed framework depends on the ability of VLMs to reason about structured visual environments, if only implicitly, while generating path-planning strategies. Misunderstandings of the environment state can propagate to planning errors. Therefore, we evaluate the performance of VLMs to understand the environment state and to generate feasible waypoints within that state. We randomly generate one hundred 10x10 grid environments, each of which consists of agents, goals, and polygonal obstacles that are encoded with integer-value vertices. To test environment understanding, a VLM is tasked with reporting the coordinates for each entity (e.g., agent, goal, or obstacle). To test feasible waypoint generation, a VLM (or LLM using a text-based encoding of the environment) is tasked with generating a feasible sequence of waypoints whose straight-line connectivity satisfies the constraints of staying within the bounds and avoiding any obstacles (see Appendix C for the prompts and an example scenario used for this evaluation).

Results for both evaluations are reported in Figure 3. The blue bars show the average accuracy of various VLM models on the task of scene interpretation over 100 environments, where accuracy

is defined as the percentage of correctly identified locations of the obstacles' vertices, the agent, and the goal (within a tolerance of 0.5 units). The red bars indicate the accuracy of various VLM and LLM models in the task of generating feasible waypoints. As can be seen from Figure 3, among various VLMs, it is observed that the accuracy in correctly identifying the spatial configuration of an environment is generally higher for newer models (e.g., `GPT-5`>`GPT4o`>`GPT4o-mini`>`GPT-4`). A similar trend is observed for the accuracy of correct waypoint generation as per the given constraints (with the exception of `GPT-4-Turbo`). For the task of waypoint generation, LLMs generally performed worse than VLMs, even when comparing the performance of the same model used as a VLM (e.g., `GPT-4-Turbo` or `GPT-5`), which suggests that the vision modality of foundations is better for spatial reasoning. These preliminary results provide a preview of what we might expect when these models are used as part of the hierarchical control framework for motion planning, which we evaluate in the next section.

6. Evaluations

The objective of these numerical evaluations is to assess the viability of foundation models as real-time, high-level planners and to compare their performance with traditional, grid-based planners. To the best of the authors' knowledge, there is no work that solves this problem *end-to-end*. Hence, we fix the hierarchical architecture of the control framework and test the viability of the foundation models as high-level planners in comparison to grid-based planners while using the same low-level controller. To assess the effectiveness of the high-level planner, we perform extensive experiments on a variety of MRS environments to measure the performance in terms of i) the reach rate or the percentage of agents reaching their goals; ii) the number of high-level interventions needed during the rollout; iii) the time taken in calling the high-level planner; and iv) the tokens used in each intervention (for assessing the cost-efficiency of using proprietary pre-trained foundation models).

6.1. Experiment setup

We perform experiments on two sets of environments, namely structured hand-crafted environments (termed "Room") with a small number of agents, i.e., $N = 5$, and unstructured maze-like environments (termed "Maze") with a large number of agents, i.e., $N = 25$ or 50 (see Figure 4).

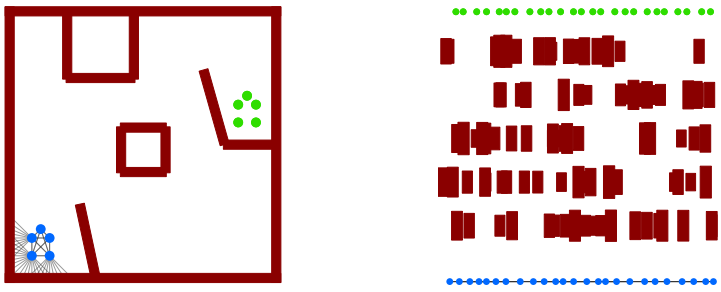


Figure 4: "Room" environment with $N = 5$ (left) and "Maze" environment with $N = 25$ (right). The agents are shown in blue, the goals in green, and the obstacles in red color.

"Room" environments The "Room" represents an enclosed warehouse or an apartment scenario where the agents are required to reach another part of the environment while remaining inside the boundary and avoiding collisions with walls and other obstacles in the room. The agents start in one corner of the room and propagate their way through the environment to reach their respective

destinations. The obstacles in the room environment are designed so that the low-level GCBF+ control policy invariably gets stuck in a deadlock, making it essential to use a high-level planner. We generate 20 environments with random angles, lengths, and locations of the walls.

"Maze" environment The "Maze" environment consists of N initial and goal locations and M rectangular obstacles of randomly generated sizes and locations. These environments capture a variety of real-world scenarios, such as narrow corridors and a large number of unknown obstacles, that MRS might encounter. For testing, we use 20 Maze environments with $N = 25$ and $M = 100$, and 11 Maze environments with $N = 50$ and $M = 375$.

The two sets of environments considered in the evaluations represent a large variety of operating environments for multi-robot systems. In particular, the first set of environments, the "Room" environments, represent structured environments where obstacles and "walls" might have a structure (although unknown) and can lead to a particular type of deadlock situation. On the other hand, the randomly generated "Maze" environments represent unstructured environments where the (unknown) obstacles can be of random shapes and sizes. Note that the locations, shapes, and sizes of the obstacles are not known to the MRS *a priori*, and the obstacles are detected by an on-board sensor such as LiDAR. An unknown obstacle environment with arbitrary shapes and sizes of obstacles requires real-time planning upon deadlock detection.

Foundation models and baselines We use various VLM models from Anthropic and OpenAI as candidates for the high-level planner. We also utilize various LLM models as baselines to evaluate which modality of the foundation best suits the problem. In all the generated environments, the low-level controller leads to a deadlock, and as a result, the reach rate for just the low-level controller is exactly 0. Hence, we do not include that as a baseline. Additionally, we use an A^*-based high-level planner (Hart et al., 1968) and a "Random" high-level planner where the leader and the waypoints are drawn randomly, as baselines for comparison. For the A^* planner, we represent the working environment of the robots as a 2D grid filled with all the obstacles seen by the MRS by the time the planner is queried. We again emphasize that there is no baseline method that can solve this complete problem and that the A^* planner is used to find a set of waypoints (similar to the foundation models), which are then followed using a low-level controller.

Roll-out for evaluation We roll out MRS trajectories for a fixed number of steps $T = 3000$ for Room environments, $T = 4000$ for the Maze environment with $N = 25$, and $T = 5000$ for the Maze environment with $N = 50$. As discussed in Section 4, the high-level planner intervenes when the MRS is stuck in a deadlock, as defined by the minimum average speed criteria. We use $\delta_v = 0.2$ and $\delta_d = 0.4$ as the threshold to define deadlocks. Once the high-level planner assigns a leader, the MRS remains in the same leader-follower configuration for $T_{LF} = 100$ steps. This helps prevent Zeno behaviors and sets an upper bound on the frequency at which the foundation model is queried.

6.2. Results

Figure 5 plots the various performance criteria for each of the test environments for the considered foundation models and baseline method. Below, we summarize our findings.

Foundation models are more effective than the grid-based and random method: An important feature we note from the experiments is that foundation models do not require any in-context examples to achieve such high performance. This corroborates our motivation for using these models due to their low data requirement. When it comes to comparison to the A^*-based planner, it is evident from all environments that foundation models achieve better performance in terms of a

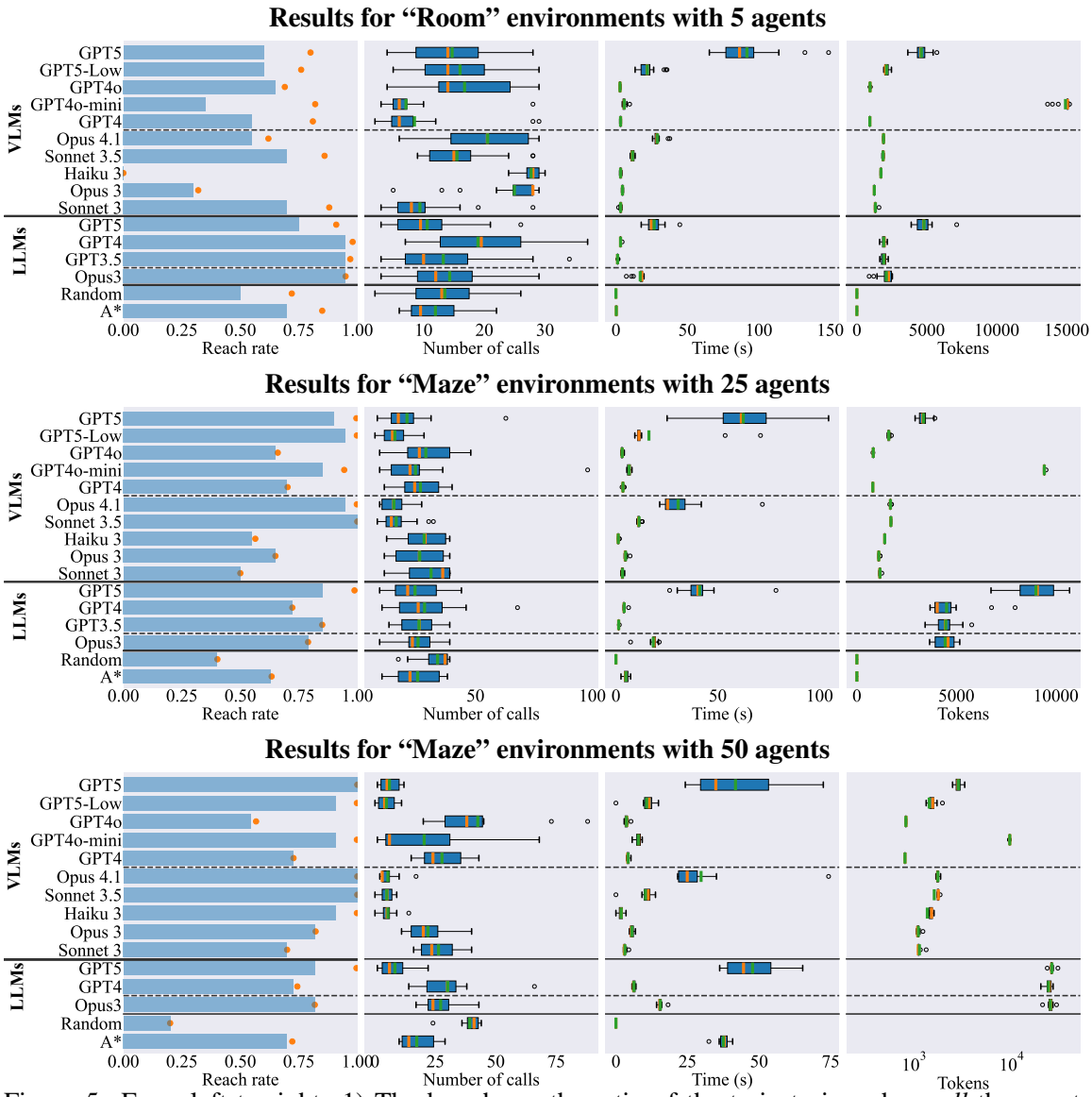


Figure 5: From left to right: 1) The bar shows the ratio of the trajectories where *all* the agents reach their goals over the total number of trajectories, and the orange dot shows the ratio of agents that reach their goals over all agents; 2) Box plot of the number of times the high-level planner intervened; 3) Box plot of the time spent for each high-level planner intervention; and 4) Box plot for the input + output tokens per intervention. In the box plots, the median values are in orange and the mean values are in green.

higher average reach rate. For the small-scale Room environments, the mean reach rate for A^* across 20 test environments is higher than VLMs but is lower than all of the LLMs. For the large-scale Maze environments, the mean reach rate for A^* is lower than most of the VLMs. The computation time of A^* increases as the size and number of obstacles increase, while that of foundation models remains the same. Note also that a randomly chosen leader, along with randomly chosen waypoints, yields a reach rate of less than 0.5 in all cases, with its performance deteriorating as the

number of agents increases. This illustrates that the foundation models provide better inference than a randomly generated one, resulting in a higher completion rate.

VLMs are cheaper and faster than LLMs for large-scale MRS: It is evident that the average number of total tokens required for image-based VLMs remains similar (≈ 1000) across the various environments (see the rightmost column of Figure 5). On the other hand, the tokens for text-based LLMs increase with the increased complexity of the environment in terms of the number of obstacles. This directly affects the cost of using proprietary foundation models, as well as the speed of inference. This is also evident when we compare the time of intervention for the same model used as VLM and LLM (e.g., `GPT-4` VLM and `GPT-4` LLM). Furthermore, the variance in the average tokens used by VLMs across experiments is much lower than that for LLMs. This is because, as the MRS collects more information about its environment (i.e., observes newer obstacles), the text-based description becomes longer. On the other hand, all the new information can be plotted on the same-sized graphic, resulting in the same-sized input to the VLMs. This feature of VLMs is advantageous as it provides predictable costs, runtime, and scalability, especially for large-scale real-time systems. One way of reducing the prompt size for LLMs is to use a portion of the available information instead of the complete available information about the obstacles. We perform ablation on the available information and report the results in Appendix E in the supplementary material.

Reasoning at the cost of time and money: For `GPT-5` model, we compared performance of the model with "reasoning-effort" low and medium (plotted as "GPT-5-VLM-Low" and "GPT-5-VLM" in Figure 5). It is clear that higher reasoning requires significantly longer inference time (up to 4 times), as well as higher output tokens (up to 2 times), but without a clear trend in performance improvement. The performance in terms of the reach rate is the same or even better when the reasoning effort is set to low for smaller environments, and is only slightly worse for the large environment. The large reasoning model, such as `GPT-5`, generates extra reasoning tokens with its main response, resulting in a higher cost and time. However, as evident from the experiments, it performs well even when the reasoning effort is set to low, making it much faster and cost-efficient.

7. Conclusions and Discussion

In this work, we test the hypothesis that VLMs can be used as high-level planners for deadlock resolution in MRS with safety and connectivity constraints. We perform extensive experiments on a variety of foundation models to understand their utility in this task. In comparison to grid-based planners, the foundation models generally performed better, resulting in an affirmative answer to our hypothesis. Our experiments also provide interesting observations and insights on the relative performance of foundation models of various modalities (LLMs and VLMs) in terms of their spatial reasoning capabilities as well as in online planning for small- and large-scale MRS problems, as well as their time and cost efficiency. The high performance of the zero-shot foundation models, particularly newer and larger models such as `GPT-5` with low reasoning effort, across various MRS environments is good evidence that they are promising high-level planners for online motion planning problems. Recently, there has been an increasing interest in automatic prompt optimization for black-box foundation models to find the best prompt design for a given task (Guo et al., 2024; Wang et al., 2024; Fernando et al., 2023) with results showing significant performance improvement over human-designed prompts. Future work includes applying such techniques to the problem of deadlock resolution to maximize the performance of VLMs as a high-level planner.

References

Boyuan Chen, Zhuo Xu, Sean Kirmani, Brian Ichter, Danny Driess, Pete Florence, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. *arXiv preprint arXiv:2401.12168*, 2024a.

Yuda Chen, Meng Guo, and Zhongkui Li. Deadlock resolution and recursive feasibility in mpc-based multi-robot trajectory generation. *IEEE Transactions on Automatic Control*, 2024b.

Jorge Cortes, Sonia Martinez, Timur Karatas, and Francesco Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on robotics and Automation*, 20(2):243–255, 2004.

Charles Dawson, Sicun Gao, and Chuchu Fan. Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods for robotics and control. *IEEE Transactions on Robotics*, 39(3):1749–1767, 2023.

Joris Dinneweth, Abderrahmane Boubezoul, René Mandiau, and Stéphane Espié. Multi-agent reinforcement learning for autonomous vehicles: a survey. *Autonomous Intelligent Systems*, 2(1):27, 2022.

Vishnu Sashank Dorbala, James F. Mullen, and Dinesh Manocha. Can an embodied agent find your "cat-shaped mug"? llm-based zero-shot object navigation. *IEEE Robotics and Automation Letters*, 9(5):4083–4090, 2024.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. *arXiv preprint arXiv:2309.16797*, 2023.

Jensen Gao, Bidipta Sarkar, Fei Xia, Ted Xiao, Jiajun Wu, Brian Ichter, Anirudha Majumdar, and Dorsa Sadigh. Physically grounded vision-language models for robotic manipulation. *arXiv preprint arXiv:2309.02561*, 2023.

Kunal Garg, Songyuan Zhang, Oswin So, Charles Dawson, and Chuchu Fan. Learning safe control for multi-robot systems: Methods, verification, and open challenges. *Annual Reviews in Control*, 57:100948, 2024.

Jaskaran Grover, Changliu Liu, and Katia Sycara. The before, during, and after of multi-robot deadlock. *The International Journal of Robotics Research*, 42(6):317–336, 2023.

Jaskaran Singh Grover, Changliu Liu, and Katia Sycara. Deadlock analysis and resolution for multi-robot systems. In *Algorithmic Foundations of Robotics XIV: Proceedings of the Fourteenth Workshop on the Algorithmic Foundations of Robotics 14*, pages 294–312. Springer, 2021.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ZG3RaNIsO8.

Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

Wenlong Huang, Fei Xia, Dhruv Shah, Danny Driess, Andy Zeng, Yao Lu, Pete Florence, Igor Mordatch, Sergey Levine, Karol Hausman, et al. Grounded decoding: Guiding text generation with grounded models for embodied agents. *Advances in Neural Information Processing Systems*, 36, 2024.

Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: General robot manipulation with multimodal prompts. In *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *ICML 2022 Workshop on Knowledge Retrieval and Language Models*, 2022. URL https://openreview.net/forum?id=6p3AuaHAFiN.

Minae Kwon, Hengyuan Hu, Vivek Myers, Siddharth Karamcheti, Anca Dragan, and Dorsa Sadigh. Toward grounded social reasoning. *arXiv preprint arXiv:2306.08651*, 2023.

Baiyu Li and Hang Ma. Double-deck multi-agent pickup and delivery: Multi-robot rearrangement in large-scale warehouses. *IEEE Robotics and Automation Letters*, 8(6):3701–3708, 2023. doi: 10.1109/LRA.2023.3272272.

Hang Ma, Wolfgang Hönig, Liron Cohen, Tansel Uras, Hong Xu, TK Satish Kumar, Nora Ayanian, and Sven Koenig. Overview: A hierarchical framework for plan generation and execution in multirobot systems. *IEEE Intelligent Systems*, 32(6):6–12, 2017.

Xiaojian Ma, Silong Yong, Zilong Zheng, Qing Li, Yitao Liang, Song-Chun Zhu, and Siyuan Huang. Sqa3d: Situated question answering in 3d scenes. In *International Conference on Learning Representations*, 2023.

James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

Farhad Mehdifar, Charalampos P Bechlioulis, Farzad Hashemzadeh, and Mahdi Baradarannia. Prescribed performance distance-based formation control of multi-agent systems. *Automatica*, 119: 109086, 2020.

Mehran Mesbahi and Magnus Egerstedt. Graph theoretic methods in multiagent networks. In *Graph Theoretic Methods in Multiagent Networks*. Princeton University Press, 2010.

Allen Z Ren, Jaden Clark, Anushri Dixit, Masha Itkina, Anirudha Majumdar, and Dorsa Sadigh. Explore until confident: Efficient exploration for embodied question answering. *arXiv preprint arXiv:2403.15941*, 2024.

Dhruv Shah, Michael Robert Equi, Błażej Osiński, Fei Xia, Brian Ichter, and Sergey Levine. Navigation with large language models: Semantic guesswork as a heuristic for planning. In *Conference on Robot Learning*, pages 2683–2699. PMLR, 2023a.

Dhruv Shah, Błażej Osiński, Sergey Levine, et al. Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action. In *Conference on robot learning*, pages 492–504. PMLR, 2023b.

Rohan Sinha, Amine Elhafsi, Christopher Agia, Matthew Foutter, Ed Schmerling, and Marco Pavone. Real-time anomaly detection and reactive planning with large language models. In *Robotics: Science and Systems*, 2024.

Yulun Tian, Katherine Liu, Kyel Ok, Loc Tran, Danette Allen, Nicholas Roy, and Jonathan P How. Search and rescue under the forest canopy using multiple uavs. *The International Journal of Robotics Research*, 39(10-11):1201–1221, 2020.

Li Wang, Aaron D Ames, and Magnus Egerstedt. Safety barrier certificates for collisions-free multirobot systems. *IEEE Transactions on Robotics*, 33(3):661–674, 2017.

Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric Xing, and Zhiting Hu. Promptagent: Strategic planning with language models enables expert-level prompt optimization. In *The Twelfth International Conference on Learning Representations*, 2024.

Licheng Wen, Xuemeng Yang, Daocheng Fu, Xiaofeng Wang, Pinlong Cai, Xin Li, Tao Ma, Yingxuan Li, Linran Xu, Dengke Shang, et al. On the road with gpt-4v (ision): Early explorations of visual-language model on autonomous driving. *arXiv preprint arXiv:2311.05332*, 2023.

Peter R Wurman, Raffaello D'Andrea, and Mick Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI magazine*, 29(1):9–9, 2008.

Quanting Xie, So Yeon Min, Pengliang Ji, Yue Yang, Tianyi Zhang, Aarav Bajaj, Ruslan Salakhutdinov, Matthew Johnson-Roberson, and Yonatan Bisk. Embodied-rag: General non-parametric embodied memory for retrieval and generation, 2024. URL https://arxiv.org/abs/2409.18313.

Michael M Zavlanos and George J Pappas. Distributed connectivity control of mobile networks. *IEEE Transactions on Robotics*, 24(6):1416–1428, 2008.

Songyuan Zhang, Kunal Garg, and Chuchu Fan. Neural graph control barrier functions guided distributed collision-avoidance multi-agent control. In *7th Annual Conference on Robot Learning*, 2023.

Songyuan Zhang, Oswin So, Mitchell Black, and Chuchu Fan. Discrete GCBF proximal policy optimization for multi-agent safe optimal control. In *The Thirteenth International Conference on Learning Representations*, 2025a.

Songyuan Zhang, Oswin So, Kunal Garg, and Chuchu Fan. GCBF+: A neural graph control barrier function framework for distributed safe multi-agent control. *IEEE Transactions on Robotics*, 41: 1533–1552, 2025b.

Supplementary Material

Appendix A. Detailed problem formulation

We start with describing the dynamics of the individual robots (referred to as agents henceforth), and then, we list the individual as well as the team objective for the system. The agent dynamics are given by $\dot{x}_i = f(x_i) + g(x_i)u_i$, where f, g are locally Lipschitz continuous functions with $x_i \in \mathcal{X} \subset \mathbb{R}^2$ denoting agents' state space and $u_i \in \mathcal{U} \subset \mathbb{R}^{n_u}$ the control constraint set for $i \in \{1, 2, \ldots, N\}$.[1]. The state x_i consists of the position $p_i \in \mathbb{R}^2$ in the global coordinates. The state space $\mathcal{X}$ consists of stationary obstacles $\mathcal{O}_l \subset \mathbb{R}^2$ for $l \in \{1, 2, \ldots, M\}$, denoting walls, blockades and other obstacles in the path of the moving agents. Each agent has a limited sensing radius $R > 0$ and the agents can only sense other agents or obstacles if it lies inside its sensing radius. The agents use LiDAR to sense the obstacles, and the observation data for each agent consists of n_{rays} evenly-spaced LiDAR rays originating from each robot and measures the relative location of obstacles. We denote the j-th ray from agent i by $y_j^{(i)} \in \mathcal{X}$ for $j \in \{1, 2, \ldots, n_{\text{rays}}\}$ that carries the relative position information of the $j-$th LiDAR hitting point to agent i, and zero padding for the rest of the states.

The time-varying connectivity graph $\mathcal{G}(t) = (\mathcal{V}(t), \mathcal{E}(t))$ dictates the network among the agents and obstacles. Here, $\mathcal{V}(t) = \mathcal{V}^a \cup \mathcal{V}^o(t)$ denotes the set of nodes, where $\mathcal{V}^a = \{1, 2, \ldots, N\}$ denotes the set of agents, $\mathcal{V}^o(t)$ is the collection of all LiDAR hitting points at time $t \geq 0$, and $\mathcal{E}(t) \subset \mathcal{V}^a(t) \times \mathcal{V}(t)$ denotes the set of edges, where $(i, j) \in \mathcal{E}(t)$ means the flow of information from node j to agent i. We denote the time-varying adjacency matrix for agents by $A(t) \in \mathbb{R}^{N \times N}$, where $A_{ij}(t) = 1$ if $(i, j) \in \mathcal{E}(t), i, j \in \mathcal{V}^a$, and 0 otherwise. The set of *all* neighbors for agent i is denoted as $\mathcal{N}_i(t) \coloneqq \{j \mid (i, j) \in \mathcal{E}(t)\}$, while the set of *agent* neighbors of agent i is denoted as $\mathcal{N}_i^a(t) \coloneqq \{j \mid A_{ij}(t) = 1\}$. The MRS is said to be connected at time t if there is a path between each pair of agents $(i, j), \forall i, j \in \mathcal{V}^a, i \neq j$ at t. One method of checking the connectivity of the MRS is through the Laplacian matrix, defined as $L(A(t)) \coloneqq D(t) - A(t)$, where $D(t)$ is the degree matrix defined as $D_{ij}(t) = \sum_j A_{ij}(t)$ when $i = j$ and 0 otherwise. From (Mesbahi and Egerstedt, 2010, Theorem 2.8), the MRS is connected at time t if and only if the second smallest eigenvalue of the Laplacian matrix is positive, i.e., $\lambda_2(L(A(t))) > 0$.

Appendix B. Leader-follower and temporary goal assignment

Here, we present the mechanism for leader-follower assignments for a large-scale MRS. As discussed in Section 4, for large-scale MRS, i.e., $N \geq N_M = 10$, a main leader is assigned for the MRS, and sub-leaders are assigned for each of the clusters. The agents in clusters will undergo a leader-follower formation with their respective sub-leaders while these sub-leaders will follow the main leader. The complete leader-follower assignment algorithm is given in Algorithm 1.

1. In this work, we consider robots modeled using single integrator dynamics operating in 2D plane, i.e., $n_x = n_u = 2$

Algorithm 1: Leader and temporary goal assignment

Data: $\{p_i\}, K, d_{min}, N, N_M$
Result: $l_M, \{l_k\}, \{p^{\text{goal}}_{\text{temp}}\}$
`/* Find the main leader */`
$l_M = \arg\min \|p_i - p_i^{\text{goal}}\|$
`/* Find the clusters */`
if $N < N_M$ **then**
  $\{\mathcal{V}_k^a\} = \{\mathcal{V}^a\}$
else
  $\{\mathcal{V}_k^a\}$ = kmeans($\{p_i\}, K$)
end
`/* Find leaders for each cluster */`
for *k in* range*(K)* **do**
  $l_k = \arg\min_{i \in \mathcal{V}_k^a} \min_{j \in \mathcal{V}_M^a} \|p_i - p_j\|$
end
`/* Assign temporary goals */`
for $k \in [1, 2, \cdots, K]$ **do**
  `/* Initial set of leaders to be followed in cluster` $\mathcal{V}_k^a$ `*/`
  $\mathcal{V}_{\text{lead}}(k) = \{l_k\}$
  `/* Assign main leader's location as the temporary goal for cluster leader */`
  $p^{\text{goal}}_{\text{temp},l_k} = p_{l_M}$
  for $i \in$ range$|\mathcal{V}_a^k|$ **do**
    `/* Find the closest agent to the set of leaders as the new follower */`
    $i_{\text{follow}} = \arg\min_{j \in \mathcal{V}_k^a \setminus \mathcal{V}_{\text{lead}}(k)} \min_{l \in \mathcal{V}_{\text{lead}}(k)} \|p_l - p_j\|$
    `/* Find leader for it */`
    $k_{\text{lead}} = \arg\min_{j \in \mathcal{V}_{\text{lead}}(k)} \|p_{i_{\text{follow}}} - p_j\|$
    `/* Assign temporary goal to it */`
    if $\|p_{i_{\text{follow}}} - p^{\text{goal}}_{i_{\text{follow}}}\| \geq d_{min}$ **then**
      $p^{\text{goal}}_{i_{\text{follow}},\text{temp}} = p_{k_{\text{lead}}}$
    else
      $p^{\text{goal}}_{i_{\text{follow}},\text{temp}} = p^{\text{goal}}_{i_{\text{follow}}}$
    end
    `/* Update the set of leaders */`
    $\mathcal{V}_{\text{lead}}(k) = \mathcal{V}_{\text{lead}}(k) \cup \{i_{\text{follow}}\}$
  end
end

Appendix C. VLM and LLM prompts

C.1. Task and output description prompts

The task description prompts used for VLMs are given in Figure 6 while those used for LLMs are given in Figure 7.

```
We are working with a multi-robot system navigating in an obstacle environment to reach their respective goal locations. The objective is to
move the robots toward their goal locations while maintaining safety with obstacles, safety with each other, and inter-agent connectivity.
Safety is based on agents maintaining a minimum inter-agent "safety radius" while connectivity is based on connected agents remaining within a
"connectivity radius".
Your role as the helpful assistant to provide a high-level command when the system gets stuck near obstacles. The high-level command is in
terms of a leader assignment for the multi-robot system and a direction of motion for the leader.
An optimal choice of leader and moving direction minimizes the traveling distance of agents toward their goals and maintains safety and
connectivity.
The input image represents a grid world where the obstacles are given in black color.
The location of the agents are given in blue color and the goal locations are given in green color.
The task is to provide a high-level command in terms of a leader assignment for the multi-robot system and a set of waypoints for the leader.
The leader assignment is an integer value in the range (1, Number of agents) and the waypoints for the leader are (x,y) coordinates. The
number of waypoints is described by the variable "Number of waypoints" = M.
The expected output is a JSON format file with the keys "Leader" and "Waypoints". The key "Leader" can take integer values in the range (1,
Number of agents) and "Waypoints" are of the form [(x1, y1), (x2, y2), ..., (xM, yM)].
The leader should be assigned as the agent that can move freely in the environment. The leader should not be assigned to an agent that is
blocked by obstacles or other agents.
The waypoints are ordered in the sequence the leader should visit them. The first point should NOT be the current location of the leader. All
the waypoints should be at least 2r distance from all the obstacles.
The consecutive waypoints should be such that the leader moves toward its goal location.
The waypoints should be such that the leader can move toward its goal location while maintaining safety with the obstacles.
The path connecting the leader and the waypoints should NOT intersect with any of the obstacles.
The waypoints should be in the free space of the environment, away from ALL the known obstacles. The obstacles can be chosen to wrap around
the obstacles to allow the leader to move toward its goal location while evading the obstacles.
The leader assignment is based on agent being able to freely move. That means there should be no obstacle or other agents in its path
connected to its goal.
If the leader cannot move directly in the direction of its goal location, the first waypoint should be to the left or right of the leader to
avoid obstacles. The consecutive waypoints should be such that the leader moves toward its goal location while maintaining safety with the
obstacles.
```

Figure 6: Description prompts used for vision-based (i.e., VLMs) high-level planners.

```
We are working with a multi-robot system navigating in an obstacle environment to reach their respective goal locations. The objective is to
move the robots toward their goal locations while maintaining safety with obstacles, safety with each other, and inter-agent connectivity.
Safety is based on agents maintaining a minimum inter-agent "safety radius" while connectivity is based on connected agents remaining within a
"connectivity radius".
Your role as the helpful assistant to provide a high-level command when the system gets stuck near obstacles. The high-level command is in
terms of a leader assignment for the multi-robot system and a direction of motion for the leader.
An optimal choice of leader and moving direction minimizes the traveling distance of agents toward their goals and maintains safety and
connectivity.
The multi-robot environment description consists of the tuple: (Number of agents, Safety radius, Connectivity radius, Agent locations, Agent
goals, Obstacles, Number of waypoints).
The environment consists of robot Agents with information ("AgentId"=id, "current state"=(x,y), "goal location"=(xg,yg)).
The obstacles are represented as a bottom-left corner, its width, and height. The obstacles are represented as a list of tuples [(x,y,w,h)].
In addition, there are global environment variables "Number of agents" = N, "Safety radius" = r, "Connectivity radius" = R.
The task is to provide a leader assignment for the multi-robot system and a set of waypoints for the leader. The leader assignment is an
integer value in the range (1, Number of agents) and the waypoints for the leader are (x,y) coordinates. The number of waypoints is described
by the variable "Number of waypoints" = M.
The expected output is a JSON format file with the keys "Leader" and "Waypoints". The key "Leader" can take integer values in the range (1,
Number of agents) and "Waypoints" are of the form [(x1, y1), (x2, y2), ..., (xM, yM)].
The waypoints are ordered in the sequence the leader should visit them. The first point should NOT be the current location of the leader. All
the waypoints should be at least 2r distance from all the obstacles.
The waypoints should be such that the leader can move toward its goal location while maintaining safety with the obstacles.
The path connecting the leader and the waypoints should NOT intersect with any of the obstacles.
The waypoints should be in the free space of the environment, away from ALL the known obstacles. The waypoints can be chosen to wrap around
the obstacles to allow the leader to move toward its goal location while evading the obstacles.
If the leader cannot move directly in the direction of its goal location, the first waypoint should be to the left or right of the leader to
avoid obstacles.
The consecutive waypoints should be such that the leader moves toward its goal location while maintaining safety with the obstacles.
```

Figure 7: Description prompts used for text-based (i.e., LLMs) high-level planners.

C.2. Environment description

The environment description prompts used for VLMs are given in Figure 8 while those used for LLM baselines are given in Figure 9. For VLMs, an additional text prompt is appended at the end with the location of the agent(s) and goal(s) provided in the image prompt to aid the VLM with waypoint assignment.

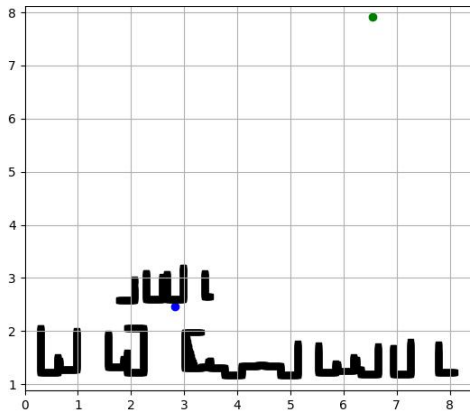


Figure 8: Environment prompt for VLM for "Maze" environment with $N = 50$ and $M = 375$.

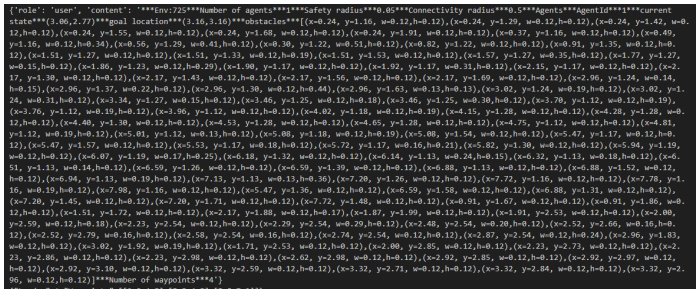
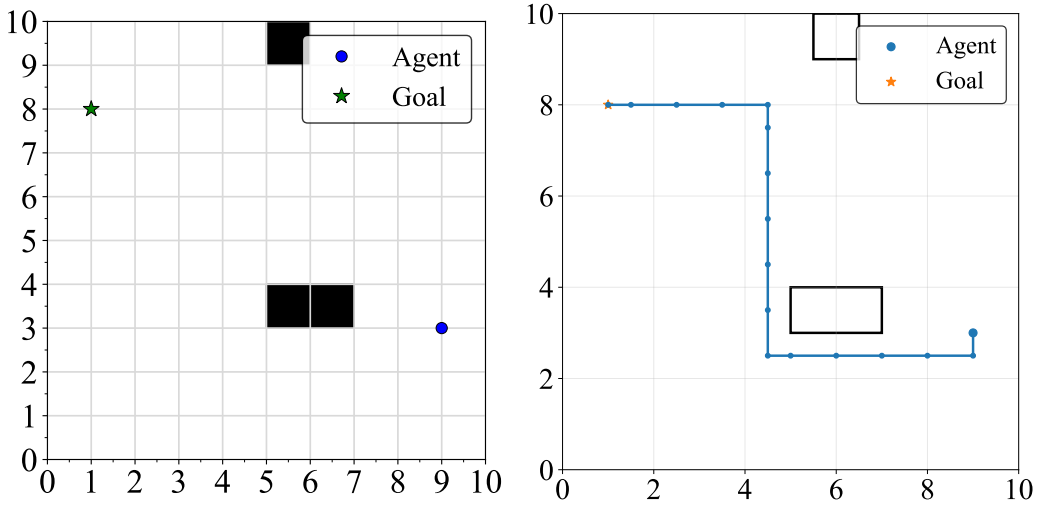

{'role': 'user', 'content': '***Env:725***Number of agents***1***Safety radius***0.05***Connectivity radius***0.5***Agents***AgentId***1***current state***(3.06,2.77)***goal location***(3.16,3.16)***obstacles***[(x-0.24, y-1.16, w=0.12,h=0.12),(x-0.24, y-1.29, w=0.12,h=0.12),(x-0.24, y-1.42, w=0.12,h=0.12),(x-0.24, y-1.55, w=0.12,h=0.12),(x-0.24, y-1.68, w=0.12,h=0.12),(x-0.24, y-1.91, w=0.12,h=0.12),(x=0.37, y-1.16, w=0.12,h=0.12),(x-0.40, y=1.16, w=0.12,h=0.34),(x=0.56, y-1.29, w=0.41,h=0.12),(x=0.30, y-1.22, w=0.51,h=0.12),(x=0.82, y-1.22, w=0.12,h=0.12),(x=0.91, y=1.35, w=0.12,h=0.12),(x=1.51, y-1.22, w=0.12,h=0.12),(x=1.51, y=1.33, w=0.12,h=0.19),(x=1.51, y=1.53, w=0.12,h=0.12),(x=1.57, y-1.22, w=0.35,h=0.12),(x=1.77, y-1.27, w=0.15,h=0.12),(x-1.86, y-1.23, w=0.12,h=0.29),(x=1.90, y-1.17, w=0.12,h=0.12),(x=1.92, y-1.17, w=0.31,h=0.12),(x-2.15, y-1.17, w=0.12,h=0.12),(x-2.17, y-1.30, w=0.12,h=0.12),(x-2.17, y-1.43, w=0.12,h=0.12),(x-2.17, y-1.56, w=0.12,h=0.12),(x-2.17, y-1.69, w=0.12,h=0.12),(x-2.96, y-1.24, w=0.14, h=0.15),(x-2.96, y-1.37, w=0.22,h=0.12),(x-2.96, y-1.30, w=0.12,h=0.44),(x-2.96, y-1.63, w=0.13,h=0.13),(x=3.02, y-1.24, w=0.19,h=0.12),(x-3.02, y-1.24, w=0.31,h=0.12),(x=3.34, y-1.27, w=0.15,h=0.12),(x=3.46, y-1.25, w=0.12,h=0.18),(x=3.46, y-1.25, w=0.30,h=0.12),(x=3.70, y-1.12, w=0.12,h=0.19),(x=3.70, y-1.12, w=0.19,h=0.12),(x=3.96, y-1.12, w=0.12,h=0.12),(x=4.02, y-1.18, w=0.12,h=0.19),(x=4.15, y-1.28, w=0.12,h=0.12),(x=4.28, y-1.28, w=0.12,h=0.12),(x=4.40, y-1.30, w=0.12,h=0.12),(x=4.53, y-1.28, w=0.12,h=0.12),(x=4.65, y-1.28, w=0.12,h=0.12),(x=4.75, y-1.12, w=0.12,h=0.12),(x=4.81, y=1.12, w=0.19,h=0.12),(x=5.01, y-1.12, w=0.13,h=0.12),(x=5.08, y-1.18, w=0.12,h=0.19),(x=5.08, y-1.54, w=0.12,h=0.12),(x=5.47, y-1.17, w=0.12,h=0.12),(x=5.47, y-1.57, w=0.12,h=0.12),(x=5.53, y-1.17, w=0.18,h=0.12),(x=5.72, y=1.17, w=0.16,h=0.21),(x=5.82, y-1.30, w=0.12,h=0.12),(x=5.94, y-1.19, w=0.12,h=0.12),(x=6.07, y-1.19, w=0.17,h=0.25),(x=6.18, y-1.32, w=0.12,h=0.12),(x=6.14, y-1.13, w=0.24,h=0.15),(x=6.32, y-1.13, w=0.18,h=0.12),(x=6.51, y-1.13, w=0.14,h=0.12),(x=6.59, y-1.26, w=0.12,h=0.12),(x=6.59, y-1.39, w=0.12,h=0.12),(x=6.88, y-1.13, w=0.12,h=0.12),(x=6.88, y-1.52, w=0.12, h=0.12),(x=6.94, y-1.13, w=0.19,h=0.12),(x-7.13, y-1.13, w=0.13,h=0.36),(x=7.20, y-1.26, w=0.12,h=0.12),(x-7.72, y-1.16, w=0.12,h=0.12),(x-7.78, y-1.16, w=0.19,h=0.12),(x-7.98, y-1.16, w=0.12,h=0.12),(x=5.47, y-1.36, w=0.12,h=0.12),(x=6.59, y-1.58, w=0.12,h=0.12),(x=6.88, y-1.31, w=0.12,h=0.12), (x-7.20, y-1.45, w=0.12,h=0.12),(x-7.20, y-1.71, w=0.12,h=0.12),(x-7.72, y-1.48, w=0.12,h=0.12),(x=0.91, y-1.67, w=0.12,h=0.12),(x=0.91, y-1.86, w=0.12,h=0.12),(x-1.51, y-1.72, w=0.12,h=0.12),(x-2.17, y-1.88, w=0.12,h=0.17),(x-1.87, y-1.99, w=0.12,h=0.12),(x=1.91, y-2.53, w=0.12,h=0.12),(x-2.00, y-2.59, w=0.12,h=0.18),(x-2.25, y-2.54, w=0.12,h=0.12),(x-2.29, y-2.54, w=0.29,h=0.12),(x=2.48, y-2.54, w=0.20,h=0.12),(x-2.52, y-2.66, w=0.16,h=0.12),(x=2.52, y-2.79, w=0.16,h=0.12),(x-2.58, y-2.54, w=0.16,h=0.12),(x=2.74, y-2.54, w=0.12,h=0.12),(x=2.87, y-2.54, w=0.12,h=0.24),(x-2.96, y-1.83, w=0.12,h=0.12),(x=3.02, y-1.92, w=0.19,h=0.12),(x=1.71, y-2.53, w=0.12,h=0.12),(x-2.00, y-2.85, w=0.12,h=0.12),(x-2.23, y-2.73, w=0.12,h=0.12),(x-2.23, y-2.86, w=0.12,h=0.12),(x-2.23, y-2.98, w=0.12,h=0.12),(x-2.62, y-2.98, w=0.12,h=0.12),(x-2.92, y-2.85, w=0.12,h=0.12),(x-2.92, y-2.97, w=0.12, h=0.12),(x-2.92, y-3.10, w=0.12,h=0.12),(x-3.32, y-2.59, w=0.12,h=0.12),(x-3.32, y-2.71, w=0.12,h=0.12),(x-3.32, y-2.84, w=0.12,h=0.12),(x-3.32, y-2.96, w=0.12,h=0.12)]***Number of waypoints***4'}

Figure 9: Environment prompt for LLM for "Maze" environment with $N = 50$ and $M = 375$.



(*a*) (*b*)

Figure 10: (a) Grid environment with randomly generated obstacles and agent-goal locations for assessing foundation models' spatial-reasoning capability and (b) The obstacles, agent and goal locations as well as the waypoints generated by `GPT-5` with low reasoning effort.

C.3. Output of foundation models

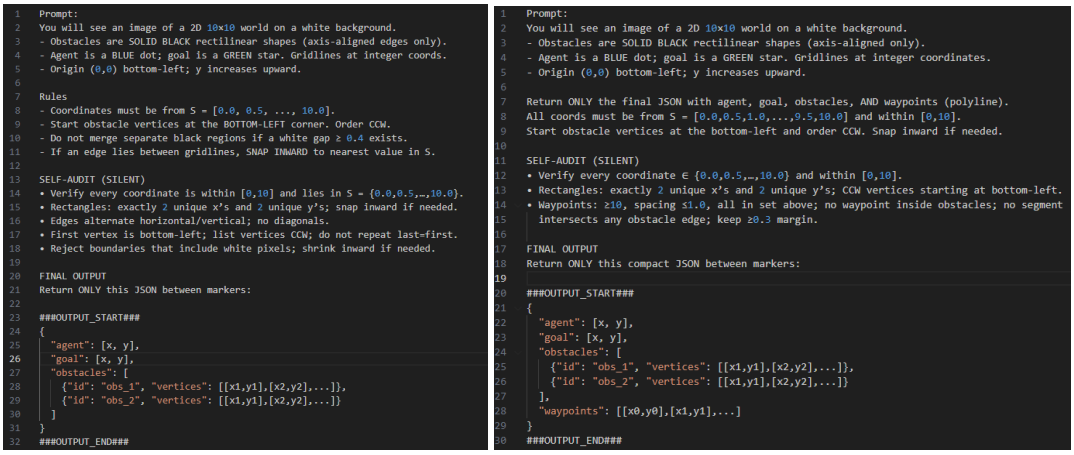
The output of the LLM or VLM is a JSON object. As an example, for the input in Figure 8, the output from `Claude3-Opus` VLM is: {"Leader" : 1, "Waypoint" : $[[2.8, 4.5], [5.5, 4.5], [5.5, 7.9]]$}.

C.4. Spatial reasoning prompts

An example environment used for assessing models' spatial reasoning is given in Figure 10. The prompts used for these evaluations are provided in Figure 11.

Appendix D. Low-level control policy

Any low-level controller that can satisfy the requirement from Problem 1 can be used as the low-level control policy, such as one from a distributed CBF-QP method (Wang et al., 2017) or a learned control policy. Since the low-level control policy is supposed to be distributed with low computational complexity, we choose to use the recently proposed learning-based GCBF+ controller from (Zhang et al., 2025b). Given sensing radius R and safety distance r, define $N_s - 1 \in \mathbb{N}$ as the maximum number of neighbors that each agent can have while all the agents in the neighborhood remain safe. Define $\tilde{\mathcal{N}}_i$ as the set of N_s closest neighboring nodes to agent i which also includes



```
Prompt:
You will see an image of a 2D 10×10 world on a white background.
- Obstacles are SOLID BLACK rectilinear shapes (axis-aligned edges only).
- Agent is a BLUE dot; goal is a GREEN star. Gridlines at integer coords.
- Origin (0,0) bottom-left; y increases upward.

Rules
- Coordinates must be from S = [0.0, 0.5, ..., 10.0].
- Start obstacle vertices at the BOTTOM-LEFT corner. Order CCW.
- Do not merge separate black regions if a white gap ≥ 0.4 exists.
- If an edge lies between gridlines, SNAP INWARD to nearest value in S.

SELF-AUDIT (SILENT)
• Verify every coordinate is within [0,10] and lies in S = {0.0,0.5,…,10.0}.
• Rectangles: exactly 2 unique x's and 2 unique y's; snap inward if needed.
• Edges alternate horizontal/vertical; no diagonals.
• First vertex is bottom-left; list vertices CCW; do not repeat last-first.
• Reject boundaries that include white pixels; shrink inward if needed.

FINAL OUTPUT
Return ONLY this JSON between markers:

###OUTPUT_START###
{
  "agent": [x, y],
  "goal": [x, y],
  "obstacles": [
    {"id": "obs_1", "vertices": [[x1,y1],[x2,y2],...]},
    {"id": "obs_2", "vertices": [[x1,y1],[x2,y2],...]}
  ]
}
###OUTPUT_END###
```

(*a*)

```
Prompt:
You will see an image of a 2D 10×10 world on a white background.
- Obstacles are SOLID BLACK rectilinear shapes (axis-aligned only).
- Agent is a BLUE dot; goal is a GREEN star. Gridlines at integer coordinates.
- Origin (0,0) bottom-left; y increases upward.

Return ONLY the final JSON with agent, goal, obstacles, AND waypoints (polyline).
All coords must be from S = [0.0,0.5,1.0,...,9.5,10.0] and within [0,10].
Start obstacle vertices at the bottom-left and order CCW. Snap inward if needed.

SELF-AUDIT (SILENT)
• Verify every coordinate ∈ {0.0,0.5,…,10.0} and within [0,10].
• Rectangles: exactly 2 unique x's and 2 unique y's; CCW vertices starting at bottom-left.
• Waypoints: ≥10, spacing ≤1.0, all in set above; no waypoint inside obstacles; no segment
  intersects any obstacle edge; keep ≥0.3 margin.

FINAL OUTPUT
Return ONLY this compact JSON between markers:

###OUTPUT_START###
{
  "agent": [x, y],
  "goal": [x, y],
  "obstacles": [
    {"id": "obs_1", "vertices": [[x1,y1],[x2,y2],...]},
    {"id": "obs_2", "vertices": [[x1,y1],[x2,y2],...]}
  ],
  "waypoints": [[x0,y0],[x1,y1],...]
}
###OUTPUT_END###
```

(*b*)

Figure 11: Prompts used for (a) scene-understanding and (b) waypoint generation tasks for spatial reasoning assessment.

agent i and $\bar{x}_{\tilde{\mathcal{N}}_i}$ as the concatenated vector of x_i and the neighbor node states with fixed size N_s that is padded with a constant vector if $|\tilde{\mathcal{N}}_i| < N_s$. Considering only collision avoidance constraints, the safe set $\mathcal{S}_N \in \mathcal{X}^N$ can be defined as:

$$\mathcal{S}_N := \Big\{ \bar{x} \in \mathcal{X}^N \;\Big|\; \Big(\Big\| y_j^{(i)} \Big\| > r,\ \forall i \in \mathcal{V}^a, \forall j \in n_{\text{rays}} \Big) \bigwedge \Big(\min_{i,j \in \mathcal{V}^a, i \neq j} \|p_i - p_j\| > 2r \Big) \Big\},$$

where $\bar{x}$ denotes the joint state vector for the MRS. The unsafe, or avoid set can be defined accordingly as $\mathcal{A}_N = \mathcal{X}^N \setminus \mathcal{S}_N$. Considering the smoothness of GCBF, we impose the condition that for a given agent $i \in V_a$, a node j where $\|p_i - p_j\| \geq R$ does not affect the GCBF h. Specifically, for any neighborhood set $\mathcal{N}_i$, let $\mathcal{N}_i^{<R}$ denote the set of neighbors in $\mathcal{N}_i$ that are strictly inside the sensing radius R as

$$\mathcal{N}_i^{<R} := \{ j : \|p_i - p_j\| < R,\ j \in \mathcal{N}_i \}. \tag{1}$$

Using these notations, GCBF is defined in (Zhang et al., 2025b) as:

Definition 1 (GCBF) *A continuously differentiable function $h : \mathcal{X}^M \to \mathbb{R}$ is termed as a Graph CBF (GCBF) if there exists an extended class-$\mathcal{K}$ function α and a control policy $\pi_i : \mathcal{X}^M \to \mathcal{U}$ for each agent $i \in V_a$ of the MAS such that, for all $\bar{x} \in \mathcal{X}^N$ with $N \geq M$,*

$$\dot{h}(\bar{x}_{\mathcal{N}_i}) + \alpha(h(\bar{x}_{\mathcal{N}_i})) \geq 0, \quad \forall i \in V_a \tag{2}$$

where

$$\dot{h}(\bar{x}_{\mathcal{N}_i}) = \sum_{j \in \mathcal{N}_i} \frac{\partial h(\bar{x}_{\mathcal{N}_i})}{\partial x_j} \left(f(x_j) + g(x_j) u_j \right), \tag{3}$$

for $u_j = \pi_j(\bar{x}_{\mathcal{N}_j})$, and the following two conditions hold:

- *The gradient of h with respect to nodes R away is 0, i.e.,*

$$\frac{\partial h}{\partial x_j}(\bar{x}_{\mathcal{N}_i}) = 0, \quad \forall j \in \mathcal{N}_i \setminus \mathcal{N}_i^{<R}. \tag{4}$$

- *The value of h does not change when restricting to neighbors that are in $\mathcal{N}_i^{<R}$, i.e.,*

$$h(\bar{x}_{\mathcal{N}_i}) = h(\bar{x}_{\mathcal{N}_i^{<R}}). \tag{5}$$

It is proved in (Zhang et al., 2025b, Theorem 1) GCBF certifies the forward invariance of its 0-superlevel set under control inputs from:

$$\mathcal{U}_{\text{safe}}^N \coloneqq \left\{ \bar{u} \in \mathcal{U}^N \;\middle|\; \dot{h}(\bar{x}_{\mathcal{N}_i}) + \alpha(h(\bar{x}_{\mathcal{N}_i})) \geq 0,\, \forall i \in V_a \right\}.$$

We start with this formulation and modify it to additionally account for the connectivity requirement, as explained below.

Following (Zhang et al., 2025b), we use graph neural networks (GNN) to parameterize GCBF. For agent i, the input features of the GNN contain the node features v_i and v_j for $j \in \mathcal{N}_i$, and edge features e_{ij} for $j \in \mathcal{N}_i$. The node features $v_i \in \mathbb{R}^{\rho_v}$ encode information specific to each node. In this work, we take $\rho_v = 3$ and use the node features v_i to one-hot encode the type of the node as either an agent node, goal node or LiDAR ray hitting point node. The edge features $e_{ij} \in \mathbb{R}^{\rho_e}$, where $\rho_e > 0$ is the edge dimension, are defined as the information shared from node j to agent i, which depends on the states of the nodes i and j. Since the safety objective depends on the relative positions, one component of the edge features is the relative position $p_{ij} = p_j - p_i$. The rest of the edge features can be chosen depending on the underlying system dynamics, e.g., relative velocities for double integrator dynamics. However, apart from the safety constraints considered in (Zhang et al., 2025b), we also consider the connectivity constraints. Therefore, the design of the node features and edge features needs to be modified for adding the connectivity information. To this end, given the desired connectivity of the MRS in terms of the *desired* adjacency matrix A^d where the desired adjacency matrix is designed such that the MRS is connected, we add the connectivity information in the edge features of GCBF. In particular, we append the edge features with $[0, 1]^\top$ in e_{ij} if the agents (i, j) are required to be connected, i.e., $A_{ij}^d = 1$, and $[1, 0]^\top$ if they are not required to be connected, i.e., $A_{ij}^d = 0$. Furthermore, we add the connectivity constraint in the GCBF by redefining the safe and the unsafe sets corresponding to the required connectivity, such that the safe set is defined as

$$\begin{aligned} \mathcal{S}_N^c \coloneqq \Big\{ \bar{x} \in \mathcal{X}^N \;\Big|\; & \Big(\left\| y_j^{(i)} \right\| > r,\; \forall i \in \mathcal{V}_a, \forall j \in n_{\text{rays}} \Big) \\ & \bigwedge \Big(\min_{i,j \in \mathcal{V}_a, i \neq j} \|p_i - p_j\| > 2r \Big) \bigwedge \\ & \Big(\max_{i,j \in \mathcal{V}_a, A_{ij}^d = 1} \|p_i - p_j\| < R \Big) \Big\}. \end{aligned} \tag{6}$$

Consequently, the unsafe, or avoid set with the connectivity constraint is defined as $\mathcal{A}_N^c = \mathcal{X}^N \setminus \mathcal{S}_N^c$. Since the GCBF h certifies the forward-invariance of its 0-superlevel set, the safety and connectivity constraints are satisfied.

The training framework is the same as (Zhang et al., 2025b). In the original GCBF+ training framework, it is essential that the initial and goal locations are safe. In the current work, we also need to make sure that the initial conditions and the goal locations sampled for training the GCBF satisfy the MRS connectivity condition, in addition to the safety condition in the original GCBF+ framework. To this end, we sample the initial and goal locations such that their corresponding

graph topology are connected, and define the desired adjacency matrix $A^d = A(0)$. The same loss function from (Zhang et al., 2025b) is used to train the distributed control policy, with the safe set definition modified as per (6).

Appendix E. Ablation studies

E.1. Effect of partial environment information

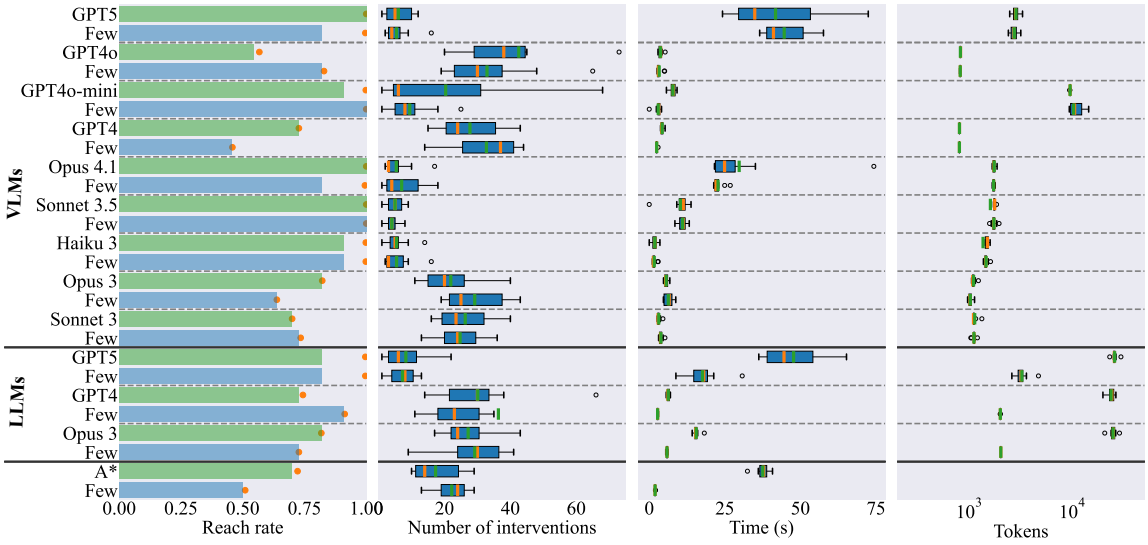


Figure 12: Performance of various high-level planners for "Maze" environments with $N = 50$ agents with all known environment information and partial information (the case with all partial environment information is indicated with the suffix "-Few", e.g., "GPT5-VLM-Few"). From left to right: 1) The bar shows the ratio of the trajectories where **all** the agents reach their goals over the total number of trajectories, and the orange dot shows the ratio of agents that reach their goals over all agents; 2) Box plot of the number of times the high-level planner intervened; 3) Box plot of the time spent for each high-level planner intervention; and 4) Box plot for the input + output token per intervention. In the box plots, the median values are in orange and the mean values are in green.

We evaluate the effect of providing partial environment information to the foundation models to study the trade-off between cost (in terms of the tokens) and performance. Figure 12 compares the performance of querying a given foundation model with all collected observations of obstacles and querying it with only the most recent observations of obstacles (50 for LLMs and 100 for VLMs). The rationale behind choosing the last few observations is twofold: 1) it reduces the number of tokens in the prompt provided to the foundation model, thereby reducing the time and cost per query, and 2) in the considered environments, the initially observed obstacles do not play much role in determining the waypoints for the leader.

GPT4o-VLM, GPT-5-LLM, and GPT4-LLM perform better with partial information We can observe that, with the exception of `GPT-4o` VLM, `GPT-5` LLM, and `GPT4` LLM, the performance drops when partial information is used. It is not entirely clear why `GPT-4o` VLM, `GPT-5` LLM, and `GPT4` LLM perform better with partial information. Our understanding is that the per-

formance of these models is poor with the complete information due to their inability to utilize the provided information to make a good decision for this problem.

Smaller models perform better with partial information From the results on `Claude3-Sonnet-VLM`, we can observe that the performance of a *smaller* model (in terms of the model parameters) improves when partial information is used. On the other hand, for *larger* models like `GPT4-VLM` and `Claude3-LLM`, the performance drops when only the partial information is used. It provides evidence in support of the intuition that smaller models do not perform well when more information is provided.

Quality of high-level commands deteriorates with less information As discussed in the main paper, the number of high-level planner interventions is generally inversely proportional to the quality of the plan they suggest. As evident from the figure, the number of interventions with partial information is, on average, more than the case when all the known information is provided to the foundation models. We infer that the quality of the plan provided drops as the amount of data provided to the foundation models decreases.

Inference cost and time improves significantly As expected, the average query time to foundation models (particularly LLMs) reduces significantly when using partial information. This provides a good trade-off metric for the user to determine how much data they should provide to the LLMs based on the desired level of performance and how much delay can be tolerated for a particular problem at hand. As stated in the beginning of the section, the motivation of conducting this ablation study is to see the cost-efficiency of providing partial information to the foundation models. While the average number of tokens used for VLMs does not change, there is a significant (at least an order of magnitude) reduction in the case of LLMs. Since the number of tokens is directly proportional to the cost associated with querying the proprietary foundation models, this trade-off study illustrates that an optimal amount of information can be provided to the foundation models to obtain desirable performance within a given cost and time budget.