M-PLAN: COMPREHENSIVE PLANNING FOR LLM AGENTS WITH MEMORY ARRANGEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM)-based agents have recently demonstrated outstanding performance in single-turn tasks, yet they continue to struggle with long-horizon tasks, particularly in environments without prior training. A key limitation of single LLMs is their difficulty in processing long contexts, which stems from weaknesses in continuous decision-making and reasoning capabilities. Although existing multi-agent systems (MAS) equipped with long-term memory bases can partially address this gap, they mostly rely on knowledge reuse or summarization—an approach that is often constrained by directly retrieval without any adaptation for the current task, while the retrieved memory may contain much history noises waiting to be preprocessed. In contrast, MAS characterized by task-specific guidance often lack long-term memory specifically designed for the guidance. To address these challenges, we propose M-Plan, a closed-loop MAS framework that integrates long-term and short-term memory. By combining global-local and past-current guidance, M-Plan enhances agents' performance on multi-turn tasks while maintaining a plug-and-play design. During task execution, agents receive task-specific milestones and insights, and are provided with local hints only when confusion arises, which form global-local guidance together. These guidance generation and utilization construct the short-term memory as well. After task completion, relevant information about the current task is optimized or reorganized before being sent to the long-term memory storage for future retrieval. The designments mentioned above build a comprehensive framework different from other individual long-term or short-term memory works. Extensive experiments on multi-turn tasks including ALFWorld, Webshop, and PDDL across diverse models demonstrate that M-Plan outperforms multiple baseline methods in both accuracy and efficiency.

1 INTRODUCTION

Large language models (LLMs) have been widely applied across numerous domains, and extensive research has demonstrated their strong capability to solve a diverse range of tasks (1). With the increasing continuity and complexity of real-world tasks in daily human activities, LLMs are required to tackle multi-turn, long-horizon tasks more frequently. Such tasks demand sustained decision-making and reasoning across multiple turns, while also requiring models to maintain and process long contexts. However, these demands remain challenging for existing LLMs, which, despite their remarkable performance on single-turn instructions, often struggle in multi-turn scenarios (2; 3). Common issues observed in LLM-based agents include forgetting information from intermediate turns, overemphasizing initial and final turns, which indicate that a single LLM is insufficient to handle increasingly complex contexts.

To address these challenges, researchers have proposed two major approaches for adapting LLM-based agents. The first exploits training to optimize agents on downstream tasks (4; 5; 6; 7; 8; 9; 10), primarily including Supervised Fine-Tuning (SFT) and Reinforcement Learning (RL), typically with human-annotated training sets, which are sometimes hard to get. The second adopts training-free methods, which typically employ a specialized multi-agent system (MAS) to process long contexts. Most existing training-free approaches integrate MAS with long term memory mechanisms (11; 12; 13; 14), usually incurring lower computational costs with greater flexibility and interpretability. For example, through setting specific agents to examine the task execution result, process successful and failed trajectories, ReasoningBank (14) can learn from historical experiences to adapt to other

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

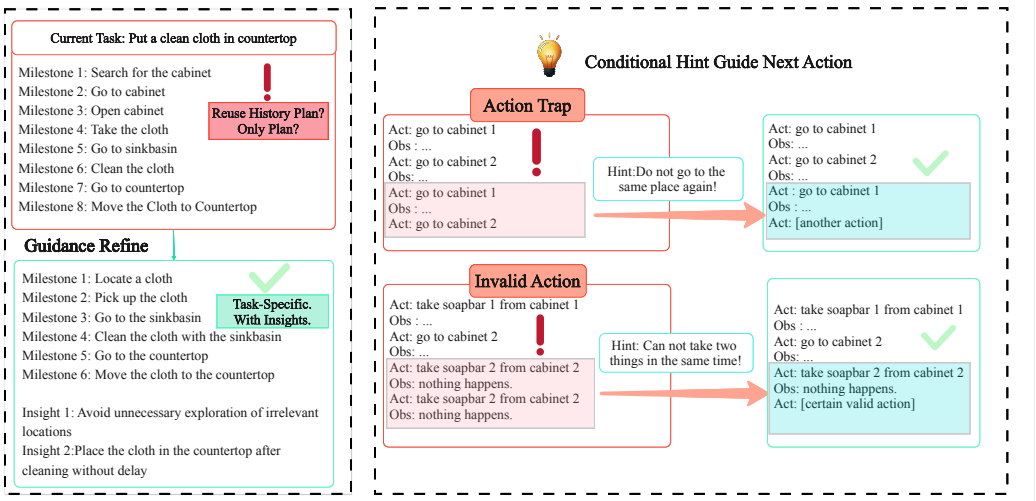


Figure 1: The task-specific guidance consists of milestones and insights: milestones guide the action agent to follow a global task-solving path, while insights provide detailed considerations and important task-relevant knowledge; conditional hint when meeting action trap (invalid action). The hint will help correct the action in the next step.

tasks, and construct a visual external memory base, which is more interpretable than training. As for short-term memory, characterized by a combination of global and step-wise guidance (15; 16; 17), those works are able to build a overall route, possibly with local hints, helping the action agent make decision more wisely.

Despite remarkable progress in MAS-based approaches, the aforementioned methods still suffer from several limitations. For long-term memory, typical approaches involve reusing historical experience or summarizing general knowledge, but these methods struggle to ensure generalization, and long historical trajectories may introduce noise and instability. For short-term memory, step-wise generation inevitably incurs substantial token consumption, whereas relying solely on global guidance often fails to provide complete task instructions, since high-level blueprints or milestones lack the necessary details and action-related knowledge required for execution. In addition, existing MAS-based planning works generally lack memory association, which weakens their self-evolution capabilities and limits their applicability to real-world scenarios.

To address these limitations, we propose M-Plan, a long and short-term memory MAS framework. The short-term memory integrates comprehensive task-specific global guidance, as shown in the left part of Fig.1 and local hints in the right part. With task-specific milestones and insights, our short-term memory can digest the retrieved information into task-specific guidance by refusing simply retrieval and adding insights to complement details, solving reuse problems in individual long-term memory. Not like the former step-wise guidance work, considering that action traps and invalid actions remain prevalent in most multi-turn tasks, our conditionally triggered hints, can reduce token consumption while helping agents correct their actions in subsequent steps. However, a short-term memory usually lacks a great partner in similar work, and our long-term memory complements short-term memory for this purpose. Precisely, to better adapt agents to new tasks, M-Plan maintains a long-term memory base for guidance, helping task-specific short-term memory further enhances the effectiveness of long-term memory during the current task. To improve the quality of information stored in the memory base and connect the short- and long-term memory, we further design an OPTIMIZER module to refine completed task trajectories before they are stored as long-term memory, helping the short- and long-term memory achieve better complementarity.

We evaluate M-Plan on three commonly used multi-turn benchmarks: ALFWorld(18), Webshop(19) and PDDL(20), which are filled with long-horizon and context-rich tasks. Experiments on two models of different scales demonstrate that M-Plan outperforms classic and typical baselines in terms of effectiveness (accuracy and reward). To assess efficiency, we report the average steps, redefined steps

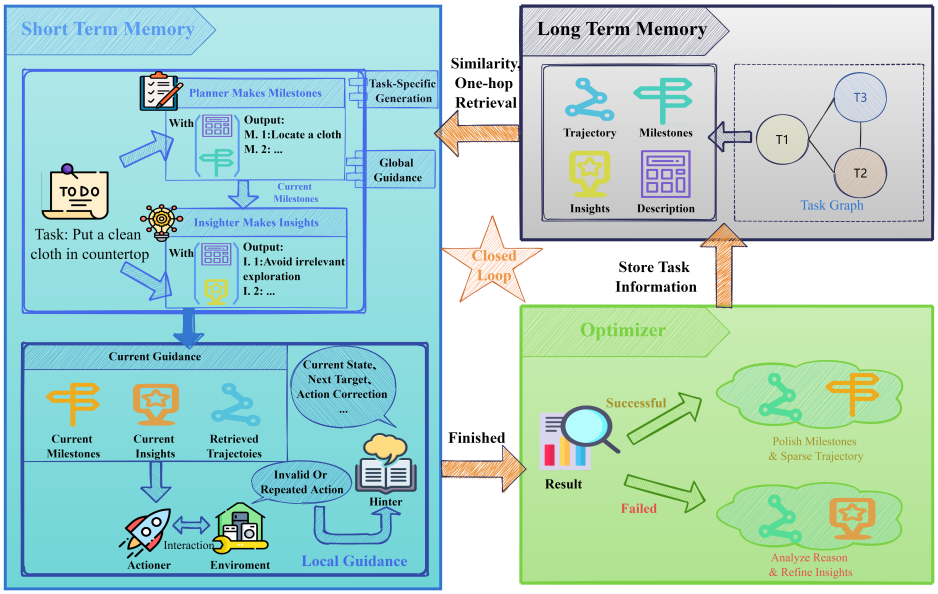


Figure 2: The whole framework of M-Plan, containing long short-term memory, global-local guidance (milestones, insights, hints), and an optimizer module to form the closed-loop.

and compare valid token consumption of each successful task. Additionally, ablation studies verify the value of each designed module in enhancing performance.

The contributions of this paper can be summarized as follows:

1. We propose M-Plan, a MAS framework that integrates long-term memory storage, short-term memory planning mechanisms, and global-local guidance. This framework eliminates the need for reusing abstract knowledge, maintains the generation of valuable guidance through long-term memory, avoids uncertainty, and controls token costs via conditional local guidance. This comprehensive closed-loop framework is effective across different models and can be theoretically extended to various multi-turn tasks.
2. We construct a closed-loop and complete system. In M-Plan, milestones and insights form a complete guidance system, with conditional local hints providing step-wise assistance when agents encounter confusion, with long term memory offering information from optimizer as generation references, which can abandon traditional reuse-based approaches for guiding agent-environment interactions.
3. We conduct experiments on challenging benchmarks, and by leveraging task-specific information, M-Plan utilizes tokens more efficiently while outperforming all baseline methods using different base models.

2 Related Work

2.1 Long Term Memory

Existing long-term memory works can be broadly categorized into parametric or non-parametric approaches. Parametric methods typically rely on large-scale trajectories or other pre-collected information to train models through reinforcement learning (RL) or supervised fine-tuning (SFT), thereby encoding long-term memory into model parameters (21; 22; 23). For example, MEMOS (21) proposes a large and complex memory operating system with parametric memory storage. Memento (9) further uses RL to improve retrieval quality. However, parametric memory is still ultimately derived from non-parametric data, such as textual knowledge or trajectories. Although training can

improve agents' performance on downstream tasks, it introduces substantial computational costs and usually requires complex data preparation in advance.

To support plug-and-play and convenient deployment, lots of works design non-parametric memory (11; 14; 13; 24; 25; 26). A typical method (27; 28; 29; 30) is to construct memory bases using vectors or textual representations for semantic retrieval. A representative paradigm is to store extracted, summarized, or general knowledge and reuse it to guide agents in action execution. For example, work (11) sets a series of agents refining, editing, and integrating general insights to be reused in future tasks, and also extracts the key steps similar to the plan of the history trajectories as in-context content. However, these methods often struggle to extract truly transferable knowledge and usually lack task-specific adaptation. Directly using retrieved memories may also introduce semantic noise from other tasks. Although historical trajectories can serve as useful examples, they are not always reliable in multi-step decision-making scenarios. To improve robustness and task-specific adaptation, M-Plan introduces a short-term memory mechanism on top of long-term memory. We further design an *Optimizer* module to compress and refine completed task experiences, forming a closed-loop process that feeds successful experiences and failure lessons back into long-term memory. In this way, retrieval and feedback integrate long-term and short-term memory into a unified system.

2.2 Short Term Memory

Agent-based short-term memory mechanisms for planning and acting have shown significant potential in multi-turn interactive tasks. Existing methods can be roughly divided into two categories. The first category relies on global guidance (31), step-wise guidance (32; 33), or their combination. For example, HiAgent (16) segments the working trajectory into subgoals to organize the context, enabling partial local compression and guidance. HiPlan (17) introduces a planning system equipped with global milestones and step-wise hints. Another common approach is to insert reasoning traces into the action trajectory, with ReAct (34) as a typical example, followed by related studies (15; 35). However, these methods overlook the fact that plans alone cannot always provide sufficient guidance for task completion. Moreover, step-wise generation or judgment can introduce substantial time and token costs. The second category adopts retrieval-augmented planning, such as (15; 17). Although these methods are convenient and effective, they often rely on few-shot examples like full trajectories or other static memory. To provide deeper and more comprehensive guidance during task execution, M-Plan introduces an *Insighter* into the short-term memory module. We further design a dynamic external memory base for task-level retrieval, enabling short-term memory to evolve across different agents.

2.3 Long Short-Term Memory

Several existing works have introduced systems that combine long-term and short-term memory (23; 36; 37). For example, MemVerse (23) adopts a sliding window to process short-range dialogue information and uses a relational graph to store task information, including episodic, core, and semantic memories in textual form. It further employs a supervised fine-tuned model to transform graph-based memories into model weights. Another work (36) constructs process and feasibility memories to organize knowledge from successful and failed trajectories, while also monitoring task progress locally. However, these methods typically require retrieval or complex memory modification at each step, which may slow down execution and substantially increase computational cost. In contrast, our system avoids frequent step-wise retrieval or examination while maintaining strong performance and further establishes a closed-loop optimization mechanism through the interaction between long-term and short-term memory.

3 Methodology

3.1 Overview

As shown in Figure 2, from the function view, M-Plan can be separated into three modules: long-term memory storage, short-term memory generation, and optimizer. Following M-Plan's workflow, there will be a closed-loop in the whole, meaning that results made by short-term memory can influence it in a future turn, and information derived from long-term memory can enlarge the base after finishing

the current task. Overall, with MAS design, whose prompts are presented in Appendix. A, M-Plan can utilize the following two kinds of information as guidance to help action agents execute the task:

1. Global guidance: A generated milestones list $M_g = [m_1, m_2, m_3, ..., m_x]$, playing the role of planning route, where each m_x represents one key subgoal need to achieve. A generated insights list $I_g = [i_1, i_2, i_3, ..., i_y]$, where each i_y represents one key insight as noticeable reference. Historically successful trajectories T_g from memory base provide demonstrations and experiences like the human recalling when meeting similar tasks.
2. Local guidance: A structural hint h_l can be included to the context to remind the action agent to get rid of the current problems, performing as a local guide. However, only under the conditions where the action agent is confused, the hint will be generated and exploited. This strategy can be likened to a wise, deep-thinking pattern with an external brain assistant, and also conforms to human thinking patterns, in which people will think more deeply to tackle traps or trouble.

3.2 Long-term Memory Storage

We first introduce the long-term memory storage module, able to break the limitation of single short-term memory in most work. This module equips agents with historical task information as references, thereby enabling more effective task-specific generation. Notably, the memory base does not rely on any pre-collected data or demonstrations. Instead, it starts from scratch and continuously absorbs external information from short-term memory during task execution. Specifically, Each memory item E_l is defined as follows:

$$E_l = \{M_l, I_l, T_l, F_l, D_l\}, \tag{1}$$

where M_l denotes task milestones, I_l represents task insights, T_l records the execution trajectory, F_l is a Boolean flag indicating whether the trajectory succeeds or fails, and D_l is the task description, which serves as the key for semantic search. These memory items are stored in a vector database. In addition, following the graph-based memory organization strategy in (11), we model the relationships among memory items in graph form, where edges are added between similar tasks to facilitate faster retrieval and better structural organization.

To better utilize long-term memory, the retrieval process can be formulated as follows:

$$\begin{aligned} &\{E_1, E_2, ...E_j\} = Ret(Q), ES = \{E_1, E_2, ...E_k\} \\ &\{E_1, E_2, ..., E_j, ..., E_k\} = Extend(ES), \end{aligned} \tag{2}$$

where $Ret(Q)$ denotes the retrieval function triggered by the task query Q, which retrieves j relevant memory items from the memory base according to semantic cosine similarity. The retrieved set is denoted as ES. To further improve diversity and robustness, we apply a one-hop graph expansion strategy to enlarge the retrieval scope and obtain the final top-k memory items. Specifically, for each item in ES, we search its directly connected one-hop neighbors in the memory graph and append them to the original retrieved set, beyond those retrieved by semantic similarity.

3.3 Short-term Memory Generation

However, current work relying solely on long-term memory may be insufficient for task-specific enhancement because historical experiences mainly provide coarse references from past tasks and cannot fully capture the dynamic states, and task-specific constraints of the current execution process. Moreover, most existing short-term planning works generate only coarse-grained plans, lacking sufficient details. And step-wise guidance at every step can also introduce substantial computational cost. To address these limitations, we introduced a multi-agent based short-term memory, including a $Planner$, an $Insighter$, a $Hinter$, and an $Actioner$. Specifically, we first retrieve two successful cases and one failed case from long-term memory. The two success trajectories serve as fundamental in-context demonstrations. The $Planner$ receives the description D_l and milestones M_l of past successful trajectories for positive reference, together with the current task description D_s, and generates a new milestone list for the current task:

$$\{m_{s1}, m_{s2}, m_{s3}, ..., m_{sx}\} = Planner(D_l, M_l, D_s). \tag{3}$$

The $Insighter$ receives the milestone list $M_s = \{m_{s1}, m_{s2}, m_{s3}, ..., m_{sx}\}$, adding more related details to enrich the content of the blueprint, and takes the task description D_l from both successful and failed cases, along with the current task description D_s, the current task milestones and their corresponding insight list I_l, to identify the information that may be valuable for the current task. It then generates a new insight list:

$$\{i_{s1}, i_{s2}, i_{s3}, ..., i_{sy}\} = Insighter(D_l, I_l, D_s). \tag{4}$$

Then, the $Hinter$ receives D_s, the ongoing trajectory T_c, the global milestone list M_s and insight list $I_s = \{i_{s1}, i_{s2}, i_{s3}, ..., i_{sy}\}$ as input. The hint-generation process is formulated as follows:

$$\{state, position, gap, correction\} = Hinter(T_c, M_s, I_s, D_s). \tag{5}$$

To reduce computational cost, we adopt a conditional step-wise hinting strategy. Instead of guiding every step, the $Hinter$ is activated only when the $Actioner$ falls into an action trap or performs an invalid action according to the environment feedback. More detailed rules of the $Hinter$ are expanded on Appendix. D. The generated hint consists of four components: $state$, which describes the agent's current state, such as objects it has already collected; $position$, which indicates the current milestone to be achieved; gap, which provides advice on how to complete the current milestone; and $correction$, which suggests an action correction for the next step.

The $Actioner$ is responsible for selecting and executing actions based on the current observation, task description, generated milestones, insights, and conditional hints.

3.4 Optimizer

Apart from long and short-term memory, there remains a problem of how to process and exploit the information from finished tasks, which is also rarely explored in current work. When $Planner$ or $Insighter$ directly receives unrefined information, their performance may be hindered, especially when the long-term memory is still in a raw stage. To address this issue, we design an $Optimizer$ module to reflect on finished task trajectories, including both successful and failed ones, and to form a closed-loop optimization process. Specifically, the $Optimizer$ works by refining the current task-specific generation, rather than entirely restarting summarization or extraction. The $Optimizer$ can be divided into three small parts: milestone list polishing for M_s, insight list complement for I_s, and trajectory sparsification for the entire task trajectory $Traj$.

In our former exploration phase, we found that a successfully executed multi-turn task may still have an imperfect M_s, which can be polished by analyzing the whole trajectory. When the task successfully finished, the $optimizer$ is more likely to discover redundant milestones during execution. However, it is harder for $optimizer$ to polish M_s when the task fails, because the failure is mostly caused by not following M_s well. In other words, M_s is usually correct, though it may not be perfect, which is why we only polish it when the task succeeds. In contrast, for I_s, it is harder to extract more valuable information from successful tasks, but notable points can often be obtained from failed ones. Moreover, given that refining I_s solely from the current failed trajectory remains challenging, we also input the most similar retrieved successful trajectory to facilitate this refinement by contrasting failed and successful rollouts. The sparsification of $Traj$ is much simpler, since it only removes repeated or invalid action-observation pairs in the trajectory. Note that sparsification is only applied to successful trajectories, since failed ones need to keep failure-related details. Overall, the outputs of the three parts are derived from the short-term memory mechanism above and will be sent to long-term memory storage as references for short-term memory generation in the next task, and then form a complete closed loop between long-term and short-term memory.

4 Experiment

We evaluate M-Plan on three widely test multi-turn benchmarks:ALFWorld (18), Webshop (19), PDDL (20), and take G-Memory (11), ReasoningBank (14) and Hiplan (17) as baselines. The detailed benchmark and baseline introduction can be seen in Appendix. C.

4.1 Metrics

Effectiveness: For ALFWorld (18), we report the success rate (`SR.`) to indicate the proportion of tasks completed successfully. For Webshop (19) and PDDL (20), in addition to `SR.`, we report the

Table 1: Results on ALFWorld, Webshop and PDDL. We report Success Rate (SR.) on all three benchmarks and Average Reward (AD) on Webshop and PDDL. Details about metrics can be seen in Sec. 4.1.

Models	Methods	ALFWorld	Webshop		PDDL	
		SR.	SR.	AD	SR.	AD
Qwen3-32B	ReasoningBank	0.694±0.000	0.268±0.012	0.535±0.025	0.239±0.010	0.473±0.014
	G-Memory	0.850±0.013	0.468±0.008	0.700±0.003	0.256±0.020	0.470±0.028
	Hiplan	0.943±0.009	0.395±0.000	0.525±0.000	0.222±0.009	0.392±0.007
	M-Plan	**0.993±0.000**	**0.565±0.000**	**0.747±0.000**	**0.528±0.009**	**0.743±0.013**
Qwen3-14B	ReasoningBank	0.657±0.000	0.282±0.006	0.607±0.017	0.172±0.009	0.379±0.003
	G-Memory	0.846±0.005	0.573±0.012	0.798±0.031	0.244±0.020	0.418±0.052
	Hiplan	0.878±0.017	0.320±0.000	0.448±0.000	0.150±0.017	0.255±0.008
	M-Plan	**0.908±0.004**	**0.682±0.003**	**0.826±0.001**	**0.461±0.019**	**0.689±0.005**

average reward to reflect the degree of attribution alignment (*i.e.*, how well the purchased objects satisfy the target requirements).

Efficiency: We report the average number of steps over all tasks (AS.) for ALFWorld (18) and PDDL (20) to directly compare the action efficiency of different agents. In addition, we separately report the average number of steps for successful tasks (ASS.), which reflects efficiency under a fixed success rate. However, a common phenomenon in these benchmarks is that a lower SR. may lead to a lower ASS., because the agent may only succeed on easier tasks that require fewer steps. Therefore, for a fairer comparison, we define Meta-Step, computed as ASS./SR., as a more comprehensive efficiency metric. For Webshop (19), unlike ALFWorld, there is no fixed number of steps for failed tasks, and most failures result from incorrect purchases made in early steps. Thus, a smaller AS. does not necessarily indicate better efficiency. Therefore, for Webshop, we mainly focus on Meta-Step.

Another important dimension of efficiency is token usage. Given that a certain number of tokens are consumed to complete a task, both gains and losses should ideally be considered. However, the loss caused by task failure is difficult to quantify in these benchmarks. Therefore, we adopt Ft as a more robust and interpretable measure of pure profit, defined as:

$$Ft = \frac{2}{\pi} * \arctan(SR/FR), \quad Ct = \frac{At}{Ft}. \tag{6}$$

where At denotes the average number of tokens consumed across all tasks and FR denotes the failure rate, and both SR and FR are normalized. When At is fixed, a lower Ct and a higher Ft indicate that the same token budget is expected to yield a greater advantage of success over failure. There will be detailed discussion about how we get this equation in the Appendix. B. Theoretically, At is positively correlated with AS under a fixed framework when each action consumes a similar number of tokens. However, we observed that in Webshop, certain actions such as examining product information incur substantially higher token overhead than simple click operations, breaking this balance. We therefore discard this metric for the Webshop environment to eliminate action-level unfairness.

4.2 Main Results

We present our main results with performing each experiment three times to evaluate the possible variances, in Table 1, Table 2, and Table 3. To improve the experience of reading, all results of step and token with standard variance are presented in Appendix. E. These results compare both effectiveness and efficiency, showing that M-Plan consistently outperforms all baselines in both aspects. We provide a detailed analysis below.

Effectiveness: Table 1 summarizes the effectiveness results in terms of success rate (SR.) and average reward (AD). M-Plan achieves state-of-the-art performance with both Qwen3-32B and Qwen3-14B backbones. On ALFWorld, M-Plan attains the highest SR. of 0.993±0.000 using Qwen3-32B, surpassing Hiplan (the second-best baseline) and substantially outperforming the memory-reuse method ReasoningBank. M-Plan maintains strong performance across diverse task categories without obvious failure modes on specific task types. On Webshop, M-Plan exhibits an

Table 2: Comparison of average steps on ALFWorld, Webshop and PDDL. We report AS., ASS., and Meta-Step, while AS. is not considered for Webshop due to its specific characteristics, as discussed in Sec. 4.1.

Models	Methods	ALFWorld			Webshop		PDDL		
		AS.	ASS.	Meta-Step	ASS.	Meta-Step	AS.	ASS.	Meta-Step
Qwen3-32B	ReasoningBank	24.99	13.96	20.12	6.14	22.90	26.61	15.76	65.90
	G-Memory	24.93	20.53	24.14	**5.10**	10.90	26.95	18.16	71.73
	Hiplan	18.38	16.46	17.46	8.09	20.48	26.33	**13.48**	60.70
	M-Plan	**12.48**	**12.20**	**12.28**	5.35	**9.47**	**22.30**	15.42	**29.22**
Qwen3-14B	ReasoningBank	26.30	13.91	21.17	5.90	20.93	28.16	19.33	112.42
	G-Memory	18.66	12.94	15.30	**5.13**	8.96	26.97	17.54	71.86
	Hiplan	20.40	16.28	18.55	8.05	25.16	27.56	**13.71**	92.18
	M-Plan	**14.63**	**11.05**	**12.17**	5.48	**8.03**	**23.33**	15.54	**33.75**

even larger performance gap compared with baselines. With Qwen3-32B, M-Plan improves SR. by roughly 20% and average reward by around 7% relative to G-Memory. When switching to the smaller Qwen3-14B backbone, the performance advantage over G-Memory narrows on ALFWorld, which we attribute to weaker capability of the smaller LLM to generate high-quality task-specific guidance. Even so, M-Plan consistently surpasses memory-reuse baselines such as G-Memory, verifying the benefits of our task-specific short-term memory design and the dedicated optimizer modules. Compared with the step-wise planning baseline Hiplan, the gains demonstrate that M-Plan can yield superior success rates without injecting hints at every single action step. On the more challenging PDDL benchmark, M-Plan achieves a clear margin over all baselines: with Qwen3-32B, its SR. reaches 0.528±0.009, more than doubling the success rate of ReasoningBank and G-Memory, suggesting that even in complex scenarios, M-plan can achieve great performance.

Step Efficiency: We analyze action efficiency using step-based metrics in Table 2. On ALFWorld, M-Plan achieves the lowest AS. and Meta-Step under both model sizes. For Qwen3-32B, M-Plan reduces Meta-Step to 12.28 , around 30% lower than Hiplan. This indicates that M-Plan's *Actioner* leverages memory and guidance effectively to complete tasks with fewer actions while maintaining high success. For Webshop, where AS. is not a reliable metric, we focus on Meta-Step. M-Plan achieves the smallest Meta-Step in both model settings. With Qwen3-32B, M-Plan cuts the redefined step overhead by over 50% relative to Hiplan and around 10% compared with G-Memory. On PDDL, M-Plan also delivers the lowest Meta-Step, showing that the planning and memory mechanism reduces redundant actions even for complex planning tasks. These step metrics collectively show that M-Plan improves action efficiency without sacrificing task success.

Token Efficiency: We further evaluate token-level efficiency with At, Ft, and Ct in Table 3. Ft measures the pure success profit, while Ct represents the normalized token cost per unit profit: a smaller Ct means the agent gains stronger success-failure advantage under the same token budget. Across ALFWorld and PDDL, M-Plan yields substantially lower Ct than all baselines. On ALFWorld with Qwen3-32B, M-Plan's Ct is nearly 70% lower than Hiplan, demonstrating that our framework extracts higher task profit at a much cheaper token cost. The step-wise baseline Hiplan suffers particularly high token consumption, especially on PDDL, which implies that per-step local guidance incurs heavy token overhead that may offset its planning benefits. In contrast, M-Plan's selective hinting and memory retrieval strategy reduce unnecessary LLM invocations, achieving better success with more economical token usage.

4.3 Ablation Study

To further demonstrate the effectiveness and efficiency of each module in M-Plan, we conduct ablation experiments on PDDL tasks using the Qwen3-32B model, with results summarized in Table 4. Compared with the full standard model, removing the *Optimizer* reduces SR. from 0.528 to 0.478 and lowers AD from 0.743 to 0.682, while Meta-Step. slightly rises. This confirms that the *Optimizer* helps refine generated plans and mitigates flawed reasoning, lifting the overall success rate and plan quality.

When the *Retrieval* module is disabled so that the short-term memory cannot access historical cases stored in long-term memory, we observe a substantial performance degradation: SR. falls to 0.425,

Table 3: Token-efficiency comparison. We report At, Ft and Ct (see in Sec. 4.1), mainly focusing on the comparison of Ct. Webshop is not discussed for the fairness in Sec. 4.1.

Models	Methods	ALFWorld			PDDL		
		At(M)	Ft	Ct(M)	At(M)	Ft	Ct(M)
Qwen3-32B	ReasoningBank	0.04	0.74	0.05	**0.07**	0.19	0.34
	G-Memory	0.05	0.93	0.06	0.12	0.21	0.59
	Hiplan	0.09	0.96	0.10	0.21	0.18	1.17
	M-Plan	**0.03**	**1.00**	**0.03**	0.17	**0.54**	**0.32**
Qwen3-14B	ReasoningBank	0.04	0.69	0.06	**0.07**	0.13	0.54
	G-Memory	0.05	0.89	0.06	0.12	0.20	0.63
	Hiplan	0.11	0.91	0.12	0.21	0.11	1.89
	M-Plan	**0.04**	**0.94**	**0.05**	0.16	**0.45**	**0.36**

Table 4: Ablation study of M-Plan on PDDL with Qwen3-32B. *Retrieval* indicates whether the short-term memory module receives cases from long-term memory.

Methods	Modules				Performance			
	Optimizer	*Retrieval*	*Insighter*	*Hinter*	`SR.`	AD	`AS.`	`Meta-Step.`
Standard	✓	✓	✓	✓	**0.528±0.009**	**0.743±0.013**	**22.30**	**29.22**
w/o *Optimizer*	✗	✓	✓	✓	0.478±0.024	0.682±0.020	22.35	29.58
w/o *Retrieval*	✓	✗	✓	✓	0.425±0.008	0.614±0.009	23.63	35.12
w/o *Insighter*	✓	✓	✗	✓	0.483±0.029	0.668±0.015	22.81	31.44
w/o *Hinter*	✓	✓	✓	✗	0.411±0.019	0.616±0.018	24.01	37.64

AD drops to 0.614, and both `AS.` and `Meta-Step.` increase markedly. This verifies that retrieved prior planning cases supply useful experience for the *Planner* and *Insighter*, reducing redundant exploration and supporting more efficient reasoning.

Ablating the *Insighter* also harms performance. The variant without *Insighter* achieves `SR.`=0.483 and AD=0.668, accompanied by higher `AS.` and `Meta-Step.`. This indicates that task-specific insights extracted by the *Insighter* offer complementary cues that regular planning modules lack, helping the model better capture task constraints and produce more reasonable plans. Finally, removing the *Hinter* yields the lowest `SR.` (0.411) and AD (0.616), together with the largest `Meta-Step.` of all ablated variants. Such a severe regression highlights the critical role of the *Hinter*: it delivers intermediate hints during iterative planning, guides the model away from dead-end paths, and suppresses excessive useless reasoning steps.

Across all variants, every proposed module contributes positively to the full M-Plan pipeline. Their combined effect improves task success, enhances plan quality measured by AD, and reduces redundant reasoning overhead in PDDL planning.

5 Conclusion

In this paper, we have proposed a comprehensive planning framework for LLM agents with memory arrangement. M-Plan combines long short-term memory and global-local guidance together, which can boost LLM agents to tackle multi-turn or other similar complex tasks. The framework can address the limitations of reusing, high cost, not fully guided by abundant global and conditional local guidance, and lacking evolution mechanisms by long short-term memory coordination with closed-loop optimization. The experiments on simulated environment benchmarks, ALFWorld, Webshop and PDDL fully show more admirable effectiveness and efficiency than baselines.

Limitation: While M-Plan is mature in long-horizon text processing, there is great potential for M-Plan to be extended to deal with real-world environments, and it can be applied with some compression and storage techniques to enhance engineering value.

REFERENCES

[1] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wensen Cheng, Qi Zhang, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huang, and Tao Gui. The rise and potential of large language model based agents: A survey, 2023.

[2] Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. Llms get lost in multi-turn conversation, 2025.

[3] Yubo Li, Xiaobin Shen, Yidi Miao, Xinyu Yao, Xueying Ding, Ramayya Krishnan, and Rema Padman. Beyond single-turn: A survey on multi-turn interactions with large language models, 2026.

[4] Saman Forouzandeh, Wei Peng, Parham Moradi, Xinghuo Yu, and Mahdi Jalili. Learning hierarchical procedural memory for llm agents through bayesian selection and contrastive refinement, 2025.

[5] Yinjie Wang, Tianbao Xie, Ke Shen, Mengdi Wang, and Ling Yang. Rlanything: Forge environment, policy, and reward model in completely dynamic rl system, 2026.

[6] Yuanzhe Li, Jianing Deng, Jingtong Hu, Tianlong Chen, Song Wang, and Huanrui Yang. Dynamic mix precision routing for efficient multi-step llm interaction, 2026.

[7] Zhiheng Xi, Jixuan Huang, Chenyang Liao, Baodai Huang, Honglin Guo, Jiaqi Liu, Rui Zheng, Junjie Ye, Jiazheng Zhang, Wenxiang Chen, Wei He, Yiwen Ding, Guanyu Li, Zehui Chen, Zhengyin Du, Xuesong Yao, Yufei Xu, Jiecao Chen, Tao Gui, Zuxuan Wu, Qi Zhang, Xuanjing Huang, and Yu-Gang Jiang. Agentgym-rl: Training llm agents for long-horizon decision making through multi-turn reinforcement learning, 2025.

[8] Jiangweizhi Peng, Yuanxin Liu, Ruida Zhou, Charles Fleming, Zhaoran Wang, Alfredo Garcia, and Mingyi Hong. Hiper: Hierarchical reinforcement learning with explicit credit assignment for large language model agents, 2026.

[9] Huichi Zhou, Yihang Chen, Siyuan Guo, Xue Yan, Kin Hei Lee, Zihan Wang, Ka Yiu Lee, Guchun Zhang, Kun Shao, Linyi Yang, and Jun Wang. Memento: Fine-tuning llm agents without fine-tuning llms, 2025.

[10] Xuhui Ren, Shaokang Dong, Chen Yang, Qing Gao, Yunbin Zhao, Yongsheng Liu, Xinwei Geng, Xiang Li, Demei Yan, Yanqing Li, Chenhao Huang, Dingwei Zhu, Junjie Ye, Boxuan Yue, Yingnan Fu, Mengzhe Lv, Zezeng Feng, Boshen Zhou, Bocheng Wang, Xuanjing Huang, Yu-Gang Jiang, Tao Gui, Qi Zhang, and Yunke Zhang. Magicagent: Towards generalized agent planning, 2026.

[11] Guibin Zhang, Muxin Fu, Guancheng Wan, Miao Yu, Kun Wang, and Shuicheng Yan. G-memory: Tracing hierarchical memory for multi-agent systems, 2025.

[12] Yu Wang and Xi Chen. Mirix: Multi-agent memory system for llm-based agents, 2025.

[13] Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. Memskill: Learning and evolving memory skills for self-evolving agents, 2026.

[14] Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. Reasoningbank: Scaling agent self-evolving with reasoning memory, 2026.

[15] Ruiwen Zhou, Yingxuan Yang, Muning Wen, Ying Wen, Wenhao Wang, Chunling Xi, Guoqiang Xu, Yong Yu, and Weinan Zhang. Trad: Enhancing llm agents with step-wise thought retrieval and aligned decision, 2024.

[16] Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. Hiagent: Hierarchical working memory management for solving long-horizon agent tasks with large language model, 2024.

[17] Ziyue Li, Yuan Chang, Gaihong Yu, and Xiaoqiu Le. Hiplan: Hierarchical planning for llm-based agents with adaptive global-local guidance, 2025.

[18] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning, 2021.

[19] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents, 2023.

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

[21] Zhiyu Li, Chenyang Xi, Chunyu Li, Ding Chen, Boyu Chen, Shichao Song, Simin Niu, Hanyu Wang, Jiawei Yang, Chen Tang, Qingchen Yu, Jihao Zhao, Yezhaohui Wang, Peng Liu, Zehao Lin, Pengyuan Wang, Jiahao Huo, Tianyi Chen, Kai Chen, Kehang Li, Zhen Tao, Huayi Lai, Hao Wu, Bo Tang, Zhengren Wang, Zhaoxin Fan, Ningyu Zhang, Linfeng Zhang, Junchi Yan, Mingchuan Yang, Tong Xu, Wei Xu, Huajun Chen, Haofen Wang, Hongkang Yang, Wentao Zhang, Zhi-Qin John Xu, Siheng Chen, and Feiyu Xiong. Memos: A memory os for ai system, 2025.

[22] Hongkang Yang, Zehao Lin, Wenjin Wang, Hao Wu, Zhiyu Li, Bo Tang, Wenqiang Wei, Jinbo Wang, Zeyun Tang, Shichao Song, Chenyang Xi, Yu Yu, Kai Chen, Feiyu Xiong, Linpeng Tang, and Weinan E. Memory^3: Language modeling with explicit memory. *Journal of Machine Learning*, 3(3):300–346, 2024.

[23] Junming Liu, Yifei Sun, Weihua Cheng, Haodong Lei, Yirong Chen, Licheng Wen, Xuemeng Yang, Daocheng Fu, Pinlong Cai, Nianchen Deng, Yi Yu, Shuyue Hu, Botian Shi, and Ding Wang. Memverse: Multimodal memory for lifelong learning agents, 2025.

[24] Xiangru Tang, Tianyu Hu, Muyang Ye, Yanjun Shao, Xunjian Yin, Siru Ouyang, Wangchunshu Zhou, Pan Lu, Zhuosheng Zhang, Yilun Zhao, Arman Cohan, and Mark Gerstein. Chemagent: Self-updating library in large language models improves chemical reasoning, 2025.

[25] Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory, 2023.

[26] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior, 2023.

[27] Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory, 2024.

[28] Sirui Liang, Pengfei Cao, Jian Zhao, Wenhao Teng, Xiangwen Liao, Jun Zhao, and Kang Liu. Learning how to remember: A meta-cognitive management method for structured and transferable agent memory, 2026.

[29] Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems, 2024.

[30] Xuechen Liang, Meiling Tao, Yinghui Xia, Jianhui Wang, Kun Li, Yijin Wang, Yangfan He, Jingsong Yang, Tianyu Shi, Yuantao Wang, Miao Zhang, and Xueqian Wang. Sage: Self-evolving agents with reflective and memory-augmented abilities. *Neurocomputing*, 647:130470, 2025.

[31] Simeng Sun, Yang Liu, Shuohang Wang, Chenguang Zhu, and Mohit Iyyer. Pearl: Prompting large language models to plan and execute actions over long documents, 2023.

[32] Lutfi Eren Erdogan, Nicholas Lee, Sehoon Kim, Suhong Moon, Hiroki Furuta, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. Plan-and-act: Improving planning of agents for long-horizon tasks, 2025.

[33] Minh Nguyen and Ehsan Shareghi. One step at a time: Language agents are stepwise planners, 2024.

[34] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023.

[35] Vishnu Sarukkai, Asanshay Gupta, James Hong, Michaël Gharbi, and Kayvon Fatahalian. Inference-time distillation: Cost-efficient agents without fine-tuning or manual prompt engineering, 2026.

[36] Bin Wen, Ruoxuan Zhang, Yang Chen, Hongxia Xie, and Lan-Zhe Guo. Aligning progress and feasibility: A neuro-symbolic dual memory framework for long-horizon llm agents, 2026.

[37] Qitong Sun, Jun Han, Tianlin Li, Zhe Tang, Sheng Chen, Fei Yang, Aishan Liu, Xianglong Liu, and Yang Liu. Kernelskill: A multi-agent framework for gpu kernel optimization, 2026.

A PROMPT LIBRARY

In each prompt, the action space comes from the benchmark official set, and one or two simple written templates can be applied to further constrain the output of the corresponding agent.

A.1 ACTIONER

```
ACTION_SYSTEM_PROMPT = """
{task_description}
"""
ACTION_USER_PROMPT="""
You can refer to the following milestone-based action guide proposed for this task to take action:
{milestones}

You can refer to the following insights for this task to take action:
{insights}

Here are examples:
{examples}

Your task and trajectories are as follows:
{task_trajectories}

You can follow the hint to take the next action:
{hint}

Now, take the next action for your task (no unnecessary explanations):
>
"""
```

A.2 PLANNER

```
MILESTONE_SYSTEM_PROMPT = """You are a professional planner specializing in breaking down complex tasks into clear, milestone-driven action guides based on expert examples.

Your instructions:
- Carefully study the provided example(s) and reproduce their style exactly in your answer.
- Do not add, remove, or rephrase steps unless strictly necessary to fit the new task.
- Organize your solution as a sequence of major milestones, each representing a key stage in accomplishing the task, just as in the example(s).
- Ignore the detailed objects, locations and other environment information in the example(s), and only care the logic.
- Each milestone should be concise and actionable, and the milestone list must align with the task action space limitations.
- Action space:
{space}
- Output Format: Milestone [number]: [content]
- Output Example:
{template}
"""
MILESTONE_USER_PROMPT = """
Example:
{retrieved_examples}

Task:
{task}

Following the provided style and format, outline a milestone-based action guide for the given task (no unnecessary explanations).
Milestone action guide:
"""
```

A.3 INSIGHTER

```
INSIGHT_SYSTEM_PROMPT = """You are an agent specializing in providing insights to help another finish the simulated environment tasks efficiently.

Your instructions:
- Carefully study the provided example(s), and generate or reuse proper insights for the current task in your answer.
- Your insights should be notices, advice for the given milestone list, and only output crucial global insights for the task.
- Each insight should be made while referring to the task action space.
- Action Space:
{space}
- Output Format: Insight [number]: [content]
- Output example:
Insight 1: xxx
Insight 2: xxx
"""
INSIGHT_USER_PROMPT = """
Example:
{retrieved_examples}

Task:
{task}

Milestone List:
{milestones}

Following the provided style and format, outline an action insights list for the given task (no unnecessary explanations).
Insights list:
"""
```

A.4 HINTER

```
HINT_SYSTEM_PROMPT = """You are an assistant guiding an agent that performs tasks in a simulated environment. Your task is to generate a step-wise hint that helps the agent complete the current milestone.

Your hint must be based on the following:
- Current Task: The overall task objective.
- Current Trajectory: The sequence of actions and observations up to the current step.
- Milestone-Based Guide: A list of milestones required to complete the task.
- Insights Guide: A list of insights as references.
- The action recommendation in your hint must be made while referring to the task action space.
- Action Space:
{space}

Analyze the agent's current state, the milestone-based guide, the insights guide, and generate the step-wise hint including only the following fields:
- Current State: Carefully trace all agent's relationships to the environment and relevant objects from the executed task trajectory, including the current obtained things, place and so on. Must be precise and never ignore any necessary agent's state information. If unrelated elements which may cause conflicts with the task space or task goal appear in the agent's state, give the reminding.
- Current Milestone: Specify which milestone in the plan the agent should currently be addressing.
- Milestone Gap: State what remains to be done to complete the current milestone.
- Action Correction (optional): Add this field only if the latest action is wrong or deviates from the milestone, or if the "Current State" contains unrelated elements which may cause conflicts with the task space or task goal. Point out the error and provide valid recovery actions from the action space to restore correct state; never propose failed actions in the same state. Omit this field when the agent's action and state are correct.

Format your output as follows (omit Action Correction if not needed):
Current State: [complete description of agent's relevant context]
Current Milestone: [the current goal that needs to be achieved. Milestone X - description]
Milestone Gap: [what still needs to be done, grounded only in observed information and milestones]
Action Correction: [only if an explicit correction is needed; only use action recommendation from the action space; otherwise omit this field]

Here are several examples:

{templates}
"""
HINT_USER_PROMPT = """
Your Input:
Current Task:
{current_task}

Current Trajectory:
{current_trajectory}

Milestone-Based Guide:
{milestone_list}

Insights Guide:
{insights_list}

Now, please generate the hint for the next action (no unnecessary explanations).
Output:
"""
```

A.5 SUCCESS OPTIMIZER

```
EXAMINE_SYSTEM_PROMPT = """You are a professional planner specializing in analyzing action-observation pairs and refine planned milestones list for a robot task.

Your instructions:
- Carefully study the provided finished trajectory and original milestones list for the task.
- Based on observation, you should delete or rewrite some unproper and redundant milestones of original list.
- Some meaningless actions should be ignored for you to refine milestones.
- Output Format: Milestone [number]: [content]
- Make the least milestones to achieve the best efficiency.
- Each milestone should be concise and actionable, and the milestone list must align with the task action space limitations.
- Action Space:
{space}

Notice: Each milestone should be as general as possible, not including any precise description.
"""
EXAMINE_USER_PROMPT = """
Task:
{task}

Trajectory:
{traj}

Original milestones list:
{miles}

Study the given task, trajectory and original milestones list, generate a new milestone-based action guide or just output the original list for the given task (no unnecessary explanations).
Milestone action guide:
"""
```

A.6 FAILED OPTIMIZER

```
REFINE_SYSTEM_PROMPT = """You are an analytical agent. You will be given a task, its failed trajectory, retrieved successfully executed trajectory and its original insights.
The reason for the failure is that the final task state does not match the required task state.

Requirements:
- You need to carefully analyze the final state of the failed trajectory and compare it with the required task state to identify the inconsistencies.
- Comparing the failed trajectory with the successful trajectory, check whether the original insights are enough for future agents to avoid making the same mistakes. If not, please modify them or add other insights.
- If the original insights cause the failure, please correct them.
- If there are not any problems with the original insights, just output them.
- The insights you summarize should not mention specific items but should instead extract general success principles applicable to similar tasks. These insights must be enlightening and provide guidance for future problems.
- Care about the task action space influencing the action agent.
- Action Space:
{space}
 Output Format:
- Start your response directly with the numbered list, no preamble or explanations.
- Each insight should be a short sentence.
- Use the following format exactly:
Insight 1:xxx
Insight 2:xxx
...
"""
REFINE_USER_PROMPT = """
### Failed Trajectory
{failed_task}
{failed_traj}

### Successful Trajectory
{success_task}
{success_traj}

## Original Insights
{ins}

Your output:
"""
```

B FORMULA REASONING

First, we assume that in a benchmark, the average reward(profit) one successful task can bring is α, and the average punishment(loss) is β, which is larger than 0 in contrast to α. To avoid that any of them appears in the formula individually and to counterbalance them, we can write an original division form to express the benefit of the success over the failure :

$$Pt' = \frac{\alpha SR}{\beta FR}, \quad (0 \leq SR < 1,\ \beta > 0), \tag{7}$$

$$FR = 1 - SR. \tag{8}$$

Since $\frac{\alpha}{\beta}$ is a constant for a fixed benchmark, we can simplify the expression to obtain:

$$Pt = \frac{SR}{FR}. \tag{9}$$

However, this direct division suffers from an edge-explosion problem as SR approaches 1. To constrain unbounded growth and normalize the upper bound, we introduce the function:

$$Ft = \frac{2}{\pi}\arctan(Pt), \quad Ct = \frac{At}{Ft}. \tag{10}$$

Treating SR as the independent variable x, we compute the first- and second-order derivatives of Ft with respect to x:

$$\frac{\mathrm{d}Ft}{\mathrm{d}x} = \frac{2}{\pi\big(x^2 + (1-x)^2\big)}, \tag{11}$$

$$\frac{\mathrm{d}^2Ft}{\mathrm{d}x^2} = \frac{4(1-2x)}{\pi\big(x^2 + (1-x)^2\big)^2}. \tag{12}$$

The second-order derivative reveals that Ft attains its maximum at $SR = 0.5$. Specifically, the first-order derivative increases when $SR < 0.5$ and decreases when $SR > 0.5$. This property encourages growth speed for low values of SR below 0.5 and mitigates the rate of increase speed for larger SR.

C EXPERIMENT SETUP

Datasets We evaluate M-Plan on three widely test multi-turn benchmarks for long-horizon planning and decision-making:

1. **ALFWorld** (18) is a simulated household environment base on text interaction. This benchmark comprises six kinds of subsets: put, put two, examine under the light, clean & put, heat & put, and cool & put, checking the capacity of an agent for understanding its own state and changes of environment via interactions. The maximum interaction step this benchmark can proceed is 50, also taken in our experiments. And we conduct tests on the officially established split of 134 out-of-distribution tasks.

2. **Webshop** (19) is a benchmark simulating an online shopping website, filled with over 1.18 million products. It will require the agent to search for target commodities and click some buttons to buy the ideal product, while reading different web pages. We conduct experiments on a set of 200 instructions. We set the max steps as 40.
3. **PDDL** (20) is a benchmark simulating four kinds of robot tasks, including making cocktails, placing blocks, moving balls and dealing with tyres. The agent is required to complete the specified goal, which typically comprises multiple subgoals within a 30-step budget.

Baselines: we compare M-Plan with the following LLM-based agent baselines, ranging from classic short-term memory reasoning to long-term memory retrieval paradigms, including:

1. **G-Memory** (11) is a representative retrieval-augmented memory approach to reuse detailed trajectories, plan-like key steps, and history summarized insights, and places many agents to refine or integrate insights for both successful and failed task trajectories.
2. **ReasoningBank** (14) is a typical extraction and reusing long-term memory method. The retrieved memory item is a series of collections of insights, comprising titles, descriptions, and content of each title, summarized by an agent, used to guide similar tasks.
3. **Hiplan** (17) is a short-term memory method equipped with a prepared memory library, containing a few trajectories to be retrieved. It also uses milestones as global guidance and step-wise hints. During the execution, similar milestones and trajectory segments will be searched to generate a hint in each step.

Reproduction Details. Since no official trajectory data are provided for Hiplan (17), we carefully construct six expert trajectories for ALFWorld, each corresponding to one of its six task types, as well as 25 task trajectories for Webshop. Following the paper, all trajectories are annotated with milestones and segments. Rather than providing an excessive number of cases, we aim to use a compact set of high-quality trajectories to improve performance. All preparation activities are completed using Qwen3-32B. For long-term memory methods and M-Plan, to ensure a fair alignment between our method and the compared approaches, we do not provide any prepared trajectory shots. The models used in the experiments are all deployed on two Nvidia A100 GPUs.

D HINTER DETAIL

Precisely, we establish two distinct principles tailored for different scenarios:

1. Loose Triggering: In ALFWorld and Webshop, agents often perform redundant revisiting and rechecking actions, which may trap them in persistent loops. Accordingly, for such occasions, once repeated actions appear in the action history, hints will be generated to provide guidance, as such repetitive behaviors tend to induce further troubles in these environments. Hints are also triggered if the most recent action is invalid.
2. Strict Triggering: In PDDL environments, hint activation follows a stricter rule: hints are triggered only when the current action duplicates the immediately preceding action or is invalid. This design permits permissible repeated actions, preserves task-solving effectiveness, and improves efficiency.

Additionally, since the environment observation can directly indicate whether the current action is valid or invalid, we use this feedback for action validation without introducing an additional agent, thereby further reducing system complexity and computational overhead. However, another issue arises with the $Hinter$ module. Although it is designed to help the $Actioner$ escape potential traps, the $Hinter$ itself may produce repetitive unhelpful or misleading guidance. For this reason, we introduce a mechanism to suspend the $Hinter$ when the next action would trigger the same hint repeatedly. After getting paused, the $Hinter$ awaits further information from the $Actioner$ to think again.

E STD RESULTS

Table 5: Comparison of average steps on ALFWorld and Webshop. We report `AS.`, `ASS.`, and `Meta-Step`, while `AS.` is not considered for Webshop due to its specific characteristics, as discussed in Sec. 4.1.

Models	Methods	ALFWorld			Webshop	
		`AS.`	`ASS.`	`Meta-Step`	`ASS.`	`Meta-Step`
Qwen3-32B	ReasoningBank	24.99±0.00	13.96±0.00	20.12±0.00	6.14±0.02	22.90±0.92
	G-Memory	24.93±0.48	20.53±0.51	24.14±0.60	**5.10±0.24**	10.90±0.69
	Hiplan	18.38±0.36	16.46±0.08	17.46±0.25	8.09±0.00	20.48±0.00
	M-Plan	**12.48±0.00**	**12.20±0.00**	**12.28±0.00**	5.35±0.00	**9.47±0.00**
Qwen3-14B	ReasoningBank	26.30±0.00	13.91±0.00	21.17±0.00	5.90±0.13	20.93±0.02
	G-Memory	18.66±0.30	12.94±0.16	15.30±0.27	**5.13±0.08**	8.96±0.31
	Hiplan	20.40±0.31	16.28±0.32	18.55±0.01	8.05±0.00	25.16±0.00
	M-Plan	**14.63±0.04**	**11.05±0.19**	**12.17±0.16**	5.48±0.06	**8.03±0.06**

Table 6: Comparison of average steps on PDDL. We report `AS.`, `ASS.`, and `Meta-Step`.

Models	Methods	PDDL		
		`AS.`	`ASS.`	`Meta-Step`
Qwen3-32B	ReasoningBank	26.61±0.31	15.76±1.84	65.90±0.15
	G-Memory	26.95±0.69	18.16±1.90	71.73±0.43
	Hiplan	26.33±0.14	**13.48±0.10**	60.70±0.07
	M-Plan	**22.30±0.13**	15.42±0.03	**29.22±0.01**
Qwen3-14B	ReasoningBank	28.16±0.16	19.33±0.30	112.42±0.25
	G-Memory 1	26.97±0.07	17.54±1.15	71.86±0.05
	Hiplan	27.56±0.28	**13.71±0.61**	92.18±0.38
	M-Plan	**23.33±0.33**	15.54±0.11	**33.75±0.05**

Table 7: Token-efficiency comparison. We report At, Ft and Ct (see in Sec. 4.1), mainly focusing on the comparison of Ct. Webshop is not discussed for the fairness in Sec. 4.1.

Models	Methods	ALFWorld			PDDL		
		At(M)	Ft	Ct(M)	At(M)	Ft	Ct(M)
Qwen3-32B	ReasoningBank	0.04±0.00	0.74±0.00	0.05±0.00	**0.07±0.00**	0.19±0.01	0.34±0.01
	G-Memory	0.05±0.00	0.93±0.01	0.06±0.00	0.12±0.00	0.21±0.02	0.59±0.07
	Hiplan	0.09±0.00	0.96±0.01	0.10±0.00	0.21±0.00	0.18±0.01	1.17±0.06
	M-Plan	**0.03±0.00**	**1.00±0.00**	**0.03±0.00**	0.17±0.00	**0.54±0.01**	**0.32±0.01**
Qwen3-14B	ReasoningBank	0.04±0.00	0.69±0.00	0.06±0.00	**0.07±0.00**	0.13±0.01	0.54±0.03
	G-Memory	0.05±0.00	0.89±0.01	0.06±0.00	0.12±0.01	0.20±0.02	0.63±0.04
	Hiplan	0.11±0.00	0.91±0.01	0.12±0.00	0.21±0.00	0.11±0.01	1.89±0.26
	M-Plan	**0.04±0.00**	**0.94±0.00**	**0.05±0.00**	0.16±0.01	**0.45±0.02**	**0.36±0.00**

Table 8: Ablation study of M-Plan on PDDL with Qwen3-32B. *Retrieval* indicates whether the short-term memory module receives cases from long-term memory.

Methods	Modules				Performance			
	Optimizer	*Retrieval*	*Insighter*	*Hinter*	SR.	*AD*	AS.	Meta-Step.
Standard	✓	✓	✓	✓	**0.528±0.009**	**0.743±0.013**	**22.30±0.13**	**29.22±0.01**
w/o *Optimizer*	✗	✓	✓	✓	0.478±0.024	0.682±0.020	22.35±0.04	29.58±0.02
w/o *Retrieval*	✓	✗	✓	✓	0.425±0.008	0.614±0.009	23.63±0.07	35.12±0.03
w/o *Insighter*	✓	✓	✗	✓	0.483±0.029	0.668±0.015	22.81±0.59	31.44±0.08
w/o *Hinter*	✓	✓	✓	✗	0.411±0.019	0.616±0.018	24.01±0.42	37.64±0.08