2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Anticipate & Act : Integrating LLMs and Classical Planning for Efficient Task Execution in Household Environments †

Raghav Arora[1*], Shivam Singh[1*], Karthik Swaminathan[1], Ahana Datta[1], Snehasis Banerjee[1,2], Brojeshwar Bhowmick[2], Krishna Murthy Jatavallabhula[3], Mohan Sridharan[4], Madhava Krishna[1]

***Abstract*— Assistive agents performing household tasks such as making the bed or cooking breakfast often compute and execute actions that accomplish one task at a time. However, efficiency can be improved by anticipating upcoming tasks and computing an action sequence that jointly achieves these tasks. State-of-the-art methods for task anticipation use data-driven deep networks and Large Language Models (LLMs), but they do so at the level of high-level tasks and/or require many training examples. Our framework leverages the generic knowledge of LLMs through a small number of prompts to perform high-level task anticipation, using the anticipated tasks as goals in a classical planning system to compute a sequence of finer-granularity actions that jointly achieve these goals. We ground and evaluate our framework's abilities in realistic scenarios in the *VirtualHome* environment and demonstrate a 31% reduction in execution time compared with a system that does not consider upcoming tasks.**

***Index Terms*— Task anticipation, large language models, classical planning, assistive agent.**

I. Introduction

Consider an agent assisting humans with daily living tasks in a home. In the scenario in Fig. 1, these tasks include making the bed and cooking breakfast, with each task requiring the agent to compute and execute a sequence of finer-granularity actions, e.g., fetch the relevant ingredients to cook breakfast. Since the list of tasks can change based on resource constraints or the human's schedule, the agent is usually asked to complete one task at a time. However, the agent can be more efficient if, similar to a human, it anticipates and prepares for upcoming tasks while computing a plan of finer-granularity actions, e.g., it can fetch the ingredients for breakfast when it fetches milk to make coffee.

State-of-the-art methods for estimating future tasks or their costs formulate them as learning problems and use data-driven deep networks [1], [2]. There is also work on using Large Language Models (LLMs) for task planning [3]–[5]. However, these methods predict sequences of high-level tasks or require a large labeled training dataset to compute a sequence of fine-grained actions. They also make it difficult to leverage domain knowledge, adapt to environmental changes, or to understand the decisions made.

We pose high-level task anticipation and finer-granularity action execution as a combined prediction and planning problem. Our framework seeks to leverage the complementary strengths of data-driven estimation based on generic prior knowledge of household tasks, and planning based on domain-specific action theories. Specifically, our framework:

1. Leverages the generic knowledge encoded in LLMs using a small number of prompts describing potential sequences of high-level household tasks, in order to predict subsequent tasks given partial sequences of tasks.

2. Uses an action language to encode finer-granularity domain-specific knowledge of household tasks in the form of domain and agent attributes, agent actions, axioms governing change, and heuristics to guide planning.

3. Adapts a symbolic and heuristic classical planner to consider both the immediate and the anticipated tasks as goals, computing a sequence of actions to jointly minimize the cost of accomplishing these goals.

We prompt existing LLMs for high-level task anticipation, use the Planning Domain Definition Language (PDDL) [6] as the action language, and use the Fast Downward (FD) solver [7] to generate fine-granularity plans for any given task. We evaluate our framework's abilities in *VirtualHome*, a realistic simulation environment [8], and in complex household scenarios involving multiple tasks, rooms, objects, and actions. We present a 31% reduction in execution time and a 12% reduction in plan length compared to a system that does not anticipate upcoming tasks.

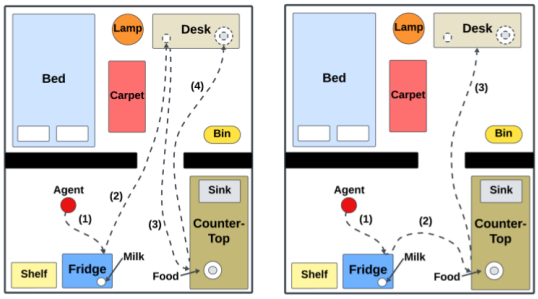

Fig. 1: Anticipation example: (a) Agent individually moves the milk and then the food to the desk; (b) Agent anticipates that milk needs to be served after food, jointly moving them to eliminate an extra trip.

II. Related Work

In this section, we review the related work to motivate our framework for task anticipation and plan generation.

LLMs for robotics: LLMs such as GPT-4 [9], PaLM [10], and Llama [11] are being used to address problems in robotics and AI. This includes generating "plans" to achieve goals [4], [12], [13], using descriptions of plans ex-

* Denotes equal contribution
[1] Robotics Research Center, IIIT Hyderabad, India
[2] TCS Research, Tata Consultancy Services, India
[3] CSAIL, Massachusetts Institute of Technology, USA
[4] IPAB, University of Edinburgh, UK
† This work was funded by TCS Research, India

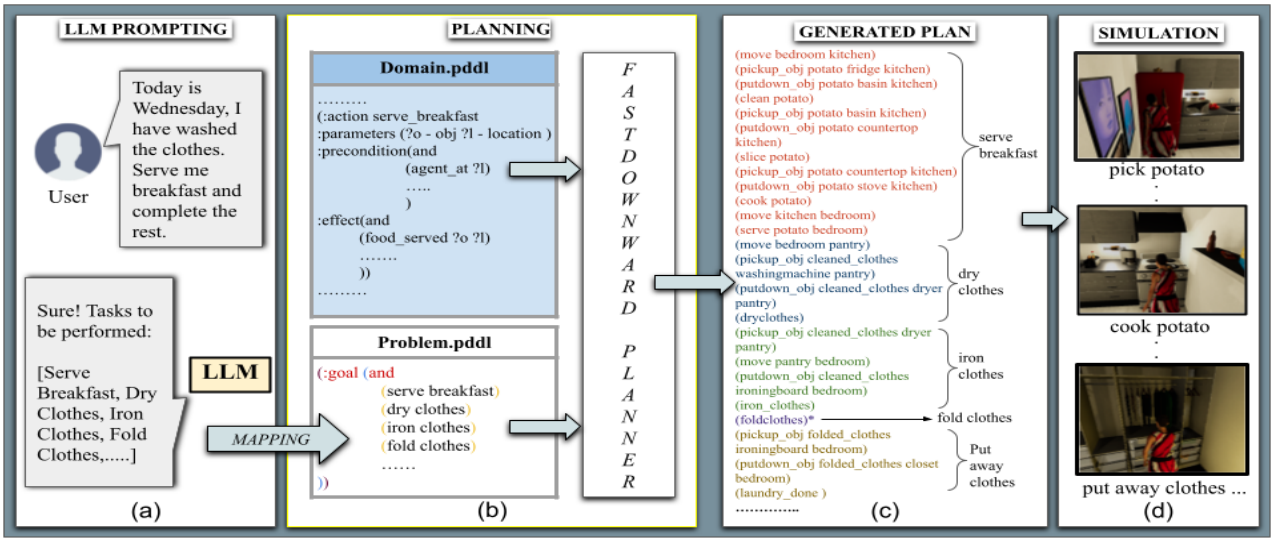

Fig. 2: Our framework's pipeline: (a) user inputs prompts with sequences of household tasks to an LLM, which then predicts high-level tasks over a time horizon; (b) the sequence of tasks is mapped to a joint goal state in a finer-granularity domain description in an action language (PDDL); (c) a heuristic planner (FD) uses this description to jointly compute the sequence of actions to be executed to complete all the tasks; and (d) the plan is executed in a realistic simulation environment.
*Actions corresponding to `fold clothes` are omitted due to space restrictions.

tracted from different sources [5], [14]–[16]. Some methods have proposed prompting strategies to validate and improve previously generated plans [17], [18]. Other methods have demonstrated that the LLM-based summarization can be used for perception and scene understanding [19], and to generate code for planning and robot manipulation [18], [20], [21]. Since LLMs are trained on a large amount of data from the Web to predict the text likely to appear next, researchers have questioned the ability of LLMs to plan (in the classical sense) based on prior domain knowledge [22], [23].

Task Anticipation: Knowledge-based and data-driven methods have been developed for task anticipation, with state-of-the-art methods using deep networks [2] and LLMs [24]. These methods predict high-level tasks or their costs in simplistic domains, with additional planning required to complete each such task, or require many examples to directly predict a sequence of finer-granularity actions to be executed. Our framework, on the other hand, leverages the complementary strengths of LLMs and classical planning. It enables an agent to operate in complex environments by anticipating high-level tasks based on limited prompts to an LLM, using these tasks as goals to be achieved jointly by planning a sequence of finer-granularity actions based on domain-specific knowledge.

Integrating LLMs with PDDL: Given the existing literature on using PDDL to encode prior knowledge for planning [6], recent papers have emphasized the need for such planning in combination with LLMs in complex domains [18], [25]. LLMs have been used to generate (or translate prior knowledge to) goal states to be achieved by a classical (PDDL-based) planner [26], [27]. However, research has also indicated that methods based on deep networks and LLMs are not well-suited for multistep, multilevel decision-making (in the classical sense) by reasoning with domain knowledge [23].

III. Problem Formulation and Framework

Consider an assistive agent asked to complete a routine, i.e., a sequence of high-level tasks $\mathcal{R} = \{\tau_1, \tau_2, ...\tau_n\}$. In a household environment, each τ_i is one of a set $\mathcal{T}$ of known tasks, e.g., *make the bed* or *make breakfast*, which requires the robot to compute and execute a plan, i.e., a sequence of finer-granularity actions $\{a_1, \ldots, a_{m_i}\}$. For example, to *water the plants*, the agent has to *bring the water hose to the garden*, *connect the hose to the tap*, and *turn the tap on*. Since the agent can be assigned different sequences of high-level tasks, it typically considers one task (in a given sequence) at a time, computing a sequence of finer-granularity actions to complete the task at a minimum cost (or time, effort). In doing so, the agent may fail to leverage an opportunity to reduce the cost of completing a subsequent task, e.g., when the agent is fetching milk from the fridge for coffee, it can also get the ingredients for making breakfast. Our framework in Figure 2 leverages the generic knowledge of LLMs to anticipate one or more high-level tasks given a partial sequence and a limited number of prompts. The anticipated tasks are considered as goals in a classical planner that computes a sequence of actions to jointly achieve these goals. We describe our framework's components below.

A. LLMs for task anticipation

LLMs like GPT-3 [28] can be tuned to predict patterns (of tasks) in specific domains. In our household domain, the high-level tasks are daily living tasks, e.g., *iron clothes* and *vacuum the house*, that are often performed at specific times and in a specific order, e.g., *wash clothes* and *dry clothes* are completed before *iron clothes*. Our framework uses an LLM to extract these task patterns from a small number of task routines provided as prompts. As described in Section IV-A, these routines can have $\approx$ 20 high-level tasks. Given a partially specified routine, the LLM can predict tasks that the agent is likely to be asked to complete next—see Figure 2(a).

```
(:action dusting
 :parameters(?o - obj ?l - location)
 :precondition(and(In_hand DustMop)
                  (agent_at ?l)
                  (not(dusted ?o ?l)))
 :effect(and(dusted ?o ?l)(increase (
     total-cost)10))
)
```

Fig. 3: Action for *Dusting* the object o at some location l.

Our choice of using the LLMs to model and predict sequences of high-level tasks is motivated by two objectives: (i) exploiting the complementary strengths of generic LLMs and domain-specific knowledge-based planning methods; and (ii) leveraging the capabilities of an LLM with limited examples of routines of interest. As described in Section IV-A, we explored the use of popular LLMs such as PaLM [10], GPT-3.5 [28], and GPT-4 [9]. We also explored the effect of using context-specific examples during training or execution.

B. Action Planning with Anticipated Goals

The tasks anticipated by the LLM are considered as goals, and our framework uses a classical planner to compute the sequence of finer-granularity actions to be executed to jointly achieve these goals. As stated earlier, the planner uses domain-specific knowledge in the form of a theory of actions; we use the STRIPS [29] subset of PDDL [6] enriched with types, negative preconditions, and action costs as the action language to describe this theory. We also focus on goal-based problems and discrete actions with deterministic effects; other action languages can be used to represent durative actions [30] or non-determinism [31].

For any given domain, statements in PDDL describe the `domain` and the `problem` to the solved. The domain description $\mathcal{D} = \langle \mathcal{S}, \mathcal{H} \rangle$ comprises a signature $\mathcal{S}$ and a theory $\mathcal{H}$ governing the domain dynamics. The signature $\mathcal{S}$ includes a specification of *types* such as *location*, *object*, *receptacle* and *agent*; *constants* such as *kitchen* and *garden* that are specific instances of the types; and *predicates* that include *fluents*, *statics*, and *actions*. Fluents such as *(agent_at ?l - location)*, *(obj_at ?o - obj ?l - location)*, and *(dropped ?o1 - obj ?r - receptacle ?l - location)* represent domain attributes whose values can change over time as a result of actions; *statics* are domain attributes whose values do not change over time; and *actions* such as *move_agent*, *cook*, *serve*, and *pickup* change the value of relevant fluents. $\mathcal{D}$ also includes a specification of each action in terms of its parameters, preconditions that need to be true for the action to be executed, effects that will be true once the action is executed, and the cost of executing it. For example, Figure 3 provides the specification of the *dusting* action; the preconditions are that the agent have a mop in its hand and be in a location with an object that needs to be dusted.

The problem description $\mathcal{P} = \langle \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$ describes the specific scenario under consideration in terms of the set $\mathcal{O}$ of specific objects, the initial state $\mathcal{I}$ comprising ground literals

Without Anticipation Plan	With Anticipation Plan
(move bedroom pantry) (pickup lawnmower pantry) (move pantry garden) (putdown lawnmower garden) (switch_on lawnmower garden) (cutting_the_grass) (move garden pantry) (pickup watering_hose pantry) (move pantry garden) (putdown watering_hose garden) (switch_on watering_hose garden) (pickup watering_hose garden) (water_the_plants) **; cost = 347 (general cost)** **; plan length = 13**	(move bedroom pantry) (pickup watering_hose pantry) (pickup lawnmower pantry) (move pantry garden) (putdown lawnmower garden) (putdown watering_hose garden) (switch_on lawnmower garden) (cutting_the_grass) (switch_on watering_hose garden) (pickup watering_hose garden) (water_the_plants) **; cost = 307 (general cost)** **; plan length = 11**

Fig. 4: Example plans produced with and without considering anticipated tasks.

of the fluents and statics, and a description $\mathcal{G}$ of the goal state in the form of relevant ground literals. The planning task is to compute a sequence of actions $\pi = (a_1, \ldots, a_K)$ that takes the system from $\mathcal{I}$ to a state where $\mathcal{G}$ is satisfied. In our case, we compute plans using the state of the art Fast Downward (FD) system in the *Autotune* configuration [7]. This heuristic planner adapts its parameters based on instances of the domain under consideration and supports different heuristics and options. We focus on minimizing the `total cost` C of the plan, i.e., if c_k^j is the cost of action a_k^j in plan π^j, the objective would be to compute:

$$\pi^* = \arg\min_{\pi^j} C(\pi^j), \quad C(\pi^j) = \sum_{k=0}^{K} c_k^j$$

where the cost of each aciton corresponds to the time taken by the agent to execute it. $\mathcal{P}$ also includes some helper statements that guide this search for plans. Recall that the agent will typically focus on computing a plan for one high-level task at a time, e.g., left panel of Figure 4; when the agent tries to compute an action sequence to jointly achieve multiple high-level goals, the overall plan length and execution cost can be reduced, e.g., right panel of Figure 4.

IV. Experimental Setup and Results

We experimentally evaluated four hypotheses:

H1: LLMs are able to accurately anticipate future tasks based on a small number of prompts of task routines.

H2: LLMs can take into account specific contextual information for task anticipation.

H3: Considering anticipated tasks as joint goals reduces plan length and plan execution time compared with considering one task at a time.

H4: Our framework allows the agent to adapt to unexpected successes and failures by interrupting plan execution and replanning if necessary.

We evaluated H1 using three LLMs: PaLM [10], GPT-3.5 [28] and GPT-4 [9]. For the other hypotheses we used GPT-4 as the default LLM. For H3, we computed plans using different configurations of the Fast-Downward system [7], and we evaluated H4 qualitatively.

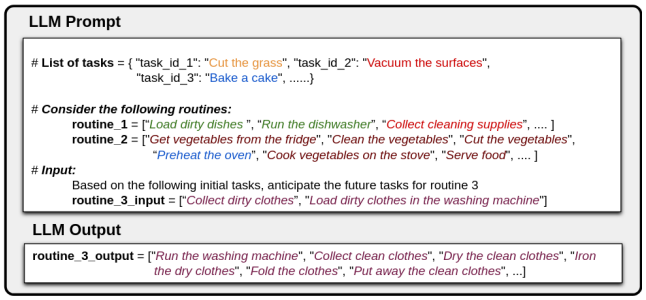

Fig. 5: LLM prompting example.

A. Experimental Setup

We first describe the setup process we followed to experimentally evaluate the hypotheses.

1) Prompting LLMs and Planning: We created a dataset $\mathcal{T}$ of high-level tasks in the household environment. These tasks belong to activities such as *Cooking, Cleaning, Washing, Baking*, and *Gardening*. We then generated multiple task routines $\mathcal{R}_i$, each with ≈ 20 tasks, by sampling tasks from different activities while preserving the relative order of tasks within each activity. This process helped us create task routines spanning activities of daily living in a home.

We conducted experiments under two configurations. In both configurations, the dataset $\mathcal{T}$ was provided to the LLMs (in JSON format) to minimize hallucinations and to ground our LLM outputs since these outputs are mapped to goal descriptions based on closed set of PDDL statements. Tasks not in $\mathcal{T}$ will be ignored and handling such tasks is beyond the scope of this paper. In the **without context** configuration, the task routines followed during two individual days were given as input (prompt) to the LLM, which was asked to complete a partially-specified routine comprising two tasks. Figure 5 shows an example prompt and LLM output at this stage. In the **with context** configuration, in addition to the two task routines (as before), we used a method similar to [18] to provide two contextual examples in form of the partially-specified task inputs and the corresponding expected LLM outputs. The difference between the two configurations was thus the additional contextual prompting provided in the second configuration to guide the LLM. We considered four task anticipation performance measures:

- *Miss Ratio (Miss.)*: ratio of number of tasks **not** anticipated to the length of the sampled sequence.
- *Partial Ordering Count (POC)*: measures capability to maintain the relative order of tasks in the routines.
- *Kendall rank correlation coefficient (KRCC) [32]*: measures match between predicted and actual task order.

$$KRCC = \frac{n_c - n_d}{\sqrt{(n_0 - n_1)(n_0 - n_2)}}$$

where: n_c is the number of concordant pairs, n_d is the number of discordant pairs, $n_0 = n(n-1)/2$ is the total number of pairs, n_1 and n_2 are the sums of ties in the first and second sequences. A pair of tasks (τ_1, τ_2), (τ_3, τ_4) is said to be concordant if $\text{rank}(\tau_1) > \text{rank}(\tau_3)$ and $\text{rank}(\tau_2) > \text{rank}(\tau_4)$ or $\text{rank}(\tau_1) < \text{rank}(\tau_3)$ and $\text{rank}(\tau_2) < \text{rank}(\tau_4)$. If the ranks disagree, the pair is discordant. A tie can occur when there is a repetition of tasks in the anticipated routine, but since we do not consider task repetition, n_2 will always be zero.
- *Success Ratio*: Fraction of experiments (i.e., prompts) for which tasks were anticipated correctly by the LLM.

Recall that to perform any high-level task, the agent has to plan a sequence of finer-granularity actions. In our experiments, the number of actions required to accomplish any given task varied from one to 16, with the initial domain description (for planning) comprising 33 independent actions, five different rooms, 33 objects distributed over 5-10 types, and 19 receptacles. *We thus set up a complex domain for experimental analysis compared with other papers that have explored task anticipation or combined LLMs with PDDL-like representations for planning.*

2) Baseline: As a baseline for the task anticipation capability of the LLMs, we sampled 100 routines of tasks and created a probability transition matrix representing the likelihood of transitioning from one task to another within the dataset: $P(\tau_j|\tau_i) = \frac{Count(\tau_i,\tau_j)}{Count(\tau_i)}$ where $P(\tau_j|\tau_i)$ is the probability of transitioning from task τ_i to task τ_j, $Count(\tau_i, \tau_j)$ denotes the number of occurrences of the transition from task τ_i to task τ_j in the dataset, and $Count(\tau_i)$ represents the total number of occurrences of task τ_i in the dataset. We then created a Markov chain of the tasks such that given an initial task, subsequent tasks are obtained by repeatedly sampling from the probability transition matrix. For the planning experiments, the baseline was planning without considering any anticipated tasks.

B. Experimental Results

Next, we describe the results of experimental evaluation.

1) Evaluating H1 and H2: To evaluate **H1**, we evaluated the ability of PaLM, GPT-3.5 and GPT-4 to anticipate future tasks, as stated in Section IV-A.1. We ran 500 experiments with tasks sampled from our household dataset, with the corresponding results summarized in Table I. Even in the absence of contextual examples, all LLMs maintained the order of tasks in a routine. However, PaLM was not able to correctly anticipate all the tasks; it missed $approx 36\%$ of the tasks. When the LLMs had access to the contextual examples, all three provided very good performance, with GPT-4 providing the correct task order 100% of the time along with a very low Miss Ratio (0.06%). Overall, the LLMs were able to anticipate tasks based on a limited number of prompts, with performance varying between the three LLMs; all three LLMs provided good task anticipation performance when they had access to the contextual examples. These results support hypotheses **H1** and **H2**.

Next, we experimented with GPT-3.5-turbo and GPT-4 under specific conditions. Specifically, we arranged tasks from our dataset in a weekly schedule. After providing the LLM with the routine of tasks during each day of the week, we posed a special prompt that deviated from the expected

LLMs	Without context			With Context		
	Miss Ratio (Miss.) ⇓	Partial Ordering Count (POC) ⇑	$KRCC$ ⇑	Miss Ratio (Miss.) ⇓	Partial Ordering Count (POC) ⇑	$KRCC$ ⇑
PaLM	0.361	0.974	0.993	0.034	0.994	0.996
GPT-3.5-turbo	0.282	0.676	0.906	0.0698	0.806	0.976
GPT-4	**0.037**	**0.960**	**0.995**	**0.0006**	**1.0**	**1.0**

TABLE I: Task anticipation performance of LLMs. We ran 500 experiments with ≈ 20 tasks per experiment. We observed a significant increase in performance after providing contextual prompts. ⇓ (⇑) implies that lower (higher) values of the corresponding measure represents better performance. Results support **H1** and **H2**.

	Success Ratio ⇑
GPT-3.5-turbo	0.65
GPT-4	**0.8**

TABLE II: Task anticipation of two LLMs with contextual information. Success ratio expressed as a fraction of tasks averaged over 20 experiments. Results support **H2**.

	Miss. ⇓	POC ⇑	KRCC ⇑	Incorr. ⇓	Repeat ⇓
GPT-4	**0.0006**	**1.0**	**1.0**	**0**	**0**
Markovian	0.413	0.364	0.908	6.28	1.49

TABLE III: Task anticipation of LLMs compared with the Markovian baseline. Values of measures expressed as an average over 500 experiments. Column "Incorr." summarizes the number of anticipated tasks that were incorrect, and "Repeat" shows the average number of repetitions per experiment. Results support **H1**.

Number of tasks anticipated →		0(Myopic)	3 (Ours)	6 (Ours)
Planner	Parameters			
AT-1	Plan Length	70.3	65.2	61.8
	Exe. Time	2051	1658	1390
LAMA	Plan Length	65.7	62.5	61.2
	Exe. Time	1835	1613	1599
AT-2	Plan Length	67.2	64.3	60.2
	Exe. Time	1847	1591	1377

TABLE IV: Planning and execution performance with three different values of the number of anticipated tasks, based on three different configurations of the Fast Downward planner; AT1, AT2, and LAMA correspond to configurations *seq-sat-fd-autotune-1*, *seq-sat-fd-autotune-2*, and *seq-sat-lama-2011* respectively. *Exe. Time* denotes the plan execution time in seconds. Results support hypothesis **H3**.

routine. One example of such a prompt is: "Today is a Monday. I have an urgent meeting in the morning." We observed that LLMs were able to respond to such prompts and respect the constraint imposed by the prompt while generating the anticipated tasks. For the specific prompt (above), the output included most of the expected tasks for Monday and two extra tasks to set up a laptop and prepare clean clothes right after breakfast. We ran 20 such experiments, and the results are summarised in Table II. These results further support **H2**.

Next, we compared the task anticipation ability of LLMs with the first-order Markovian baseline described in Section IV-A.2. For the baseline, we generated the routine of tasks based on the transition probability function of the baseline, with the results summarized in Table III. Since the baseline was based on a Markov chain, there was no way for the system to recover if it reached a faulty state; it just continued to sample tasks from the faulty state. The deviation of the baseline from the correct routine (and the LLM's output) is shown in the columns labeled *Incorr.* and *Repeat* in Table III. Unlike this baseline, the LLMs were able to learn from limited prompts representing domain-specific preferences. These results further support **H1**.

2) Evaluating hypothesis H3: To measure the impact of considering the anticipated tasks during planning, we measured the cost of executing the computed sequence of finer-granularity actions. Specifically, we prompted the LLM with contextual examples and asked it to provide different number of anticipated tasks for a partially-specified routine. These anticipated tasks were considered as joint goals by the planning system, with the resultant plan of actions being executed by the agent. Since the cost of the actions (in the domain description) was based on the execution time, we used the total cost of any executed plan as the execution time (in seconds) of the plan. The results of these experiments are summarized in Table IV.

In these experiments, we varied the number of anticipated tasks from zero ("Myopic"), with the agent planning to perform one task at a time, to six, i.e., with six tasks anticipated and considered jointly during planning. As the number of anticipated tasks increased, the search time limit provided to the planner was increased by units of 30 seconds. For example, when the agent jointly planned an action sequence for the current task and the next (anticipated) task, the search time limit was set to 60 seconds. In addition, we organized these experiments as paired trials with the initial conditions and the overall sequence of assigned tasks being the same in each paired trial across the different number of anticipated tasks. Each value reported in Table IV is the average of 10 repetitions of the corresponding experiment.

Table IV shows the plan length and execution time as the number of anticipated tasks changes from zero to three and six, under three configuration options supported by FD: *seq-sat-fd-autotune-1, seq-sat-lama-2011*, and *seq-sat-fd-autotune-2*. For each configuration, as the number of anticipated tasks increased, the plan length decreased. There may be positive interaction between goals, with actions executed to achieve one goal leading to conditions that are essential preconditions for actions to be executed to achieve a subsequent goal. In such cases, planning jointly for the two goals will not make a big difference compared with pursuing one goal at a time. However, positive interaction between goals was unlikely in our experiments because tasks were anticipated by the LLM at a high level of abstraction. The observed reduction in plan length in Table IV thus indicates that anticipating and planning to jointly achieve multiple tasks enables the agent to complete all the tasks by executing fewer actions compared with not considering anticipated tasks. Furthermore, the plan execution time decreased with an increase in the number of anticipated tasks, indicating that the agent became more efficient when it interleaved the actions for different tasks. In these planning trials, the agent often had to use the available planning time limit to compute

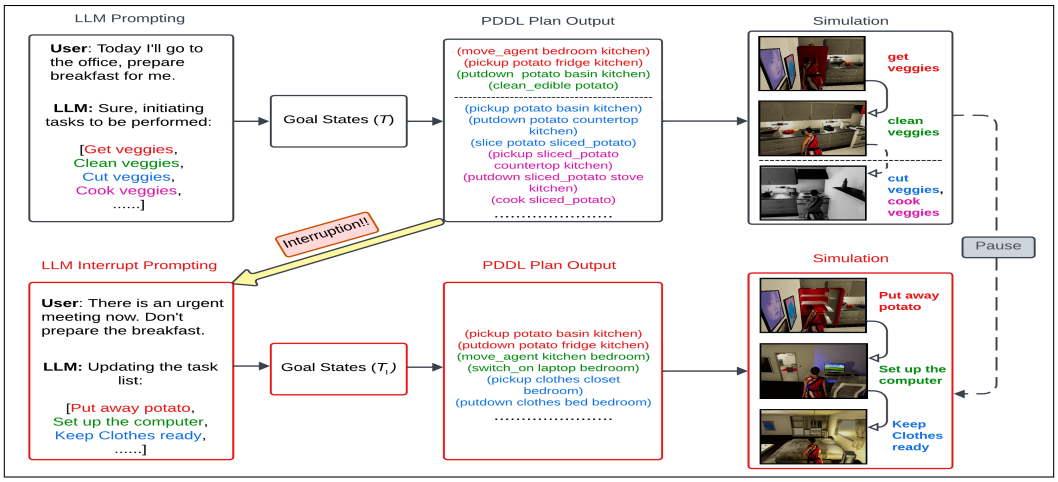

Fig. 6: An illustrative use case that involved an interruption during the execution of a plan computed to jointly accomplish some anticipated tasks; the agent was able to revise the anticipated tasks and replan as appropriate.

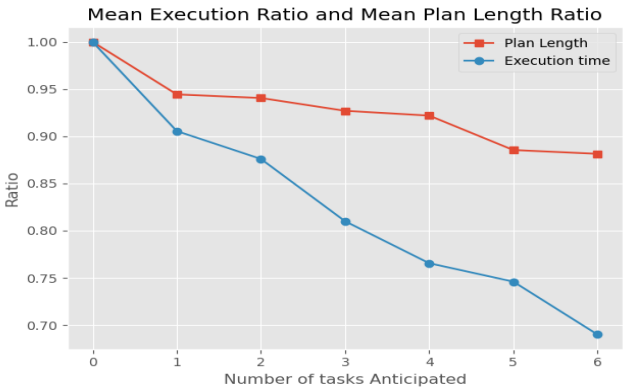

Fig. 7: Plan execution time and plan length with the number of anticipated tasks ranging from 1-6. Values expressed as ratio of the corresponding values in the myopic case (x=0). We observed a reduction of 31% in execution cost, and 12% in plan length.

the plans, particularly when the task required it to sequence multiple actions and when multiple anticipated tasks had to be considered. As a result, although there was a reduction in plan length and execution time, the planning time reached a plateau and did not change much during the experiments summarized in Table IV. These results support **H3**.

Figure 7 further illustrates the results of these planning experiments for the *seq-sat-fd-autotune-1* configuration. Since the initial state and set of tasks varied between each set of paired trials, averaging the numbers (e.g., for execution time) across these trials may not be meaningful. In each paired trial, we thus expressed the execution time and plan length of each instance that involved one or more anticipated tasks, as a fraction of the corresponding execution time and plan length (respectively) obtained without any task anticipation (i.e., "myopic"). The average of these fractions over 10 repetitions is shown in Figure 7. We observed a substantial ($\approx 31\%$) reduction in execution time and plan length ($\approx 12\%$) as the number of anticipated tasks increased to six. We thus conclude that anticipating future tasks and considering them during planning leads to more efficient performance; these results further support **H3**.

3) Evaluating hypothesis H4: Finally, we qualitatively evaluated H4. In these experiments, the execution of a plan generated by a planner was randomly interrupted by the user. Each interrupt was accompanied by a prompt by the user, leading to a change in the agent's operation. In particular, the state of the environment before the interrupt and the prompt (used for interrupting) were sent to the LLM to generate a new routine of tasks to be accomplished by the agent. While some actions in our domain are irreversible, e.g., we cannot put the pieces of a cut fruit back together, our combination of LLMs and action planning was able to undo the effects of some actions when that is appropriate. For example, in Figure 6, action execution was interrupted by the prompt "There is an urgent meeting now. Do not prepare breakfast". In this situation, the agent was able to plan suitable actions and put the vegetables back in their original location because it was no longer necessary to cook a hot breakfast; the agent instead computed action sequences to prepare the laptop and suitable clothes for the meeting.

V. Conclusion and Future Work

This paper described a framework for task anticipation and action execution by an agent in complex household environments. The framework leverages the generic knowledge encoded in LLMs for high-level task anticipation based on limited prompts, and plans a sequence of finer-granularity actions based on the domain-specific knowledge encoded in PDDL to jointly accomplish the anticipated tasks. We demonstrate a substantial reduction in action execution time and plan length in comparison with a planning system that does not consider anticipated actions. In future work, we intend to explore scalability of this framework to more complex domains, incorporate probabilistic planning, and explore implementation on a physical robot assistant. We will also explore the ability of LLMs to automatically adapt task anticipation to the preferences and needs of specific individuals in the household in which the agent is operating.

References

[1] P. Schydlo, M. Rakovic, L. Jamone, and J. Santos-Victor, "Anticipation in human-robot cooperation: A recurrent neural network approach for multiple action sequences prediction," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 5909–5914.

[2] R. Dhakal, M. R. H. Talukder, and G. J. Stein, "Anticipatory Planning: Improving Long-Lived Planning by Estimating Expected Cost of Future Tasks," in *IEEE International Conference on Robotics and Automation*, London, UK, 2023, pp. 11 538–11 545.

[3] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, P. Sermanet, T. Jackson, N. Brown, L. Luu, S. Levine, K. Hausman, and b. ichter, "Inner monologue: Embodied reasoning through planning with language models," in *International Conference on Robot Learning*, 14–18 Dec 2023, pp. 1769–1782.

[4] Y. Ding, X. Zhang, C. Paxton, and S. Zhang, "Task and motion planning with large language models for object rearrangement," 2023.

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

[6] M. Ghallab, C. Knoblock, D. Wilkins, A. Barrett, D. Christianson, M. Friedman, C. Kwok, K. Golden, S. Penberthy, D. Smith, Y. Sun, and D. Weld, "Pddl - the planning domain definition language," 08 1998.

[7] M. Helmert, "The fast downward planning system," *Journal of Artificial Intelligence Research*, vol. 26, pp. 191–246, jul 2006. [Online]. Available: https://doi.org/10.1613%2Fjair.1705

[8] X. Puig, K. Ra, M. Boben, J. Li, T. Wang, S. Fidler, and A. Torralba, "Virtualhome: Simulating household activities via programs," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8494–8502.

[9] OpenAI, "Gpt-4 technical report," *ArXiv*, vol. abs/2303.08774, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:257532815

[10] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel, "Palm: Scaling language modeling with pathways," 2022.

[11] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozire, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," 2023.

[12] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, P. Sermanet, N. Brown, T. Jackson, L. Luu, S. Levine, K. Hausman, and B. Ichter, "Inner monologue: Embodied reasoning through planning with language models," 2022.

[13] P. Sharma, A. Torralba, and J. Andreas, "Skill induction and planning with latent language," in *Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 1713–1726. [Online]. Available: https://aclanthology.org/2022.acl-long.120

[14] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. J. Joshi, R. C. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. M. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, and M. Yan, "Do as i can, not as i say: Grounding language in robotic affordances," in *Conference on Robot Learning*, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:247939706

[15] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," *International Conference on Machine Learning*. [Online]. Available: https://par.nsf.gov/biblio/10366294

[16] B. Y. Lin, Y. Fu, K. Yang, P. Ammanabrolu, F. Brahman, S. Huang, C. Bhagavatula, Y. Choi, and X. Ren, "Swiftsage: A generative agent with fast and slow thinking for complex interactive tasks," *ArXiv preprint*, vol. abs/2305.17390, 2023. [Online]. Available: https://arxiv.org/abs/2305.17390

[17] S. S. Raman, V. Cohen, E. Rosen, I. Idrees, D. Paulius, and S. Tellex, "Planning with large language models via corrective re-prompting," 2022.

[18] T. Silver, S. Dan, K. Srinivas, J. B. Tenenbaum, L. P. Kaelbling, and M. Katz, "Generalized planning in pddl domains with pretrained large language models," 2023.

[19] J. Wu, R. Antonova, A. Kan, M. Lepert, A. Zeng, S. Song, J. Bohg, S. Rusinkiewicz, and T. Funkhouser, "Tidybot: Personalized robot assistance with large language models," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[20] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, "Progprompt: Generating situated robot task plans using large language models," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 11 523–11 530.

[21] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, "Voxposer: Composable 3d value maps for robotic manipulation with language models," *arXiv preprint arXiv:2307.05973*, 2023.

[22] K. M. Collins, C. Wong, J. Feng, M. Wei, and J. B. Tenenbaum, "Structured, flexible, and robust: benchmarking and improving large language models towards more human-like behavior in out-of-distribution reasoning tasks," 2022.

[23] K. Valmeekam, A. Olmo, S. Sreedharan, and S. Kambhampati, "Large language models still can't plan (a benchmark for llms on planning and reasoning about change)," 2023.

[24] Q. Zhao, C. Zhang, S. Wang, C. Fu, N. Agarwal, K. Lee, and C. Sun, "Antgpt: Can large language models help long-term action anticipation from videos?" 2023.

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

[26] 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," 2023.

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

[28] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.

[Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

[29] R. E. Fikes and N. J. Nilsson, "Strips: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence*, vol. 2, no. 3, pp. 189–208, 1971. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0004370271900105

[30] A. Gerevini, A. Saetti, and I. Serina, "An empirical analysis of some heuristic features for planning through local search and action graphs," *Fundamental Information*, vol. 107, no. 2-3, pp. 167–197, 2011.

[31] M. Sridharan, M. Gelfond, S. Zhang, and J. Wyatt, "REBA: A Refinement-Based Architecture for Knowledge Representation and Reasoning in Robotics," *Journal of Artificial Intelligence Research*, vol. 65, pp. 87–180, May 2019.

[32] M. G. Kendall, "The treatment of ties in ranking problems," *Biometrika*, vol. 33, no. 3, pp. 239–251, 1945. [Online]. Available: http://www.jstor.org/stable/2332303