arXiv:2309.15943v1 [cs.RO] 27 Sep 2023

Scalable Multi-Robot Collaboration with Large Language Models: Centralized or Decentralized Systems?

Yongchao Chen[1,2], Jacob Arkin[1], Yang Zhang[3], Nicholas Roy[1], and Chuchu Fan[1]

Abstract— **A flurry of recent work has demonstrated that pre-trained large language models (LLMs) can be effective task planners for a variety of single-robot tasks. The planning performance of LLMs is significantly improved via prompting techniques, such as in-context learning or re-prompting with state feedback, placing new importance on the token budget for the context window. An under-explored but natural next direction is to investigate LLMs as multi-robot task planners. However, long-horizon, heterogeneous multi-robot planning introduces new challenges of coordination while also pushing up against the limits of context window length. It is therefore critical to find token-efficient LLM planning frameworks that are also able to reason about the complexities of multi-robot coordination. In this work, we compare the task success rate and token efficiency of four multi-agent communication frameworks (centralized, decentralized, and two hybrid) as applied to four coordination-dependent multi-agent 2D task scenarios for increasing numbers of agents. We find that a hybrid framework achieves better task success rates across all four tasks and scales better to more agents. We further demonstrate the hybrid frameworks in 3D simulations where the vision-to-text problem and dynamical errors are considered. See our project website[4] for prompts, videos, and code.**

I. INTRODUCTION

Multi-robot systems have great potential as a tool for operations that require the completion of many tasks, such as warehouse management. Planning for these systems is often challenging due to heterogeneous robot capabilities, coordination during tasks requiring multiple robots, inter-dependencies of separate tasks, and general safety considerations (e.g. collision avoidance). Further, the difficulty scales with the number of robots. Previous work has used algorithm-based [1], [2], [3] or learning-based [4], [5] methods to control multi-robot systems. Such approaches are typically tuned for a specific scenario, requiring significant engineering effort that limits generalization into novel tasks or scenarios.

Motivated by the ability of pre-trained large language models (LLMs) to generalize to new task domains [6], [7], there have been many recent efforts to use them for single-agent task planning [8], [9]. Planning performance is significantly improved through clever use of the context provided to the LLM, whether via techniques for initial prompts (e.g., in-context learning, chain-of-thought) or iterative re-prompting with feedback (e.g., environment state changes, detected errors). Given this success, there is new interest in investigating LLMs as task planners for multi-robot systems [10], [11]. These recent efforts address systems consisting of two or three robots; they assign an LLM to each robot and have the models engage in collaborative dialogue rounds to try to find good plans.

Scaling to systems of many robots and tasks with longer horizons is an issue for approaches that assign each robot its own LLM agent. First, both the number of possible coordinating actions and the possible action inter-dependencies grow exponentially with the number of agents, making the reasoning more difficult for the language models. Second, the context provided to each LLM contains the responses of each other LLM for the current round of dialogue in addition to the history of dialogue, actions, and states from prior rounds; so, scaling the number of agents also scales the context token length requirements toward their modern limits and increases the runtime of LLM inference (and API costs). Moreover, the immediately relevant information in the context can become diluted in longer prompts. These limitations are beyond the scope of prior work [10], [11].

Our goal is to preserve the generalizability of LLMs as task planners for multi-robot settings while addressing the challenges of scaling to many agents. We argue that different frameworks for integrating LLM planners into multi-robot task planning can improve both scalability and task planning success rates. In this work, we compare four different frameworks (Figure 3) of cooperative dialogue for task planning among multiple LLMs for increasing numbers of robots. For each, planning is performed incrementally in which the LLMs collaborate to find the next action to take for each agent in the system. The first approach (DMAS) uses a decentralized communication framework in which each robot is provided its own LLM agent and dialogue proceeds in rounds of turn-taking. The second approach (CMAS) uses a centralized framework in which a single LLM produces the next action for all robots in the system. We also propose two hybrid versions of these two approaches: (1) a variant of DMAS that adds a central LLM responsible for providing an initial plan to prime the dialogue (HMAS-1) and (2) a variant of CMAS that gives each robot an LLM with which to provide robot-local feedback to the central LLM planner. To further address issues of token length due to historical dialogue and planning context, we also propose a truncated prompt that only includes state-action information from prior dialogue rounds. We evaluate the performance of each approach in four different task planning environments inspired by warehouse settings. To further demonstrate LLMs as multi-robot planners, we apply these approaches to a simulated 3D manipulation task that requires coordination among the manipulators.

---

[1] Massachusetts Institute of Technology. jarkin@mit.edu, nickroy@csail.mit.edu, chuchu@mit.edu

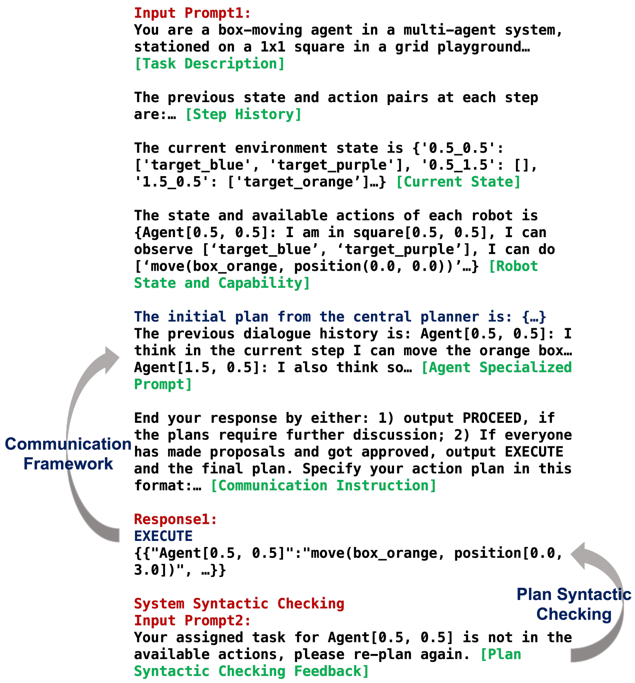
[2] Harvard University. yongchaochen@fas.harvard.edu

[3] MIT-IBM Watson AI Lab. yang.zhang2@ibm.com

[4] https://yongchao98.github.io/MIT-REALM-Multi-Robot/

II. PROBLEM DESCRIPTION

This work focuses on task planning for multi-robot systems. We consider a cooperative multi-robot task scenario with N robots and M LLM agents. We assume that each LLM agent has full knowledge of the environment and each robot's capabilities. The robot capabilities can be heterogeneous, requiring the planners to assign tasks to robots accordingly. In order to provide each LLM with the task goals and observations, we manually define functions to translate them into text prompts. We also define functions to map the output of the LLM planners into pre-defined robot actions. Planning is performed iteratively, choosing the next action for each robot to take. At each iteration, the M LLM agents engage in collaborative dialogue to find a consensus for the next set of robot actions. Given the next action, the robots act in the environment, and the resulting new state is provided as context to the LLMs for the next planning iteration.

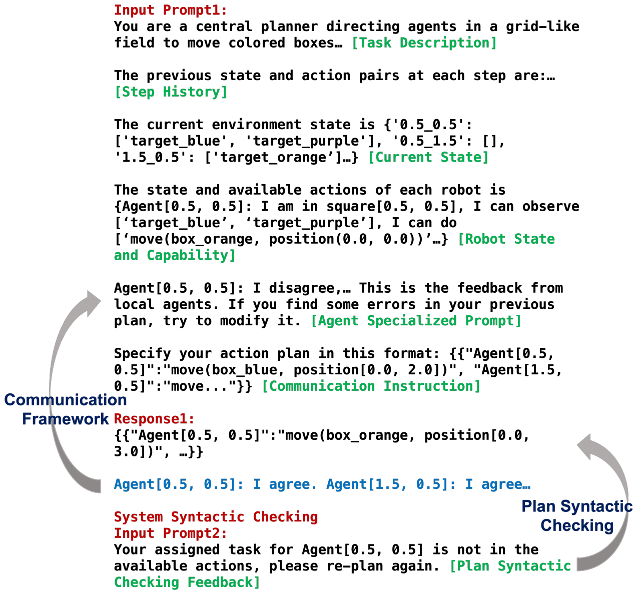


```
Input Prompt1:
You are a box-moving agent in a multi-agent system,
stationed on a 1x1 square in a grid playground…
[Task Description]

The previous state and action pairs at each step
are:… [Step History]

The current environment state is {'0.5_0.5':
['target_blue', 'target_purple'], '0.5_1.5': [],
'1.5_0.5': ['target_orange’]…} [Current State]

The state and available actions of each robot is
{Agent[0.5, 0.5]: I am in square[0.5, 0.5], I can
observe [‘target_blue’, ‘target_purple’], I can do
[‘move(box_orange, position(0.0, 0.0))’…} [Robot
State and Capability]

The initial plan from the central planner is: {…}
The previous dialogue history is: Agent[0.5, 0.5]: I
think in the current step I can move the orange box…
Agent[1.5, 0.5]: I also think so… [Agent Specialized
Prompt]

End your response by either: 1) output PROCEED, if
the plans require further discussion; 2) If everyone
has made proposals and got approved, output EXECUTE
and the final plan. Specify your action plan in this
format:… [Communication Instruction]

Response1:
EXECUTE
{{"Agent[0.5, 0.5]":"move(box_orange, position[0.0,
3.0])", …}}

System Syntactic Checking
Input Prompt2:
Your assigned task for Agent[0.5, 0.5] is not in the
available actions, please re-plan again. [Plan
Syntactic Checking Feedback]
```

Fig. 1. Simplified prompt example of the HMAS-1 local agent. The acquired 'Response1' acts as the initial plan, or otherwise sent to the next local agent for further discussion.

III. METHODS

Given the goal and the current environment state as text, the LLM agents engage in dialogue per the communication framework (Section III-B) in order to generate an initial set of actions for the robots to take. Before execution, this action set is checked by an external rules-based verifier for syntax errors; any errors are provided as feedback to re-prompt for correction. Given a syntactically correct set of actions, the robots then execute those actions in the environment, resulting in a new environment state. We show examples of the prompt structure for the HMAS-1 and HMAS-2 approaches in Figure 1 and Figure 2 respectively. We describe the main components of the initial prompt in the next subsection.



```
Input Prompt1:
You are a central planner directing agents in a grid-like
field to move colored boxes… [Task Description]

The previous state and action pairs at each step are:…
[Step History]

The current environment state is {'0.5_0.5':
['target_blue', 'target_purple'], '0.5_1.5': [],
'1.5_0.5': ['target_orange’]…} [Current State]

The state and available actions of each robot is
{Agent[0.5, 0.5]: I am in square[0.5, 0.5], I can observe
[‘target_blue’, ‘target_purple’], I can do
[‘move(box_orange, position(0.0, 0.0))’…} [Robot State
and Capability]

Agent[0.5, 0.5]: I disagree,… This is the feedback from
local agents. If you find some errors in your previous
plan, try to modify it. [Agent Specialized Prompt]

Specify your action plan in this format: {{"Agent[0.5,
0.5]":"move(box_blue, position[0.0, 2.0])", "Agent[1.5,
0.5]":"move..."}} [Communication Instruction]

Response1:
{{"Agent[0.5, 0.5]":"move(box_orange, position[0.0,
3.0])", …}}

Agent[0.5, 0.5]: I agree. Agent[1.5, 0.5]: I agree…

System Syntactic Checking
Input Prompt2:
Your assigned task for Agent[0.5, 0.5] is not in the
available actions, please re-plan again. [Plan Syntactic
Checking Feedback]
```

Fig. 2. Simplified prompt example of the HMAS-2 central agent. The generated 'Response1' is sent to local agents for feedback. Once the central-local iteration terminates, the output plan is checked for syntactic correctness.

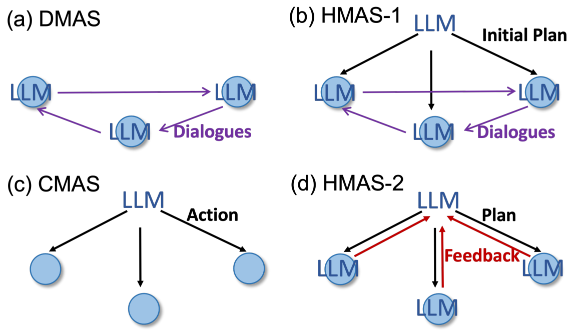


Fig. 3. Four LLM-based multi-agent communication frameworks compared in this work. The circles represent robots that may have actions in the current step and the 'LLM' text represents each LLM agent. The overlap between one circle and one 'LLM' text means that the robot is delegated with one LLM agent to express its special opinions to other agents. The 'LLM' text without the overlapped circle represents a central planning agent.

A. *Main Components of LLM Prompt*

We use the same basic structure to prompt each LLM agent, but the specifics of the prompt depend on the individual agent's role. The prompt structure consists of the following main components:

- **Task Description**: the requirements and constraints of the task for the multi-robot system to accomplish.
- **Step History**: the history of dialogue, environment states, and actions from previous steps in the iterative planning process. We describe this in more detail in Section III-C.
- **Current State**: the objects in the environment (boxes) and their properties (position and volume).

- **Robot State & Capability**: the capabilities (available actions) of each robot and their current location. This prompt component is synthesized by our pre-defined functions. Note that the available actions include possible collisions; it is the responsibility of the planner to find safe plans.
- **Agent Specialized Prompt**: the prompt for each local agent emphasizes its own state and indicates the responses and initial plans of the other agents. For frameworks with a central agent, the prompt for that agent includes feedback from the local agents. Further, each agent is provided a persona.
- **Communication Instruction**: the instruction for how to respond to other agents & how to format the output.
- **Plan Syntactic Checking Feedback**: (optional) explanation of syntax errors in the generated output. The syntactic checking ensures that the output is formatted correctly and uses available actions.

B. Communication Frameworks for Sub-task Plan

We compare the four LLM-based multi-robot planning frameworks shown in Figure 3. The Decentralized Multi-agent System framework (DMAS) is shown in Figure 3(a) and is the framework used in previous works on LLMs as multi-robot planners [10], [11]. Each robot is assigned an LLM planner and another agent to whom it should send its comments. The agents use a turn-taking approach for dialogue, as illustrated. The comments from prior agents in the dialogue are concatenated and included as part of the prompt for the next agent; thus, the prompt length increases over the duration of the dialogue for the current planning iteration. The dialogue ends once the current agent outputs “EXECUTE" followed by the action for each agent.

The Centralized Multi-agent System framework (CMAS) is shown in Figure 3(c). This approach incorporates only a single LLM as a central planner that is responsible for assigning the actions for each robot at each planning iteration.

We propose two Hybrid Multi-agent System frameworks, HMAS-1 (Figure 3(b)) & HMAS-2 (Figure 3(d)), that are variants of DMAS and CMAS respectively. In HMAS-1, a central LLM planner proposes an initial set of actions for the current planning iteration that is provided to each of the robots' LLM planners; the robots' LLMs then proceed as done in DMAS. In HMAS-2, a central LLM planner generates an initial set of actions for each robot, as done in CMAS; however, each robot has an LLM agent that checks its assigned action and provides feedback to the central planner. In the case of a local agent disagreeing with its assigned action, the central agent will re-plan. This process repeats until each robot's LLM agrees with its assigned action. Note that in both HMAS-1 and HMAS-2, only the agents that will take an action participate in the dialogue, thus reducing the duration of dialogue and the corresponding number of tokens in the prompts.

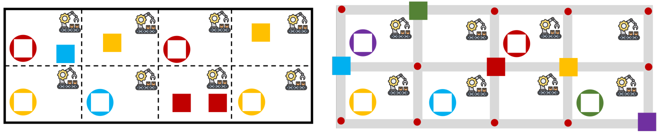
C. Step History

Including the full history of the dialogue, environment states, and actions rapidly exhausts the context token budget for the LLM planners, constraining the performance of these frameworks. We therefore compare three approaches in an ablation study of the historical information included in the context: (1) no historical information, (2) only state-action pair history (no dialogue), and (3) the full history. We only include the results for all three approaches in the ablation study; since we found that (2) has the best trade-off between task performance and token efficiency (see Section IV-C), all other experiments are performed with only state-action pair history.

D. Token Length Constraint

We use gpt-4-0613 and gpt-3.5-turbo-0613 in this work, which have context token limits of 8192 and 4097, respectively. To make sure the total token length (prompt + response) does not surpass these limits, we employ a sliding context window over the step history part of the prompt; the step history will include as many of the most recent steps as permissible without surpassing a total prompt length of 3500 tokens.

(a) BoxNet1 (b) BoxNet2



(c) Warehouse (d) BoxLift

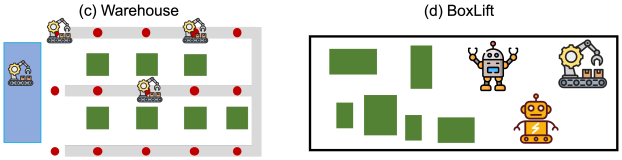


Fig. 4. Four multi-robot task planning environments.

IV. EXPERIMENTS

A. Testing Environments

To compare the four different LLM-based planning frameworks, we design four multi-robot task planning environments inspired by a warehouse setting. In order to evaluate how these frameworks scale to many robots, we instantiate each environment with increasing numbers of robots. For BoxNet1 and BoxNet2, we run trials of 4, 8, 16, and 32 robots. For Warehouse and BoxLift, we run trials 4, 6, 8, and 10 robots. For each number of robots in each environment, we perform 10 trials with varied initial conditions, resulting in 40 total trials per environment.

We track whether each trial resulted in task execution. A task is considered a failure in the following conditions: (1) the dialogue among agents results in a context length beyond the token limit, (2) the agents do not reach consensus before a pre-specified limit of dialogue rounds, (3) the syntactic checking iterates beyond a pre-specified limit, (4) the number of planning iterations exceeds a limit before reaching the goal, and (5) the plan results in a collision. We choose the limits for (1), (2), (3), and (4) such that the failure is very likely a result of endless dialogue or actions. Note that only BoxNet2 and Warehouse can have collision.

BoxNet1 Figure 4(a) shows the BoxNet1 environment. The environment consists of cell regions, robot arms, colored boxes, and colored goal locations (circles) for each box. The goal is to move each box into its associated goal location in the fewest time steps. The robot arms are confined to the cell they occupy. Each arm has three possible actions: (1) move a box within its cell to a neighboring cell, (2) move a box within its cell to a goal location within its cell, and (3) do nothing. We assume no collisions.

BoxNet2 Figure 4(b) shows the BoxNet2 environment, which is similar to BoxNet1. In this environment, each box can only be moved between cells by being placed at a corner (red circles), and a given corner can only hold one box at a time; we treat placing two or more boxes on the same corner as a collision (and thus task failure). Each arm in this environment has three possible actions: (1) move a box from a corner to a different corner of the cell, (2) move a box from a corner to a goal location within the its cell, and (3) do nothing. The constraint on box movement and possibility of collision makes this scenario more challenging than BoxNet1.

Warehouse Figure 4(c) shows the Warehouse environment. In this environment, mobile manipulators are tasked with moving all of the boxes (green) to the target region (blue) in the fewest time steps. Each robot can only move between permissible locations (red) by traveling along the gray paths; in a single time step, a robot cannot move beyond an adjacent permissible location. We treat two robots occupying the same location as a collision, resulting in task failure. A robot can pick up a box only when at a permissible location that is immediately adjacent to it. Each robot has six possible actions: (1) & (2) move left or right (if a permissible location exists), (3) pick up an adjacent box, (4) place a box in the target region, (5) move from the target region to any of the adjacent permissible locations, and (6) do nothing.

BoxLift Figure 4(d) shows the BoxLift environment. In this environment, robots are tasked to lift each box (green) in the fewest time steps. The robots are able to lift different amounts of weight, and the boxes have different sizes and weights. In a single time step, multiple robots can be assigned to lift the same box. The box is lifted if the total capability of the robots lifting is greater than the box's weight. As an additional challenge, the LLM agents are only able to observe the size of each box, not their weight. This is meant to simulate real situations in which box size roughly correlates with weight. The size and weight of each box is roughly proportional, but we introduce some variability. The LLM agents are provided feedback about whether or not the box was successfully lifted. This environment attempts to test the LLM planner's ability to efficiently assign heterogeneous robots to collaboritve tasks and also incorporate prior experience when planning.

B. *Metrics*

To measure how well each framework is able to plan, we report the average task success rate and average number of steps per plan. To measure the token efficiency and API usage, we also report the average number of tokens used per plan and the average number of API calls per plan. The average number of steps per plan, the average number of tokens per plan, and the average number of API calls per plan only include plans that were successful. We therefore report normalized values for those three metrics. Let $\mathcal{M}$ be the set of values for a given metric M, e.g. average API calls, such that $m_i \in \mathcal{M}$ is the value of metric M for the ith framework. Let $\hat{\mathcal{M}}$ be the set of values for the normalized metric $\hat{M}$ such that the normalized metric value $\hat{m}_i \in \hat{\mathcal{M}}$ for the ith framework is:

$$\hat{m}_i = \frac{m_i}{min(\mathcal{M})} \tag{1}$$

The best value for the normalized metrics is 1.0. The framework that performs best for one of those metrics will thus have a value of 1.0.

C. *Results*

Table I shows the experimental results for the four LLM-based multi-robot planning frameworks.

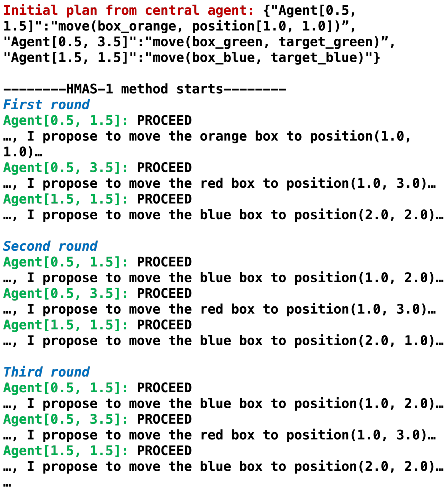
Communication Frameworks We note a few key results. The HMAS-2 framework outputs plans with highest quality since it achieves highest success rates and the fewest actions per plan. The CMAS framework has the fewest API calls and uses the fewest tokens; this is expected as it uses a single LLM and only requires one API call to generate the plan (assuming no syntax errors). The DMAS framework uses the most API calls and tokens, and also has the lowest task success rate. We observe that the LLM agents in DMAS often take many rounds of dialogue per planning step to decide to act, resulting in long dialogues. During the dialogue, the agents often repeat what previous agents have said without contributing anything new; or, agents will repeatedly propose the same action, diluting the context of important information [12]. HMAS-1, our hybrid variant of DMAS, primes the dialogue with an initial plan from a central LLM planner. This modification significantly improves the performance of the dialgoue that follows. We hypothesize that the initial plan serves as better starting point than DMAS, thus leading to better performance metrics. However, HMAS-2 outperforms HMAS-1 in all metrics. We show one example of HMAS-1 dialogue in Figure 5. It shows that the LLM agents can get stuck on their proposed action, leading to inefficient dialogue.

We also report the trend of task success rates as a function of increasing numbers of agents, as shown in Figure 6. For low numbers of agents, CMAS is competitive with HMAS-2; however, CMAS scales significantly worse to more agents. For more challenging tasks like Warehouse, CMAS performs worse than HMAS-2 for all numbers of agents, indicating that a single central LLM planner tends to generate unreasonable plans for more complex multi-robot task scenarios. Unlike CMAS, HMAS-2 is able to check and correct for errors

TABLE I

EVALUATION RESULTS ON FOUR TASKS. WE REPORT AVERAGE SUCCESS RATES (↑) OVER 40 RUNS ACROSS 4 AGENT NUMBERS PER TASK, AVERAGE NUMBER OF STEPS PER PLAN (↓), AVERAGE API CALLS (↓), AND AVERAGE NUMBER OF TOKENS ACROSS ALL RUNS (↓). APART FROM SUCCESS RATE, THE LATER THREE VALUES ARE NORMALIZED OVER SUCCESSFUL RUNS.

		DMAS	HMAS-1	CMAS	HMAS-2	HMAS-2 w/o History	HMAS-2 w/ All History	CMAS (GPT-3)	HMAS-2 (GPT-3)
BoxNet 1	Success	25.0%	52.5%	75.0%	**82.5%**	77.5%	75.0%	12.5%	27.5%
	Steps	2.25	1.58	1.92	**1.33**	1.45	1.34	2.76	1.74
	API Calls	16.00	5.06	**1.89**	4.62	4.61	4.41	3.21	6.33
	Tokens	48.56	13.80	**4.09**	9.00	4.18	8.95	6.73	10.02
BoxNet 2	Success	0.0%	7.5%	27.5%	**57.5%**	27.5%	32.5%	0.0%	5.0%
	Steps	-	1.38	1.22	**1.21**	1.69	1.24	-	1.29
	API Calls	-	8.24	**1.00**	3.35	4.20	2.39	-	3.97
	Tokens	-	9.32	**1.00**	5.22	3.89	3.16	-	5.83
Warehouse	Success	0.0%	5.0%	15.0%	**62.5%**	45.0%	52.5%	0.0%	10.0%
	Steps	-	1.62	1.73	**1.16**	1.23	1.34	-	1.19
	API Calls	-	8.52	**1.00**	5.48	5.62	5.71	-	6.33
	Tokens	-	14.02	**1.00**	9.21	9.98	10.56	-	10.09
BoxLift	Success	52.5%	67.5%	90.0%	**100.0%**	0.0%	90.0%	12.5%	20.0%
	Steps	1.42	1.33	1.99	**1.11**	-	1.17	2.82	1.65
	API Calls	37.13	18.99	**1.00**	14.90	-	15.21	1.76	18.92
	Tokens	50.34	24.54	**1.00**	21.52	-	28.02	2.04	25.42



```
Initial plan from central agent: {"Agent[0.5,
1.5]":"move(box_orange, position[1.0, 1.0])",
"Agent[0.5, 3.5]":"move(box_green, target_green)",
"Agent[1.5, 1.5]":"move(box_blue, target_blue)"}

--------HMAS-1 method starts--------
First round
Agent[0.5, 1.5]: PROCEED
…, I propose to move the orange box to position(1.0,
1.0)…
Agent[0.5, 3.5]: PROCEED
…, I propose to move the red box to position(1.0, 3.0)…
Agent[1.5, 1.5]: PROCEED
…, I propose to move the blue box to position(2.0, 2.0)…

Second round
Agent[0.5, 1.5]: PROCEED
…, I propose to move the blue box to position(1.0, 2.0)…
Agent[0.5, 3.5]: PROCEED
…, I propose to move the red box to position(1.0, 3.0)…
Agent[1.5, 1.5]: PROCEED
…, I propose to move the blue box to position(2.0, 1.0)…

Third round
Agent[0.5, 1.5]: PROCEED
…, I propose to move the blue box to position(1.0, 2.0)…
Agent[0.5, 3.5]: PROCEED
…, I propose to move the red box to position(1.0, 3.0)…
Agent[1.5, 1.5]: PROCEED
…, I propose to move the blue box to position(2.0, 2.0)…
…
```

Fig. 5. Simplified communication example of HMAS-1. The local agents hesitate on possible actions, making the dialogue endless.

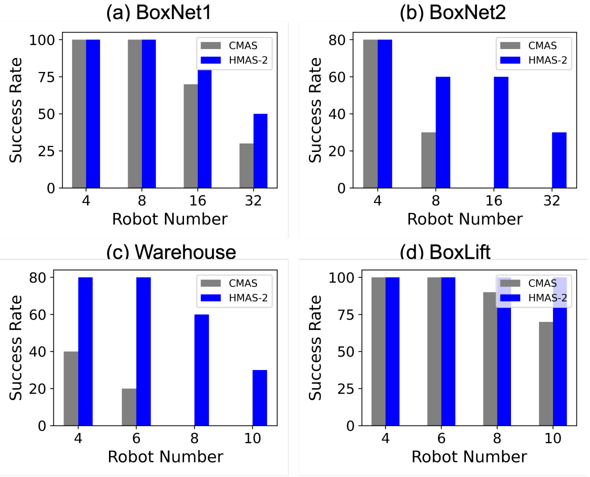


Fig. 6. Success rate vs. robot number for CMAS and HMAS-2 methods in four testing environments.

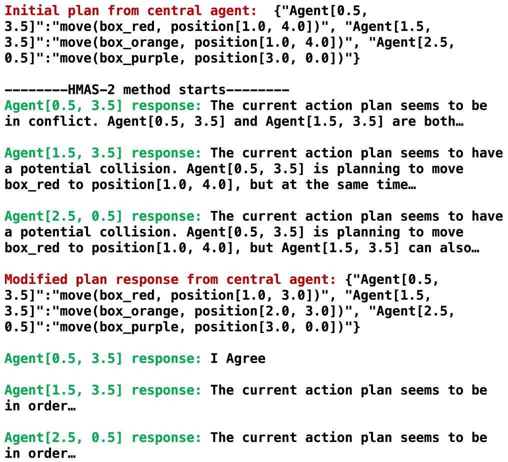
in plans, such as identifying actions that would result in a collision. Figure 7 shows one example of dialogue correcting the flawed plan via feedback from local LLM agents.

Step History Method In Table I, we report the results of HMAS-2 for different step history prompts, as described in Section III-C. The framework performs much worse when provided no history of prior actions or dialogue rounds than when provided with the state-action pair history; this is consistent with our intution that the past actions provide useful information for future decisions. The framework performs a bit worse when provided the full history than when provided with the state-action pair history. We hypothesize that this is a result of context dilution [12] from long dialogue histories.

GPT-3 Performance A common trend among pre-trained LLM evaluations is that some capabilities do not emerge until a model reaches sufficient size or is trained on sufficient amounts of quality data [9]. We report the results of CMAS and HMAS-2 when using GPT-3 as the LLM and find that it performs significantly worse than GPT-4. It is a useful reminder that the quality of LLM-based planners depends on the quality and capability of the underlying LLM.

D. 3D Simulation

In addition to the 2D scenarios for our experiments, we also perform experiments in a 3D environment simulated using Pybullet [13], as illustrated in Figure 8. The task and environment are similar to BoxNet1 and BoxNet2. The environment consists of colored boxes, colored bowls, and robot arms. The goal is to move each colored box into its associated bowl of the same color in the fewest actions. Each arm is immobile and confined to actions within its workspace (indicated by the dotted blue lines). Arms can only pick and place boxes that are within its workspace or on the border.



```
Initial plan from central agent:  {"Agent[0.5,
3.5]":"move(box_red, position[1.0, 4.0])", "Agent[1.5,
3.5]":"move(box_orange, position[1.0, 4.0])", "Agent[2.5,
0.5]":"move(box_purple, position[3.0, 0.0])"}

--------HMAS-2 method starts--------
Agent[0.5, 3.5] response: The current action plan seems to be
in conflict. Agent[0.5, 3.5] and Agent[1.5, 3.5] are both…

Agent[1.5, 3.5] response: The current action plan seems to have
a potential collision. Agent[0.5, 3.5] is planning to move
box_red to position[1.0, 4.0], but at the same time…

Agent[2.5, 0.5] response: The current action plan seems to have
a potential collision. Agent[0.5, 3.5] is planning to move
box_red to position[1.0, 4.0], but Agent[1.5, 3.5] can also…

Modified plan response from central agent: {"Agent[0.5,
3.5]":"move(box_red, position[1.0, 3.0])", "Agent[1.5,
3.5]":"move(box_orange, position[2.0, 3.0])", "Agent[2.5,
0.5]":"move(box_purple, position[3.0, 0.0])"}

Agent[0.5, 3.5] response: I Agree

Agent[1.5, 3.5] response: The current action plan seems to be
in order…

Agent[2.5, 0.5] response: The current action plan seems to be
in order…
```

Fig. 7. Simplified communication example of HMAS-2. The local agents detect the collision risk and report it to the central agent.

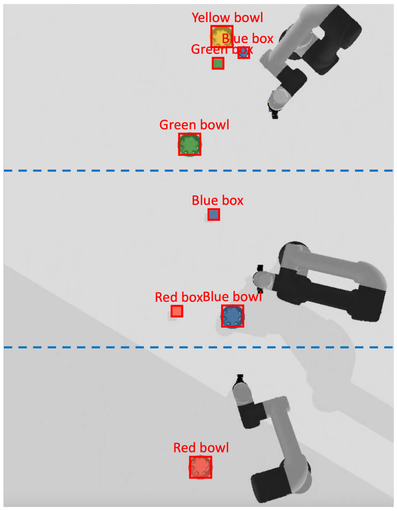


Fig. 8. 3D simulation environments: robot arms collaborate to move all the boxes into the same colored bowls. Each robot arm has a limited workspace and can only move within its assigned region (divided by the blue lines).

Pick and place actions are executed via pre-defined motion primitives. Unlike BoxNet2, boxes can be placed anywhere that is reachable along the boundary, so collisions are possible but unlikely. We again test the scalability to more agents and instantiate the environment with either three or six arms (each has its own workspace). The 3D environment has an additional complexity of using an image-to-text model (ViLD [14]) to provide bounding boxes and text descriptions for each object. Further, the 3D simulation has a richer environment model that permits action execution errors due to dynamical factors (e.g., a box slips out of a gripper) that require re-planning. The iterative nature of the LLM-based planning frameworks in this work naturally handles such instances of replanning.

We do ten runs for each scenario. Table II shows the results of the experiments with three and six agents, respectively. Both CMAS and HMAS-2 achieve 100% success rates. CMAS used more action steps than HMAS-2 in the six robot situation, consistent with our results from the 2D environments that CMAS performs worse than HMAS-2 in more complex tasks.

TABLE II

EVALUATION RESULTS ON 3D SIMULATIONS. WE REPORT AVERAGE VALUES OVER 10 RUNS. REPORTED NUMBERS OF ACTION STEPS, NUMBER OF API CALLS, AND NUMBER OF TOKENS ARE NORMALIZED VALUES.

	Three Robots				Six Robots			
	Success	Steps	API Calls	Tokens	Success	Steps	API Calls	Tokens
CMAS	100%	1.0	1.0	1.0	100%	1.39	1.0	1.18
HMAS-2	100%	1.03	3.45	3.28	100%	1.03	2.65	1.31

V. RELATED WORK

LLMs for Robotics A representative set of prior work [15], [6], [16], [7], [17] uses LLMs to select actions from pre-defined skill primitives and complete the tasks step by step with texts or codes as the intermediate, such as SayCan [8], Inner Monologue [18], Code-As-Policy [19], and ProgGPT [20]. Regarding to connect task planning and motion planning, prior work such as Text2Motion [21] and AutoTAMP [2] studies integrating LLMs with traditional Task and Motion Planners. Other work explores querying LLMs to output rewards of robot actions so that independent reward-based planners can be connected. The reward formats can be real values [22], [23], temporal logics [24], [2], [25], or patterns [26]. The recent two work [10], [11] firstly extend LLMs into multi-robot situations, while the robot number is limited to two or three and the scalability of frameworks and step history approaches is not considered. A recent work [27] considers the scalability of LLM-based single-robot planning in broader environments with more objects.

Dialogues and Debates of LLMs Outside the Robotics domain, LLM-based multi-agent discussion has shown impressive capability to promote the research in social behaviors [28], [23], dialogue-based games[29], and software development [30]. Recent work shows that discussion among multiple LLM agents can improve factuality and accuracy[31], [32]. Prior work focuses more on understanding LLM behaviors or improving the solution for a single question.

Multi-Robot Collaboration Multi-robot collaboration has been extensively studied many decades, especially on multi-arm and multi-drone motion planning [33], [34], [35]. The traditional methods rely on sampling-based methods for trajectory generation [36], or formal methods to optimize Task and Motion Planning [37], [3], [35]. Recent work also explored learning-based methods as alternatives [38], [39].

VI. CONCLUSION

Our work considers the scalability of LLM-based multi-robot task planning for long-horizon tasks to systems with many robots with heterogeneous capabilities. We propose several new frameworks for collaborative LLM dialogue and find that hybrid approaches with both central and local LLM planners produce the most successful plans and scale best to large number of agents. Future work can explore more complex tasks with more hierarchical frameworks of robot groups, e.g., each agent for each specialized robot sub-group.

REFERENCES

[1] W. Liu, K. Leahy, Z. Serlin, and C. Belta, “Robust multi-agent coordination from catl+ specifications,” in *2023 American Control Conference (ACC)*. IEEE, 2023, pp. 3529–3534.

[2] Y. Chen, J. Arkin, Y. Zhang, N. Roy, and C. Fan, “Autotamp: Autoregressive task and motion planning with llms as translators and checkers,” *arXiv preprint arXiv:2306.06531*, 2023.

[3] M. Cavorsi, B. Capelli, L. Sabattini, and S. Gil, “Multi-robot adversarial resilience using control barrier functions,” in *Robotics: Science and Systems*, 2022.

[4] F. Zhang, C. Jia, Y.-C. Li, L. Yuan, Y. Yu, and Z. Zhang, “Discovering generalizable multi-agent coordination skills from multi-task offline data,” in *The Eleventh International Conference on Learning Representations*, 2022.

[5] M. Samvelyan, T. Rashid, C. S. De Witt, G. Farquhar, N. Nardelli, T. G. Rudner, C.-M. Hung, P. H. Torr, J. Foerster, and S. Whiteson, “The starcraft multi-agent challenge,” *arXiv preprint arXiv:1902.04043*, 2019.

[6] 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.

[7] 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

[8] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, *et al.*, “Do as i can, not as i say: Grounding language in robotic affordances,” *arXiv preprint arXiv:2204.01691*, 2022.

[9] 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.

[10] H. Zhang, W. Du, J. Shan, Q. Zhou, Y. Du, J. B. Tenenbaum, T. Shu, and C. Gan, “Building cooperative embodied agents modularly with large language models,” *arXiv preprint arXiv:2307.02485*, 2023.

[11] Z. Mandi, S. Jain, and S. Song, “Roco: Dialectic multi-robot collaboration with large language models,” *arXiv preprint arXiv:2307.04738*, 2023.

[12] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang, “Lost in the middle: How language models use long contexts,” *arXiv preprint arXiv:2307.03172*, 2023.

[13] E. Coumans and Y. Bai, “Pybullet, a python module for physics simulation for games, robotics and machine learning,” http://pybullet.org, 2016–2021.

[14] X. Gu, T.-Y. Lin, W. Kuo, and Y. Cui, “Open-vocabulary object detection via vision and language knowledge distillation,” *arXiv preprint arXiv:2104.13921*, 2021.

[15] N. Wake, A. Kanehira, K. Sasabuchi, J. Takamatsu, and K. Ikeuchi, “Chatgpt empowered long-step robot control in various environments: A case application,” *arXiv preprint arXiv:2304.03893*, 2023.

[16] M. Skreta, N. Yoshikawa, S. Arellano-Rubach, Z. Ji, L. B. Kristensen, K. Darvish, A. Aspuru-Guzik, F. Shkurti, and A. Garg, “Errors are useful prompts: Instruction guided task programming with verifier-assisted iterative prompting,” *arXiv preprint arXiv:2303.14100*, 2023.

[17] L. Guan, K. Valmeekam, S. Sreedharan, and S. Kambhampati, “Leveraging pre-trained large language models to construct and utilize world models for model-based task planning,” *arXiv preprint arXiv:2305.14909*, 2023.

[18] 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,” *arXiv preprint arXiv:2207.05608*, 2022.

[19] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, “Code as policies: Language model programs for embodied control,” *arXiv preprint arXiv:2209.07753*, 2022.

[20] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, “ProgPrompt: Generating situated robot task plans using large language models,” in *International Conference on Robotics and Automation (ICRA)*, 2023. [Online]. Available: https://arxiv.org/abs/2209.11302

[21] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg, “Text2motion: From natural language instructions to feasible plans,” *arXiv preprint arXiv:2303.12153*, 2023.

[22] W. Yu, N. Gileadi, C. Fu, S. Kirmani, K.-H. Lee, M. G. Arenas, H.-T. L. Chiang, T. Erez, L. Hasenclever, J. Humplik, *et al.*, “Language to rewards for robotic skill synthesis,” *arXiv preprint arXiv:2306.08647*, 2023.

[23] S. Tan, B. Ivanovic, X. Weng, M. Pavone, and P. Kraehenbuehl, “Language conditioned traffic generation,” *arXiv preprint arXiv:2307.07947*, 2023.

[24] Y. Chen, R. Gandhi, Y. Zhang, and C. Fan, “Nl2tl: Transforming natural languages to temporal logics using large language models,” *arXiv preprint arXiv:2305.07766*, 2023.

[25] J. X. Liu, Z. Yang, B. Schornstein, S. Liang, I. Idrees, S. Tellex, and A. Shah, “Lang2LTL: Translating natural language commands to temporal specification with large language models,” in *Workshop on Language and Robotics at CoRL 2022*, 2022. [Online]. Available: https://openreview.net/forum?id=VxfjGZzrdn

[26] S. Mirchandani, F. Xia, P. Florence, B. Ichter, D. Driess, M. G. Arenas, K. Rao, D. Sadigh, and A. Zeng, “Large language models as general pattern machines,” *arXiv preprint arXiv:2307.04721*, 2023.

[27] K. Rana, J. Haviland, S. Garg, J. Abou-Chakra, I. Reid, and N. Suenderhauf, “Sayplan: Grounding large language models using 3d scene graphs for scalable task planning,” *arXiv preprint arXiv:2307.06135*, 2023.

[28] G. Li, H. A. A. K. Hammoud, H. Itani, D. Khizbullin, and B. Ghanem, “Camel: Communicative agents for" mind" exploration of large scale language model society,” *arXiv preprint arXiv:2303.17760*, 2023.

[29] D. Schlangen, “Dialogue games for benchmarking language understanding: Motivation, taxonomy, strategy,” *arXiv preprint arXiv:2304.07007*, 2023.

[30] C. Qian, X. Cong, C. Yang, W. Chen, Y. Su, J. Xu, Z. Liu, and M. Sun, “Communicative agents for software development,” *arXiv preprint arXiv:2307.07924*, 2023.

[31] Y. Du, S. Li, A. Torralba, J. B. Tenenbaum, and I. Mordatch, “Improving factuality and reasoning in language models through multiagent debate,” *arXiv preprint arXiv:2305.14325*, 2023.

[32] Z. Wang, S. Mao, W. Wu, T. Ge, F. Wei, and H. Ji, “Unleashing cognitive synergy in large language models: A task-solving agent through multi-persona self-collaboration,” *arXiv preprint arXiv:2307.05300*, 2023.

[33] Y. Koga and J.-C. Latombe, “On multi-arm manipulation planning,” in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*. IEEE, 1994, pp. 945–952.

[34] B. Williams, “Multi-agent path finding for precedence-constrained goal sequences,” in *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2022.

[35] V. N. Hartmann, A. Orthey, D. Driess, O. S. Oguz, and M. Toussaint, “Long-horizon multi-robot rearrangement planning for construction assembly,” *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 239–252, 2022.

[36] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning,” *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[37] Z. Liu, M. Guo, and Z. Li, “Time minimization and online synchronization for multi-agent systems under collaborative temporal tasks,” *arXiv preprint arXiv:2208.07756*, 2022.

[38] C.-C. Wong, S.-Y. Chien, H.-M. Feng, and H. Aoyama, “Motion planning for dual-arm robot based on soft actor-critic,” *IEEE Access*, vol. 9, pp. 26 871–26 885, 2021.

[39] H. Ha, J. Xu, and S. Song, “Learning a decentralized multi-arm motion planner,” *arXiv preprint arXiv:2011.02608*, 2020.