# Probabilistically Correct Language-Based Multi-Robot Planning Using Conformal Prediction

Jun Wang, Guocheng He, and Yiannis Kantaros, *Member, IEEE*

***Abstract*—This paper addresses task planning problems for language-instructed robot teams. Tasks are expressed in natural language (NL), requiring the robots to apply their skills at various locations and semantic objects. Several recent works have addressed similar planning problems by leveraging pre-trained Large Language Models (LLMs) to design effective multi-robot plans. However, these approaches lack performance guarantees. To address this challenge, we introduce a new distributed LLM-based planner, called S-ATLAS for Safe plAnning for Teams of Language-instructed AgentS, that can achieve user-defined mission success rates. This is accomplished by leveraging conformal prediction (CP), a distribution-free uncertainty quantification tool. CP allows the proposed multi-robot planner to reason about its inherent uncertainty, due to imperfections of LLMs, in a distributed fashion, enabling robots to make local decisions when they are sufficiently confident and seek help otherwise. We show, both theoretically and empirically, that the proposed planner can achieve user-specified task success rates, assuming successful plan execution, while minimizing the average number of help requests. We provide comparative experiments against related works showing that our method is significantly more computational efficient and achieves lower help rates.**

***Index Terms*—Multi-robot systems, task and motion planning, AI-enabled robotics, planning under uncertainty.**

## I. Introduction

DESIGNING robots with advanced task planning abilities has been a fundamental goal in robotics [1]. To this end, several task and motion planners have been proposed recently, which, however, often require significant user expertise for mission specification, using formal languages [2] or reward functions [3]. Natural Language (NL) has emerged as a more user-friendly alternative to specify robot missions. Motivated by the remarkable generalization abilities of pre-trained Large Language Models (LLMs) across diverse domains, there has been increasing attention on utilizing LLMs for NL-based planning. Early efforts primarily focused on single-robot task planning problems [4], [5], [6], [7], [8], [9], [10] while recent extensions to multi-robot systems are presented in [11], [12], [13], [14], [15], [16], [17] and the references therein. A detailed survey can be found in [18]. A major challenge with current LLM-based planners is that they confidently generate incorrect and possibly unsafe outputs.

This paper focuses on enhancing the reliability of LLM-based multi-robot planners. We consider teams of robots possessing various skills (e.g., mobility and manipulation) tasked with missions expressed in NL. Mission completion requires the robots to apply their skills to various known semantic objects that exist in the environment. Our overarching goal is to design LLM-based planners capable of computing multi-robot plans (i.e., sequences of robot actions) with a user-specified probability of correctness; this ensures desired mission success rates assuming successful plan execution. This requires developing LLM-based planners that can reason about uncertainties of the employed LLMs, enabling robots to make decisions when sufficiently confident and seek help otherwise.

To address this problem, we propose a new distributed LLM-based planner, called S-ATLAS for Safe plAnning for Teams of Language-instructed AgentS; see Fig. 1. In our framework, at each time step, the robots select actions sequentially while considering actions chosen by other robots as e.g., in [11]. Each robot is delegated to a pre-trained LLM agent that is responsible for decision-making. This coordinate-descent approach enables the distributed construction of robot plans. To choose an action for a robot, we present the action selection problem to the corresponding LLM agent as a multiple-choice question-answering (MCQA) scenario [7]. The 'question' corresponds to the textual task description and the history of past decisions, while the set of available 'choices' represents the skills that the selected robot can apply. The MCQA setup ensures that, unlike other multi-robot planners, the LLM only chooses from valid choices, partially mitigating the risk of hallucination, where nonsensical actions might be generated. Instead of selecting the action with the highest logit score, inspired by [19], we employ conformal prediction (CP) to quantify the uncertainty of the employed LLM [20]; this is crucial as LLMs often confidently produce incorrect outputs. CP enables the design of individual prediction sets for each robot that contain the correct action with high confidence. This allows each LLM-driven robot to determine when it is uncertain about its decisions, which is a key step toward achieving desired mission success rates. In cases of high uncertainty, indicated by non-singleton prediction sets, the respective robots seek assistance from other robots and/or users; otherwise, they execute the action in their singleton sets.

*Related Works:* As discussed earlier, several planners for language-driven robot teams have been proposed recently which, unlike the proposed method, lack performance guarantees. The closest works to ours are [19], [21], [22], which

Received 26 June 2024; accepted 1 November 2024. Date of publication 21 November 2024; date of current version 29 November 2024. This article was recommended for publication by Associate Editor L. Liu and Editor M. A. Hsieh upon evaluation of the reviewers' comments. This work was supported in part by ARL under Grant DCIST CRA W911NF-17-2-0181 and in part by NSF under Grant CNS #2231257 and Grant CCF #2403758. *(Corresponding author: Yiannis Kantaros.)*

Jun Wang and Yiannis Kantaros are with the Preston M. Green Department of Electrical and Systems Engineering, Washington University in St. Louis (WashU), St. Louis, MO 63130 USA (e-mail: junw@wustl.edu; ioannisk@wustl.edu).

Guocheng He was with the Preston M. Green Department of Electrical and Systems Engineering, Washington University in St. Louis (WashU), St. Louis, MO 63130 USA. He is now with the Department of Computer Science, Vanderbilt University, Nashville, TN 37240 USA (e-mail: guocheng@wustl.edu).

Digital Object Identifier 10.1109/LRA.2024.3504233


Fig. 1. The proposed distributed planner can determine when and which LLM-driven robot is uncertain about correctness of its next action. In cases of high uncertainty, the respective robots seek assistance. This enables the planner to achieve desired mission success rates.

apply CP for aligning LLM uncertainty. In [21], CP is applied to single-step MCQA tasks while [19], [22] build upon that approach to address more general NL tasks, modeled as multi-step MCQA problems, with desired success rates. Notably, the inspiring work in [19] is the first to demonstrate how CP can determine when a robot needs help from a user to efficiently resolve task ambiguities. These works, however, focus on single-robot planning problems. We note that [19] can be extended to multi-robot settings by treating the multi-robot system as a single robot performing multiple actions simultaneously. However, we show that these centralized implementations do not scale well with increasing team size due to increasing computational costs and/or the diminishing ability of a single LLM to effectively manage large robot teams. Moreover, centralized implementations of [19] generate multi-robot/global prediction sets, making it harder to determine which specific robot requires assistance. Our method addresses these challenges through its distributed nature, constructing local prediction sets, for each robot, allowing users to easily identify which robot needs help and when, while also ensuring desired team-wide mission success rates. Comparisons against these centralized approaches demonstrate that our method is significantly more computationally efficient and requires fewer user interventions, particularly as team size and mission complexity grow. Finally, we note that CP has been applied to various autonomy tasks [23], [24], [25], [26]. To the best of our knowledge, this paper presents the first CP application in NL-based multi-robot planning, particularly in *distributed* settings.

*Contribution:* The contribution of the paper can be summarized as follows. **(i)** We propose the first *distributed* planner for language-instructed multi-robot systems that can achieve *user-specified task success rates* (assuming successful execution of robot skills). **(ii)** We show how CP can be applied in a distributed fashion to construct local prediction sets for each individual robot. **(iii)** We provide comparative experiments showing that our planner outperforms related language-based planners in terms of efficiency and help rates.

## II. Problem Formulation

*Robot Team:* Consider a team of $N \geq 1$ robots. Each robot is governed by known dynamics: $\mathbf{p}_j(t+1) = \mathbf{f}_j(\mathbf{p}_j(t), \mathbf{u}_j(t))$, $j \in \{1, 2, \ldots, N\}$, where $\mathbf{p}_j(t)$ and $\mathbf{u}_j(t)$ stand for the state the control input of robot $j$ at discrete time $t$, respectively. Each robot possesses $A > 0$ skills collected in a set $\mathcal{A} \in \{1, \ldots, A\}$. These skills $a \in \mathcal{A}$ are represented textually (e.g., 'take a picture', 'grab', or 'go to'). The application of skill $a$ by robot $j$ to an object/region at location $\mathbf{x}$ at time $t$ is represented as $s_j(a, \mathbf{x}, t)$. For simplicity, we assume homogeneity among the robots, meaning they all share the same skill set $\mathcal{A}$ and can apply skill $a$ at any given location $\mathbf{x}$ associated with an object/region. When it is clear from the context, we denote $s_j(a, \mathbf{x}, t)$ by $s_j(t)$ for brevity. We also define the multi-robot decision at time $t$ as $\mathbf{s}(t) = [s_1(t), \ldots, s_N(t)]$. The time step $t$ is increased by one, once $\mathbf{s}$ is completed/executed. Also, we assume that every robot has access to low-level error-free controllers to execute the skills in $\mathcal{A}$ [19].

*Environment:* The robot team resides in a known environment $\Omega$ that contains $M > 0$ semantic objects. Each object $e$ is defined by its location $\mathbf{x}_e$ and a semantic label $o_e$. Using the action space $\mathcal{A}$ and the available semantic objects, we can construct a set $\mathcal{S}$ that collects all possible decisions $s_j$ that a robot $j$ can make. Notice that since we consider homogeneous robots, the set $\mathcal{S}$ is the same for all robots $j$.

*Mission Specification:* The team is assigned a coordinated task $\phi$, expressed in NL, that is defined over objects/regions of interest in $\Omega$. This task may comprise multiple sub-tasks that are not necessarily pre-assigned to specific robots. Thus, achieving $\phi$ involves determining which actions each robot should apply, when, where, and in what order. Our goal is to design multi-robot plans that accomplish $\phi$, defined as $\tau = \mathbf{s}(1), \mathbf{s}(2), \ldots, \mathbf{s}(H)$, for some horizon $H$. We formalize this by considering a distribution $\mathcal{D}$ over scenarios $\xi_i = \{N_i, \mathcal{A}_i, \phi_i, H_i, \Omega_i\}$, with corresponding ground truth plans $\tau_i$.[1] Recall that $N_i, \mathcal{A}_i, \phi_i, H_i,$ and$\Omega_i$ refer to the number of robots, the robot skills, the NL-based task, the mission horizon, and the semantic environment, respectively, associated with $\xi_i$. The horizon $H_i$ determines an upper bound on the number of steps $t$ required to accomplish $\phi_i$. The subscript $i$ is used to emphasize that these parameters can vary across scenarios. We assume that all scenarios $\xi_i$ drawn from $\mathcal{D}$ are feasible. When it is clear from the context, we drop the dependence on $i$. Note that $\mathcal{D}$ is unknown but we assume that we can sample i.i.d. scenarios from it. Our goal is to design probabilistically correct language-based planners, i.e., planners that can generate correct/feasible plans $\tau_i$ satisfying $\xi_i$, in at least $(1-\alpha) \cdot 100\%$ of scenarios $\xi_i$ drawn from $\mathcal{D}$, for a user-specified $\alpha \in (0, 1)$.[2]

[1] Accounting for heterogeneous robots requires defining more complex distributions that can generate individual action sets. The proposed planner remains applicable. We abstain from this presentation for simplicity.

[2] We note that probabilistic task satisfaction arises solely due to imperfections of LLMs employed to process NL. Our future work will consider imperfections of robot skills as well.

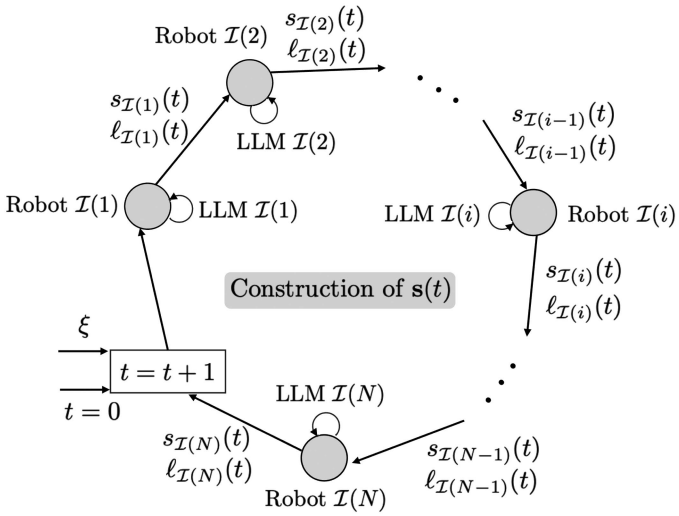


Fig. 2. Given a task scenario $\xi$, the robots select decisions at each time $t$ sequentially as per $\mathcal{I}$. Each robot $j = \mathcal{I}(i)$ (gray disk) is delegated to an LLM that generates a decision $s_j(t)$ given textual context $\ell_{\mathcal{I}(i-1)}(t)$ containing the task description and past robot decisions provided by the previous robot $\mathcal{I}(i-1)$. If the LLM for robot $j$ is not certain enough about what the correct decision $s_j(t)$ is, the robots seek assistance (not shown).

**Algorithm 1:** S-ATLAS.

1: **Input**: Scenario $\xi$; Coverage level $\alpha$; Upper bound $W$
2: Initialize an ordered set $\mathcal{I}$;
3: Initialize prompt $\ell_{\mathcal{I}(1)}(1)$ with empty history of actions
4: **for** $t = 1$ to $H$ **do**
5: Initialize help counter $w = 0$
6: $\mathbf{s}(t) = [\varnothing, \ldots, \varnothing]$
7: **for** $i \in \mathcal{I}$ **do**
8: Next robot $j = \mathcal{I}(i)$
9: Compute $\ell_j(t)$ using $\ell_{j-1}(t)$
10: Compute the local prediction set $\mathcal{C}(\ell_j(t))$ as in (8)
11: **if** $|\mathcal{C}(\ell_j(t))| > 1$ **then**
12: $w = w + 1$
13: **if** $w > W$ **then**
14: Obtain $s_j(t)$ from human operator
15: **else**
16: Get new $\mathcal{I}'$, set $\mathcal{I} = \mathcal{I}'$, and go to line 6
17: **else**
18: Pick the (unique) decision $s_j(t) \in \mathcal{C}(\ell_j(t))$
19: Update $\ell_j(t) = \ell_j(t) + s_j(t)$
20: Send $\ell_j(t)$ to robot $\mathcal{I}(i+1)$
21: Construct $\mathbf{s}(t) = [s_1(t), \ldots, s_N(t)]$
22: Append $\mathbf{s}(t)$ to the multi-robot plan $\tau$

*Problem 1:* Design a planner that generates multi-robot plans $\tau$, which complete missions $\xi$, drawn from an unknown distribution $\mathcal{D}$, with probability greater than a user-specified threshold $1 - \alpha \in (0, 1)$, assuming successful execution of $\tau$.

## III. Probabilistically Correct Language-Based Multi-Robot Planning

In this section, we propose a new task planner that harnesses pre-trained LLMs to address Problem 1. In Section III-A, we present a centralized planner, as a potential solution to Problem 1, building upon [7], [19], which, however, cannot effectively scale to multi-robot settings. To address this challenge, in Sections III-B-III-D, we present S-ATLAS, our proposed distributed planner; see Algorithm 1 and Fig. 2.

### A. A Centralized Planning Framework Using LLMs

In this section, we present a centralized planner building on the single-robot planners from [7], [19]. In [7], the task planning problem is modeled as a sequence of MCQA scenarios over time steps $t \in \{1, \ldots, H\}$, where the question represents the task $\phi$ and mission progress, and the choices represent decisions $s_j(t) \in \mathcal{S}$ that (the single) robot $j$ can make. This approach can be extended to multi-robot systems by treating the team as a single robot executing $N$ actions simultaneously, leading to $S^N$ choices in the MCQA framework. Based on [7], the action with the highest LLM confidence score is selected; assuming perfect robot skills (i.e., affordance functions that always return 1). However, these scores are uncertainty-agnostic and uncalibrated. Therefore, this approach cannot effectively solve Problem 1 [19]. Instead, we apply the conformal prediction (CP) approach from [19] allowing the team to make decisions when sufficiently certain and seek help from users otherwise. As shown in [19], this centralized approach can achieve desired mission success rates. However, application of CP in this setting becomes computationally intractable as $N$ increases due to the exponential growth of the number $S^N$ of options that need to be considered; see Section V-B.

*Remark 3.1 (Enhancing Computational Efficiency):* A more efficient centralized approach would be to consider only $X \ll S^N$ options (i.e., multi-robot decisions) in every MCQA scenario, that are generated on-the-fly by an LLM as proposed in [19]. As in [19], these options should be augmented with an additional choice 'other option, not listed' to account for cases where all LLM options are nonsensical. A challenge in this case is that as $N$ increases, LLMs struggle to provide valid options, often leading to 'incomplete' plans ending with 'other option, not listed'; see Section V-B.

### B. Distributed Multi-Robot Planning Using LLMs

In this section, we propose S-ATLAS, a distributed planning framework to address Problem 1 and overcome the limitations of the centralized approaches discussed in Section III-A, making it more effective for larger-scale planning problems. The key idea is that, given a scenario $\xi$ drawn from $\mathcal{D}$, the robots pick decisions at time $t$ successively (as opposed to jointly in Section III-A). Each robot $j$ is delegated to an LLM that picks a decision $s_j(t)$, using an MCQA setup, while incorporating decisions of past robot decisions. If the LLM for robot $j$ is not certain enough about what the correct $s_j(t)$ is, it seeks assistance from other robots and/or human operators. This process gives rise to the multi-robot decision $\mathbf{s}(t)$ at time $t$. The above is repeated to construct $\mathbf{s}(t+1), \ldots, \mathbf{s}(H)$. Next, we describe our proposed distributed planner in more detail.

Consider a scenario $\xi$ drawn from $\mathcal{D}$ and an ordered set $\mathcal{I}$, initialized by a user, collecting robot indices $j \in \{1, \ldots, N\}$ so that each robot index appears exactly once; e.g., $\mathcal{I} = \{1, \ldots, N\}$ [line 2, Alg. 1]. We denote the $i-$th element in $\mathcal{I}$ by $\mathcal{I}(i)$. At each $t \in \{1 \ldots H\}$, the robots pick actions sequentially as per $\mathcal{I}$ [lines 3-22, Alg. 1]; see also Fig. 2. Specifically, robot $j = \mathcal{I}(i)$ selects $s_j(t)$ while considering, the NL task description as well as all

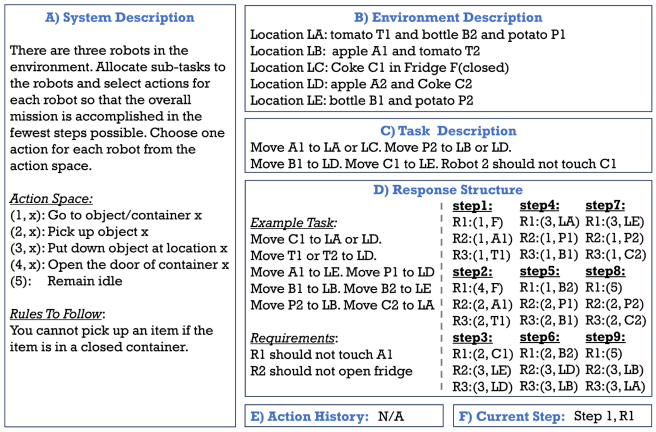


Fig. 3. Example of the constructed prompt in Section V for GPT 3.5. This prompt refers to $t = 1$ when the action history is empty.

decisions that have been made until $t-1$ and all decisions made by the robots $\mathcal{I}(1), \ldots, \mathcal{I}(i-1)$ at time $t$. This information, denoted by $\ell_j(t)$, is described textually. Given the context $\ell_j(t)$, the LLM assigned to robot $j$ generates $s_j(t)$. Hereafter, for simplicity of notation, we assume that $\mathcal{I} = \{1, \ldots, N\}$ so that $\mathcal{I}(j) = j$. We also assume that all agents share the same LLM model; this assumption is made only for ease of notation as it will be discussed later. Next, we first describe how $\ell_j(t)$ is structured and then we discuss how $s_j(t)$ is computed given $\ell_j(t)$.

*Prompt Construction:* The prompt $\ell_j(t)$ consists of the following parts (see Fig. 3): (a) *System description* that defines the action space and the objective of the LLM. (b) *Environment description* that describes the semantic objects that exist in the environment; (c) *Task description* that includes the task description $\phi$; (d) *Response structure* describing the desired structure of the LLM output for an example task. (e) *History of actions* that includes the sequence of decisions made by all robots up to time $t-1$ and for all robots $1, \ldots, j-1$ up to time $t$. (f) *Current time step $t$ and index to the robot $j$* that is now responsible for picking an action.

*Plan Design:* Assume that at time $t$, it is robot's $j$ turn to select $s_j(t)$ as per $\mathcal{I}$. First, robot $j$ constructs $\ell_j(t)$ using the prompt $\ell_{j-1}(t)$ that the previous robot $j-1$ constructed (or $\ell_N(t-1)$, if $j = 1$) [line 8-9, Alg. 1]. Parts (a)-(c) in $\ell_j(t)$ are the same for all $j$ and $t \geq 1$ (since the environment is static/known); part (d) is the same for all $j$ and $t \geq 1$; part (e) includes the history of actions and can be found in part (e) of $\ell_{j-1}(t)$; and part (f) is constructed as discussed above. The initial prompt $\ell_1(1)$ is manually constructed given a scenario $\xi$ with part (d) being empty [line 3, Alg. 1].

Given $\ell_j(t)$, selection of $s_j(t)$ is represented as an MCQA problem that is solved by the LLM for robot $j$. Specifically, given $\ell_j(t)$ ('question' in MCQA), the LLM $j$ has to pick decision $s_j(t)$ among all available ones included in part (a) ('choices' in MCQA). Given any $s_j(t) \in \mathcal{S}$, LLMs can provide a confidence score $g(s_j(t)|\ell_j(t))$; the higher the score, the more likely the decision $s_j(t)$ is a valid next step to positively progress the task [7]. Note that if the robots do not share the same LLM model, then $g(s_j(t)|\ell_j(t))$ should be replaced by $g_j(s_j(t)|\ell_j(t))$ throughout the paper. To get the scores $g(s_j(t)|\ell_j(t))$ for all $s_j(t) \in \mathcal{S}$, we query the model over all potential decisions $s_j$. Using these scores, a possible approach to select the $s_j(t)$ is by simply choosing the decision with the highest score, i.e., $s_j(t) = \arg\max_{s_j \in \mathcal{S}} g(s_j|\ell_j(t))$. However, this point-prediction approach is uncertainty-agnostic as these scores do not represent calibrated confidence [19]. A much preferred approach would be to generate a set of actions (called, hereafter, *prediction set*), denoted by $\mathcal{C}(\ell_j(t))$, that contains the ground truth action with a user-specified high-probability [line 10, Alg. 1]. Hereafter, we assume that such prediction sets are provided. We defer their construction to Section III-D (see (8)) but we emphasize that these sets are critical to achieving desired $1-\alpha$ mission completion rates.

Given $\mathcal{C}(\ell_j(t))$, we select decisions $s_j(t)$ as follows. If $\mathcal{C}(\ell_j(t))$ is singleton, then we select the action included in $\mathcal{C}(\ell_j(t))$ as it contributes to mission progress with high probability [line 18, Alg. 1]. Otherwise, robot $j$ seeks assistance from its teammates asking all robots to select new decisions for time $t$ as a per a different order $\mathcal{I}$. If this does not result in singleton prediction sets, robot $j$ asks a human operator to select $s_j(t)$ [lines 11-14, Alg. 1]. The process of seeking assistance is discussed in more detail in Section III-C. Once $s_j(t)$ is selected, robot $j$ records $s_j(t)$ into part (e) of $\ell_j(t)$. With slight abuse of notation, we denote this prompt update by $\ell_{j+1}(t) = \ell_j(t) + s_j(t)$, where the summation means concatenation of text [line 19, Alg. 1]. Then, robot $j$ sends the resulting prompt $\ell_j(t)$ to the next robot $j+1$ [line 20, Alg. 1] We repeat the above procedure sequentially over all robots $j$ as per $\mathcal{I}$. This way, we construct the multi-robot action $\mathbf{s}(t)$ [lines 21-22, Alg. 1]. Once $\mathbf{s}(t)$ is constructed the current time step is updated to $t+1$. The above process is repeated to design $\mathbf{s}(t+1)$ and terminates at $t = H$. This way, we generate a plan $\tau = \mathbf{s}(1), \ldots, \mathbf{s}(t), \ldots, \mathbf{s}(H)$. The robots execute the plan $\tau$ synchronously once it is fully constructed.

### C. Seeking Assistance From Teammates and Human Operators

As discussed in Section III-B, if $\mathcal{C}(\ell_j(t))$ is singleton, then this means that the LLM is sufficiently certain that the decision $s_j(t) \in \mathcal{C}(\ell_j(t))$ will contribute to making mission progress. Otherwise, the corresponding robot $j$ should seek assistance in the following two ways [lines 11-14, Alg. 1]. First, robot $j$ generates a new order $\mathcal{I}$ which is forwarded to all robots along with a message to redesign $\mathbf{s}(t)$ from scratch, as per the new ordered set $\mathcal{I}$ [lines 11-13, Alg. 1]. Our empirical analysis has shown that $\mathcal{I}$ can potentially affect the size of prediction sets even though this change affects only part (f) in the constructed prompts; see Section V. We remark that this step can significantly increase computational complexity of our method, and, therefore, it may be neglected for large robot teams and sets $\mathcal{S}$. The second type of assistance is pursued when a suitable set $\mathcal{I}$ cannot be found within a user-specified number $W \geq 0$ of attempts. Then, robot $j$ requests help from human operators [lines 13-14, Alg. 1]. Particularly, robot $j$ presents to a user the set $\mathcal{C}(\ell_j(t))$ (along with the prompt $\ell_j(t)$) and asks them to choose an action from it. The user either picks an action for the corresponding robot or decides to halt the operation; see also Remark 4.3.

### D. Constructing Local Prediction Sets Using CP

In this section, we discuss how the prediction sets $\mathcal{C}(\ell_j(t))$, introduced in Section III-B, are constructed, given a required task success rate $1-\alpha$ (see Problem 1). The MCQA setup allows

us to apply conformal prediction (CP) to construct these sets. To illustrate the challenges of their construction, we consider the following cases: (i) single-robot & single-step plans and (ii) multi-robot & multi-step plans.

*Single-robot & Single-step Plans:* Initially, we focus on scenarios with $N = 1$ and $H = 1$. First, we sample $M$ i.i.d. scenarios from $\mathcal{D}$ called, hereafter, calibration scenarios. For each calibration scenario $i$, we construct its equivalent prompt $\ell^i_{j,\text{calib}}$ associated with (the single) robot $j$. For each prompt, we (manually) compute the ground truth plan $\tau^i_{\text{calib}} = s^i_{j,\text{calib}}(1)$ accomplishing this task. For simplicity, we assume that there exists a unique correct decision $s^i_{j,\text{calib}}$ for each $\ell^i_{j,\text{calib}}$; this assumption is relaxed in Remark 3.4. Hereafter, we drop the dependence of the robot decisions and prompts on the time step, since we consider single-step plans. This way we construct a calibration dataset $\mathcal{M} = \{(\ell^i_{j,\text{calib}}, \tau^i_{\text{calib}})\}_{i=1}^M$.

Consider a new scenario drawn from $\mathcal{D}$, called validation/test scenario. We convert this scenario into its equivalent prompt $\ell_{j,\text{test}}$. Since the calibration and the validation scenario are i.i.d., CP can generate a prediction set $\mathcal{C}(\ell_{j,\text{test}})$ of decisions $s_j$ containing the correct one, denoted by $s_{j,\text{test}}$, with probability greater than a desired value $1-\alpha \in (0,1)$, i.e.,

$$P(s_{j,\text{test}} \in \mathcal{C}(\ell_{j,\text{test}})) \geq 1-\alpha. \tag{1}$$

To generate $\mathcal{C}(\ell_{j,\text{test}})$, CP first uses the LLM's confidence $g$ to compute the set of non-conformity scores (NCS) $\{r^i = 1 - g(s^i_{j,\text{calib}} \,|\, \ell^i_{j,\text{calib}})\}_{i=1}^M$ over the calibration set. The higher the score is, the less each calibration point conforms to the data used for training the LLM. Then we perform calibration by computing the $\frac{(M+1)(1-\alpha)}{M}$ empirical quantile of $r_1, \ldots, r_M$ denoted by $q$. Using $q$, CP constructs the prediction set

$$\mathcal{C}(\ell_{j,\text{test}}) = \{s_j \in \mathcal{S} \,|\, g(s_j|\ell_{j,\text{test}}) > 1-q\}, \tag{2}$$

that includes all decisions that the model is at least $1-q$ confident in. This set is used for decision making as per Section III-B for single-robot/step tasks. By construction of these sets, we have that $s_j = \arg\max_{s_j \in \mathcal{S}} g(s_j|\ell_{j,\text{test}}) \in \mathcal{C}(\ell_{j,\text{test}})$.

*Multi-robot & Multi-step Plans:* Next, we generalize the above result to the case where $N \geq 1$ and $H \geq 1$. Here we cannot apply directly (1)-(2) to compute individual sets $\mathcal{C}(\ell_{j,\text{test}}(t))$ for each robot, as this violates the i.i.d. assumption required to apply CP. Particularly, the distribution of prompts $\ell_{j,\text{test}}(t)$ depends on the previous prompts i.e., the multi-robot decisions at $t' < t$ as well as the decisions of robots $1, \ldots, j-1$ at time $t$. Inspired by [19], we address this challenge by (i) lifting the data to sequences, and (ii) performing calibration at the sequence level using a carefully designed NCS.

We sample $M \geq 1$ independent calibration scenarios from $\mathcal{D}$. Each scenario corresponds to various tasks $\phi_i$, numbers of robots $N_i$, and mission horizons $H_i$. Each task $\phi_i$ is broken into a sequence of $T_i = H_i \cdot N_i \geq 1$ prompts defined as in Section III-B. This sequence of $T_i$ prompts is denoted by

$$\begin{aligned}\bar{\ell}^i_{\text{calib}} = &\underbrace{\ell^i_{1,\text{calib}}(1), \ldots, \ell^i_{j,\text{calib}}(1), \ldots, \ell^i_{N_i,\text{calib}}(1)}_{t=1}, \ldots, \\ &\underbrace{\ell^i_{1,\text{calib}}(t'), \ldots, \ell^i_{j,\text{calib}}(t'), \ldots, \ell^i_{N_i,\text{calib}}(t')}_{t=t'}, \ldots, \\ &\underbrace{\ell^i_{1,\text{calib}}(H_i), \ldots, \ell^i_{j,\text{calib}}(H_i), \ldots, \ell^i_{N_i,\text{calib}}(H_i)}_{t=H_i}.\end{aligned} \tag{3}$$

where, by construction, each prompt $\bar{\ell}^i_{\text{calib}}$ contains a history of ground truth decisions made so far. We define the corresponding sequence of ground truth decisions as:[3]

$$\begin{aligned}\tau^i_{\text{calib}} = &\underbrace{s_{1,\text{calib}}(1), \ldots, s_{j,\text{calib}}(1), \ldots, s_{N_i,\text{calib}}(1)}_{=\mathbf{s}_{\text{calib}}(1)}, \ldots, \\ &\underbrace{s_{1,\text{calib}}(t'), \ldots, s_{j,\text{calib}}(t'), \ldots, s_{N_i,\text{calib}}(t')}_{=\mathbf{s}_{\text{calib}}(t')}, \ldots, \\ &\underbrace{s_{1,\text{calib}}(H_i), \ldots, s_{j,\text{calib}}(H_i), \ldots, s_{N_i,\text{calib}}(H_i)}_{=\mathbf{s}_{\text{calib}}(H_i)}.\end{aligned} \tag{4}$$

This gives rise to a calibration set $\mathcal{M} = \{(\bar{\ell}^i_{\text{calib}}, \tau^i_{\text{calib}})\}_{i=1}^M$. As before, for simplicity, we assume that each context $\bar{\ell}^i_{\text{calib}}$ has a unique correct plan $\tau^i_{\text{calib}}$; this assumption is relaxed in Remark 3.4. We denote by $\bar{\ell}^i_{\text{calib}}(k)$ and $\tau^i_{\text{calib}}(k)$ the $k$-th entry in $\bar{\ell}^i_{\text{calib}}$ and $\tau^i_{\text{calib}}$, respectively, where $k \in \{1, \ldots, T_i\}$. Note that the iterations $k$ are different from the time steps $t \in \{1, \ldots, H_i\}$. Each iteration $k$ refers to the prompt/decision corresponding to a specific robot index $j$ at time step $t$.

Next, for each calibration sequence, we define the NCS similarly to the single-robot-single-step plan case. Specifically, the NCS of the $i$-th calibration sequence, denoted by $\bar{r}_i$, is computed based on the lowest NCS over $k \in \{1, \ldots, T_i\}$, i.e., $\bar{r}_i = 1 - \bar{g}(\tau^i_{\text{calib}} | \bar{\ell}^i_{\text{calib}})$, where

$$\bar{g}(\tau^i_{\text{calib}} \,|\, \bar{\ell}^i_{\text{calib}}) = \min_{k \in \{1,\ldots,T\}} g(s^i_{\text{calib}}(k) \,|\, \bar{\ell}^i_{\text{calib}}(k)). \tag{5}$$

Consider a new validation scenario drawn from $\mathcal{D}$ associated with a task $\phi_{\text{test}}$ that is defined over $N_{\text{test}}$ robots and horizon $H_{\text{test}}$ corresponding to a sequence of prompts

$$\bar{\ell}_{\text{test}} = \bar{\ell}_{\text{test}}(1), \ldots, \bar{\ell}_{\text{test}}(k), \ldots, \bar{\ell}_{\text{test}}(T_{\text{test}}),$$

where $T_{\text{test}} = H_{\text{test}} \cdot N_{\text{test}}$. Using the set $\mathcal{M}$, CP can generate a prediction set $\bar{\mathcal{C}}(\bar{\ell}_{\text{test}})$ of plans $\tau$, containing the correct one, denoted by $\tau_{\text{test}}$, with probability greater than $1-\alpha$, i.e.,

$$P(\tau_{\text{test}} \in \bar{\mathcal{C}}(\bar{\ell}_{\text{test}})) \geq 1-\alpha. \tag{6}$$

The prediction set $\bar{\mathcal{C}}(\bar{\ell}_{\text{test}})$ is defined as:

$$\bar{\mathcal{C}}(\bar{\ell}_{\text{test}}) = \{\tau \,|\, \bar{g}(\tau|\bar{\ell}_{\text{test}}) > 1-\bar{q}\}, \tag{7}$$

where $\bar{q}$ is the $\frac{(M+1)(1-\alpha)}{M}$ empirical quantile of $\bar{r}_1, \ldots, \bar{r}_M$. The size of the prediction sets can be used to evaluate uncertainty of the LLM. Specifically, if $|\bar{\mathcal{C}}(\bar{\ell}_{\text{test}})| = 1$, then this means that the LLM is certain with probability at least $1-\alpha$ that the designed plan accomplishes the assigned task.

On-the-fly and Local Construction: Notice that $\bar{\mathcal{C}}(\bar{\ell}_{\text{test}})$ in (7) is constructed after the entire sequence $\bar{\ell}_{\text{test}} = \bar{\ell}_{\text{test}}(1), \ldots, \bar{\ell}_{\text{test}}(T_{\text{test}})$ is obtained. However, at test time, we do not see the entire

[3]The distribution $\mathcal{D}$ induces a distribution over data sequences (3) [19]. These data sequences are equivalent representations of the sampled scenarios augmented with the correct decisions. Observe that in these sequences the order of robots is determined by an ordered set $\mathcal{I}_i$ (generated by the induced distribution). For ease of notation, in this section, we assume that the ordered set is defined as $\mathcal{I}_i = \{1, \ldots, N_i\}$ and $\mathcal{I}_{\text{test}} = \{1, \ldots, N_{\text{test}}\}$ for all calibration and validation sequences, respectively, and for all $t$. However, in general, these ordered sets do not need to be the same across the calibration sequences and/or across the time steps $t$ within the $i$-th calibration sequence. For instance, the induced distribution may randomly pick a set $\mathcal{I}$ from a finite set of possible sets $\mathcal{I}$ for each time $t$. Then, in Section III-C, when 'help from robots' is needed, a new set $\mathcal{I}$ from this finite set can be randomly selected.

sequence of prompts all at once; instead, the contexts $\bar{\ell}_{\text{test}}(k)$ are revealed sequentially over iterations $k$, as the robots pick their actions. Thus, next we construct the prediction set $\bar{\mathcal{C}}(\bar{\ell}_{\text{test}})$ on-the-fly and incrementally, using only the current and past information, as the robots take turns in querying their LLMs for action selection. At every iteration $k \in \{1, \ldots, T\}$, we construct the local prediction set associated with a robot $j$ and a time step $t$,

$$\mathcal{C}(\bar{\ell}_{\text{test}}(k)) = \{\tau_{\text{test}}(k) \mid g(\tau_{\text{test}}(k)|\bar{\ell}_{\text{test}}(k))) > 1 - \bar{q}\}, \quad (8)$$

where $g$ refers to the (uncalibrated) LLM score used in Section III-B. The prediction set $\mathcal{C}(\bar{\ell}_{\text{test}}(k)))$ in (8) should be used by the respective robot $j$ to select a decision $s_j(t)$. Essentially, $\mathcal{C}(\bar{\ell}_{\text{test}}(k)))$ corresponds to the set $\mathcal{C}(\ell_j(t)))$ used in Section III-B and in [line 10, Alg. 1]. For instance, the set $\mathcal{C}(\bar{\ell}_{\text{test}}(k))$ with $k = N_{\text{test}} + 1$ refers to the prediction set of robot $\mathcal{I}(1)$ at time $t = 2$. As it will be shown in Section IV, it holds that $\bar{\mathcal{C}}(\bar{\ell}_{\text{test}}) = \hat{\mathcal{C}}(\bar{\ell}_{\text{test}})$, where

$$\hat{\mathcal{C}}(\bar{\ell}_{\text{test}}) = \mathcal{C}(\bar{\ell}_{\text{test}}(1)) \times \cdots \times \mathcal{C}(\bar{\ell}_{\text{test}}(k)) \cdots \times \mathcal{C}(\bar{\ell}_{\text{test}}(T_{\text{test}})). \quad (9)$$

*Remark 3.2 (Prediction sets):* Construction of prediction sets requires generating a new calibration set for every new test scenario to ensure the desired coverage level. A dataset-conditional guarantee which holds for a fixed calibration set can also be applied [27]. Also, obtaining a non-empty prediction set for a fixed $\alpha$, requires $M \geq \lceil (M+1)(1-\alpha) \rceil$.

*Remark 3.3 (Efficiency Benefits):* Given a quantile $\bar{q}$ and the scores for each decision, the complexity of constructing the prediction sets $\mathcal{C}(\ell_j(t))$ for all robots $j$ at time $t$ is $O(N \cdot |\mathcal{S}|)$. This is notably more efficient than the centralized approach of Section III-A, where the corresponding complexity of constructing multi-robot/global prediction sets is $O(|\mathcal{S}|^N)$.

*Remark 3.4 (Multiple Feasible Solutions):* The above CP analysis assumes that $\mathcal{D}$ generates scenarios with a unique solution. To relax this assumption, the key modification lies in the construction of the calibration dataset. Specifically, for each calibration mission scenario, among all feasible plans, we select the one constructed by picking the decision with the highest LLM confidence score at each planning step. CP can then be applied as usual, yielding prediction sets that contain, with user-specified probability, the plan $\tau_{\text{test}}$ consisting of the decisions with the highest LLM confidence scores among all feasible decisions. This can also be formally shown by following the same steps as in the Appendices A3-A4 of [19].

*Remark 3.5 (i.i.d. Assumption):* The guarantees in (6) hold only if calibration and test scenarios are i.i.d. This can be relaxed by employing robust CP to obtain valid prediction sets for all distributions $\mathcal{D}'$ that are 'close' to $D$ (based on the $f-$divergence) [28].

## IV. Theoretical Mission Success Rate Guarantees

In this section, we show that Algorithm 1 achieves user-specified $1 - \alpha$ mission success rates. To show this, we need first to state the following result; its proof can be found in [29] and follows the same logic as the proof of Claim 1 in [19].

*Proposition 4.1:* The prediction set $\bar{\mathcal{C}}(\bar{\ell}_{\text{test}})$ defined in (7) is the same as the on-the-fly constructed prediction set $\hat{\mathcal{C}}(\bar{\ell}_{\text{test}})$ defined in (9), i.e., $\bar{\mathcal{C}}(\bar{\ell}_{\text{test}}) = \hat{\mathcal{C}}(\bar{\ell}_{\text{test}})$.

*Theorem 4.2 (Mission Success Rate):* Assume that prediction sets are constructed on-the-fly/causally with coverage level $1 - \alpha$ and that the robots seek help from a user whenever the local prediction set $\mathcal{C}(\bar{\ell}_{\text{test}}(k))$ - defined in (8) - is not singleton after $W$ attempts; see Section III-C. (a) Assuming error-free execution of the designed plans, the mission success rate over new test scenarios (and the randomness of the calibration sets) drawn from $\mathcal{D}$ is at least $1 - \alpha$. (b) If $\bar{g}(\tau|\bar{\ell}_{\text{test}})$, used in (8), models true conditional probabilities, then the average amount of user help, modeled by the average size of the prediction sets $\bar{\mathcal{C}}(\bar{\ell}_{\text{test}})$, is minimized among possible prediction schemes that achieve $1 - \alpha$ mission success rates.

*Proof:* (a) To show this result, we consider the following cases. Case I: We have that $|\mathcal{C}(\bar{\ell}_{\text{test}}(k))| = 1, \forall k \in \{1, \ldots, T_{\text{test}}\}$ (with or without the robots 'helping' each other) and $\tau_{\text{test}} \in \hat{C}(\bar{\ell}_{\text{test}})$ where $\tau_{\text{test}}$ is the correct plan and $\hat{C}(\bar{\ell}_{\text{test}})$ is defined as in (9). In this case, the robots will select the correct plan. Case II: We have that $\tau_{\text{test}} \in \hat{C}(\bar{\ell}_{\text{test}})$, regardless of the size of the local prediction sets $\mathcal{C}(\bar{\ell}_{\text{test}}(k)), \forall k \in \{1, \ldots, T_{\text{test}}\}$. Here, the robots will select the correct plan assuming users who faithfully provide help; otherwise a distribution shift may occur as calibration sequences are constructed using correct decisions. Case III: We have that $\tau_{\text{test}} \notin \hat{C}(\bar{\ell}_{\text{test}})$. The latter means that there exists at least one iteration $k$ such that $\tau_{\text{test}}(k) \notin \mathcal{C}(\bar{\ell}_{\text{test}}(k))$. In this case, the robots will compute an incorrect plan. Observe that the probability that either Case I or II will occur is equivalent to the probability $P(\tau_{\text{test}} \in \hat{C}(\bar{\ell}_{\text{test}}))$. Due to Proposition 4.1 and (6), we have that $P(\tau_{\text{test}} \in \hat{C}(\bar{\ell}_{\text{test}})) \geq 1 - \alpha$. Thus, either of Case I and II will occur with probability that is at least equal to $1 - \alpha$. Since Cases I-III are mutually and collectively exhaustive, we conclude that the probability that Case III will occur is less than $\alpha$. This implies that the mission success rate is at least $1 - \alpha$. (b) This result holds directly due to Theorem 1 in [30]. ■

*Remark 4.3 (Multiple Feasible Solutions):* The above results hold in case of multiple feasible solutions too. The proof of Theorem 4.2 follows the same steps, with the only differences being that (i) the correct plan $\tau_{\text{test}}$ refers to the feasible plan comprising the decisions with the highest LLM confidence scores at each step, and (ii) when user help is needed, the user selects the feasible action with the highest LLM confidence score from the prediction set. Thus, the robot must return to the user the scores for each action in the set.

## V. Experiments

In this section, we empirically validate S-ATLAS using the AI2THOR simulator [31] and compare it to the centralized baselines from Section III-A. Comparisons against planners that do not allow robots to ask for help can be found in the Appendix of [29]. In all case studies, we pick GPT-3.5, Llama-2-7b, and Llama-3-8b as the LLM for all robots.

### A. Empirical Validation of Mission Success Rates

We consider home service tasks defined over 12 semantic objects: 'Apple', 'Kettle', 'Tomato', 'Bread', 'Potato', and 'Knife'. The set $\mathcal{A}$ includes the actions 'go to, grab object, put object down, open door, remain idle'. The action 'remain idle' is useful when all sub-tasks in $\phi$ have been assigned to other robots or when the task has been accomplished before the given horizon $H$. Thus, we have that $|\mathcal{S}| = 28$.

We generate 110 test scenarios $\xi$ with $N \in \{1, 3, 10, 15\}$ in various environments. Each mission consists of $K$ sub-tasks and at most one safety requirement. Each sub-task requires moving

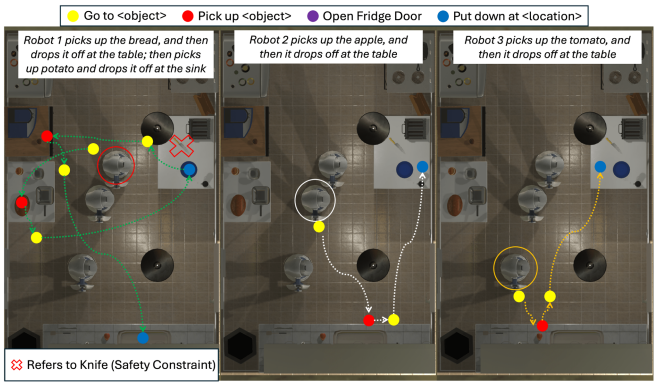


Fig. 4. Execution of a three-robot plan for the mission: 'Deliver the bread, apple and tomato to the table; Deliver the potato to the sink. Also, Robot 1 should never pick up a knife'. Each snapshot illustrates the plan, generated by S-ATLAS, followed by each robot to accomplish the task.

a specific object, possibly located inside closed containers (e.g., drawers and fridge), to either a desired location or one of several possible destinations. The safety constraint requires certain robots to avoid approaching/grabbing specific objects. We select $K \leq 4$ for $N = 1$, $4 \leq K \leq 8$ for $N \in \{3, 10\}$, and $K = 10$ when $N = 15$. Observe that $N \leq K$ implies that a robot will have to accomplish at least one sub-task while $N \leq K$ means that at least one robot will stay idle throughout the mission. An example of a task for $N = 3$, along with the plan designed by Algorithm 1 coupled with GPT 3.5, is shown in Fig. 4; see also [32]. Notice that the considered tasks may have multiple feasible plans and, therefore, CP is applied as in Remarks 3.4 and 4.3.

In what follows, we empirically validate the mission success rate guarantees provided in Section IV on our validation scenarios. For each validation scenario, we generate 30 calibration sequences to construct prediction sets. Then, for each scenario, we compute the plan using Algorithm 1 with $W = 0$ (to minimize API calls) and manually check if it is the correct one. We compute the ratio of how many of the corresponding 110 generated plans are the correct ones. We repeat this 50 times. The average ratio across all experiments is the (empirical) mission success rate. When $1 - \alpha = 0.90$ and $1 - \alpha = 0.95$, the mission success rate was 94.59% and 96.57% respectively, using GPT 3.5, validating Theorem 4.2. The average runtime to design a plan was 0.4 minutes. When $1 - \alpha = 0.90$ and $1 - \alpha = 0.95$, help was requested in making 4.44% and 8.1% of all decisions $s_j(t)$, respectively. As expected, the frequency of help requests increases as $1 - \alpha$ increases. Help rates also depend on the selected LLM. For instance, when our planner is paired with Llama 3-8b, which is significantly smaller and, therefore, less effective than GPT 3.5, the help rate increases to 8.9% and 15.6% for $1 - \alpha = 0.90$ and $1 - \alpha = 0.95$, respectively. When Llama 2-7b, which preceded Llama 3-8b, is considered, the help rates increase to 25% and 40% for $1 - \alpha = 0.90$ and $1 - \alpha = 0.95$, respectively.

### B. Comparisons Against Conformal Centralized Baselines

First, we compare our planner against the centralized baseline discussed in Section III-A, in terms of help rates and computational efficiency. Evaluating the baseline in multi-robot scenarios is computationally challenging and impractical due to the large number $|\mathcal{S}|^N$ of multi-robot decisions that need to be considered; e.g., for $N = 15$, we have $|\mathcal{S}|^N = 5.0977 \cdot 10^{21}$. Specifically, the baseline designs $\mathbf{s}(t)$ by solving a single MCQA problem with $|\mathcal{S}|^N$ choices. In contrast, our planner solves sequentially $N$ MCQA problems with $|\mathcal{S}|$ choices each. Also, recall that application of CP requires obtaining the LLM confidence scores for each option in the MCQA problem. To obtain them, our implementation requires one LLM query per option. Thus, the baseline requires $|\mathcal{S}|^N$ queries to apply CP at time $t$ (while ours require $N \cdot |\mathcal{S}|$). This results in prohibitively high API costs (using GPT 3.5) and further compromises the computational efficiency of the baseline (using either model). To enable comparisons, we consider small teams with $N = 2$ robots and $|\mathcal{S}| = 28$ using Llama 2-7b and Llama 3-8b; see Table I. We generate 20 test scenarios and, for each scenario, we collect 20 calibration sequences. We compute plans using S-ATLAS (with $W = 0$) and the baseline. We repeat this 50 times. Both methods were exposed to the same validation and calibration data. Observe in Table I that our method (i) requires significantly lower help rates from users than the baseline,[4] and (ii) computes plans faster.[5] We attribute these to the distributed nature of our planner enabling it to solve 'smaller' MCQA scenarios, compared to the baseline, to design $\mathbf{s}(t)$. [6]

TABLE I
COMPARISON OF S-ATLAS AGAINST THE BASELINE (B) FOR $N = 2$

| | Llama2-7b | | Llama3-8b | |
|---|---|---|---|---|
| | Ours | B | Ours | B |
| Runtime for plan design (mins) | 3 | 23 | 4 | 24 |
| Help Rate($1 - \alpha = 80\%$) | 16% | 62% | 1.4% | 13.4% |
| Help Rate($1 - \alpha = 90\%$) | 29% | 88% | 8% | 30% |

Second, we evaluate the baseline on the previously considered case studies, using the more efficient setup discussed in Remark 3.1. We query GPT-3.5 to generate a set of $X = 4$ multi-robot decisions as in [19]. Our empirical results show that the LLM's ability to generate options containing a valid choice decreases as $N$ increases. For $N = 1, 2, 3$ and 10, valid options were included in the LLM-generated set in only 95%, 85%, 48%, and 20% of the planning steps, leading to incomplete plans. Note that increasing $X$ does not necessarily yield better results; e.g., for $N = 3$, the corresponding percentages for $X = 6, 8$, and 10 are 52%, 48%, and 48%.

### C. Asking for Help From Robot Teammates

In this section, we select five scenarios from Section V-A, with $N = 3$, where non-singleton prediction sets were constructed using GPT 3.5. We show that re-computing $\mathbf{s}(t)$ using a different set $\mathcal{I}$ can potentially result in smaller prediction sets. In each of these scenarios, we select $W = 1$, and we observe that two scenarios switched to singleton prediction sets. In the first scenario, the task requires the robots to move a tomato and a water bottle to the sink, put the kettle to stove burner, and the

[4]We observed that larger sets of options tend to increase help rates.

[5]Note that these runtimes are implementation-specific. For example, our method (and, similarly, the baseline) can be accelerated by parallelizing the extraction of $|\mathcal{S}|$ LLM confidence scores during the construction of $\mathcal{C}(\ell_j(t))$.

[6]Our implementation of S-ATLAS, paired with Llama 3-8b, in an unseen test scenario with $N = 2$ robots, can be found on this https://drive.google.com/file/d/1Ds5wulsNRdXPb1KE6gJyMU9LZlUkcs1N/view?usp=sharinglink.

bread at the table. We initialize the ordered set of robot indices as $\mathcal{I} = \{1, 2, 3\}$. In this scenario, the bottle of water is inside the closed fridge. The prediction set $\mathcal{C}(\ell_1(1))$ for the robot $\mathcal{I}(1) = 1$ at $t = 1$ is non-singleton and contains the actions: (go to location of water bottle), (go to the location of the fridge). In this case, robot 1 (randomly) generates a new ordered set: $\mathcal{I}' = \{2, 1, 3\}$ and asks all robots to select their decisions as per $\mathcal{I}'$. Thus, the robot $\mathcal{I}'(1) = 2$ will select first an action. In this case, the corresponding prediction set is singleton defined as (go to the location of the fridge). Similar observations were made in the second scenario. We note again here that changing the set $\mathcal{I}$ may not always result in smaller prediction sets, as this minimally affects the prompts.

## VI. Conclusions and Future Work

We proposed S-ATLAS, a new multi-robot LLM-based planner that can achieve desired mission success rates. In our future work we plan to relax the assumption of perfect robot skills by leveraging CP to reason simultaneously about uncertainties arising from LLMs and robot skill execution.

## References


[1] C. R. Garrett et al., "Integrated task and motion planning," *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 4, no. 1, pp. 265–293, 2021.

[2] Z. Chen, Z. Zhou, S. Wang, J. Li, and Z. Kan, "Fast temporal logic mission planning of multiple robots: A planning decision tree approach," *IEEE Robot. Automat. Lett.*, vol. 9, no. 7, pp. 6146–6153, Jul. 2024.

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[4] I. Singh et al., "ProgPrompt: Generating situated robot task plans using large language models," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 11523–11530.

[5] J. Liang et al., "Code as policies: Language model programs for embodied control," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 9493–9500.

[6] S. Li et al., "Pre-trained language models for interactive decision-making," *Adv. Neural Inf. Process. Syst.*, vol. 35, pp. 31199–31212, 2022.

[7] M. Ahn et al., "Do as I can, not as I say: Grounding language in robotic affordances," in *Proc. 6th Conf. Robot Learn.*, 2023, vol. 205, pp. 287–318.

[8] Z. Yang et al., "Text2Reaction: Enabling reactive task planning using large language models," *IEEE Robot. Automat. Lett.*, vol. 9, no. 5, pp. 4003–4010, May 2024.

[9] C. Tang, D. Huang, W. Ge, W. Liu, and H. Zhang, "GraspGPT: Leveraging semantic knowledge from a large language model for task-oriented grasping," *IEEE Robot. Automat. Lett.*, vol. 8, no. 11, pp. 7551–7558, Nov. 2023.

[10] 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 robot task planning," in *Proc. 7th Conf. Robot Learn.*, 2023, vol. 229, pp. 23–72.

[11] Z. Mandi, S. Jain, and S. Song, "RoCo: Dialectic multi-robot collaboration with large language models," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2024, pp. 286–299.

[12] H. Zhang et al., "Building cooperative embodied agents modularly with large language models," in *Proc. Int. Conf. Learn. Representations*, 2024.

[13] S. Hong et al., "Metagpt: Meta programming for multi-agent collaborative framework," in *Proc. Int. Conf. Learn. Representations*, 2024. [Online]. Available: https://iclr.cc/virtual/2024/poster/18491

[14] Y. Chen, J. Arkin, Y. Zhang, N. Roy, and C. Fan, "Scalable multi-robot collaboration with large language models: Centralized or decentralized systems?," in *Proc. Int. Conf. Robot. Automat.*, 2024, pp. 4311–4317.

[15] B. Zhang et al., "Controlling large language model-based agents for large-scale decision-making: An actor-critic approach," in *Proc. Int. Conf. Learn. Representations*, 2024. [Online]. Available: https://iclr.cc/virtual/2024/22201?utm_source=chatgpt.com

[16] S. S. Kannan, V. L. Venkatesh, and B.-C. Min, "Smart-LLM: Smart multi-agent robot task planning using large language models," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, Abu Dhabi, UAE, Oct. 2024. [Online]. Available: https://iros2024-abudhabi.org/accepted-papers

[17] X. Liu, P. Li, W. Yang, D. Guo, and H. Liu, "Leveraging large language model for heterogeneous ad hoc teamwork collaboration," in *Robot.: Sci. Syst.*, 2024. [Online]. Available: https://www.roboticsproceedings.org/rss20/p033.html

[18] F. Zeng, W. Gan, Y. Wang, N. Liu, and P. S. Yu, "Large language models for robotics: A survey," 2023, *arXiv:2311.07226*.

[19] A. Z. Ren et al., "Robots that ask for help: Uncertainty alignment for large language model planners," in *Proc. 7th Conf. Robot Learn.*, 2023, vol. 229, pp. 661–682.

[20] A. N. Angelopoulos et al., "Conformal prediction: A gentle introduction," *Found. Trends Mach. Learn.*, vol. 16, no. 4, pp. 494–591, 2023.

[21] B. Kumar et al., "Conformal prediction with large language models for multi-choice question answering," in *Proc. ICML (Neural Conversational AI TEACH) Workshop*, 2023.

[22] J. Wang, J. Tong, K. Tan, Y. Vorobeychik, and Y. Kantaros, "Conformal temporal logic planning using large language models," 2023, *arXiv:2309.10092*.

[23] S. Su et al., "Collaborative multi-object tracking with conformal uncertainty propagation," *IEEE Robot. Automat. Lett.*, vol. 9, no. 4, pp. 3323–3330, Apr. 2024.

[24] J. Sun, Y. Jiang, J. Qiu, P. T. Nobel, M. Kochenderfer, and M. Schwager, "Conformal prediction for uncertainty-aware planning with diffusion dynamics model," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, vol. 36, pp. 80324–80337.

[25] L. Lindemann, M. Cleaveland, G. Shim, and G. J. Pappas, "Safe planning in dynamic environments using conformal prediction," *IEEE Robot. Automat. Lett.*, vol. 8, no. 8, pp. 5116–5123, Aug. 2023.

[26] A. Dixit, Z. Mei, M. Booker, M. Storey-Matsutani, A. Z. Ren, and A. Majumdar, "Perceive with confidence: Statistical safety assurances for navigation with learning-based perception," in *Proc. 8th Annu. Conf. Robot Learn.*, 2024.

[27] V. Vovk, "Conditional validity of inductive conformal predictors," in *Proc. Asian Conf. Mach. Learn.*, 2012, pp. 475–490.

[28] M. Cauchois, S. Gupta, A. Ali, and J. C. Duchi, "Robust validation: Confident predictions even when distributions shift," *J. Amer. Stat. Assoc.*, pp. 1–66, 2024. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/01621459.2023.2298037?scroll=top&needAccess=true

[29] J. Wang, G. He, and Y. Kantaros, "Safe task planning for language-instructed multi-robot systems using conformal prediction," 2024, *arXiv:2402.15368*.

[30] M. Sadinle, J. Lei, and L. Wasserman, "Least ambiguous set-valued classifiers with bounded error levels," *J. Amer. Stat. Assoc.*, vol. 114, no. 525, pp. 223–234, 2019.

[31] E. Kolve et al., "AI2-thor: An interactive 3D environment for visual AI," 2017, *arXiv:1712.05474*.

[32] *Demonstrations*, 2024. [Online]. Available: https://vimeo.com/962767783