

PIP-LLM: Integrating PDDL-Integer Programming with LLMs for Coordinating Multi-Robot Teams Using Natural Language

Guangyao Shi^{1†}, Yuwei Wu², Vijay Kumar², Gaurav S. Sukhatme¹

Abstract—Enabling robot teams to execute natural language commands requires translating high-level instructions into feasible, efficient multi-robot plans. While Large Language Models (LLMs) combined with Planning Domain Description Language (PDDL) offer promise for single-robot scenarios, existing approaches struggle with multi-robot coordination due to brittle task decomposition, poor scalability, and low coordination efficiency. We introduce PIP-LLM, a language-based coordination framework that consists of PDDL-based team-level planning and Integer Programming (IP) based robot-level planning. PIP-LLMs first decomposes the command by translating the command into a team-level PDDL problem and solves it to obtain a team-level plan, abstracting away robot assignment. Each team-level action represents a subtask to be finished by the team. Next, this plan is translated into a dependency graph representing the subtasks’ dependency structure. Such a dependency graph is then used to guide the robot-level planning, in which each subtask node will be formulated as an IP-based task allocation problem, explicitly optimizing travel costs and workload while respecting robot capabilities and user-defined constraints. This separation of planning from assignment allows PIP-LLM to avoid the pitfalls of syntax-based decomposition and scale to larger teams. Experiments across diverse tasks show that PIP-LLM improves plan success rate, reduces maximum and average travel costs, and achieves better load balancing compared to state-of-the-art baselines.

I. INTRODUCTION

A long-standing goal in multi-robot research is to create systems that take concise human language commands and automatically generate executable coordination plans for robot teams. Achieving this level of autonomy requires accurate interpretation of human commands, decomposition of these commands into executable subtasks, and effective allocation of tasks across robots [1]. Recent advances in large language models (LLMs) offer a promising path toward this objective.

LLMs, trained on large-scale text corpora, exhibit common-sense reasoning abilities and can process a wide range of tasks expressed in natural language. They have achieved notable success in domains such as robotic manipulation [2, 3], navigation [4], and locomotion [5]. Some studies have directly employed LLMs as task planners, showing moderate success in single-robot sequencing problems. However, in more complex scenarios, LLMs alone often produce suboptimal or infeasible solutions [6, 7], which is especially problematic in multi-robot coordination

where feasibility and optimality are equally critical. Consequently, many recent works integrate LLMs with classical planning [8], for example, by translating natural language (NL) commands into Planning Domain Description Language (PDDL) and leveraging existing solvers [9, 10]. Yet, most of these approaches remain limited to single-robot domains, and extending them to multi-robot contexts introduces additional challenges. First, formulating multi-robot problems into a single structured representation is significantly more complex, and LLMs frequently produce erroneous or incomplete encodings (“hallucinations” [11]–[15]) when handling large teams and complex tasks. Second, even with correct formulations, standard solvers for PDDL often fail to scale to the large state and action spaces of multi-robot planning. Third, numeric objectives and constraints (e.g., energy minimization or resource-constrained task allocation) are difficult to specify in PDDL, and the corresponding solvers remain underdeveloped.

Several efforts have attempted to extend the LLM+classic planning paradigm to multi-robot systems. Bai et al. [16] propose decomposing PDDL goals into robot-specific subgoals to decouple planning across robots. However, their method does not generalize well to teams with more than two robots or scenarios where decomposition is non-trivial. Zhang et al. [17] introduce LaMMA-P, which extends the architecture proposed by Kannan et al. [6] by incorporating PDDL planning for each robot. Both frameworks rely heavily on pure syntactic segmentation and analysis of natural language commands for task decomposition, which is inherently unreliable. For instance, suppose the command is “put five apples in the fridge”, the naive decomposition based on syntactic segmentation and analysis is to decompose the task into five subtasks, each of which is to put an apple in the fridge. However, some robots may be capable of moving two apples at a time, and some robots move one. How to decompose the task depends not only on textual information but also on the robots’ capabilities and their availability. Similarly, a command such as “put all fruits and vegetables on the table into fridges 1 and 2” requires contextual reasoning about object quantities and container capacities, which cannot be captured by naive language segmentation. These limitations hinder their applicability to large-scale, heterogeneous, or long-horizon tasks. Moreover, their frameworks implicitly assume a special case of multi-robot task allocation (MRTA) [18] where the number of subtasks is no greater than the number of robots and allocation is determined solely by skill requirements without

¹Guangyao Shi and Gaurav S. Sukhatme are with the Department of Computer Science, University of Southern California, Los Angeles, CA 90089, USA. Email: {shig, gaurav}@usc.edu.

²Yuwei Wu and Vijay Kumar are with the GRASP Lab, University of Pennsylvania, Philadelphia, PA 19104, USA. Email: {yuweiwu, kumar}@seas.upenn.edu.

† Corresponding author.

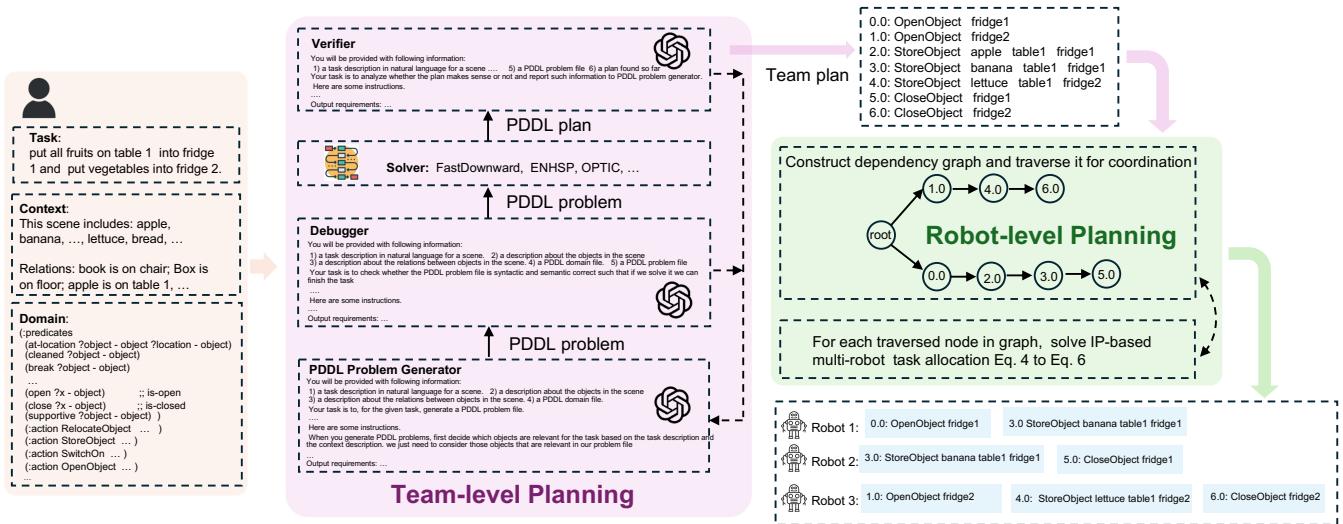


Fig. 1. Overview of PIP-LLM. Users provide the task, context, and PDDL domain description in natural language (orange box). Then, LLMs will analyze the relevant information to construct a PDDL problem instance (*PDDL Problem Generator*). Next, the result is fed to a *Verifier*, which will check the syntax of PDDL problems and the correctness of the solution and feed the information back to the problem generation LLM. The generation process may be repeated several rounds. Each action in the output plan corresponds to a subtask that the robots as a team can do. Then, PIP-LLM uses this plan to construct a dependency graph to identify the dependencies between subtasks and facilitate parallel execution, which will then be used to allocate and schedule robots to finish all sub-tasks (green box, *Robot-level Planning*).

explicitly considering task costs (e.g., travel distance) and the transition costs between tasks. Generally, however, the number of subtasks may exceed the number of robots in MRTA; tasks may vary in duration; and robots may need to transit between multiple tasks. Without explicitly considering task costs, solutions can be highly suboptimal; without proper coordination, workload imbalances lead to large travel costs of a particular robot, i.e., some robots are busy all the time while others keep idling.

To address these challenges, we propose a novel framework, **PIP-LLM**, for language-driven multi-robot coordination (Fig. 1). The key idea is to form a hierarchical planning structure: using LLM+PDDL for team-level task understanding and then using Integer Programming (IP) for robot assignments. Specifically, given a natural language command, our framework first formulates a team-level PDDL problem that captures what the robot team as a whole must accomplish, independent of specific robot assignments. The resulting plan is then represented as a dependency graph by analyzing action dependencies, which will then be used to schedule the robotic team. Unlike prior work that relies on syntactic segmentation, this approach provides semantically grounded and formal task decomposition by formulating and solving PDDL problems. In the second stage, a skill-based allocation framework using IP assigns tasks to robots based on their capabilities and task costs, ensuring both feasibility and efficiency. This two-level design enables the framework to robustly handle complex commands, scale efficiently to large numbers of robots and objects, and achieve more balanced and near-optimal execution. Furthermore, user preferences can be seamlessly incorporated into task allocation; for example, constraints expressed in natural language, such as excluding a particular robot from certain tasks.

The main contributions of this paper are as follows:

- We propose a novel multi-robot task planning framework that integrates PDDL-based team-level planning with IP-based robot-level allocation, enabling flexible and efficient coordination of robot teams using natural language commands.
- We introduce a benchmark dataset for evaluating multi-robot task planning in large-scale, long-horizon scenarios involving many robots and objects.
- We provide extensive simulation results and demonstrate real-world applications to validate the effectiveness and scalability of the proposed framework.

II. RELATED WORK

Recent advances have demonstrated the potential of integrating LLMs with individual robotic platforms to enable natural human–robot interaction and high-level task planning. By leveraging the strengths of LLMs in language understanding and contextual reasoning, these systems translate free-form instructions into executable action sequences. Because natural language is inherently contextual and semantic, LLMs are well-suited for navigation, exploration, and instruction following; in such settings, they help ground linguistic inputs into spatial and semantic representations that guide behavior in complex, partially observed environments [19]–[21]. In addition, modules for real-time residency [22] and anomaly detection [23] have been integrated into LLM-based robotic pipelines to improve robustness in dynamic and unforeseen conditions.

Built on this, researchers are increasingly exploring how LLMs can facilitate coordination for multi-robot teams [6, 7, 24]–[27]. Some pioneering works seek to design prompt frameworks to solve multi-robot coordination problems using mainly LLMs [6, 7]. Zhao et al. [7] proposed a dialogue-based framework to coordinate multiple robots. Such ap-

proaches are not suitable to solve long-horizon and large-scale multi-robot problems because such problems will require robots to communicate many rounds with redundant (possibly misleading as the number of rounds increases) information. LLMs tend to get lost and hallucinate. Chen et al. [25] did an extensive evaluation of communication frameworks (centralized, decentralized, or hybrid) to find a scalable solution for multi-robot coordination. Another representative work along this direction is [6], in which the authors propose a framework for multi-robot task planning based on the canonical decomposition of multi-robot task planning, i.e., task decomposition, coalition formation, and task allocation [1]. Such a framework does not use any external tools and purely relies on LLMs’ reasoning capability to solve problems. However, it cannot be used to coordinate large-scale robot teams, and it does not consider the optimality aspects of the coordination.

Another line of work uses LLMs to translate natural language into structured intermediate representations [17, 24, 28, 29]. Xu et al. [24] use hierarchical temporal logic as an intermediate representation for temporally dependent multi-robot task planning. Zhang et al. [17] extend [6] by adding a PDDL representation for each robot’s individual planning. Our work aligns with this direction. The proposed framework adopts a hierarchical structure that employs two distinct representations at different levels: PDDL for team-level planning and IP for robot-level planning. Our framework can cover both the feasibility and optimality aspects of multi-robot coordination and is very suitable for long-horizon tasks that involve a large number of robots.

III. PROBLEM FORMULATION

A. Preliminaries

PDDL: As a standard, declarative language for specifying automated planning tasks [30, 31], it separates the domain, which specifies types, predicates, and parameterized actions with preconditions and effects, from the problem, which enumerates concrete objects, an initial state, and goals. Beyond the basic STRIPS subset, PDDL supports richer constructs such as conditional and quantified effects, numeric fluents, and durative actions.

Robot Model: We consider a set of N (possibly) heterogeneous robots, $\mathcal{R} = \{R^1, R^2, \dots, R^N\}$. Let Δ denote the set of all skills or actions that a robot may be capable of performing. We assume that skills in Δ are either pre-implemented in the system or accessible through available API calls. Each agent R^n possesses a subset of skills, $S^n \subseteq \Delta$, subject to specific constraints. For example, the robot skill `PickUpObject` may be limited by the maximum mass that the robot can lift.

Environment Model: We assume a static and deterministic environment. The state of the environment can be explicitly represented. The number of objects and robots is fixed, and the robot’s action will have a deterministic effect on the state of the environment.

B. Problem Statement

Given a high-level natural-language instruction I , our aim is to parse the instruction, decompose it into the requisite subtasks, and synthesize an executable plan. Subtasks are scheduled to maximize robot utilization, enabling parallel execution when precedence and resource constraints allow. Environment E contains multiple robots and objects, and I is assumed to be feasible within E .

IV. PROPOSED FRAMEWORK

A. Overview

The proposed framework is shown in Fig. 1, which hierarchically conducts task planning for multi-robot systems.

Team-Level Planning. Given a task description in natural language, LLMs will first analyze the context information, relevant predicates, and goals that need to be achieved. Based on this analysis, it will construct a PDDL problem instance. Then, the problem instance will be fed to a Debugger and a *Verifier*, which will check the syntax of PDDL problems and the correctness of the solution and feed the information back to the problem generation LLM. The generation process will be repeated until the set loop limit is reached. Each action in the output plan corresponds to a subtask that the robots as a team can do. Then, we will use this plan to construct a dependency graph, which helps us to identify the dependencies between subtasks and facilitate parallel execution. Details on how to construct such task trees will be discussed in Sec. IV-B. The team-level planning part of PIP-LLM is, in essence, doing task decomposition. But it is fundamentally different from those in [6, 17] which rely on syntactic language segmentation to decompose tasks. Instead, our framework uses LLMs to construct a formal high-level plan and represent tasks as a structured format, a dependency tree. It should be noted that our team-level PDDL planning problem is different from the robot-specific PDDL planning problem [16, 17] in the sense that the action is about the team as a whole can without referring to any specific robots. Such an abstract reduces the information fed to LLMs and allows them to do high-level reasoning more accurately.

Algorithm 1: Dependency Graph Scheduling

```

Input : A subtask dependency graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ 
Output: Ordered robot actions
1 create empty stack  $H$  # stack for BFS frontier
2  $H.\text{insert}(root)$ 
3 while schedule is triggered and  $H$  is not empty do
4   current_subtasks  $\leftarrow H.\text{next\_task}()$ 
5    $\text{sol} \leftarrow \text{skill\_MRTA}(\text{current\_subtasks})$ 
6   allocate subtasks to robots using  $\text{sol}$ 
7   add child nodes of current_subtasks in  $\mathcal{G}$  to  $H$ 
8 end

```

Robot-Level Planning. The constructed dependency graph will then be used to allocate and schedule robots to finish all sub-tasks. Specifically, we will traverse the task tree in

a Breadth-First-Search (BFS) fashion. We maintain a BFS frontier in a stack. The task nodes in this frontier can be executed in parallel. When we need to allocate robots for the task, we pop one node from the frontier and update the stack. More details are given in Algo. 1. Whenever we traverse a node in the dependency graph, we will solve a skill-based MRTA problem to allocate the subtask to specific robots to finish. Details will be discussed in Sec. IV-C. Moreover, we may incorporate human preference constraints into the MRTA formulation to accommodate humans' extra requirements of robots.

Combining both team-level planning and robot-level planning, as we will show in the experiments, such a hierarchy planning structure in our framework can not only achieve a high success rate in fulfilling diverse tasks, but also improve the overall coordination efficiency of the multi-robot systems compared to baselines.

B. From Team-Level Plan to Subtask Dependency Graph

The output from the PDDL planner is a sequence of team actions. If we execute the actions in order, we can finish the tasks. However, it is not necessary to strictly follow the order because some actions in the plan may be parallelizable. An example is shown in Fig. 1. Given the domain, we can get a team plan as shown at the upper right of Fig. 1. It should be noted that for the first action (`OpenObject fridge1`) in the plan, its effect will not affect any preconditions in the next action (`OpenObject fridge2`). Similarly, there is no conflict in preconditions and effects between `StoreObject apple table1 fridge1` and `StoreObject lettuce table1 fridge2`. In such a case, we can actually execute these two actions in parallel. Inspired by this, we propose a simple action parallelization algorithm as shown in Algo. 2 to parallelize the team actions obtained from our team-level planning. The resulting dependency graph of the plan is also in the robot-level planning block of Fig. 1.

C. Skill-based Multi-Robot Task Allocation

We consider a team of N (potentially heterogeneous) robots, where each robot is characterized by a vector of continuous capability *skills* [32]. The skills of robot i are

$$q^{(i)} = [q_1^{(i)}, q_2^{(i)}, \dots, q_U^{(i)}] \in \mathbb{R}_{\geq 0}^U, \quad (1)$$

where $q_u^{(i)} \in \mathbb{R}_{\geq 0}$ denotes the level of the u -th skill. If robot i lacks skill u (e.g., a car-like robot cannot open a refrigerator door), then $q_u^{(i)} = 0$; otherwise $q_u^{(i)} > 0$ quantifies its capacity. For example, if a robot can pick up 2 units of objects, its pickup skill is 2.0. Stacking the robot skill vectors gives the team skill matrix $\mathbf{Q} = [q^{(1)^T}, \dots, q^{(N)^T}] \in \mathbb{R}_+^{U \times N}$ whose i -th column and u -th row corresponds to robot i and skill u , respectively.

Each task is specified by the skill requirements needed to complete it. For example, the task `put a lunch box in the fridge` may require the skills `open_door`: 1.0, `close_door`: 1.0, `pick_up`: 1.2, and `drop`: 1.2, where 1.2 encodes a weight affordance. Formally, the desired task

Algorithm 2: Dependency Graph Generation

Input :

- A PDDL domain file, a problem file
- A plan file \mathcal{P} as an ordered list

Output: A subtask dependency graph \mathcal{G}

```

1  $\mathcal{G} \leftarrow$  root node
2 for action_index, action in  $\mathcal{P}$  do
3   # trace back to find a proper parent node
4   for  $i$  in range(action_index-1, 1, -1) do
5     flag  $\leftarrow$  Check_dependency(action,  $\mathcal{P}[i]$ )
6     if flag and no parent node added then
7       add action as child node of  $\mathcal{P}[i]$  in  $\mathcal{G}$ 
8       break
9     end
10   end
11   if no parent node added then
12     add action as child node of root of  $\mathcal{G}$ 
13   end
14 end

```

skills are $\mathbf{Y} = [y_1, y_2, \dots, y_U]^T \in \mathbb{R}_{\geq 0}^U$, with $y_u = 0$ if skill u is unnecessary and $y_u > 0$ otherwise.

Assigning robot i to a task incurs a cost $c_i \in \mathbb{R}_{\geq 0}$; in this work, we take c_i to be the total travel distance of robot i . Given the task skill requirement, robot skill matrix, and the cost, we will solve the following lexicographic multi-objective IP problem to decide robot task allocation,

$$\min \quad \{\max_i c_i x_i, \sum_i c_i x_i\} \quad (2a)$$

$$\text{s.t. } \mathbf{Qx} \geq \mathbf{Y}, \quad x_i \in \{0, 1\}, \quad (2b)$$

$$\text{customized constraints,} \quad (2c)$$

where x_i is a binary variable: $x_i = 1$ if robot i is assigned to the task, otherwise 0; $\mathbf{x} = [x_1, x_2, \dots, x_N]^T$. The first objective $\max_i c_i x_i$ denotes the maximum cost of the team, and the second objective $\sum_i c_i x_i$ denotes the sum of the team cost. Eq. (2c) denotes other domain-specific constraints or preference constraints specified by users (an example is given in Tab. III, ID=5). Such problems can be solved using existing solvers like Gurobi.

V. EXPERIMENTS

We conducted two types of simulation experiments to evaluate the performance of the proposed PIP-LLM framework. In the first type of experiment, we use the AI2THOR simulator [33] as [6, 17], which is designed for domestic tasks. The goal of this type of experiment is to evaluate whether the proposed framework can improve the team coordination efficiency in terms of travel costs while not compromising the success rate. In the second type of experiment, we used a customized warehouse environment in Gazebo, which involved coordinating many robots to rearrange many objects. The goal of the second type of experiment is to show the scalability and flexibility of the

proposed framework in coordinating a large number of robots with concise language commands. Moreover, we validate our results with a hardware demonstration.

A. AI2THOR Benchmark Dataset

To evaluate the performance of PIP-LLM and facilitate comparisons with baseline methods, we created a benchmark dataset by inheriting and modifying those from [6, 17]. Specifically, the dataset consists of three categories of tasks to better evaluate the performance of the proposed framework with an increasing level of complexity.

- **Compound Tasks** involve multiple objects and can be decomposed into sequential or parallel subtasks. The robot team is homogeneous, and each robot has the necessary skills to finish all tasks. We need to properly allocate and schedule tasks among robots. For example, Slice the lettuce, trash the mug and switch off the light.
- **Complex Tasks** are intended for heterogeneous robot teams. It resembles Compound Tasks in their characteristics, like task decomposition, multi-robot engagement, and the presence of multiple objects. However, unlike Compound Tasks, where individual robots can perform subtasks independently, robots may need to perform a subtask cooperatively due to limitations in their skills, e.g., a robot does not have the skill to open the fridge/microwave; a robot can only afford one unit of weight and cannot move an object of two units of weight¹. We create a set of such tasks by either setting the robots' corresponding skill to zero or increasing the weight of objects. Such tasks require strategic team coordination for effective task completion.
- **Vague Tasks** These tasks present additional challenges with ambiguous natural language instructions, which require the robots to infer missing details. For example, gather up 3 school supplies on the bed; It is very bright and we do not need light.

Our dataset includes 41 compound tasks, 23 complex tasks, and 25 vague command tasks to evaluate task decomposition, allocation, and execution efficiency.

B. Evaluation Baselines and Metrics

We use the following evaluation metrics: Success Rate (SR), Robot Utilization (RU), Goal Condition Recall (GCR), Executability (Exe), Sum of Travel Cost (TC-sum), Max of Travel Cost (TC-max), following [6, 17, 34]. Our evaluations are based on the dataset's final ground truth demonstrated by humans.

- *Exe* is the fraction of actions in the task plan that can be executed, regardless of their impact on task completion.

¹AI2THOR does not support cooperative manipulation. When dealing with such tasks, we manually inspect the plan and cost output from frameworks, but do not visualize plans in the simulator.

- *RU* measures the planning efficiency of the action sequence by comparing the total transitions from all successful executions to the overall corresponding ground truth count.
- *GCR* is computed using the set difference between ground truth goal conditions and final conditions achieved, divided by the total number of ground truth goal conditions.
- *SR* is calculated as the ratio of successful executions to the total number of tasks.
- *TC-avg* is the average travel cost for robots to finish all the tasks. It is a metric for the coordination efficiency of the team.
- *TC-max* is the maximum total travel cost for robots to finish all the tasks. It is a metric for the team's load-balancing efficiency.

We use SMART-LLM [6], LaMMA-P [17], and a basic Chain of Thought (CoT) as our baselines using GPT-4o. We evaluate our framework PIP-LLM using both GPT-4o and Qwen-32B.

C. Results for AI2-THOR Environment

1) *Qualitative results:* An illustrative example is shown in Fig. 2. The task description is given in the caption. The first row presents screenshots of ours during the task execution, and the second row corresponds to baselines. The travel cost of each robot is shown in Fig. 2d and 2h, respectively. As we can see from the figures, our approach, by using object information and robot cost to optimize the task allocation and scheduling, results in a smaller travel cost (shorter paths for each robot), i.e., better coordination efficiency.

2) *Quantitative results:* The main statistical results are shown in Tab. I. Overall, in all GPT-4o-based frameworks, our framework shows significant improvement in travel cost-related metrics (TC-sum and TC-max) compared to the state-of-the-art baselines and shows a substantial increase in the success rate, especially in the vague tasks group. Such results suggest that our framework can not only improve the team coordination efficiency but also augment LLMs' capability to understand and analyze the tasks. We attribute such improvements to the following reasons. First, our framework introduces a hierarchical structure: team first, then robots. On the team level, LLMs will be guided by the PDDL domain and do not need to go into details of each robot, thus being less likely to be distracted by redundant information to hallucinate. Besides, we introduce the debugger and checker structure to allow the LLMs to correct themselves if the first round of generation is not right. Second, on the robot level, we incorporate an explicit cost-aware task allocation and scheduling structure, which will inherently improve the coordination efficiency. Tab. I also shows the result of using Qwen-32B for our framework. In such a case, the performance of our framework, in terms of both success rate and travel cost, remains significantly better than the naive CoT framework with GPT-4o. These results demonstrate the effectiveness of the proposed framework.

TABLE I
EVALUATION OF PIP-LLM AND BASELINES IN THE AI2-THOR SIMULATOR.

Methods	Compound					Complex					Vague				
	SR ↑	TC max/avg ↓	GCR ↑	RU ↑	Exe ↑	SR ↑	TC max/avg ↓	GCR ↑	RU ↑	Exe ↑	SR ↑	TC max/avg ↓	GCR ↑	RU ↑	Exe ↑
CoT (GPT-4o)	0.37	6.5/5.8	0.42	0.72	0.67	0.00	—	0.16	0.00	0.45	0.00	—	0.00	0.00	0.00
SMART-LLM (GPT-4o)	0.71	6.3/5.3	0.82	0.78	0.93	0.26	7.3/6.0	0.39	0.63	0.75	0.00	—	0.12	0.00	0.67
LaMMA-P (GPT-4o)	0.93	6.3/5.0	0.94	0.91	0.92	0.78	7.0/5.8	0.88	0.87	1.00	0.52	7.0/5.5	0.56	0.71	0.91
PIP-LLM (ours, GPT-4o)	0.93	5.3/4.3	0.95	1.00	0.87	5.8/4.5	0.92	0.90	1.00	0.68	6.0/4.8	0.77	0.82	0.91	
PIP-LLM (ours, Qwen-32B)	0.63	5.8/4.8	0.72	1.00	0.93	0.48	6.5/5.5	0.45	0.81	0.64	0.40	0.14/0.14	0.35	0.77	0.54

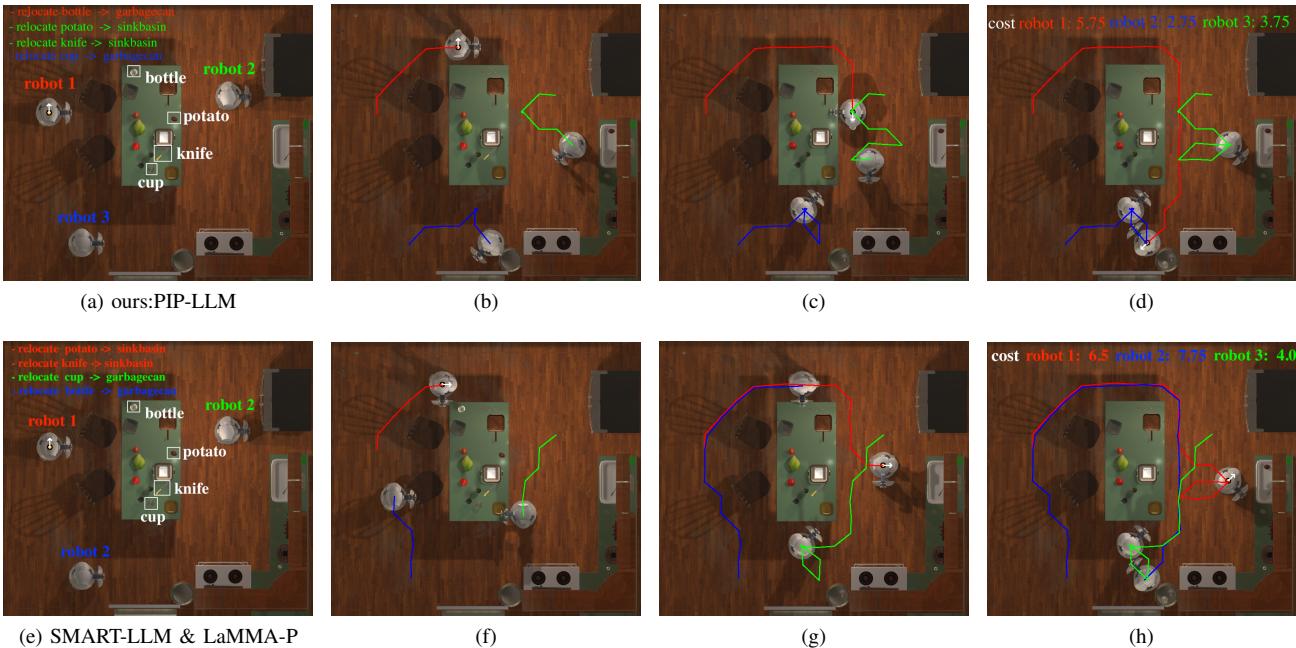


Fig. 2. An illustrative example to show the differences between our framework, PIP-LLM, and baselines (SMART-LLM and LaMMA-P). The task is put potato and knife in sinkbasin and trash the bottle and cup. (a)-(d) show the execution of results from PIP-LLM. (e)-(h) show the execution of baselines. The top left of (a) and (e) shows the allocated tasks of all robots. The top of (d) and (h) shows the cost of all robots. Ours achieves much smaller maximum/average cost. Red, blue, and green colors correspond to robots 1, 2, and 3, respectively.

Methods	Compound		Complex		Vague	
	SR	TC max/avg	SR	TC max/avg	SR	TC max/avg
PIP-LLM w/o \mathcal{D}	0.49	5.8/5.0	0.35	5.8/4.8	0.16	6.8/5.3
PIP-LLM w/o $\mathcal{B} \& \mathcal{V}$	0.68	5.5/4.3	0.43	6.5/5.0	0.32	6.3/4.8
PIP-LLM w/ $\mathcal{N} = 1$	0.85	5.5/4.8	0.74	5.3/4.5	0.44	6.0/4.8
PIP-LLM w/ $\mathcal{N} = 3$	0.90	5.8/4.8	0.83	6.0/4.8	0.60	6.0/4.5
PIP-LLM w/ $\mathcal{N} = 4$	0.93	5.3/4.3	0.87	5.8/4.5	0.68	6.0/4.8
PIP-LLM w/o \mathcal{AS}	0.93	6.5/5.3	0.87	7.0/5.8	0.68	7.3/6.5

TABLE II

ABLATIONS ACROSS THREE CATEGORIES. WE CONSIDER THE FOLLOWING KEY COMPONENTS IN FIG. 1: TEAM-LEVEL PLANNING PDDL DOMAIN (\mathcal{D}), DEBUGGER (\mathcal{B}), VERIFIER (\mathcal{V}), THE NUMBER OF ALLOWED REFINING LOOPS (\mathcal{N}), AND THE TASK ALLOCATION AND SCHEDULING COMPONENT (\mathcal{AS}). EVALUATION IS BASED ON GPT-4o.

D. Ablation Study for AI2-THOR Environment

We conducted an ablation study to assess the contribution of individual components to the performance of PIP-LLMs in Tab. II. The following key components were considered: the team-level planning PDDL domain (\mathcal{D}), the debugger (\mathcal{B}), the verifier (\mathcal{V}), the number of allowed refinement loops (\mathcal{N}), and the task allocation and scheduling module (\mathcal{AS}). In the first row, we exclude the PDDL domain \mathcal{D} and allow the LLMs to design team-level actions autonomously, performing

planning without generating explicit PDDL problems. As shown in the table, the success rate drops substantially across all three task categories, underscoring the critical role of the PDDL domain in guiding task understanding and decomposition. In the second row, we remove the solution refinement loop, i.e., the debugger and verifier components (\mathcal{B} and \mathcal{V}). This results in a pronounced decline in success rate, particularly for the complex and vague task groups, highlighting the effectiveness of \mathcal{B} and \mathcal{V} . To further examine their impact, we vary the number of refinement loops (\mathcal{N}) from the third to the fifth rows. The results clearly indicate that increasing \mathcal{N} consistently improves performance. We adopt $\mathcal{N} = 4$ for the results reported in Tab. I. Finally, in the last row, we replace our task allocation and scheduling module with a purely LLM-based approach. This modification led to a substantial increase in travel-related costs, demonstrating the efficiency of the proposed \mathcal{AS} component in optimizing resource utilization.

E. Results for Gazebo Warehouse Environment

Since AI2THOR is designed for a small-scale household setup, we cannot use it to test scalable multi-robot coordination. To this end, we create a customized warehouse environment in Gazebo for the second part of our experiments,

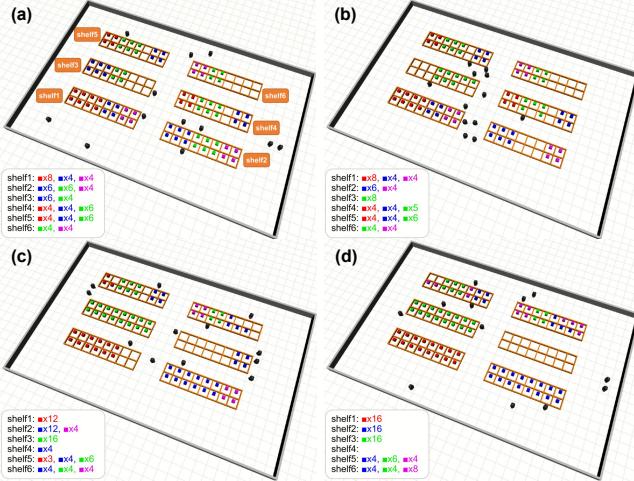


Fig. 3. A team of 12 robots conducting tasks in the warehouse. Products 1, 2, 3, and 4 are in red, blue, green, and magenta, respectively. The task command is There should be full of product 1 in shelf 1; full of product 2 in shelf 2; full of product 3 in shelf 3; and shelf 4 should be empty. The team-level action plan is 0.0: (move-product shelf2 shelf3 product3 magnitude6) 1.0: (move-product shelf3 shelf2 product2 magnitude6), ..., 8.0: (move-product shelf4 shelf2 product2 magnitude4) (a)-(d) show the changes in product locations over time.

```
domain file
inventory management domain
(define (domain inventory-management)
(:requirements :typing :fluent)
(:types shelf product magnitude)
(:predicates
  (has-product ?s - shelf ?p - product)
  (different-shelves ?s1 - shelf ?s2 - shelf) )
(:functions
  (amount ?s - shelf ?p - product)
  (free-space ?s - shelf)
  (value-of ?m - magnitude) )
(:action move-product
:parameters (?from - shelf ?to - shelf ?p - product
                           ?m - magnitude)
:precondition (and
  (different-shelves ?from ?to)
  (> (value-of ?m) 0)
  (>= (amount ?from ?p) (value-of ?m))
  (>= (free-space ?to) (value-of ?m)))
:effect (and
  (decrease (amount ?from ?p) (value-of ?m))
  (increase (amount ?to ?p) (value-of ?m))
  (decrease (free-space ?to) (value-of ?m))
  (increase (free-space ?from) (value-of ?m)) ) ) )
```

Fig. 4. PDDL domain used for the Gazebo warehouse environment.

as shown in Fig. 3, in which a team of robots needs to move many objects between different shelves based on the input of language instructions. We use blocks of different colors to represent different objects that need to be moved. We assume that the states of robots and objects are known. The team-level planning PDDL domain file is shown in Fig. 4, which focuses on only moving products from one shelf to another shelf without referring to any specific robots. An illustrative task example is given in the caption of Fig. 3. The PDDL plan, i.e., a sequence of team actions, will be generated and fed to the task allocation and scheduling part to decide which robots should be used to accomplish the team action. The team can efficiently rearrange a large number of objects.

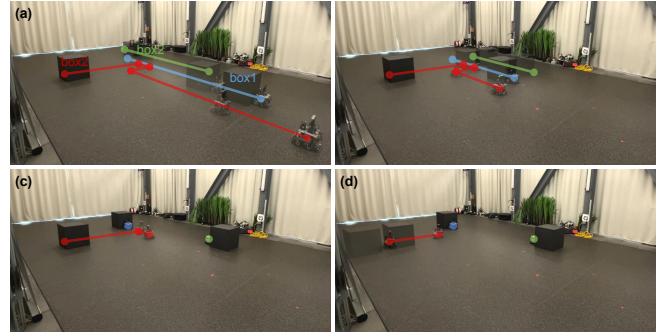


Fig. 5. Snapshots of hardware experiments. The task is Push boxes to their corresponding goals. The planned paths for robots are marked as blue, green, and red dots, respectively, in (a). Three robots are available for the task. The transparent robots and boxes along the paths denote their future locations.

Tab. III shows ten representative tasks for the warehouse environment. A task is considered finished successfully if the product distribution on the shelves is the same as the ground truth annotated by a human. To estimate the success rate, we execute each task ten times from varied initial robot locations, explicitly accounting for GPT-4o's stochastic output. As shown in Tab. III, our framework is able to achieve a significantly higher success rate meanwhile inducing a much lower travel cost. Baselines include CoT, SMART-LLM, and LaMMA-P. We observe that these baselines succeed only in the first task. This is because in task 1, the initial product distribution is very close to the desired distribution: we just need to move two products 1 from shelf 1 to shelf 3. The baseline frameworks can manage such “one-step” simple planning. For other tasks, they all involve moving products back and forth between shelves, which causes all baselines to fail. These results indicate that our framework can effectively understand high-level instructions to coordinate a large number of robots with a high coordination efficiency. Moreover, in tasks 6 and 7, we introduce noise to the language commands by using misspellings. We observe that such small noise can still lead the framework to fail occasionally. For example, in task 6, product 3 is written as *prodcut3* on purpose and LLMs may generate a PDDL problem by claiming *prodcut3* as a new product instead of product 3. Similarly, LLMs may claim *shelve4* as a new shelf in the generated PDDL file.

Hardware Demonstration. We employ multiple mobile ground robots to perform object transfer operations, where each robot is tasked with pushing boxes to designated goal locations. Snapshots of a representative experiment at different time frames are shown in Fig. 5.

Limitations. The proposed framework relies on a pre-defined team-level PDDL planning domain. Such a domain encapsulates all possible transitions in the environment and can effectively guide LLMs to find team-level solutions. Currently, it is still very challenging for LLMs to automatically design such a domain given a general problem context description. Another limitation is that we assume the world is closed and static, as in the classic AI planning. If there exists uncertainty in the environment (e.g., the number and type of objects

TABLE III
STATISTICAL RESULTS FROM TABLETOP EXPERIMENTS.

Task	Success rate (%)		Travel cost (m)		
	ID	ours	baselines	ours (max/avg)	baselines
1	100	100	—	28.5/33.0	34.1/40.3
2	100	0	—	164.7/141.1	—
3	100	0	—	151.2/111.6	—
4	100	0	—	321.3/267.1	—
5	90	0	—	194.3/167.1	—
6	80	0	—	201.4/157.4	—
7	70	0	—	304.2/271.4	—
8	100	0	—	111.6/97.4	—
9	100	0	—	221.2/177.4	—
10	100	0	—	288.3/248.3	—

ID	Task description
1	There should be 7 product 1 in shelf 3.
2	There should be 3 product2 in shelf4, 5 product2 in shelf2.
3	There should be 10 product1 in shelf3, 9 product1 in shelf2. We can only use robot1 to move product to [location] of the shelf1
4	There should be 2 product2, 7 product3, 8 product1 in shelf3, 3 product2, 5 product3, 12 product1 in shelf2.
5	There should be 20 product1 in shelf2, 16 product1 in shelf4, 2 product1 in shelf3. But robot 0 should not be used to move product1 from one shelf to another.
6	On shelf4, there oughta be 14 prodcut3, 2 prodcut1, and 1 prduct. (<i>include misspellings by design</i>)
7	Theyre supposed to be 17 prodcut1 on shelve4, I think.
8.	shelf4 has ten plus seven product1
9.	shelf2 stocks a quartet of product2 and a trio of product3
10.	shelf4 stores a dozen product3 and a quintet of product1.

are changing or the execution of actions succeeds with a probability), some replanning mechanism should be included to close the loop for PIP-LLM.

VI. CONCLUSION

This work presents a hierarchical framework, PIP-LLM, that integrates LLMs with both PDDL and IP to support scalable, long-horizon multi-robot task planning. By leveraging the complementary strengths of PDDL and IP, and using LLMs to bridge natural language inputs with formal representations, the framework lowers the barrier to entry for non-experts while maintaining the rigor needed for complex problem-solving. Through comprehensive simulations and hardware validation, we demonstrate the framework's effectiveness and potential in enabling intuitive deployment of multi-robot coordination systems across diverse scenarios. The code and dataset will be released after acceptance.

REFERENCES

- [1] Y. Rizk, M. Awad, and E. W. Tunstel, “Cooperative heterogeneous multi-robot systems: A survey,” *ACM Computing Surveys (CSUR)*, vol. 52, no. 2, pp. 1–31, 2019.
- [2] H. Liu, Y. Zhu, K. Kato, A. Tsukahara, I. Kondo, T. Aoyama, and Y. Hasegawa, “Enhancing the llm-based robot manipulation through human-robot collaboration,” *IEEE Robotics and Automation Letters*, vol. 9, no. 8, pp. 6904–6911, 2024.
- [3] Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, L. Fan, and A. Anandkumar, “Eureka: Human-level reward design via coding large language models,” in *The Twelfth International Conference on Learning Representations*, 2024.
- [4] B. Yu, H. Kasaei, and M. Cao, “L3mvn: Leveraging large language models for visual target navigation,” in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 3554–3560.
- [5] Y. Tang, W. Yu, J. Tan, H. Zen, A. Faust, and T. Harada, “Saytap: Language to quadrupedal locomotion,” in *Conference on Robot Learning*. PMLR, 2023, pp. 3556–3570.
- [6] S. S. Kannan, V. L. Venkatesh, and B.-C. Min, “Smart-llm: Smart multi-agent robot task planning using large language models,” in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 12140–12147.
- [7] Z. Mandi, S. Jain, and S. Song, “Roco: Dialectic multi-robot collaboration with large language models,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 286–299.
- [8] M. Ghallab, D. Nau, and P. Traverso, *Automated Planning: theory and practice*. Elsevier, 2004.
- [9] Y. Liu, L. Palmieri, S. Koch, I. Georgievski, and M. Aiello, “Delta: Decomposed efficient long-term robot task planning using large language models,” in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 10995–11001.
- [10] 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.
- [11] Z. Li, X. Wu, H. Du, F. Liu, H. Nghiem, and G. Shi, “A survey of state of the art large vision language models: Benchmark evaluations and challenges,” in *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR) Workshops*, June 2025, pp. 1587–1606.
- [12] Z. Ji, Y. Tiezheng, Y. Xu, N. Lee, E. Ishii, and P. Fung, “Towards mitigating llm hallucination via self reflection,” in *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.
- [13] Z. Li, X. Wu, G. Shi, Y. Qin, H. Du, T. Zhou, D. Manocha, and J. L. Boyd-Graber, “Videohallu: Evaluating and mitigating multimodal hallucinations on synthetic video understanding,” *arXiv preprint arXiv:2505.01481*, 2025.
- [14] Z. Li, W. Yu, C. Huang, R. Liu, Z. Liang, F. Liu, J. Che, D. Yu, J. Boyd-Graber, H. Mi *et al.*, “Self-rewarding vision-language model via reasoning decomposition,” *arXiv preprint arXiv:2508.19652*, 2025.
- [15] T. Guan, F. Liu, X. Wu, R. Xian, Z. Li, X. Liu, X. Wang, L. Chen, F. Huang, Y. Yacoob *et al.*, “Hallusionbench: an advanced diagnostic suite for entangled language hallucination and visual illusion in large vision-language models,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 14375–14385.
- [16] D. Bai, I. Singh, D. Traum, and J. Thomason, “Twostep: Multi-agent task planning using classical planners and large language models,” *arXiv preprint arXiv:2403.17246*, 2024.
- [17] X. Zhang, H. Qin, F. Wang, Y. Dong, and J. Li, “Lamma-p: Generalizable multi-agent long-horizon task allocation and planning with lm-driven pdl planner,” *arXiv preprint arXiv:2409.20560*, 2024.
- [18] B. P. Gerkey and M. J. Matarić, “A formal analysis and taxonomy of task allocation in multi-robot systems,” *The International journal of robotics research*, vol. 23, no. 9, pp. 939–954, 2004.
- [19] R. Schumann, W. Zhu, W. Feng, T.-J. Fu, S. Riezler, and W. Y. Wang, “Velma: verbalization embodiment of llm agents for vision and language navigation in street view,” in *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence*, 2024.
- [20] Z. Ravichandran, V. Murali, M. Tzes, G. J. Pappas, and V. Kumar, “Spine: Online semantic planning for missions with incomplete natural language specifications in unstructured environments,” *International Conference on Robotics and Automation (ICRA)*, 2025.
- [21] D. Shah, B. Osinski, B. Ichter, and S. Levine, “LM-nav: Robotic navigation with large pre-trained models of language, vision, and action,” in *6th Annual Conference on Robot Learning*, 2022.
- [22] A. Tagliabue, K. Kondo, T. Zhao, M. Peterson, C. T. Tewari, and J. P. How, “Real: Resilience and adaptation using large language models on autonomous aerial robots,” in *2024 IEEE 63rd Conference on Decision and Control (CDC)*, 2024, pp. 1539–1546.
- [23] R. Sinha, A. Elhafsi, C. Agia, M. Foutter, E. Schmerling, and M. Pavone, “Real-time anomaly detection and reactive planning with large language models,” in *Robotics: Science and Systems (RSS)*, 2024.

- [24] S. Xu, X. Luo, Y. Huang, L. Leng, R. Liu, and C. Liu, “Nl2htl2plan: Scaling up natural language understanding for multi-robots through hierarchical temporal logic task specifications,” *IEEE Robotics and Automation Letters*, 2025.
- [25] Y. Chen, J. Arkin, Y. Zhang, N. Roy, and C. Fan, “Scalable multi-robot collaboration with large language models: Centralized or decentralized systems?” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 4311–4317.
- [26] J. Wang, G. He, and Y. Kantaros, “Probabilistically correct language-based multi-robot planning using conformal prediction,” *IEEE Robotics and Automation Letters*, 2024.
- [27] B. Yu, H. Kasaei, and M. Cao, “Co-navgpt: Multi-robot cooperative visual semantic navigation using large language models,” *arXiv preprint arXiv:2310.07937*, 2023.
- [28] K. Obata, T. Aoki, T. Horii, T. Taniguchi, and T. Nagai, “Lip-llm: Integrating linear programming and dependency graph with large language models for multi-robot task planning,” *IEEE Robotics and Automation Letters*, 2024.
- [29] A. Lykov, M. Dronova, N. Naglov, M. Litvinov, S. Satsevich, A. Bazhenov, V. Berman, A. Shcherbak, and D. Tsetserukou, “Llm-mars: Large language model for behavior tree generation and nlp-enhanced dialogue in multi-agent robot systems,” *arXiv preprint arXiv:2312.09348*, 2023.
- [30] P. Haslum, N. Lipovetzky, D. Magazzeni, C. Muise, R. Brachman, F. Rossi, and P. Stone, *An introduction to the planning domain definition language*. Springer, 2019, vol. 13.
- [31] M. Fox and D. Long, “Pddl2. 1: An extension to pddl for expressing temporal planning domains,” *Journal of artificial intelligence research*, vol. 20, pp. 61–124, 2003.
- [32] H. Ravichandar, K. Shaw, and S. Chernova, “Strata: unified framework for task assignments in large teams of heterogeneous agents,” *Autonomous Agents and Multi-Agent Systems*, vol. 34, no. 2, p. 38, 2020.
- [33] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi, “AI2-THOR: An Interactive 3D Environment for Visual AI,” *arXiv*, 2017.
- [34] 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 *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 11 523–11 530.