HiAgent: Hierarchical Working Memory Management for Solving Long-Horizon Agent Tasks with Large Language Model

Mengkang Hu♠ **Tianxing Chen**♠ **Qiguang Chen**♡ **Yao Mu**♠
Wenqi Shao♢ **Ping Luo**♠*
♠ The University of Hong Kong ♡ Harbin Institution of Technology
♢ Shanghai Artificial Intelligence Laboratory
https://github.com/HiAgent2024/HiAgent

Abstract

Large Language Model (LLM)-based agents exhibit significant potential across various domains, operating as interactive systems that process environmental observations to generate executable actions for target tasks. The effectiveness of these agents is significantly influenced by their memory mechanism, which records historical experiences as sequences of action-observation pairs. We categorize memory into two types: cross-trial memory, accumulated across multiple attempts, and in-trial memory (*working memory*), accumulated within a single attempt. While considerable research has optimized performance through cross-trial memory, the enhancement of agent performance through improved working memory utilization remains underexplored. Instead, existing approaches often involve directly inputting entire historical action-observation pairs into LLMs, leading to redundancy in long-horizon tasks. Inspired by human problem-solving strategies, this paper introduces HIAGENT, a framework that leverages subgoals as memory chunks to manage the working memory of LLM-based agents hierarchically. Specifically, HIAGENT prompts LLMs to formulate subgoals before generating executable actions and enables LLMs to decide proactively to replace previous subgoals with summarized observations, retaining only the action-observation pairs relevant to the current subgoal. Experimental results across five long-horizon tasks demonstrate that HIAGENT achieves a twofold increase in success rate and reduces the average number of steps required by 3.8. Additionally, our analysis shows that HIAGENT consistently improves performance across various steps, highlighting its robustness and generalizability.

1 Introduction

Owing to the development of powerful reasoning capabilities of Large Language Models (LLMs)

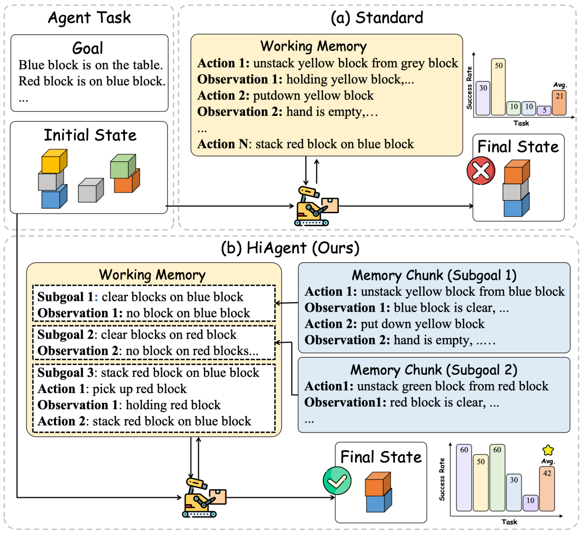

Figure 1: **Top right**: A commonly adopted paradigm STANDARD for LLM-based agents includes: i) prompts LLMs to generate one action; ii) executes the generated action and then append the obtained observation to the LLM's context (working memory); and iii) generates the next action. **Bottom**: Instead of incorporating all historical action-observation pairs into the working memory, HIAGENT leverage subgoals as memory chunks, with a summarized observation as the observation for each memory chunk. HIAGENT achieves an average success rate improvement of twofold (42 vs. 21) across five long-horizon tasks.

in recent years (OpenAI, 2022, 2023; Meta AI, 2024; Touvron et al., 2023; Jiang et al., 2023), LLM-based agents have demonstrated significant potential in various applications (Xie et al., 2023; Wang et al., 2024; Xi et al., 2023), such as software development (Hong et al., 2023; Bairi et al., 2024), robotic planning (Yao et al., 2022b; Puig et al., 2018; Singh et al., 2023; Huang et al., 2022a), simulating human behavior (Park et al., 2023), etc. Typically, an LLM-based agent refers to an interactive system that processes environmental observations, maintains context across multiple rounds of dialogue, and outputs executable actions tailored to completing a given task. *Memory* is one of the

*Corresponding Author (pluo.lhi@gmail.com).

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 32779–32798
July 27 - August 1, 2025 ©2025 Association for Computational Linguistics

critical components of LLM-based agents, involving how agents store and utilize past experiences. When handling a specific task, an agent's memory can be divided into cross-trial and in-trial memory (also as known as *working memory*). Cross-trial memory typically consists of the historical trajectory information accumulated across multiple attempts at the current task. In contrast, in-trial memory pertains to the information relevant to the current trial. While many papers have explored leveraging cross-trial memory to optimize agent performance (Shinn et al., 2024; Zhao et al., 2024; Guo et al., 2023), few have investigated ways to better utilize working memory. Existing LLM-based agent literature primarily employs the STANDARD strategy illustrated in Figure 1, where all action-observation pairs in working memory are directly incorporated into the context when prompting LLMs (Liu et al., 2023c; Ma et al., 2024; Yao et al., 2022b). Although this approach transmits the historical information to the LLM as comprehensively as possible, it encounters issues in *long-horizon agent tasks*. Such tasks typically require the agent to perform numerous actions to complete the task, resulting in an extensive working memory. This lengthy working memory creates a redundant context, hindering LLMs from maintaining coherent strategies and making accurate predictions over extended periods.

Drawing on principles of cognitive science (Newell et al., 1972; Anderson, 2013), humans typically decompose a complex problem into multiple subproblems, addressing each individually. Each subproblem is treated as a memory "chunk," thereby reducing the cognitive load on working memory (Miller, 1956). By focusing on the results of completed subproblems rather than their detailed execution, humans effectively manage cognitive resources and improve their efficiency in solving complex, long-horizon tasks. Inspired by human cognition and problem-solving strategies, we propose a sophisticated hierarchical working memory management framework **HIAGENT** tailored for long-horizon agent tasks. The core idea of HIAGENT is to trigger LLMs to generate subgoals, with each subgoal serving as a chunk of the working memory. Specifically, as shown in Figure 2, we first prompt the LLM to generate a subgoal, then create actions to achieve the subgoal and store the corresponding action-observation pairs in a memory chunk. Once the subgoal is completed, we summarize the memory chunk and append the subgoal-observation pair to the working memory. In a word, HIAGENT triggers LLMs to proactively decide to replace previous subgoals with summarized observations while retaining only the action-observation pairs relevant to the current subgoal. To provide more flexible working memory management, we also introduce a trajectory retrieval module, which can retrieve the detailed trajectory information of specific past subgoals when necessary.

To validate the effectiveness and efficiency of HIAGENT, we conduct experiments on five long-horizon agent tasks from AgentBoard (Ma et al., 2024). The experimental results show that the success rate of HIAGENT is twice that of the STANDARD strategy, and it exceeds the STANDARD strategy by 23.94% in progress rate. Additionally, HIAGENT is more efficient than STANDARD strategy, reducing the average number of steps to complete tasks by 3.8, the context length by 35.02%, and the run time by 19.42%. Furthermore, to demonstrate that redundant context impairs the performance of LLM-based agents in long-horizon tasks, we compare HIAGENT to a method that generates subgoals without disregarding the detailed trajectory information of past subgoals. Experimental results show that HIAGENT improves the success rate by 20% while reducing both runtime and the number of steps. By analyzing model performance across varying step counts, we find that HIAGENT not only consistently outperform STANDARD on progress rate but also show a higher likelihood of generating executable actions as the number of steps increased.

2 Preliminary

2.1 Large Language Model based Agent

Large Language Model (LLM) based agents are intelligent autonomous systems designed to perform complex tasks. These tasks can be formalized as a partially observable Markov decision process (POMDP), characterized by the tuple (S, O, A, T, R), where: S denotes the state space; O represents the observation space; A signifies the action space; $T : S \times A \to S$ embodies the transition function; $R : S \times A \to R$ encapsulates the reward function; An LLM-based agent operates as a policy $\pi(a_t|I, o_t, a_{t-1}, o_{t-1}, \ldots, a_0, o_0)$, which, given the historical action-observation pairs and instructions I (encompassing in-context examples, environmental descriptions, etc.), generates an executable action $a_t \in A$. Each action precipitates

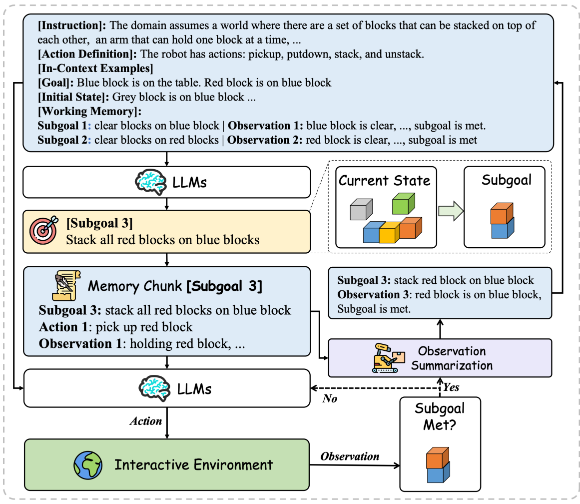

Figure 2: An overview of the process of HIAGENT.

a new state $s_{t+1} \in S$ and a subsequent observation $o_{t+1} \in O$. This iterative interaction persists until either task completion or the agent reaches a predetermined maximum number of steps.

2.2 Working Memory

From the cognitive science perspective, working memory enables individuals to hold and manipulate information in real-time, facilitating complex cognitive tasks such as reasoning, comprehension, and learning (Newell et al., 1972; Anderson, 2013). In LLM-based agents, we define working memory as the essential historical information required by the LLM at a given moment t to complete the current task. Effective working memory management allows for better integrating past experiences and current stimuli, leading to more informed and accurate decisions. It can be likened to the human process of attentional control and cognitive updating, which involves selectively focusing on relevant information, filtering out distractions, and continually updating the mental workspace with new and pertinent data. The STANDARD approach in Figure 1 stores all historical action-observation pairs in working memory, i.e., $m_t^{std} = (o_t, a_{t-1}, o_{t-1}, \ldots, a_0, o_0)$. Although this provides the LLM with comprehensive information, it also introduces redundancy, complicating the LLM's processing.

3 Methodology

3.1 Overview

The core idea of HIAGENT is to employ subgoals for hierarchical management of working memory. More specifically, as is shown in Figure 2, the process of HIAGENT can be described as follows: **(1)** Before generating specific grounded actions, we prompt the LLM to first formulate a subgoal g_i. Each subgoal serves as a milestone within the overall task. **(2)** Subsequently, the LLM generates precise actions to accomplish this subgoal. **(3)** Upon the LLM's determination that a particular subgoal has been fulfilled, we synthesize the corresponding action-observation pairs into a summarized observation s_i (§3.3). We then obscure the action-observation pairs within the context, substituting them with s_i. Consequently, the working memory of HIAGENT can be formalized as $m_t = (g_0, s_0, ..., g_{n-1}, s_{n-1}, g_n, a_{n0}, o_{n1}, ...)$. **(4)** Additionally, we have incorporated a retrieval module to facilitate more flexible memory management(§3.4). For instance, if the q_{th} subgoal is retrieved, we input the detailed action-observation pairs into the context rather than the summarized observation, i.e., $m'_t = (g_0, s_0, ..., g_q, a_{q0}, a_{q0}, ..., g_n, a_{n0}, o_{n0}, ...)$.

3.2 Subgoal-based Hierarchical Working Memory

As is shown in Figure 2, at each time step, the LLM can either generate the next action for the current subgoal or generate a new subgoal when it determines that the existing subgoal has been accomplished. For the current subgoal, the agent retains all action-observation pairs, providing a detailed context for immediate decision-making. For past subgoals, only a summarized version of the observations is kept. This subgoal-based hierarchical management approach in HIAGENT is deeply motivated by cognitive science principles, drawing parallels with human cognition and problem-solving strategies (Newell et al., 1972; Anderson, 2013). Employing subgoals to compartmentalize action-observation pairs can be conceptualized as a form of chunking methodology. In human cognition, chunking allows individuals to group related information into meaningful units, thereby overcoming working memory limitations (Miller, 1956). Similarly, HIAGENT utilizes subgoals as cognitive chunks, encapsulating related actions and observations. This chunking mechanism enables the system to handle complex sequences of information more effectively, reducing cognitive load and enhancing overall performance. Furthermore, by generating subgoals before specific actions, the system mimics the human tendency to break down larger objectives into more manageable components. This methodology enhances computational efficiency and aligns with established theories of human information processing.

3.3 Observation Summarization

The process of observation summarization can be formalized as $s_i = S(g_i, o_0, a_0, ..., o_t)$, where S can be implemented using either a Large Language Model (LLM) or alternative text summarization models. This function encapsulates the synthesis of historical observations and actions, contextualized by the current subgoal, to produce a concise representation of the agent's state. Furthermore, a crucial component of the summarized observation is assessing whether the current subgoal has been achieved. This evaluation serves as a pivotal guide for future subgoal generation, facilitating adaptive and goal-oriented behavior in the agent's decision-making process. By doing so, the agent can maintain a condensed yet informative context, balancing the need for historical information with efficiency. The example prompt can be found in Appendix **??**

3.4 Trajectory Retrieval

Despite the summarization, there may be instances where detailed past trajectory information becomes crucial for immediate decision-making. For instance, when a past subgoal execution fails, we need detailed trajectory information to determine the cause of failure. Moreover, reviewing past successful experiences can also increase the likelihood of success when facing novel challenges and scenarios. To address this, we introduce a trajectory retrieval module. When the LLM determines that detailed information from a past subgoal is necessary, it generates a retrieval function to recall the complete action-observation pairs for that subgoal, analogous to the way to generate actions. This selective retrieval allows the agent to access detailed historical data on demand without consistently carrying the full context.

4 Experiments

4.1 Experimental Setup

Evaluation Tasks We conduct the experiments on five long-horizon agent tasks, which typically require more than 20 steps: *(i)* **Blocksworld** requires the model to arrange the blocks into a specified target configuration by executing a series of moves; *(ii)* **Gripper** involves moving objects between different rooms; *(iii)* **Tyreworld** simulates changing a car tire, including removing the flat tire, replacing it with a spare, and installing the new tire; *(iv)* **Barman** emulates a bartender's tasks in mixing cocktails, including combining various ingredients, shakers, and garnishing drinks; *(v)* **Jericho** (Hausknecht et al., 2020) is a suite of text-based adventure game environments designed to evaluate agents' ability to navigate and interact with fictional worlds. More details can be found in Appendix A.

Evaluation Metrics We use multiple metrics to evaluate both the effectiveness and efficiency of LLM-based agents in solving long-horizon tasks: *(i)* **Progress Rate** (Ma et al., 2024) evaluates the advancement toward task completion. Specifically, a task consists of multiple goal conditions, and the progress rate is the proportion of goal conditions fulfilled by the model out of the total number of goal conditions. *(ii)* **Success Rate** measures the percentage of successful task completions. The success rate is 1 when the progress rate is 1. *(iii)* **Average Steps** counts the steps taken to complete the task; *(iv)* **Context Efficiency** is defined as the mean number of tokens in the in-trial context across all steps required to complete a given task. *(v)* **Run Time** evaluates the time required to complete tasks.

Baselines STANDARD prompting strategy is a predominantly used method in current LLM-based agent literature (Yao et al., 2022b; Ma et al., 2024; Liu et al., 2023c). It operates by taking one action followed by one observation, providing a comparative baseline for evaluating the performance of HIAGENT.

Implementation Details The implementation of evaluation tasks is based on AgentBoard (Ma et al., 2024). We set a maximum step limit of 30 for task configuration and provide one in-context example for each task. We employ GPT-4 (gpt-4-turbo)[1] as the LLM backbone for our experiments, serving both as the agent policy and the observation summarization model. We set the *temperature* hyperparameter for LLM inference to 0 and *topp* to 1. Detailed prompt examples are provided in the Appendix B.

4.2 Main Results

As shown in Table 1, HIAGENT demonstrated substantial advancements over STANDARD. Overall, in terms of effectiveness, it increased the success rate

[1]We utilized the model via OpenAI API service.

Table 1: Performance of STANDARD and HIAGENT on 5 long-horizon agent tasks. We report on four metrics: Success Rate (**SR**), Progress Rate (**PR**), Average Steps (**Steps**), and Context Efficiency (**Context**), Run Time (**Time**). The symbol ↑ indicates that a higher value for the metric is preferable, while ↓ signifies that a lower value is considered better. In the ***Overall*** section, the result is obtained by averaging the values of a certain metric across various tasks.

	SR ↑	PR ↑	Steps ↓	Context ↓	Time ↓
Blocksworld					
STANDARD	30.00	35.00	25.00	100%	100%
HIAGENT	**60.00** +30.00	**80.00** +45.00	**18.60** -6.40	**67.46%** -32.54%	**63.47%** -36.53%
Gripper					
STANDARD	**50.00**	**87.75**	25.20	100%	100%
HIAGENT	**50.00** +0.00	86.25 -1.50	**24.80** -0.40	**49.99%** -50.01%	**70.46%** -29.54%
Tyreworld					
STANDARD	10.00	39.28	28.40	100%	100%
HIAGENT	**60.00** +50.00	**75.83** +36.55	**19.00** -9.4	**73.58%** -26.42%	**77.58%** -22.42%
Barman					
STANDARD	10.00	17.50	26.85	100%	100%
HIAGENT	**30.00** +20.00	**40.83** +23.33	**24.5** -2.35	**67.02%** -32.98%	**95.54%** -4.46%
Jericho					
STANDARD	5.00	13.51	26.60	100%	100%
HIAGENT	**10.00** +5.00	**29.85** +16.34	**26.15** -0.45	**66.86%** -33.14%	**95.85%** -4.15%
Overall					
STANDARD	21.00	38.61	26.41	100%	100%
HIAGENT	**42.00** +21.00	**62.55** +23.94	**22.61** -3.80	**64.98%** -35.02%	**80.58%** -19.42%

by 21% and the progress rate by 23.94%. Regarding task execution efficiency, it reduced the average number of steps to completion by 3.8, decreased the number of context tokens consumed by 35%, and reduced the run time by 19.42%. Furthermore, in certain tasks (blocksworld, barman, jericho), HIAGENT even achieved more than double the progress rate improvement while maintaining efficiency. In tyreworld, the model not only achieved a 50% improvement in success rate but also reduced the average number of steps by 9.4. Although the progress rate slightly decreased by 1.5% in the gripper task, context token usage was reduced by over 50%.
We can draw several conclusions from previous discussions:
(1) HIAGENT is more **effective** than STANDARD, achieving huge improvements on both success rate and progress rate.
(2) HIAGENT is also more **efficient** than STANDARD, requiring fewer steps to complete tasks, utilizing shorter context lengths, and achieving faster runtime.

5 Analysis

To gain deeper insights into our approach, we explored the following research questions:
(1) *Are all modules effective for* HIAGENT*?*
(2) *Is* HIAGENT *consistently superior to the baseline at different steps?*
(3) *Is improvement of* HIAGENT *solely derived from task decomposition?*
(4) *How effective are the frameworks in generating executable actions?*
(5) *Are the observed performance improvements in* HIAGENT *statistically significant compared to* STANDARD*?*

5.1 Answer 1: All Modules in HIAGENT are Effective for HIAGENT

In this section, we conducted an ablation study to explore whether *Observation Summarization* and *Trajectory Retrieval* are effective.

Observation Summarization is effective. We heuristically use the observation corresponding to the last action as the summarized observation when removing the *Observation Summarization* module. As is shown in Table 2 ("*w/o OS*"), there is a significant decline in performance across all metrics. Specifically, the success rate and progress rate were significantly impacted, decreasing by 30% and 7.6%, respectively. It indicates that the observation summarization module can comprehensively aggregate the detailed information within a trajectory, thereby aiding the reasoning of an LLM-based agent.

Trajectory Retrieval is also crucial for performance enhancement. We hide all the detailed trajectory information of previous subgoals at each time step to verify the effectiveness of *Trajectory Retrieval*. According to the results in Table 2 ("*w/o TR*"), the success rate decreased by 10%, and the average steps increased by 1.2. This is because, while trajectory retrieval lengthens the reasoning steps of the LLM, it allows the agent to flexibly retrieve past trajectories under certain subgoals, which is more beneficial for identifying errors in previous actions.

The combination of Observation Summarization and Trajectory Retrieval yields significant improvement. We conducted an experiment where both modules were removed to validate the functionality and effectiveness of the combined *Observation Summarization* and *Trajectory Retrieval* modules. As shown in Table 2 ("*w/o OS & TR*"), there is a noticeable performance decline compared to HIAGENT, with the success rate decreasing by 20%. This decline is also evident when compared to the individual ablations of the *Observation Summarization* and *Trajectory Retrieval* modules, highlighting a substantial reduction in progress rate in their absence.

5.2 Answer 2: HIAGENT is consistently superior to STANDARD at different steps

To conduct a more granular study of HIAGENT's performance, we present the progress rate at different step counts (in intervals of 5 steps) in Figure 3. The experimental results indicate that overall, HIAGENT consistently achieves a higher progress rate at each step than STANDARD *(f)*. Additionally, it is noteworthy that HIAGENT benefits more from an increased number of steps, whereas STANDARD does not. For example, in the blocksworld task *(a)* and barman task *(b)*, STANDARD shows no progress rate increase between steps 15-25, whereas HIAGENT exhibits continuous growth. This further demonstrates HIAGENT's advantage in handling long-horizon agent tasks.

5.3 Answer 3: The improvement in HIAGENT is not solely attributed to task decomposition

Using LLMs to generate subgoals has been employed in numerous studies and has demonstrated considerable performance advantages (Zhou et al., 2022; Yin et al., 2023). Therefore, a pertinent question arises: "*Is the performance improvement attributed to* HIAGENT *merely related to task decomposition, rather than efficient working memory management?*" To address this question, we implemented a new method that prompts the LLM to generate a subgoal before generating executable actions, followed by generating actions to achieve this subgoal. Unlike HIAGENT, this approach does not obscure the detailed trajectory information of previous subgoals. The experimental results, detailed in Table 3, indicate that although task decomposition can lead to a performance improvement (30% in success rate), the success rate is still 20% lower than HIAGENT. Additionally, solely using task decomposition introduces inefficiencies, increasing runtime by 5.7% and context length by 12.8%. In summary, HIAGENT is more efficient and effective than task decomposition alone.

5.4 Answer 4: HIAGENT is effective in generating executable actions even under long steps

LLM-based agents sometimes generate actions that cannot be executed, such as attempting to retrieve objects from a closed container. This is typically due to LLMs' poor reasoning abilities. To investigate this, we calculated the proportion of executable actions generated by the model at each timestep, referred to as *executability*. As shown in Figure 4, HIAGENT is more likely to generate executable actions than STANDARD, further demonstrating the effectiveness of HIAGENT. Additionally, we observed that STANDARD is more prone to generating non-executable actions when the steps are longer (e.g., in the blocksworld, when the steps exceed 20, executability drops below 10%). This is because, as the working memory increases, the ability of LLMs to generate executable actions decreases. In contrast, HIAGENT maintains over 80% executability even with longer steps, indicating that the robustness to long steps is a key factor in the strong performance on long-horizon tasks.

5.5 Answer 5: The observed performance improvements in HIAGENT are statistically significant compared to STANDARD

To validate the statistical significance of the improvements in both effectiveness and efficiency, we selected the *Progress Rate* and *Average Steps* metrics for analysis. We employed the Wilcoxon signed-rank test (Woolson, 2005) for this purpose due to its suitability for comparing paired samples. This non-parametric test helps assess whether the

Table 2: Ablation study of HIAGENT on tyreworld. "*w/o OS*" refers to removing the ***O**bservation **S**ummarization* module introduced by Section 3.3. "*w/o TR*" refers to removing the ***T**rajectory **R**etrieval* module introduced by Section 3.4. "*w/o TR & OS*" refers to removing both modules.

Model	SR ↑	PR ↑	Steps ↓	Context ↓	Time ↓
HIAGENT	**60.0**	75.8	**19.0**	**100.0%**	**100.0%**
w/o OS	30.0 -30.0	68.2 -7.6	24.2 +5.2	110.8% +10.8%	122.5% +22.5%
w/o TR	50.0 -10.0	**76.9** +1.1	21.2 +2.2	105.0% +5.0%	107.5% +7.5%
w/o OS & TR	30.0 -30.0	62.4 -13.4	26.2 +7.2	107.2% +7.2%	121.2% +21.2%

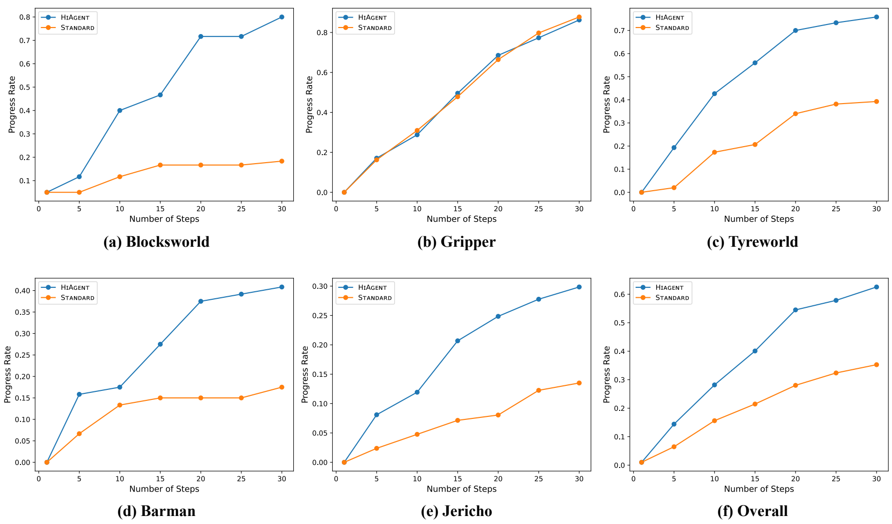

Figure 3: Progress rate at different steps.

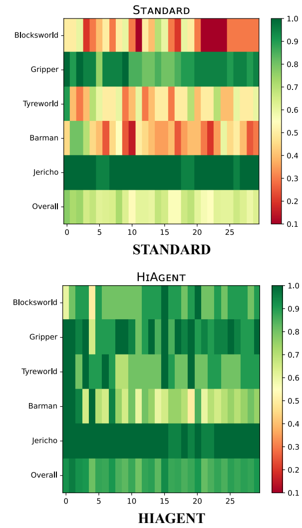

Figure 4: Executability of actions at different steps.

observed differences are likely due to chance or represent a genuine effect. The results of our analysis are as follows: *(i)* For the Progress Rate, the test statistic is 144.0 with a p-value of 2.38×10^{-5}, indicating a statistically significant difference between HIAGENT and STANDARD; *(ii)* For the Average Steps, the test statistic is 112.5 with a p-value of 0.0016, also demonstrating a statistically significant difference. These results confirm that the observed improvements in both effectiveness and efficiency are not due to random variation, underscoring the superiority of HIAGENT.

6 Related Work

Large Language Model based-Agent. Large Language Models (LLMs) have revolutionized the field of language agents, endowing them with the prowess to tackle intricate challenges through a logical sequence of actions (Xie et al., 2023; Xi et al., 2023; Wang et al., 2024). A series of works explored various applications of LLM-based agents, such as code generation (Wang et al., 2023b; Lin et al., 2018), web browsing (Yao et al., 2022a; Zhou et al., 2023a; Pan et al., 2024; Li and Waldo, 2024), robotics (Shridhar et al., 2020; Mu et al., 2024a,b), tool use (Li et al., 2023b; Wu et al., 2024; Qin et al., 2023), reasoning (Yang et al., 2024; Chen et al., 2025), planning (Xie et al., 2024), conducting research (Kang and Xiong, 2024), chip design and more. In addition, a great deal of work has explored the application of LLM-based agents in the field of multi-agent systems (Hong et al., 2023; Zhang et al., 2023a; Wu et al., 2023; Li et al., 2023a; Chen et al., 2023).

This paper introduces a working memory management framework HIAGENT that can be universally applied to enhance the performance of other agent frameworks. For example, ReAct (Yao et al., 2022b) introduces a method where the LLM generates a chain of thought (Wei et al., 2022) before generating actions, and the trajectory formed by the triplet of *"(thought, action, observation)"* can be managed using HIAGENT. Additionally, HIAGENT has the potential to alleviate information management challenges in multi-agent frameworks (Hong et al., 2023).

Table 3: Experimental results on tyreworld. "w. TD" refers to *Task Decomposition*, i.e., having the LLM generate subgoals without concealing detailed trajectory information of previous subgoals.

Model	SR ↑	PR ↑	Steps ↓	Context ↓	Time ↓
STANDARD	10.0	39.3	28.4	100%	100%
w. TD	40.0 +30.0	67.4 +28.1	22.8 -5.6	112.8% +12.8%	105.7% +5.7%
w. HIAGENT	**60.0** +50.0	**75.8** +36.5	**19.0** -9.4	**73.6%** -26.4%	**77.6%** -22.4%

Planning. Planning is a cornerstone of human intelligence (Chen et al., 2024b, 2025), representing a systematic approach to achieving goals through a series of deliberate actions (Yao et al., 2024; Zhang et al., 2023b; Song et al., 2023; Huang et al., 2023, 2022b; Liu et al., 2023a; Hu et al., 2023b; Ruan et al., 2023; Aghzal et al., 2023; Hu et al., 2024). It involves breaking down complex tasks into manageable sub-tasks, searching for potential solutions, and achieving a desired goal. Least-to-most (Zhou et al., 2022) and Plan-and-solve (Wang et al., 2023a) propose decomposing a complex question into a series of sub-questions. Lumos (Yin et al., 2023) and XAgent (Team, 2023) introduce an independent planning module for generating subgoals and use full context in the grounding module to complete each subgoal.

HIAGENT distinguishes itself from the literature by not only utilizing planning to enhance task performance but also by using subgoals as memory chunks to manage working memory hierarchically. This approach brings context efficiency and surpasses methods that rely solely on planning, as discussed in Section 5.3.

Memory. The memory module in LLM-based agents is analogous to the human memory system, which is responsible for encoding, storing, and retrieving information (Zhang et al., 2024). The memory modules are typically divided into long-term memory and short-term memory. Long-term memory can usually be stored in an external database, while short-term memory (also known as working memory) is typically used directly as the context input of LLMs. Most current research papers primarily focus on managing long-term memory (Alonso et al., 2024; Maharana et al., 2024; Chen et al., 2024a; Xiao et al., 2024; Yuan et al., 2023; Wang et al., 2023c; Majumder et al., 2023; Hu et al., 2023a; Hao et al., 2024; Tu et al., 2023; Liang et al., 2023; Kagaya et al., 2024). Pioneer works include Memorybank (Zhong et al., 2024), with its global-level summaries, has made significant strides in distilling conversations into coherent narratives. Other works, such as Think-in-memory (Liu et al., 2023b) and the Retroformer (Yao et al., 2023), incorporated summary modules to manage long-term memories. Unlike these works, our study investigates how optimizing the management of working memory can enhance agent performance. Another line of research involves modifying the structure of transformers to enable LLMs to process longer contexts, thereby extending their working memory capabilities (Zhou et al., 2023b; Chevalier et al., 2023; Bertsch et al., 2024; Ruoss et al., 2023; Beltagy et al., 2020; An et al., 2023).

However, existing research has identified that LLMs encounter attention loss issues with lengthy texts (Liu et al., 2024). Consequently, we believe that investigating more efficient management of working memory remains a valuable endeavor.

7 Conclusion

This paper proposes HIAGENT, a flexible framework that utilizes subgoals to manage the working memory of LLM-based agents. Experimental results from five long-horizon agent tasks demonstrate that HIAGENT outperforms the baseline model across all tasks, with an overall success rate more than double that of the baseline model. Furthermore, HIAGENT is more efficient, accomplishing tasks with fewer steps, in less runtime, and using shorter context. In the future, we hope HIAGENT can inspire more creative ideas on effectively managing the working memory of LLM-based agents.

Limitation

While HIAGENT reduces redundant context, it may still face challenges in extremely long-horizon tasks where memory constraints persist. Future work could explore more advanced retrieval strategies to further optimize memory efficiency. Moreover, our experiments primarily focus on benchmark tasks; extending the evaluation to more diverse real-world applications would provide deeper insights into the generalizability of our method.

References

Mohamed Aghzal, Erion Plaku, and Ziyu Yao. 2023. Can large language models be good path planners? a benchmark and investigation on spatial-temporal reasoning. *arXiv preprint arXiv:2310.03249*.

Nick Alonso, Tomás Figliolia, Anthony Ndirango, and Beren Millidge. 2024. Toward conversational agents with context and time sensitive long-term memory. *arXiv preprint arXiv:2406.00057*.

Chenxin An, Shansan Gong, Ming Zhong, Mukai Li, Jun Zhang, Lingpeng Kong, and Xipeng Qiu. 2023. L-eval: Instituting standardized evaluation for long context language models. *arXiv preprint arXiv:2307.11088*.

John R Anderson. 2013. *The architecture of cognition*. Psychology Press.

Ramakrishna Bairi, Atharv Sonwane, Aditya Kanade, Arun Iyer, Suresh Parthasarathy, Sriram Rajamani, B Ashok, and Shashank Shet. 2024. Codeplan: Repository-level coding using llms and planning. *Proceedings of the ACM on Software Engineering*, 1(FSE):675–698.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Amanda Bertsch, Uri Alon, Graham Neubig, and Matthew Gormley. 2024. Unlimiformer: Long-range transformers with unlimited length input. *Advances in Neural Information Processing Systems*, 36.

Nuo Chen, Hongguang Li, Juhua Huang, Baoyuan Wang, and Jia Li. 2024a. Compress to impress: Unleashing the potential of compressive memory in real-world long-term conversations. *arXiv preprint arXiv:2402.11975*.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. 2025. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*.

Qiguang Chen, Libo Qin, Jiaqi Wang, Jingxuan Zhou, and Wanxiang Che. 2024b. Unlocking the capabilities of thought: A reasoning boundary framework to quantify and optimize chain-of-thought. *Advances in Neural Information Processing Systems*, 37:54872–54904.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chen Qian, Chi-Min Chan, Yujia Qin, Yaxi Lu, Ruobing Xie, et al. 2023. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors in agents. *arXiv preprint arXiv:2308.10848*.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. *arXiv preprint arXiv:2305.14788*.

Jing Guo, Nan Li, Jianchuan Qi, Hang Yang, Ruiqiao Li, Yuzhen Feng, Si Zhang, and Ming Xu. 2023. Empowering working memory for large language model agents. *arXiv preprint arXiv:2312.17259*.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. 2024. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings. *Advances in neural information processing systems*, 36.

Matthew Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. 2020. Interactive fiction games: A colossal adventure. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7903–7910.

Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, et al. 2023. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*.

Chenxu Hu, Jie Fu, Chenzhuang Du, Simian Luo, Junbo Zhao, and Hang Zhao. 2023a. Chatdb: Augmenting llms with databases as their symbolic memory. *arXiv preprint arXiv:2306.03901*.

Mengkang Hu, Yao Mu, Xinmiao Yu, Mingyu Ding, Shiguang Wu, Wenqi Shao, Qiguang Chen, Bin Wang, Yu Qiao, and Ping Luo. 2023b. Tree-planner: Efficient close-loop task planning with large language models. *arXiv preprint arXiv:2310.08582*.

Mengkang Hu, Pu Zhao, Can Xu, Qingfeng Sun, Jianguang Lou, Qingwei Lin, Ping Luo, Saravan Rajmohan, and Dongmei Zhang. 2024. Agentgen: Enhancing planning abilities for large language model based agent via environment and task generation. *arXiv preprint arXiv:2408.00764*.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022a. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, pages 9118–9147. PMLR.

Wenlong Huang, Fei Xia, Dhruv Shah, Danny Driess, Andy Zeng, Yao Lu, Pete Florence, Igor Mordatch, Sergey Levine, Karol Hausman, and Brian Ichter. 2023. Grounded decoding: Guiding text generation with grounded models for robot control. *Preprint*, arXiv:2303.00855.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. 2022b. Inner monologue: Embodied reasoning through planning with language models. *Preprint*, arXiv:2207.05608.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Tomoyuki Kagaya, Thong Jing Yuan, Yuxuan Lou, Jayashree Karlekar, Sugiri Pranata, Akira Kinose, Koki Oguri, Felix Wick, and Yang You. 2024. Rap: Retrieval-augmented planning with contextual memory for multimodal llm agents. *arXiv preprint arXiv:2402.03610*.

Hao Kang and Chenyan Xiong. 2024. Researcharena: Benchmarking llms' ability to collect and organize information as research agents. *arXiv preprint arXiv:2406.10291*.

Eric Li and Jim Waldo. 2024. Websuite: Systematically evaluating why web agents fail. *arXiv preprint arXiv:2406.01623*.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023a. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in Neural Information Processing Systems*, 36:51991–52008.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023b. Api-bank: A comprehensive benchmark for tool-augmented llms. *arXiv preprint arXiv:2304.08244*.

Xinnian Liang, Bing Wang, Hui Huang, Shuangzhi Wu, Peihao Wu, Lu Lu, Zejun Ma, and Zhoujun Li. 2023. Unleashing infinite-length input capacity for large-scale language models with self-controlled memory system. *arXiv e-prints*, pages arXiv–2304.

Xi Victoria Lin, Chenglong Wang, Luke Zettlemoyer, and Michael D Ernst. 2018. Nl2bash: A corpus and semantic parser for natural language interface to the linux operating system. *arXiv preprint arXiv:1802.08979*.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. 2023a. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*.

Lei Liu, Xiaoyan Yang, Yue Shen, Binbin Hu, Zhiqiang Zhang, Jinjie Gu, and Guannan Zhang. 2023b. Think-in-memory: Recalling and post-thinking enable llms with long-term memory. *arXiv preprint arXiv:2311.08719*.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. 2023c. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. 2024. Agentboard: An analytical evaluation board of multi-turn llm agents. *arXiv preprint arXiv:2401.13178*.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of llm agents. *arXiv preprint arXiv:2402.17753*.

Bodhisattwa Prasad Majumder, Bhavana Dalvi Mishra, Peter Jansen, Oyvind Tafjord, Niket Tandon, Li Zhang, Chris Callison-Burch, and Peter Clark. 2023. Clin: A continually learning language agent for rapid task adaptation and generalization. *arXiv preprint arXiv:2310.10134*.

Meta AI. 2024. Introducing meta Llama 3: The most capable openly available LLM to date. Accessed: 2024-04-18.

George A Miller. 1956. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, 63(2):81.

Yao Mu, Junting Chen, Qinglong Zhang, Shoufa Chen, Qiaojun Yu, Chongjian Ge, Runjian Chen, Zhixuan Liang, Mengkang Hu, Chaofan Tao, et al. 2024a. Robocodex: Multimodal code generation for robotic behavior synthesis. *arXiv preprint arXiv:2402.16117*.

Yao Mu, Qinglong Zhang, Mengkang Hu, Wenhai Wang, Mingyu Ding, Jun Jin, Bin Wang, Jifeng Dai, Yu Qiao, and Ping Luo. 2024b. Embodiedgpt: Vision-language pre-training via embodied chain of thought. *Advances in Neural Information Processing Systems*, 36.

Allen Newell, Herbert Alexander Simon, et al. 1972. *Human problem solving*, volume 104. Prentice-hall Englewood Cliffs, NJ.

OpenAI. 2022. Openai: Introducing chatgpt.

OpenAI. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Yichen Pan, Dehan Kong, Sida Zhou, Cheng Cui, Yifei Leng, Bing Jiang, Hangyu Liu, Yanyi Shang, Shuyan Zhou, Tongshuang Wu, et al. 2024. Webcanvas: Benchmarking web agents in online environments. *arXiv preprint arXiv:2406.12373*.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22.

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. 2018. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8494–8502.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

Jingqing Ruan, Yihong Chen, Bin Zhang, Zhiwei Xu, Tianpeng Bao, Guoqing Du, Shiwei Shi, Hangyu Mao, Xingyu Zeng, and Rui Zhao. 2023. Tptu: Task planning and tool usage of large language model-based ai agents. *arXiv preprint arXiv:2308.03427*.

Anian Ruoss, Grégoire Delétang, Tim Genewein, Jordi Grau-Moya, Róbert Csordás, Mehdi Bennani, Shane Legg, and Joel Veness. 2023. Randomized positional encodings boost length generalization of transformers. *arXiv preprint arXiv:2305.16843*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2020. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. 2023. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530. IEEE.

Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M. Sadler, Wei-Lun Chao, and Yu Su. 2023. Llm-planner: Few-shot grounded planning for embodied agents with large language models. *Preprint*, arXiv:2212.04088.

XAgent Team. 2023. Xagent: An autonomous agent for complex task solving.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Shangqing Tu, Chunyang Li, Jifan Yu, Xiaozhi Wang, Lei Hou, and Juanzi Li. 2023. Chatlog: Recording and analyzing chatgpt across time. *arXiv preprint arXiv:2304.14106*.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023a. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. *arXiv preprint arXiv:2305.04091*.

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. 2023b. Mint: Evaluating llms in multi-turn interaction with tools and language feedback. *arXiv preprint arXiv:2309.10691*.

Zihao Wang, Shaofei Cai, Anji Liu, Yonggang Jin, Jinbing Hou, Bowei Zhang, Haowei Lin, Zhaofeng He, Zilong Zheng, Yaodong Yang, et al. 2023c. Jarvis-1: Open-world multi-task agents with memory-augmented multimodal language models. *arXiv preprint arXiv:2311.05997*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Robert F Woolson. 2005. Wilcoxon signed-rank test. *Encyclopedia of Biostatistics*, 8.

Mengsong Wu, Tong Zhu, Han Han, Chuanyuan Tan, Xiang Zhang, and Wenliang Chen. 2024. Seal-tools: Self-instruct tool learning dataset for agent tuning and detailed benchmark. *arXiv preprint arXiv:2405.08355*.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. 2023. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, Song Han, and Maosong Sun. 2024. Infllm: Unveiling the intrinsic capacity of llms for understanding extremely long sequences with training-free memory. *arXiv preprint arXiv:2402.04617*.

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. 2024. Travelplanner: A benchmark for real-world planning with language agents. *arXiv preprint arXiv:2402.01622*.

Tianbao Xie, Fan Zhou, Zhoujun Cheng, Peng Shi, Luoxuan Weng, Yitao Liu, Toh Jing Hua, Junning Zhao, Qian Liu, Che Liu, et al. 2023. Openagents: An open platform for language agents in the wild. *arXiv preprint arXiv:2310.10634*.

Siwei Yang, Bingchen Zhao, and Cihang Xie. 2024. Aqa-bench: An interactive benchmark for evaluating llms' sequential reasoning ability. *arXiv preprint arXiv:2402.09404*.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022a. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022b. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Weiran Yao, Shelby Heinecke, Juan Carlos Niebles, Zhiwei Liu, Yihao Feng, Le Xue, Rithesh Murthy, Zeyuan Chen, Jianguo Zhang, Devansh Arpit, et al. 2023. Retroformer: Retrospective large language agents with policy gradient optimization. *arXiv preprint arXiv:2308.02151*.

Da Yin, Faeze Brahman, Abhilasha Ravichander, Khyathi Chandu, Kai-Wei Chang, Yejin Choi, and Bill Yuchen Lin. 2023. Lumos: Learning agents with unified data, modular design, and open-source llms. *arXiv preprint arXiv:2311.05657*.

Ruifeng Yuan, Shichao Sun, Zili Wang, Ziqiang Cao, and Wenjie Li. 2023. Evolving large language model assistant with long-term conditional memory. *arXiv preprint arXiv:2312.17257*.

Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B Tenenbaum, Tianmin Shu, and Chuang Gan. 2023a. Building cooperative embodied agents modularly with large language models. *arXiv preprint arXiv:2307.02485*.

Jiajie Zhang, Shulin Cao, Tingjia Zhang, Xin Lv, Jiaxin Shi, Qi Tian, Juanzi Li, and Lei Hou. 2023b. Reasoning over hierarchical question decomposition tree for explainable question answering. *arXiv preprint arXiv:2305.15056*.

Zeyu Zhang, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Quanyu Dai, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. 2024. A survey on the memory mechanism of large language model based agents. *arXiv preprint arXiv:2404.13501*.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19632–19642.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19724–19731.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, et al. 2023a. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.

Wangchunshu Zhou, Yuchen Eleanor Jiang, Peng Cui, Tiannan Wang, Zhenxin Xiao, Yifan Hou, Ryan Cotterell, and Mrinmaya Sachan. 2023b. Recurrentgpt: Interactive generation of (arbitrarily) long text. *arXiv preprint arXiv:2305.13304*.

A More Details on Evaluation Tasks

A.1 Blocksworld

Action List

1. pickup <block>: allows the arm to pick up a block from the table if it is clear and the arm is empty. After the pickup action, the arm will be holding the block, and the block will no longer be on the table or clear.
2. putdown <block>: allows the arm to put down a block on the table if it is holding a block. After the putdown action, the arm will be empty, and the block will be on the table and clear.
3. stack <block> <block>: allows the arm to stack a block on top of another block if the arm is holding the top block and the bottom block is clear. After the stack action, the arm will be empty, the top block will be on top of the bottom block, and the bottom block will no longer be clear.
4. unstack <block> <block>: allows the arm to unstack a block from on top of another block if the arm is empty and the top block is clear. After the unstack action, the arm will be holding the top block, the top block will no longer be on top of the bottom block, and the bottom block will be clear.

Goal example

b1 is on b2., b2 is on b3.

Observation example

b1 is on the table. b2 is on the table. B3 is on the table. Robot arm is empty. The b1 is clear. The b2 is clear. The b3 is clear.

Action example

pickup b2.

A.2 Gripper

Action List

1. move <room1> <room2>: This action allows the robot to move from one room to another.The action has a single precondition, which is that the robot is currently in a room. The effect of this action is to move the robot to another room and to remove the fact that it is in the original room.
2. pick <obj> <room> <gripper>: This action allows the robot to pick up an object using the gripper. The action has three preconditions: (1) the object is located in a room (2) the robot is currently in the same room and (3) the gripper is free (i.e., not holding any object). The effect of this action is to update the state of the world to show that the robot is carrying the object using the gripper, the object is no longer in the room, and the gripper is no longer free.
3. drop <obj> <room> <gripper>: This action allows the robot to drop an object that it is carrying. The action has two preconditions: (1) the robot is currently carrying the object using the gripper, and (2) the robot is currently in a room. The effect of this action is to update the state of the world to show that the robot is no longer carrying the object using the gripper, the object is now located in the room, and the gripper is now free.

Goal example

ball1 is at roomb. , ball2 is at roomb. , ball3 is at roomb. , ball4 is at room.

Observation example

Ball1 is a ball. Ball1 is carrying right. Ball2 is a ball. Ball2 is at rooma. Ball3 is a ball. Ball3 is at rooma. Ball4 is a ball. Ball4 is at rooma. Left is a gripper. Left is free. Right is a gripper. Robby is at rooma. Room rooma Room roomb.

Action example

Pick up ball1 at rooma with arm right.

A.3 Tyreworld

Action List

1. open <container>: The precondition for this action is that the container is unlocked and closed. The effect of this action is that the container is open and not closed.
2. close <container>: The precondition for this action is that the container is open. The effect of this action is that the container is closed and not open.
3. fetch <object> <container>: The precondition for this action is that the object is inside the container and the container is open. The effect of this action is that the object is held by the agent and not inside the container.
4. put-away <object> <container>: The precondition for this action is that the object is held by the agent and the container is open. The effect of this action is that the object is inside the container and not held by the agent.
5. loosen <nut> <hub>: The precondition for this action is that the agent has a wrench, the nut on hub is tight, and the hub is on the ground. The effect of this action is that the nut on hub is loose

and not tight.
6. tighten <nut> <hub>: The precondition for this action is that the agent has a wrench, the nut on hub is loose, and the hub is on the ground. The effect of this action is that the nut on hub is tight and not loose.
7. jack-up <hub>: This action represents the process of lifting a hub off the ground using a jack. It requires the agent to have a jack and for the hub to be on the ground. After performing this action, the hub will no longer be on the ground and the agent will no longer have the jack.
8. jack-down <hub>: This action represents the process of lowering a hub back to the ground from an elevated position using a jack. It requires the agent to have the hub off the ground. After performing this action, the hub will be back on the ground and the agent will have the jack.
9. undo <nut> <hub>: This action undo the fastening of a nut on a hub. The preconditions are the hub is not on the ground (i.e., it has been jacked up), the hub is fastened, the agent has a wrench and the nut is loose. The effects are the agent has the nut, the hub is unfastened, the hub is no longer loose and the hub is not fastened anymore.
10. do-up <nut> <hub>: This action fasten a nut on a hub. The preconditions are the agent has a wrench, the hub is unfastened, the hub is not on the ground (i.e., it has been jacked up) and the agent has the nut to be fastened. The effects are the nut is now loose on the hub, the hub is fastened, the hub is no longer unfastened and the agent no longer has the nut.
11. remove-wheel <wheel> <hub>: This action removes a wheel from a hub. It can only be performed if the hub is not on the ground, the wheel is currently on the hub, and the hub is unfastened. After the action is performed, the agent will have the removed wheel and the hub will be free, meaning that the wheel is no longer on the hub.
12. put-on-wheel <wheel> <hub>: This action puts a wheel onto a hub. It can only be performed if the agent has the wheel, the hub is free, the hub is unfastened, and the hub is not on the ground. After the action is performed, the wheel will be on the hub, the hub will no longer be free, and the agent will no longer have the wheel.
13. inflate <wheel>: This action inflates a wheel using a pump. It can only be performed if the agent has a pump, the wheel is not inflated, and the wheel is intact. After the action is performed, the wheel will be inflated.

Goal example
w1 is in boot.

Observation example
Boot is closed. Boot is unlocked. Hub the-hub1 is fastened. Hub the-hub1 is on the ground. Jack is in boot. Pump is in boot. R1 is in boot. The nut nuts1 on the hub the-hub1 is tight. Wheel r1 is intact. Wheel r1 is not inflated. Wheel w1 is on hub the-hub1. Wrench is in boot.

Action example
Open boot.

A.4 Barman

Action List
1. <hand> grasp <container>: Grasp a container
2. <hand> leave <container>: Leave a container on the table
3. fill-shot <shot> <ingredient> <hand1> <hand2> <dispenser>: Fill a shot glass with an ingredient from dispenser
4. refill-shot <shot> <ingredient> <hand1> <hand2> <dispenser>: Refill a shot glass with an ingredient from dispenser
5. empty-shot <hand> <shot> <beverage>: Empty a shot glass 6. clean-shot <shot> <beverage> <hand1> <hand2>: Clean a shot glass
7. pour-shot-to-clean-shaker <shot> <ingredient> <shaker> <hand1> <level1> <level2>: Pour an ingredient from a shot glass to a clean shaker from level1 to level2
8. pour-shot-to-used-shaker <shot> <ingredient> <shaker> <hand1> <level1> <level2>: Pour an ingredient from a shot glass to a used shaker from level1 to level2
9. empty-shaker <hand> <shaker> <cocktail> <level1> <level2>: Empty a shaker containing cocktail from level1 to level2
10. clean-shaker <hand1> <hand2> <shaker>: Clean a shaker 11. shake <cocktail> <ingredient1> <ingredient2> <shaker> <hand1> <hand2>: Shake a cocktail in a shaker
12. pour-shaker-to-shot <beverage> <shot> <hand> <shaker> <level1> <level2>: Pour a beverage from a shaker to a shot glass from level1 to level2

Goal example
shot1 contains cocktail1.

Observation example
Cocktail1 part1 ingredient is ingredient1. Cocktail1 part2 ingredient is ingredient3. Cocktail2 part1 ingredient is ingredient2. Cocktail2 part2 ingredient is ingredient3. Cocktail3 part1 ingredient is ingredient1. Cocktail3 part2 ingredient is ingredient2. Dispenser1 dispenses ingredient1. Dispenser2 dispenses ingredient2. Dispenser3 dispenses ingredient3. Left hand is empty. Level l0 is next to level l1. Level l1 is next to level l2. Right hand is empty. Shaker1 is at empty level l0. Shaker1 is at level l0. Shaker1 is clean. Shaker1 is empty. Shaker1 is on the table. Shot1 is clean. Shot1 is empty. Shot1 is on the table. Shot2 is clean. Shot2 is empty. Shot2 is on the table. Shot3 is clean. Shot3 is empty. Shot3 is on the table. Shot4 is clean. Shot4 is empty. Shot4 is on the table.

Action example
right grasp shot1.

A.5 Jericho

Action List
1. Inventory: check things you are carrying
2. Look: check your surroundings
3. Examine <place/obj>: check the details of something
4. Take <obj>: pickup obj
5. Put down <obj>: leave a obj at your current place.
6. Drop <obj>
7. Check valid actions: Check actions you can use
8. South: go south
9. North: go north
10. East: go east
11. West: go west
12. Up: go up
13. Down: go down
14. Check valid actions (Other available actions)

Goal example
You are the warrior Link that needs to save the princess from the castle.

Observation example
You are at the path leading to the castle. The castle is to your north. There is a barrel in front of you.

Action example
Examine barrel

B Prompt Examples

B.1 Standard

Environment Implementation

Your goal is to replace flat tyres with intact tyres on the hubs. Remember to open boot first to get tools you need. Intact tyres should be inflated. The nuts should be tight on the hubs. The flat tyres, wrench, jack, and pump should be in the boot. The boot should be closed.

There are 13 actions defined in this domain:
open <container>: The precondition for this action is that the container is unlocked and closed. The effect of this action is that the container is open and not closed.
close <container>: The precondition for this action is that the container is open. The effect of this action is that the container is closed and not open.
fetch <object> <container>: The precondition for this action is that the object is inside the container and the container is open. The effect of this action is that the object is held by the agent and not inside the container.
put-away <object> <container>: The precondition for this action is that the object is held by the agent and the container is open. The effect of this action is that the object is inside the container and not held by the agent.
loosen <nut> <hub>: The precondition for this action is that the agent has a wrench, the nut on hub is tight, and the hub is on the ground. The effect of this action is that the nut on hub is loose and not tight.
tighten <nut> <hub>: The precondition for this action is that the agent has a wrench, the nut on hub is loose, and the hub is on the ground. The effect of this action is that the nut on hub is tight and not loose.
jack-up <hub>: This action represents the process of lifting a hub off the ground using a jack. It requires the agent to have a jack

and for the hub to be on the ground. After performing this action, the hub will no longer be on the ground and the agent will no longer have the jack.
jack-down <hub>: This action represents the process of lowering a hub back to the ground from an elevated position using a jack. It requires the agent to have the hub off the ground. After performing this action, the hub will be back on the ground and the agent will have the jack.
undo <nut> <hub>: This action undo the fastening of a nut on a hub. The preconditions are the hub is not on the ground (i.e., it has been jacked up), the hub is fastened, the agent has a wrench and the nut is loose. The effects are the agent has the nut, the hub is unfastened, the hub is no longer loose and the hub is not fastened anymore.
do-up <nut> <hub>: This action fasten a nut on a hub. The preconditions are the agent has a wrench, the hub is unfastened, the hub is not on the ground (i.e., it has been jacked up) and the agent has the nut to be fastened. The effects are the nut is now loose on the hub, the hub is fastened, the hub is no longer unfastened and the agent no longer has the nut.
remove-wheel <wheel> <hub>: This action removes a wheel from a hub. It can only be performed if the hub is not on the ground, the wheel is currently on the hub, and the hub is unfastened. After the action is performed, the agent will have the removed wheel and the hub will be free, meaning that the wheel is no longer on the hub.
put-on-wheel <wheel> <hub>: This action puts a wheel onto a hub. It can only be performed if the agent has the wheel, the hub is free, the hub is unfastened, and the hub is not on the ground. After the action is performed, the wheel will be on the hub, the hub will no longer be free, and the agent will no longer have the wheel.
inflate <wheel>: This action inflates a wheel using a pump. It can only be performed if the agent has a pump, the wheel is not inflated, and the wheel is intact. After the action is performed, the wheel will be inflated.

Here are examples:

The goal is to satisfy the following conditions: w1 is in boot. (Note you need to open boot first so that you can extract tools from it.)

Observation: Boot is closed. Boot is unlocked. Hub the-hub1 is fastened. Hub the-hub1 is on the ground. Jack is in boot. Pump is in boot. R1 is in boot. The nut nuts1 on the hub the-hub1 is tight. Wheel r1 is intact. Wheel r1 is not inflated. Wheel w1 is on hub the-hub1. Wrench is in boot.

Action: check valid actions.
Observation: Valid actions are: Open boot. Check valid actions. Look around.
Action: Open boot.
Observation: Boot is open.
Action: fetch wrench from boot
Observation: You have wrench.
Action: check valid actions
Observation: Valid actions are: Fetch pump from boot., Fetch r1 from boot., Put-away wrench in boot., Close boot., Fetch jack from boot., Loosen the nut nuts1 on the hub the-hub1., check valid actions
Action: Loosen the nut nuts1 on the hub the-hub1.
Observation: The nut nuts1 on the hub the-hub1 is loose.
Action: fetch jack from boot
Observation: You have jack.
Action: jack-up the hub the-hub1
Observation: Hub the-hub1 is not on the ground.
Action: Undo the fastening of the nut nuts1 on the hub the-hub1.
Observation: Hub the-hub1 is unfastened.
Action: Remove-wheel w1 from the hub the-hub1.
Observation: You have w1.
Action: Put-away w1 in boot
Observation: W1 is in boot. Goal is completed.
You should perform actions to accomplish the goal: The goal is to satisfy the following conditions: Wheel r1 is inflated., r1 is on the-hub1., w1 is in boot.

You should use the following commands for help when your action cannot be understood: check valid actions
Observation: Boot is closed. Boot is unlocked. Hub the-hub1 is fastened. Hub the-hub1 is on the ground. Jack is in boot. Pump is in boot. R1 is in boot. The nut nuts1 on the hub the-hub1 is tight. W1 is on the-hub1. Wheel r1 is intact. Wheel r1 is not inflated. Wrench is in boot.
Action: Open boot.
Observation: Boot is open.
Action: jack-up the-hub1
Observation: The action is not valid and therefore takes no effect. Please check valid actions. Action: jack-up the hub the-hub1
Observation: The action is not valid and therefore takes no effect. Please check valid actions.
Action: Loosen the nut nuts1 on the hub the-hub1.
Observation: The action is not valid and therefore takes no effect. Please check valid actions.
Action: Loosen the nut nuts1 on the hub the-hub1.
Observation: The action is not valid and therefore takes no effect. Please check valid actions.
Action: Loosen the nut nuts1 on the hub the-hub1.
Observation: The action is not valid and therefore takes no effect. Please check valid actions.
Action: Fetch wrench from boot.
Observation: You have wrench.

B.2 HiAgent

Environment Implementation

Your goal is to replace flat tyres with intact tyres on the hubs. Remember to open boot first to get tools you need. Intact tyres should be inflated. The nuts should be tight on the hubs. The flat tyres, wrench, jack, and pump should be in the boot. The boot should be closed.

There are 13 actions defined in this domain:
open <container>: The precondition for this action is that the container is unlocked and closed. The effect of this action is that the container is open and not closed.
close <container>: The precondition for this action is that the container is open. The effect of this action is that the container is closed and not open.
fetch <object> <container>: The precondition for this action is that the object is inside the container and the container is open. The effect of this action is that the object is held by the agent and not inside the container.
put-away <object> <container>: The precondition for this action is that the object is held by the agent and the container is open. The effect of this action is that the object is inside the container and not held by the agent.
loosen <nut> <hub>: The precondition for this action is that the agent has a wrench, the nut on hub is tight, and the hub is on the ground. The effect of this action is that the nut on hub is loose and not tight.
tighten <nut> <hub>: The precondition for this action is that the agent has a wrench, the nut on hub is loose, and the hub is on the ground. The effect of this action is that the nut on hub is tight and not loose.
jack-up <hub>: This action represents the process of lifting a hub off the ground using a jack. It requires the agent to have a jack and for the hub to be on the ground. After performing this action, the hub will no longer be on the ground and the agent will no longer have the jack.
jack-down <hub>: This action represents the process of lowering a hub back to the ground from an elevated position using a jack. It requires the agent to have the hub off the ground. After performing this action, the hub will be back on the ground and the agent will have the jack.
undo <nut> <hub>: This action undo the fastening of a nut on a hub. The preconditions are the hub is not on the ground (i.e., it has been jacked up), the hub is fastened, the agent has a wrench and the

nut is loose. The effects are the agent has the nut, the hub is unfastened, the hub is no longer loose and the hub is not fastened anymore.
do-up <nut> <hub>: This action fasten a nut on a hub. The preconditions are the agent has a wrench, the hub is unfastened, the hub is not on the ground (i.e., it has been jacked up) and the agent has the nut to be fastened. The effects are the nut is now loose on the hub, the hub is fastened, the hub is no longer unfastened and the agent no longer has the nut.
remove-wheel <wheel> <hub>: This action removes a wheel from a hub. It can only be performed if the hub is not on the ground, the wheel is currently on the hub, and the hub is unfastened. After the action is performed, the agent will have the removed wheel and the hub will be free, meaning that the wheel is no longer on the hub.
put-on-wheel <wheel> <hub>: This action puts a wheel onto a hub. It can only be performed if the agent has the wheel, the hub is free, the hub is unfastened, and the hub is not on the ground. After the action is performed, the wheel will be on the hub, the hub will no longer be free, and the agent will no longer have the wheel.
inflate <wheel>: This action inflates a wheel using a pump. It can only be performed if the agent has a pump, the wheel is not inflated, and the wheel is intact. After the action is performed, the wheel will be inflated.

Note: A subgoal is a milestone goal that you need to complete in order to achieve the final goal. When there is an unfinished subgoal, you need to ground the given subgoal to corresponding executable actions for solving the given task in the following format: "Action: action". When there is no current subgoal or you believe the previous subgoal has been completed (based on past actions and observations), you need to output the next subgoal to be completed and its first action in the following format: "Subgoal: subgoal Action: action". You cannot output two subgoals consecutively. Detailed trajectory information (action-observation pair) of previously satisfied subgoals will be hidden for context efficiency. If you believe that the detailed trajectory information of a particular subgoal is crucial for the current subgoal, you can use Action: "retrieve(subgoal_id)" to obtain the detailed trajectory information.

Here are examples:

The goal is to satisfy the following conditions: w1 is in boot. (Note you need to open boot first so that you can extract tools from it.)

Observation: Boot is closed. Boot is unlocked. Hub the-hub1 is fastened. Hub the-hub1 is on the ground. Jack is in boot. Pump is in boot. R1 is in boot. The nut nuts1 on the hub the-hub1 is tight. Wheel r1 is intact. Wheel r1 is not inflated. Wheel w1 is on hub the-hub1. Wrench is in boot.

1 Subgoal: Open the boot to access the tools.
Action: check valid actions.
Observation: Valid actions are: Open boot. Check valid actions. Look around.
Action: Open boot.
Observation: Boot is open. Subgoal is met.

2 Subgoal: Fetch the wrench from the boot.
Action: fetch wrench from boot
Observation: You have wrench. Subgoal is met.

3 Subgoal: Loosen the nut on the hub.
Action: Loosen the nut nuts2 on the hub the-hub1.
Observation: The action is not valid. Subgoal is not met.

4 Subgoal: Examine errors and try to loosen the nut on the hub again.
Action: retrieve(3)
Observation: trajectory of Subgoal 3 is retrieved.
Action: check valid actions

Observation: Valid actions are: Fetch pump from boot., Fetch r1 from boot., Put-away wrench in boot., Close boot., Fetch jack from boot., Loosen the nut nuts1 on the hub the-hub1., check valid actions
Action: Loosen the nut nuts on the hub the-hub1.
Observation: The nut nuts1 on the hub the-hub1 is loose. Subgoal is met.

5 Subgoal: Fetch the jack from the boot and jack up the hub. Action: fetch jack from boot
Observation: You have jack.
Action: jack-up the hub the-hub1
Observation: Hub the-hub1 is not on the ground. Subgoal is met.

6 Subgoal: Unfasten the nut nuts1 and remove wheel w1.
Action: Undo the fastening of the nut nuts1 on the hub the-hub1.
Observation: Hub the-hub1 is unfastened.
Action: Remove-wheel w1 from the hub the-hub1.
Observation: You have w1. Subgoal is met.

7 Subgoal: Put away the wheel w1 in the boot.
Action: Put-away w1 in boot
Observation: W1 is in boot. Goal is completed. You should perform actions to accomplish the goal: The goal is to satisfy the following conditions: Wheel r1 is inflated., r1 is on the-hub1., w1 is in boot.

You should use the following commands for help when your action cannot be understood: check valid actions

Observation: Boot is closed. Boot is unlocked. Hub the-hub1 is fastened. Hub the-hub1 is on the ground. Jack is in boot. Pump is in boot. R1 is in boot. The nut nuts1 on the hub the-hub1 is tight. W1 is on the-hub1. Wheel r1 is intact. Wheel r1 is not inflated. Wrench is in boot.

1 Subgoal: Open the boot to access the tools.
Observation: Boot is open.

2 Subgoal: Fetch the wrench from the boot.
Observation: You have wrench.

3 Subgoal: Loosen the nut on the hub the-hub1.
Observation: The nut nuts1 on the hub the-hub1 is loose.

4 Subgoal: Fetch the jack from the boot and jack up the hub the-hub1.
Observation: You have jack and hub the-hub1 is elevated. Subgoal is met.

5 Subgoal: Unfasten the nut nuts1 and remove wheel w1 from the hub the-hub1.
Action: undo nuts1 on the-hub1
Observation: Hub the-hub1 is unfastened. You have nuts1.
Action: Remove-wheel w1 from the-hub1
Observation: The-hub1 is free. You have w1.

C More details on Observation Summarization

C.1 Prompt Example

```
You are an advanced AI system tasked with summarizing and analyzing a series of action-observation pairs (trajectories) and determining whether a specific subgoal has been met.
Your goal is to create a summary that captures all essential information, decisions, and outcomes from the given trajectories, and indicate whether the subgoal has been met based on the summarized observations.
If there are no valid actions taken, you need to analyze the reason.

### Instructions:
1. Provide a summarized observation related to the subgoal in a concise manner.
2. Determine whether the subgoal has been met.
3. Do not output anything except whether summary and subgoal are met. Your output should be only one line. Do not output things like '##Summary', '##Summary and Analysis'.

{example}

##Trajectory
{formatted_trajectory}
```

```
##Subgoal:
{subgoal}
###Output:
```

C.2 Comparison with Other Task Planning Methods

To further validate the effectiveness of HIAGENT, we conducted comprehensive comparisons with other state-of-the-art efficient task planning methods, including Least-to-Most (Zhou et al., 2022) and Tree-Planner (Hu et al., 2023b).

Comparison with Least-to-Most. As shown in Table 2, we adapted the Least-to-Most approach to agent scenarios as our Task Decomposition baseline. The results demonstrate that HIAGENT achieves a 20% higher success rate while maintaining better context efficiency. This improvement can be attributed to our hierarchical memory management approach that enables more effective reasoning over long-horizon tasks.

Comparison with Tree-Planner. We also compared HIAGENT with Tree-Planner, another efficient task planning method, on the Tyreworld environment. The results are shown in Table 4.

Table 4: Performance comparison between HIAGENT and Tree-Planner on Tyreworld.

Method	Success Rate	Context Efficiency
HIAGENT	**60.0%**	**100.0%**
Tree-Planner	40.0%	94.7%

While Tree-Planner maintains relatively efficient context usage (94.7%), it falls short of HIAGENT's performance on long-horizon tasks. This performance gap can be attributed to our more effective hierarchical memory management approach, which enables better reasoning over extended action sequences.

These comprehensive comparisons with state-of-the-art methods further validate the effectiveness of our approach in handling complex task planning scenarios.