Foundation Models to the Rescue: Deadlock Resolution in Connected Multi-Robot Systems

Kunal Garg Songyuan Zhang Jacob Arkin Chuchu Fan

***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. Utilizing the generalizability and low data requirements of foundation models, this paper explores the possibility of using text-based models, i.e., large language models (LLMs), and text-and-image-based models, i.e., 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 to the MRS along with a set of waypoints for the MRS leader. Then, a low-level distributed control policy based on graph neural networks 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 LLMs and VLMs. We compare their performance with a graph-based planner in terms of effectiveness in helping the MRS reach their target locations and computational time. Our results illustrate that, compared to grid-based planners, the foundation models perform better in terms of the goal-reaching rate and computational time for complex environments, which helps us conclude that foundation models can assist MRS operating in complex obstacle-cluttered environments to resolve deadlocks efficiently.**

I. Introduction

Multi-agent robotic systems (MRS) are widely used in various applications today, such as warehouse operations [1], [2], self-driving cars [3], and coordinated drone navigation in a dense forest for search-and-rescue missions [4], among others. In various MRS applications for navigating in unknown environments, such as coverage [5] and formation control [6], robot agents must remain within the communication region of each other, or, in other words, 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 often 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.

In recent years, learning-based methods have shown promising results in computing a low-level control policy for complex robotic systems [7]–[9]. The recent work [10] proposed a new notion termed graph control barrier function (GCBF) for encoding safety in arbitrarily large MRS, and a framework, GCBF+, for learning the GCBF and a safe distributed control policy. However, that work focuses on safety, i.e., collision avoidance, and does not incorporate connectivity maintenance or goal-reaching requirements. In this work, we modify the GCBF-based distributed low-level control policy so that the new policy can maintain both safety and connectivity for arbitrarily large MRS. However, the resulting low-level control policy is still prone to failure modes such as deadlocks in obstacle environments. Many works have been proposed to solve the problem of detecting and moving out of deadlocks under safety constraints [11]–[14]. However, these works do not consider connectivity constraints. To this end, we extend the GCBF+ framework from [10] to incorporate robot connectivity in addition to safety. The GCBF+ framework uses a graph neural network (GNN) to learn a distributed control policy for collision avoidance, where edge features encode the essential information for maintaining safety (i.e., relative state information). In this work, we add edge features with the required connectivity information and modify the definition of the safe set to include connectivity requirements.

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 takes the environment information available to the MRS so far and proposes a high-level command in terms of a leader assignment for the MRS and a set of waypoints for it to navigate safely in obstacle environments. Motivated by the generalizability and low data requirements of pre-trained foundation models [15] as well as the recent success of foundation models in assisting a control framework for complex robotics problems [16], [17], we explore the possibility of using text-based large language models (LLMs) and text-and-image-based vision language models (VLMs) as high-level planners to resolve deadlocks in MRS. Instead of *proactively* using these models, whether directly for planning, translation, or reward design, we are instead interested in using foundation models *reactively* to resolve a class of failure modes in low-level controllers for MRS, namely deadlocks. This also helps to ensure that the VLM or LLM does not lead to a violation of safety as it is taken care of by the provably safe low-level

The authors are with the Department of Aeronautics and Astronautics at MIT, {kgarg, jarkin, szhang21, chuchu}@mit.edu. Project website: https://mit-realm.github.io/LLM-gcbfplus-website/

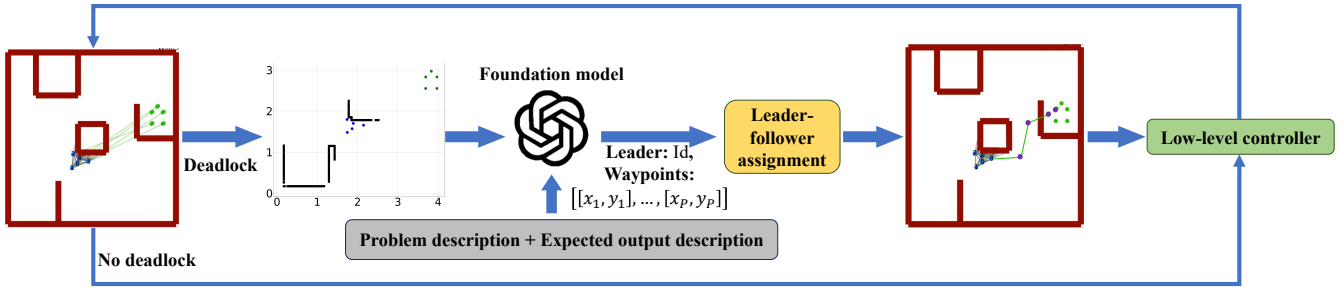

Fig. 1: Overview of the hierarchical control framework using a foundation model as a high-level planner. In the image on the left, the robots are shown in blue, their goals in green, and obstacles in red. The planner assigns a leader and waypoints (shown in purple) for the leader of the MRS, resulting in a leader-follower formation. A GNN-based low-level controller provides a distributed control policy for safety and connectivity while the leader helps evade the deadlock.

control policy. When a deadlock is detected, a VLM or an LLM is prompted to decide on a leader for the MRS and a set of waypoints for navigation to resolve it, conditioned on a top-view image-based description (for VLM) or text-based description (for LLM) of the so far observed environment. The MRS then reconfigures into a leader-follower formation to evade the deadlock situation. 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 to resolve deadlocks in an obstacle-cluttered environment for MRS (see Figure 1). For large-scale MRS, we propose a hierarchy of leaders for efficient deadlock evasion. 2) We compare the performance of various text-based LLMs and text-and-image-based VLMs in efficient deadlock resolution for MRS. We perform extensive experiments on a variety of MRS environments varying the number of agents from 5 to 50 with various foundation models. Our results demonstrate that both VLM and LLM-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 grid-based planners such as A*. Based on our observations, 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.

II. Related Work

Pre-trained LLMs have been shown to exhibit generalization to novel tasks without requiring updates to the underlying model parameters [15], [18]. While originally intended for language tasks, pre-trained LLMs have since been adopted for use in robotics for planning and control design. For planning, a class of approaches has investigated the direct use of LLMs as planners by prompting them to generate sequences of actions by which a robot could accomplish a given task [19]–[23]; some methods rank the possible next action according to the probability of the LLM generating that action combined with the likelihood that the action will succeed [20], even iterating between LLM action proposals and estimates of individual action success probabilities to handle long-horizon tasks [21]. Instead, another class of methods relies on LLMs to translate from a natural language task description to a formal representation that can be provided as input to existing planners [24]–[28]. More recently, VLMs have become popular in robotic applications due to their strong semantic reasoning capabilities [16]. VLMs have shown promising results in reasoning about future actions of robotic systems with partial environment information [29]–[34]. In particular, VLMs have been successfully employed in robot navigation tasks [35]–[37], even when long-horizon reasoning is required [17].

III. Problem Formulation

In this work, we design a distributed control framework for large-scale multi-robot systems (MRS) with multiple objectives. 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 radius $R > r$ such that the agents can only sense other agents or obstacles if they lie within their sensing 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 (see Appendix for more details). 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 $\mathcal{E}(t) \subset \mathcal{V}^a \times \mathcal{V}$ 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 so that robots can build team knowledge and share information, where given a communication radius $R > 0$, two agents i, j are connected if $\|p_i - p_j\| \leq R$ (see Appendix for more details). The formal problem statement studied in this paper is described in the following.

Problem 1. *Consider the multi-agent system with connected initial topology $\mathcal{G}(0)$, safety parameters $r > 0$, a sensing*

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**: The 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 $\|y_j^{(i)}(t)\| > r,\ \forall j = 1, \ldots, n_{\text{rays}}$ for all $t \geq 0$;*
- ***Connectivity**: The graph $\mathcal{G}(t)$ remains connected at all times;*
- ***Performance**: The agents reach their respective goals, i.e., $\lim_{t \to T_F} \|p_i(t) - p_i^{\text{goal}}\| = 0$.*

IV. 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. In this work, 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., $\frac{\sum_i \|p_i - p_i^{\text{goal}}\|}{N} > \delta_d$. These criteria imply that the MRS is stuck in a deadlock due to the obstacles as the agents are not near their goals. 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. We provide the details of the VLM/LLM-based high-level planner in Section IV-A. Once a leader and its waypoints are obtained, the MRS reconfigures into a leader-follower formation. The leader-follower assignment is done by sequentially assigning the closest unassigned follower to its closest assigned agent as its leader. The MRS remains in the leader-follower mode for a fixed time $T_{LF} > 0$, which is a user-defined hyper-parameter. The complete leader-follower assignment algorithm is described in Appendix B.

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 complete 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 [38]. 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 $\mathcal{V}_M^a$ (see Appendix B for more details). 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.

A. Foundation model based high-level planner

To initiate the leader-follower assignment process as explained in the previous section, it is necessary to designate a leader for the MRS. 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 leader and 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 D.

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. For VLMs, at any given time instant t_q 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 (see Section IV-B for more details). 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. For LLMs, the environment state is represented in text as the tuple: (Number of agents, Safety radius, Connectivity radius, Agent locations, Agent goals, Locations of observed obstacles), where the agent locations are $\{p_i(t_q)\}$, the goals $\{p_i^{\text{goal}}\}$, and the observed obstacles $\{o_i(t)\}_{t \leq t_q}$.

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

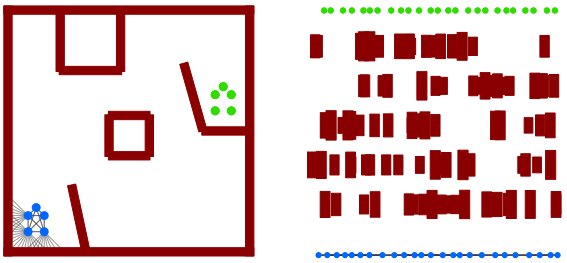

Fig. 2: "Room" environment with $N = 5$ (left) and "Maze" environment with $N = 25$ (right). The agents are shown in blue, the goals in green, and the obstacles in red color.

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 fields "Leader" and "Waypoints", with an expected output of the form $\{\text{"Leader"} : \text{Id}, \text{"Waypoints"} : [[x_1, y_1], \ldots, [x_P, y_P]]\}$.

B. 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 [10]. Since the main focus of this paper is on the high-level planner, the details of the low-level controller are provided in Appendix . We also show that the proposed hierarchical architecture does not get stuck in any deadlocks in the Appendix H.

V. Evaluations

To assess the effectiveness of the high-level planner, we perform extensive experiments on a variety of MRS environments with the objective of measuring 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).

A. 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 2).

"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 obstacle environment to reach their respective destinations. The obstacles in the room environment are designed in such a manner 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. For testing, we use 20 Maze environments with $N = 25$ and $M = 100$, and 11 Maze environments with $N = 50$ and $M = 375$.

Foundation models and baselines We use `Claude3-Sonnet` (or simply, `Claude3S`)[1], `Claude3-Opus` (or simply, `Claude3O`), `GPT-4o`[2] and `GPT4-Turbo` (or simply, `GPT4`) models as VLMs for the high-level planner, and `GPT4`, `GPT3.5`, `Claude2`[3], and `Claude3O` as the LLMs for the high-level planner. 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. We use an A*-based high-level planner and a "Random" high-level planner where the leader and the waypoints are drawn randomly, as baselines for comparison.

Roll-out for evaluation To evaluate the effectiveness of the high-level planner, we roll out MRS trajectories for a fixed number of steps $T = 4000$ (i.e., $T_F = 1200s$) for the Maze environment with $N = 25$ and $T = 5000$ (i.e., $T_F = 1500s$) for the Maze environment with $N = 50$, and $T = 3000$ (i.e., $T_F = 900s$) for Room environments. As discussed in Section IV, the high-level planner intervenes when the MRS is stuck in a deadlock, defined according to the minimum average speed criteria. In this work, 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 is useful as it helps prevent Zeno behaviors, and sets an upper bound on the frequency at which the foundation model is queried, making the proposed framework applicable to real-time robotic applications.

B. Results

Figure 3 plots the various performance criteria for each of the test environments for the considered foundation models and baseline method. The broad observations and conclusions are summarized below. For the large-scale Maze environments with $N = 50$ and $M = 375$, the prompts for LLM-based planners `GPT3.5` and `Claude2` exceed the maximum allowed context window. Videos from the experiments and the code for running the experiments are included in the Supplementary material.

Foundation models are more effective than the grid-based and random method: An important feature we

[1]https://www.anthropic.com/news/claude-3-family
[2]https://platform.openai.com/docs/models
[3]https://www.anthropic.com/news/claude-2

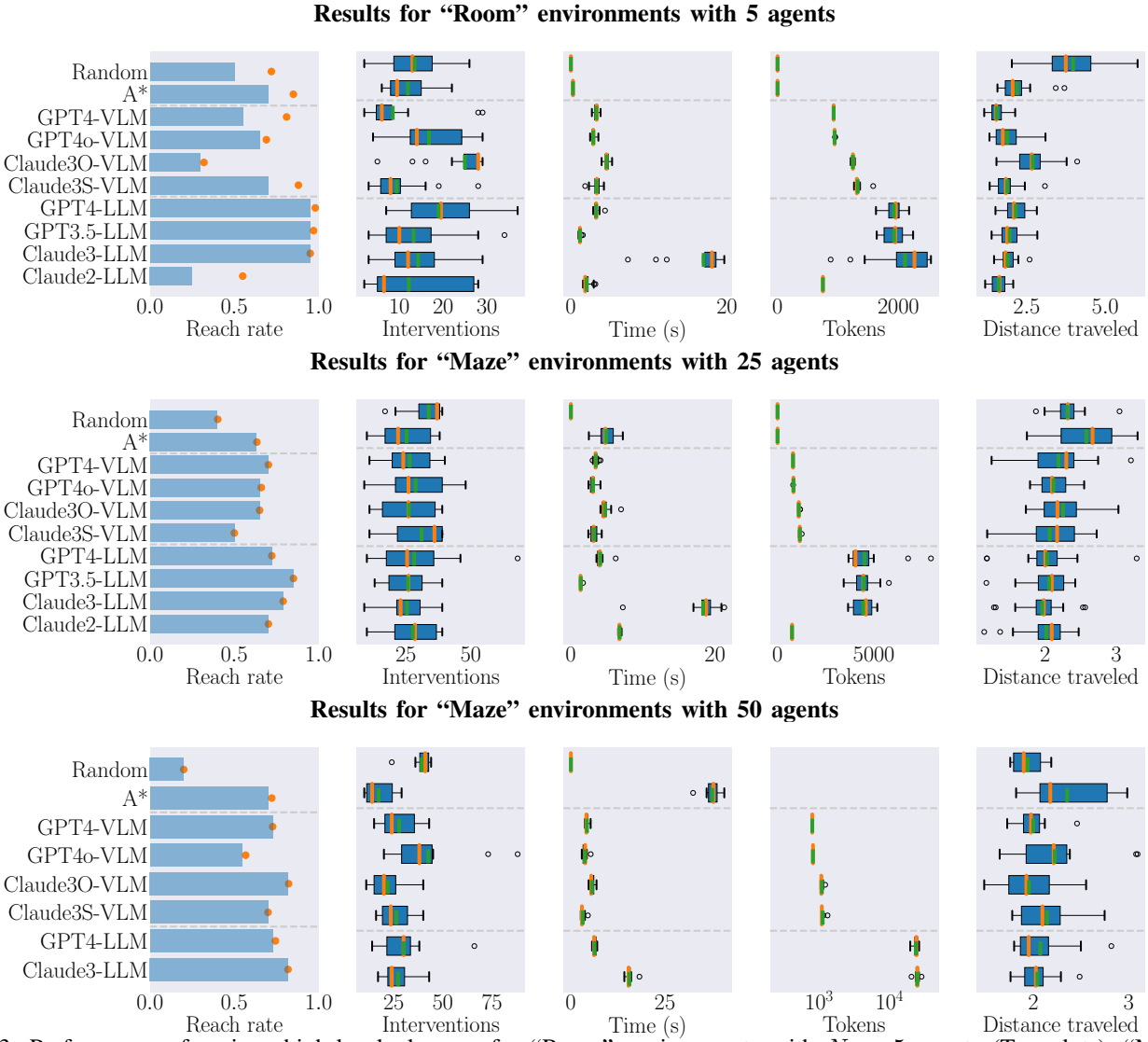

Fig. 3: Performance of various high-level planners for "Room" environments with $N = 5$ agents (Top plots), "Maze" environments with $N = 25$ agents (Middle plots), and "Maze" environments with $N = 50$ agents (Bottom plots). From left to right: 1) The bar shows the ratio of the trajectories where **all** 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.

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 higher average reach rate. For the small-scale Room environments, the mean reach rate for A* across 20 test environments is lower than LLMs such as `GPT4`, `GPT3.5` and `Claude3`, as well as `Claude3-Opus` LLM. For the large-scale Maze environments, the mean reach rate for A* is lower than most of the foundation models. Note that the computation time of A* increases as the complexity of the environment in terms of its size and number of obstacles increases, while that of foundation models remains the same. Note also that a randomly chosen leader along with randomly chosen waypoints results in less than 0.5 reach rage in all cases, with its performance dropping as the number of agents becomes larger. This illustrates that the foundation models provide much better inference than a randomly generated one, resulting in a higher completion rate.

LLMs outperform VLMs: We observe that the text-based foundation models outperform text-and-image-based foundation models when it comes to performance in terms of reach rate. Moreover, from the number of high-level interventions, we infer that LLMs achieve similar or better performance than VLMs while requiring a lesser number

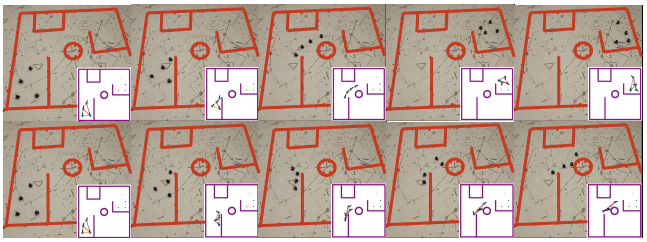

Fig. 4: Snapshots from robot experiments with the proposed hierarchical control (top figures) and only the low-level controller (bottom figures). The hierarchical architecture leads to a successful run while just the low-level controller results in a deadlock.

of interventions. That signifies better leader and waypoint assignment by LLMs than VLMs, resulting in the completion of tasks with fewer interventions.

VLMs are cheaper and faster than LLMs for large-scale MRS: It is evident that the average number of total tokens required for VLMs remains similar (≈ 1000) across the various environments. On the other hand, the tokens for 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 (e.g., `GPT4`) used as VLM and LLM. Furthermore, the variance in the average tokens used by VLMs across experiments is close to zero, while LLMs have a significant amount of variance. 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 as well as on the utility of multi-leader assignment over just one leader in large-scale MRS and report the results in Appendix G.

a) Hardware demonstration: We also validate our approach in hardware experiments on the first scenario, i.e., the room environment, using four Turtlebot3 ground robots. We use an off-board ground computer for sending control commands to the robots via ROS, while the state information of the robots is obtained using a Vicon motion capture system. Figure 4 shows the snapshots from the experiments under the proposed framework with `GPT3.5` LLM as the high-level planner as well as just the low-level controller without any high-level planner intervention. The experiments confirm the fact that only the low-level controller leads to deadlocks, while the proposed method is capable of completing the tasks.

VI. Conclusions and Discussion

In this work, we tested the hypothesis that text-based and text-and-image-based foundation models can be used as high-level planners for deadlock resolution in MRS with safety and connectivity constraints. We performed 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 provided interesting observations and insights on the relative performance of LLMs and VLMs for small- and large-scale MRS problems, as well as their time and cost efficiency.

The high performance of the zero-shot foundation model across various MRS environments is good evidence that they are promising high-level planners for deadlock resolution. We are encouraged that in many cases LLM-based planners need to intervene less frequently, implying better intervention quality. However, the prompt size for LLMs increases significantly as the complexity of the environment increases. The prompt design used in this work is the result of manual trial and error while following generally accepted practices for prompt engineering. We note that the text-based environment description has much more flexibility as compared to the image-based description. One implication of this is the freedom to modify and optimize the prompt to obtain better performance. Another implication is the increased variability and, as a result, the high sensitivity of the performance of the model to the prompt design. Recently, there has been an increasing interest in automatic prompt optimization for black-box LLMs to find the best prompt design for a given task [39]–[41] 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 LLMs as a high-level planner.

VII. Limitations

One limitation of using a proprietary foundation model as a high-level planner is the time it takes to query the model and receive a response. For real-time intervention, fast performance can often be a prerequisite. Future work involves investigating possible methods to improve the runtime, such as using smaller, open-source models, running locally, fine-tuned to the specific task, or using embedding models that can capture semantic context with significantly lower runtime. In this work, the high-level planner intervenes according to a heuristic threshold of average agent velocity, which may not be an optimal indicator of the failure mode of the low-level controller. If the system can better anticipate deadlocks, the high-level planner can intervene earlier. Future work includes embedding the local environment information, either as text or image, to classify whether a deadlock is likely to occur.

REFERENCES

[1] B. Li and H. Ma, "Double-deck multi-agent pickup and delivery: Multi-robot rearrangement in large-scale warehouses," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3701–3708, 2023.

[2] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," *AI magazine*, vol. 29, no. 1, pp. 9–9, 2008.

[3] J. Dinneweth, A. Boubezoul, R. Mandiau, and S. Espié, "Multi-agent reinforcement learning for autonomous vehicles: a survey," *Autonomous Intelligent Systems*, vol. 2, no. 1, p. 27, 2022.

[4] Y. Tian, K. Liu, K. Ok, L. Tran, D. Allen, N. Roy, and J. P. How, "Search and rescue under the forest canopy using multiple uavs," *The International Journal of Robotics Research*, vol. 39, no. 10-11, pp. 1201–1221, 2020.

[5] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[6] F. Mehdifar, C. P. Bechlioulis, F. Hashemzadeh, and M. Baradarannia, "Prescribed performance distance-based formation control of multi-agent systems," *Automatica*, vol. 119, p. 109086, 2020.

[7] C. Dawson, S. Gao, and C. Fan, "Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods for robotics and control," *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 1749–1767, 2023.

[8] S. Zhang, K. Garg, and C. Fan, "Neural graph control barrier functions guided distributed collision-avoidance multi-agent control," in *7th Annual Conference on Robot Learning*, 2023.

[9] K. Garg, S. Zhang, O. So, C. Dawson, and C. Fan, "Learning safe control for multi-robot systems: Methods, verification, and open challenges," *Annual Reviews in Control*, vol. 57, p. 100948, 2024.

[10] S. Zhang, O. So, K. Garg, and C. Fan, "Gcbf+: A neural graph control barrier function framework for distributed safe multi-agent control," *arXiv preprint arXiv:2401.14554*, 2024.

[11] Y. Zhou, H. Hu, Y. Liu, S.-W. Lin, and Z. Ding, "A distributed method to avoid higher-order deadlocks in multi-robot systems," *Automatica*, vol. 112, p. 108706, 2020.

[12] J. S. Grover, C. Liu, and K. 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*. Springer, 2021, pp. 294–312.

[13] J. Grover, C. Liu, and K. Sycara, "The before, during, and after of multi-robot deadlock," *The International Journal of Robotics Research*, vol. 42, no. 6, pp. 317–336, 2023.

[14] Y. Chen, M. Guo, and Z. Li, "Deadlock resolution and recursive feasibility in mpc-based multi-robot trajectory generation," *IEEE Transactions on Automatic Control*, 2024.

[15] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," in *ICML 2022 Workshop on Knowledge Retrieval and Language Models*, 2022. [Online]. Available: https://openreview.net/forum?id=6p3AuaHAFiN

[16] A. Z. Ren, J. Clark, A. Dixit, M. Itkina, A. Majumdar, and D. Sadigh, "Explore until confident: Efficient exploration for embodied question answering," *arXiv preprint arXiv:2403.15941*, 2024.

[17] W. Huang, F. Xia, D. Shah, D. Driess, A. Zeng, Y. Lu, P. Florence, I. Mordatch, S. Levine, K. Hausman *et al.*, "Grounded decoding: Guiding text generation with grounded models for embodied agents," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[18] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[19] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," in *International Conference on Machine Learning*. PMLR, 2022, pp. 9118–9147.

[20] A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," in *Conference on robot learning*. PMLR, 2023, pp. 287–318.

[21] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg, "Text2motion: From natural language instructions to feasible plans," *Autonomous Robots*, vol. 47, no. 8, pp. 1345–1365, 2023.

[22] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar *et al.*, "Inner monologue: Embodied reasoning through planning with language models," in *Conference on Robot Learning*. PMLR, 2023, pp. 1769–1782.

[23] Y. Chen, J. Arkin, Y. Zhang, N. Roy, and C. Fan, "Scalable multi-robot collaboration with large language models: Centralized or decentralized systems?" in *International Conference on Robotics and Automation*. IEEE, 2024.

[24] ——, "Autotamp: Autoregressive task and motion planning with llms as translators and checkers," in *International Conference on Robotics and Automation*, 2024.

[25] Y. Chen, R. Gandhi, Y. Zhang, and C. Fan, "NL2TL: Transforming natural languages to temporal logics using large language models," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, p. 15880–15903.

[26] Y. Xie, C. Yu, T. Zhu, J. Bai, Z. Gong, and H. Soh, "Translating natural language to planning goals with large-language models," *arXiv preprint arXiv:2302.05128*, 2023.

[27] T. Silver, V. Hariprasad, R. S. Shuttleworth, N. Kumar, T. Lozano-Pérez, and L. P. Kaelbling, "PDDL planning with pretrained large language models," in *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022. [Online]. Available: https://openreview.net/forum?id=1QMMUB4zfl

[28] B. Liu, Y. Jiang, X. Zhang, Q. Liu, S. Zhang, J. Biswas, and P. Stone, "Llm+ p: Empowering large language models with optimal planning proficiency," *arXiv preprint arXiv:2304.11477*, 2023.

[29] M. Kwon, H. Hu, V. Myers, S. Karamcheti, A. Dragan, and D. Sadigh, "Toward grounded social reasoning," *arXiv preprint arXiv:2306.08651*, 2023.

[30] X. Ma, S. Yong, Z. Zheng, Q. Li, Y. Liang, S.-C. Zhu, and S. Huang, "Sqa3d: Situated question answering in 3d scenes," in *International Conference on Learning Representations*, 2023.

[31] L. Wen, X. Yang, D. Fu, X. Wang, P. Cai, X. Li, T. Ma, Y. Li, L. Xu, D. 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.

[32] B. Chen, Z. Xu, S. Kirmani, B. Ichter, D. Driess, P. Florence, D. Sadigh, L. Guibas, and F. Xia, "Spatialvlm: Endowing vision-language models with spatial reasoning capabilities," *arXiv preprint arXiv:2401.12168*, 2024.

[33] J. Gao, B. Sarkar, F. Xia, T. Xiao, J. Wu, B. Ichter, A. Majumdar, and D. Sadigh, "Physically grounded vision-language models for robotic manipulation," *arXiv preprint arXiv:2309.02561*, 2023.

[34] Y. Jiang, A. Gupta, Z. Zhang, G. Wang, Y. Dou, Y. Chen, L. Fei-Fei, A. Anandkumar, Y. Zhu, and L. Fan, "Vima: General robot manipulation with multimodal prompts," in *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022.

[35] D. Shah, M. R. Equi, B. Osiński, F. Xia, B. Ichter, and S. Levine, "Navigation with large language models: Semantic guesswork as a heuristic for planning," in *Conference on Robot Learning*. PMLR, 2023, pp. 2683–2699.

[36] V. S. Dorbala, J. F. Mullen, and D. Manocha, "Can an embodied agent find your "cat-shaped mug"? llm-based zero-shot object navigation," *IEEE Robotics and Automation Letters*, vol. 9, no. 5, pp. 4083–4090, 2024.

[37] D. Shah, B. Osiński, S. Levine *et al.*, "Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action," in *Conference on robot learning*. PMLR, 2023, pp. 492–504.

[38] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.

[39] Q. Guo, R. Wang, J. Guo, B. Li, K. Song, X. Tan, G. Liu, J. Bian, and Y. Yang, "Connecting large language models with evolutionary algorithms yields powerful prompt optimizers," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=ZG3RaNIsO8

[40] X. Wang, C. Li, Z. Wang, F. Bai, H. Luo, J. Zhang, N. Jojic, E. Xing, and Z. Hu, "Promptagent: Strategic planning with language models enables expert-level prompt optimization," in *The Twelfth International Conference on Learning Representations*, 2024.

[41] C. Fernando, D. Banarse, H. Michalewski, S. Osindero, and T. Rocktäschel, "Promptbreeder: Self-referential self-improvement via prompt evolution," *arXiv preprint arXiv:2309.16797*, 2023.

[42] M. Mesbahi and M. Egerstedt, "Graph theoretic methods in multiagent networks," in *Graph Theoretic Methods in Multiagent Networks*. Princeton University Press, 2010.

[43] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, 2017.

APPENDIX

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 $x_i \in \mathcal{X} \subset \mathbb{R}^{n_x}$ denoting agents' state space and $u_i \in \mathcal{U} \subset \mathbb{R}^{n_u}$ the control constraint set for $i \in \{1, 2, \ldots, N\}$.[4]. The state x_i consists of the position $p_i \in \mathbb{R}^2$ in the global coordinates along with other states, such as the orientation and the velocity of the agent i. 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 $\mathcal{E}(t) \subset \mathcal{V}^a \times \mathcal{V}$ 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 $(i, j), i, j \in \mathcal{V}^a$ 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 [42, 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., $\lambda_2(L(A(t))) > 0$.

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 [43] 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 [10]. Here, we briefly review the GCBF+ controller and present the details of how the safety and connectivity constraints are encoded.

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

A. Graph control barrier functions (GCBF)

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 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:

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

where $\bar{x}$ denotes the joint state vector for the MRS. The unsafe, or avoid set can be defined accordingly as $\mathcal{A}_N = \mathcal{X}^N \setminus \mathcal{S}_N$. We now introduce the notion of GCBF for encoding safety for MAS. 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

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

Using these notations, the notion of GCBF is defined in [10] 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 [10, Theorem 1] GCBF certifies the forward invariance of its 0-superlevel set under control inputs in the set

$$\mathcal{U}_{\text{safe}}^N \coloneqq \Big\{ \bar{u} \in \mathcal{U}^N \;\Big|\; \dot{h}(\bar{x}_{\mathcal{N}_i}) + \alpha(h(\bar{x}_{\mathcal{N}_i})) \geq 0,\ \forall i \in V_a \Big\}. \tag{6}$$

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

B. Learning GCBF with safety and connectivity constraints

Following [10], 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 [10], 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., $A^d_{ij} = 1$, and $[1, 0]^\top$ if they are not required to be connected, i.e., $A^d_{ij} = 0$. 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

$$\begin{aligned}\mathcal{S}^c_N := \Big\{ \bar{x} \in \mathcal{X}^N \;\Big|\; \Big(\Big\| y^{(i)}_j \Big\| > r,\ \forall i \in \mathcal{V}_a, \forall j \in n_{\text{rays}} \Big) \\ \bigwedge \Big(\min_{i,j \in \mathcal{V}_a, i \neq j} \|p_i - p_j\| > 2r \Big) \\ \bigwedge \Big(\max_{i,j \in \mathcal{V}_a, A^d_{ij}=1} \|p_i - p_j\| < R \Big) \Big\}. \end{aligned} \tag{7}$$

Consequently, the unsafe, or avoid set with the connectivity constraint is defined as $\mathcal{A}^c_N = \mathcal{X}^N \setminus \mathcal{S}^c_N$. 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 [10]. 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 [10] is used to train the distributed control policy, with the safe set definition modified as per (7).

C. Leader-follower for small MRS

The follower assignment is carried out in an iterative way. 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

$$i_{\text{follow},k} = \underset{j \in \mathcal{V} \setminus \mathcal{V}_{\text{lead}}(t,k-1)}{\arg\min} \ \min_{i \in \mathcal{V}_{\text{lead}}(t,k-1)} \|p_i - p_j\|, \tag{8}$$

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 a given minimum distance away from its goal, i.e., if $\|p_i - p^{\text{goal}}_i\| \geq d_{\min}$ for some $d_{min} > 0$, their temporary goal is chosen as the location of their leaders, i.e., $\bar{p}^{\text{goal}}_i = p_{i_{\text{lead}}}$.

D. Leader-follower for large MRS

As discussed in Section IV, 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.

Algorithm 1: Leader-follower 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 using k-means clustering */
if $N < N_M$ **then**
  $\{\mathcal{V}_k^a\} = \{\mathcal{V}^a\}$
else
  $\{\mathcal{V}_k^a\}$ = kmeans$(\{p_i\}, K)$
end
/* Find sub-leaders for each of the 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 to each agent */
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 closest agent to the set of leader 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 this follower */
    $k_{\text{lead}} = \arg\min_{j \in \mathcal{V}_{\text{lead}}(k)} \|p_{i_{\text{follow}}} - p_j\|$
    /* Assign temporary goal to the follower */
    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

E. Task and output description prompts

The task description prompts used for VLMs are given in Figure 5 while those used for LLMs are given in Figure 6.

```
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.
```

Fig. 5: 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.
```

Fig. 6: Description prompts used for text-based (i.e., LLMs) high-level planners.

F. Environment description

The environment description prompts used for VLMs are given in Figure 7 while those used for LLMs are given in Figure 8. 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.

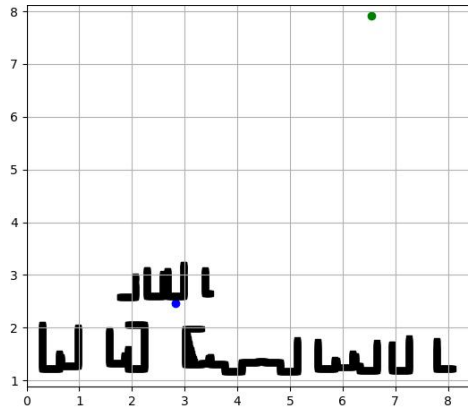

Fig. 7: Environment prompt for VLM for "Maze" environment with $N = 50$ and $M = 375$.

{'role': 'user', 'content': '***Env:725***Number of agents***1***Safety radius***0.05***Connectivity radius***0.5***Agents***AgentId***1***Current state***(3.86,2.77)***Goal location***(3.16,3.16)***Obstacles***[(x=0.24, y=1.15, 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=0.49, y=1.16, w=0.12,h=0.34),(x=0.56, y=1.29, w=0.41,h=0.12),(x=0.38, 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=1.27, 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=1.27, 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=1.38, 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=0.38,h=0.12),(x=3.70, y=1.12, w=0.12,h=0.19),(x=3.76, y=1.12, w=0.10,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=1.38, 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=1.95, 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=2.23, 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=2.21, y=2.86, w=0.12,h=0.12),(x=2.21, 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'}

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

G. Output of foundation models

The output of the LLM or VLM is a JSON object. As an example, for the input in Figure 7, the output from `Claude3-Opus` VLM is: {"Leader" : 1, "Waypoint" : $[[2.8, 4.5], [5.5, 4.5], [5.5, 7.9]]$}.

In Figure 9, we report the mean distance traveled by the MRS under various high-level planners normalized by the mean initial distance of MRS from the goal locations. For "Room" environment, "random" baseline has the highest mean distance traveled among all methods, while having a relatively lower reach rate, compared to some of the foundation models. There does not seem to be much variation in the distance traveled among LLM-based high-level planners for "Maze" environments. While GPT4-VLM has a relatively higher mean traveled distance, the A*-based baseline has the highest distance traveled for "Maze" environment with $N = 25$. For "Room" environment, `Claude3O-VLM` has the highest mean distance traveled by MRS. An interesting observation here is that unlike other performance metric which seem to have a stronger correlation among themselves, the mean distance traveled does not seem to have such property. As an example, for "Room" environment and "Maze" environment with $N = 50$ with VLMs as high-level planners, the distance traveled seems to follow the same pattern as the number of calls. However, the same cannot be said about "Maze" environment with $N = 25$.

H. Effect of partial environment infromation

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 10 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 and GPT4-LLM perform better with partial information We can observe that, with the exception of `GPT-4o` VLM and `GPT4` LLM, the performance drops when partial information is used. It is not entirely clear why `GPT-4o` VLM and `GPT4` LLM perform better with partial information. Our understanding is that the performance of `GPT-4o` VLM and `GPT4` LLM is poor with the complete information due to their ability (or in this case, inability) of utilizing 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.

I. Effect of multi-leader assignment

Additionally, we evaluate the effect of using just one leader for large-scale MRS instead of the proposed multi-leader assignment as described in Section D.

As can be seen from the figure, the performance in terms of reach rate drops significantly when only one leader is used and all the other robots in MRS directly follow that leader as compared to the proposed multi-leader assignment where robots follow their local leaders. The number of interventions needed by the high-level planner is also higher when one leader is used instead of the proposed multi-leader framework. This illustrates the efficacy of the multi-leader assignment in large-scale systems.

Recall that a deadlock is defined as the existence of an interval $\tau \subset \mathbb{R}$, given as $\tau = [a, b)$ where $a, b \in \mathbb{R}$, where the average speed of the MRS $\frac{1}{N}\sum_i |\dot{p}_i|(t) = 0$ for all $t \in \tau$ and the Lebesgue measure of the interval $\mu(\tau) > 0$, where $\mu(\tau) = |\tau| \coloneqq b - a$. Let $a \in \mathbb{R}$ be a time instant when the average speed of the MRS is zero, i.e., $\frac{1}{N}\sum_i |\dot{p}_i|(a) = 0$. Per the leader-assignment condition based on the average speed of the MRS, the condition $\frac{1}{N}\sum_i \|\dot{p}_i(t)\| < u_{min}$ triggers a leader assignment step at time t. The leader is chosen as the agent that has the minimum value of $\frac{\|p_i - p_{gi}\|}{\|\dot{p}_i|_{p_{gi,\text{temp}}}\|}$. There are two cases possible: $\|\dot{p}_i|_{p_{gi,\text{temp}}}\| = 0$ for all $i \in \mathcal{V}$ or there exists at least one $i \in \mathcal{V}$ such that $\|\dot{p}_i|_{p_{gi,\text{temp}}}\| > 0$. The latter case leads to a non-zero average speed of the MRS, leading to $b = a$ in $\tau = [a, b)$, and thus, a deadlock cannot occur in this case. Next, we prove that

$$\|\dot{p}_i|_{p_{gi,\text{temp}}}\| = 0$$

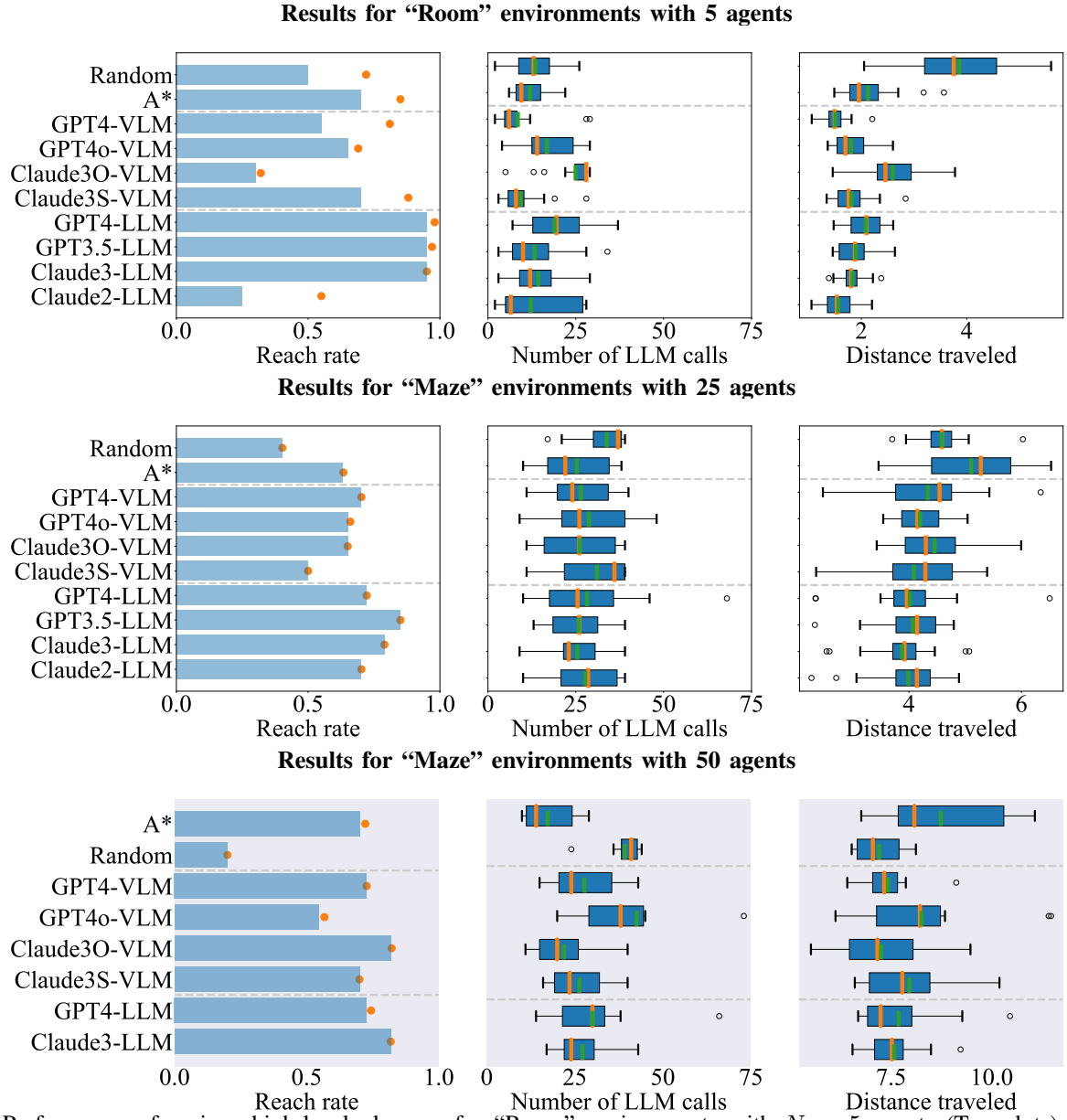

Fig. 9: Performance of various high-level planners for "Room" environments with $N = 5$ agents (Top plots), "Maze" environments with $N = 25$ agents (Middle plots), and "Maze" environments with $N = 50$ agents (Bottom plots). From left to right: 1) The bar shows the ratio of the trajectories where **all** 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; and 3) Box plot for the mean traveled distance by the multi-robot system normalized by the mean initial distance from the goal locations. In the box plots, the median values are in orange and the mean values are in green.

for all $i \in \mathcal{V}$ is only possible at $t = 0$. Note that $\|\dot{p}_i|_{p_{gi,\text{temp}}}\| = 0$ for all $i \in \mathcal{V}$ is possible only if *all* the agents are occluded with obstacles and none of the agents have any free space to move. This is only possible at $a = 0$, i.e., the MRS is initialized in a location that is occluded by obstacles such that it cannot move. For $a > 0$, since the MRS can reach such a location where the average speed becomes lower than the leader-assignment threshold, there exists at least one agent that has *free* space around it to move and hence, $\|\dot{p}_i|_{p_{gi,\text{temp}}}\| = 0$ is not possible for all $i \in \mathcal{V}$ for any $a > 0$ and hence, the MRS cannot get stuck in a deadlock.

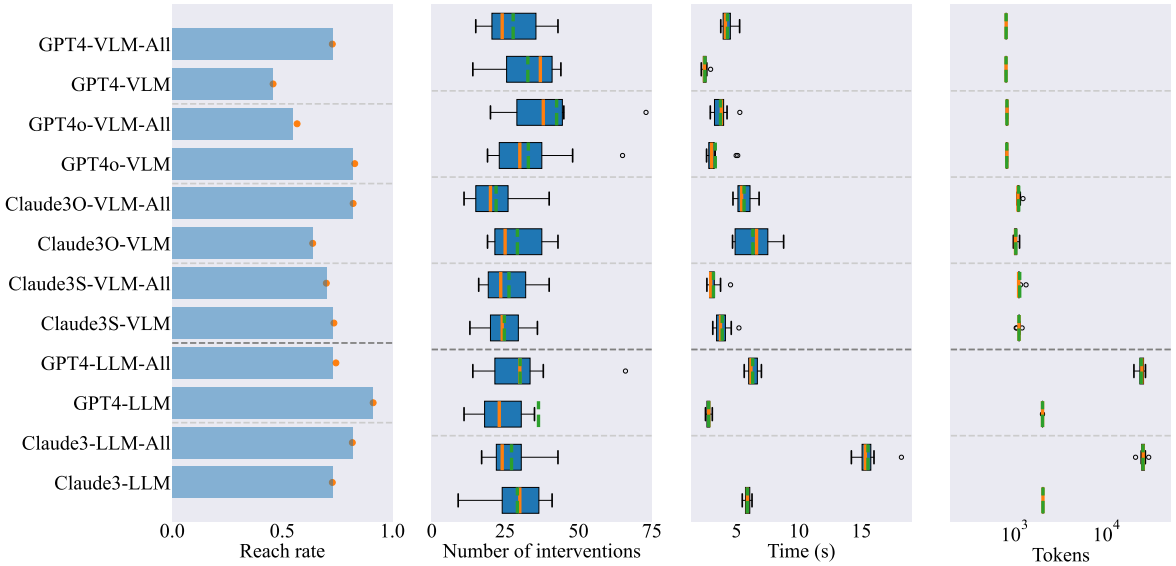

Fig. 10: Performance of various high-level planners for "Maze" environments with $N = 50$ agents with all known environment information and partial information (the case with all known environment information is indicated with the suffix "-All", e.g. "GPT4-VLM-All"). From left to right: 1) The bar shows 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.

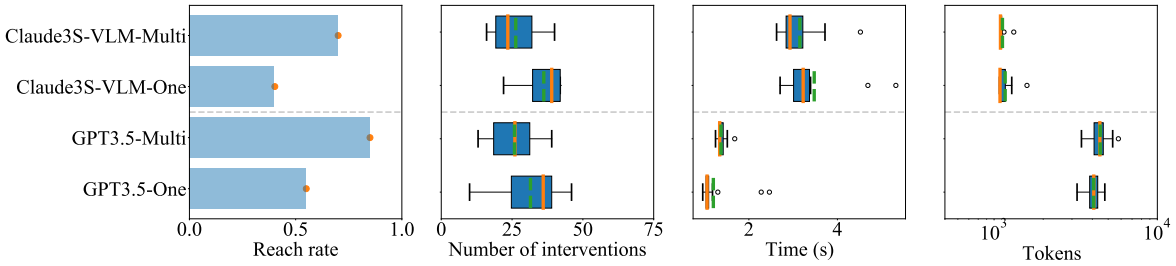

Fig. 11: Performance of `Claude3-Sonnet-VLM` planner for "Maze" environments with $N = 50$ agents and `GPT3.5-LLM` for "Maze" environment with $N = 25$ with a single leader and multi-leader assignment (the case with one leader is indicated with suffix "-One", and that with multi-leader with "-Multi". 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.