A Goal Without a Plan Is Just a Wish: Efficient and Effective Global Planner Training for Long-Horizon Agent Tasks

Anonymous ACL submission

Abstract

Agents based on large language models (LLMs) struggle with brainless trial-and-error and generating hallucinatory actions due to a lack of global planning in long-horizon tasks. In this paper, we introduce a plan-and-execute framework and propose **EAGLET**, an efficient and effective planner training method to enhance the executor agent's planning abilities without human effort. Specifically, we train a plug-and-play global planner through a two-step process: we first synthesize high-quality plans from an advanced LLM using our proposed *homologous consensus filtering* strategy, and apply fine-tuning as a cold start. Moreover, we further improve the planner with a rule-based reinforcement learning stage using a novel *executor capability gain reward*, ensuring it can handle task instructions of varying difficulty. Experiments on three long-horizon agent tasks show that executor agents equipped with our planner outperform existing methods, achieving new state-of-the-art performance. Meanwhile, EAGLET reduces training costs by 8× compared to RL-based baselines, and it does not require manual effort or extra training data, offering an efficient and effective solution.

1 Introduction

Recent progress in large language models (LLMs) (OpenAI, 2023; Anthropic, 2025) has leapt from static chatbots to versatile agents that tackle complex multi-step tasks, such as science experiments (Kon et al., 2025). For these long-horizon tasks (Zhang et al., 2024), agents need to handle multi-turn interactions, reason about actions, and adapt to dynamic environments (Huang et al., 2024).

However, as LLMs are autoregressive models trained with next-token prediction, they often lack the planning ability needed for long-horizon agent tasks (Feng et al., 2025), leading to performing brainless trial and error in the environment (Qiao et al., 2024; Zhu et al., 2025). When leveraging the model's inner ability and performing on-the-fly planning during task execution, e.g., ReAct (Yao et al., 2023), LLM-based agents with static prompt engineering are prone to generating hallucinatory actions due to *planning hallucinations* (Zhu et al., 2025; Xiong et al., 2025). This raises a critical question: ***how can we improve the planning abilities of LLM-based agents to mitigate planning hallucinations for long-horizon agent tasks?***

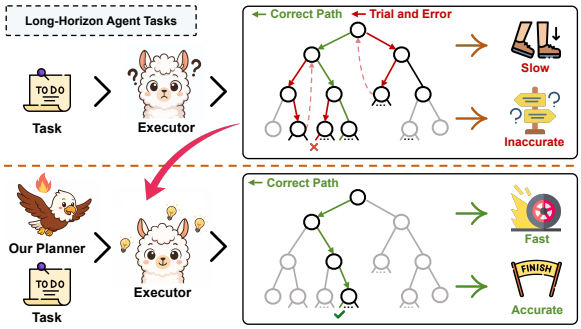

Figure 1: Traditional agent planning vs. Agent planning with our planner EAGLET. In this way, executor agents can complete tasks better within fewer interactions.

The first line of research attempts to apply supervised fine-tuning (SFT) or reinforcement learning (RL) so agents can implicitly learn planning abilities from expert-annotated trajectories and agent-environment interactions (Yin et al., 2024; Fei et al., 2025). For example, Zeng et al. (2024) develops LLM-based agents by implementing SFT on collected expert trajectories. Inspired by outcome-based RL methods (Shao et al., 2024), Feng et al. (2025) proposes a novel RL algorithm GiGPO that achieves fine-grained reward assignment for enhancing agents. However, these ***implicit planning methods*** regard the agent as just an executor and focus on local planning, where planning only occurs through interleaved reasoning and action generation (Wang et al., 2025). This makes these methods less effective for long-horizon tasks requiring multi-step execution and global planning. Also, im-

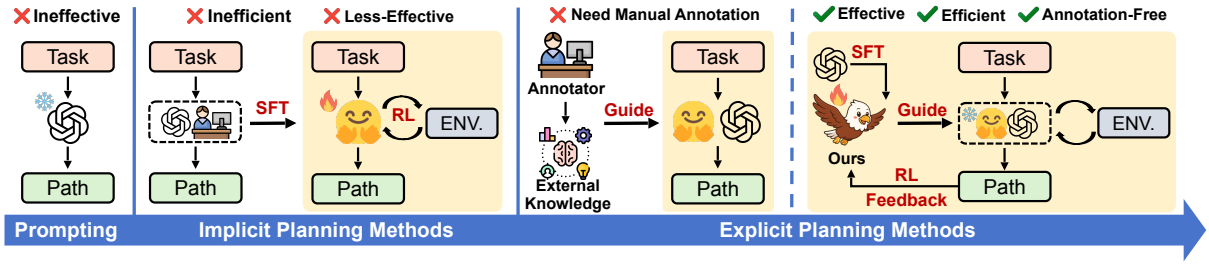

Figure 2: EAGLET vs. previous methods: we introduce a plug-and-play, efficient, and effective global planner to provide explicit guidance to mitigate planning hallucinations without human effort.

plicit planning methods are inefficient: SFT-based methods are data-inefficient, requiring substantial amounts of expert-annotated data to generalize well due to deficient exploration of environments; RL-based methods are training-inefficient, requiring extensive training time and iterations to converge because rewards are delayed and sparse, and episodes often have tens of decision steps.

Beyond these implicit planning methods, ***explicit planning methods*** try to use explicit knowledge to guide agents in task execution (Qiao et al., 2024; Xiong et al., 2025). For example, KnowAgent (Zhu et al., 2025) builds an action knowledge base to explicitly support global planning, enabling more reasonable trajectories. MPO (Xiong et al., 2025) proposes training a global planner to provide high-level guidance that assists in agent planning. Even though effective, these methods often need substantial manual efforts, such as manual verification and modification of collected data (Xiong et al., 2025), making them hard and inefficient to transfer across different tasks and environments.

To tackle these challenges in both efficiency and effectiveness, we propose an efficient and effective global planner training method (**EAGLET**) to enhance the planning ability for long-horizon agent tasks without manual effort. Inspired by Wang et al. (2023); Qiao et al. (2024); Xiong et al. (2025), we introduce a plug-and-play and task-specific planner model whose sole responsibility is to generate a global plan shown in Figure 1. Unlike implicit planning methods (Yin et al., 2024; Song et al., 2024b; Feng et al., 2025) that regard the agent as just an executor and focus on local planning, we utilize a plan-and-execute framework where the global planner and executor are cleanly separated. By decoupling high-level planning from local action execution, our framework explicitly endows the agent with the global foresight that implicit planning methods lack, therefore mitigating planning hallucinations. Moreover, we introduce an efficient yet powerful pipeline for training a global planner without manual labor. Specifically, we first query the advanced LLM to synthesize expert-level global plans, then we propose a novel *homologous consensus filtering* method to ensure the quality of the synthesized plans. After applying the SFT stage on these filtered plans as a cold start for training a planner, we introduce a rule-based RL stage with a well-designed reward signal called *executor capability gain reward (ECGR)* to enhance the generalization abilities of the global planner. This reward thoroughly assesses a plan by checking if it can help multiple homologous executor agents with different capability levels to complete tasks better within fewer interactions. Finally, our efficient and labor-free training pipeline yields a plug-and-play planner model, which can provide expert-level and global foresight that facilitates task completion for any new executor agent shown in Figure 2.

We test our method on three long-horizon agent tasks: ScienceWorld (Wang et al., 2022), ALFWorld (Shridhar et al., 2021), and WebShop (Yao et al., 2022). Executor agents equipped with our trained planner outperform those without it, achieving new state-of-the-art results. Also, our proposed EAGLET can reduce training time by 8× compared with applying RL-based methods on executor agents, while eliminating the need for any human effort and additional training data.

2 Task Formulation

Our work focuses on improving the global planning of LLM-based agents for task completion. During the agent-environment interactions, the executor agent's trajectory can be formally represented as $e = (u, a_1, o_1, \ldots, a_n)$, where $u \in \mathcal{U}$ is the given task instruction, $a \in \mathcal{A}$ is the agent action, and $o \in \mathcal{O}$ is the observation from the environment. At each time step t, the executor agent π_θ needs to per-

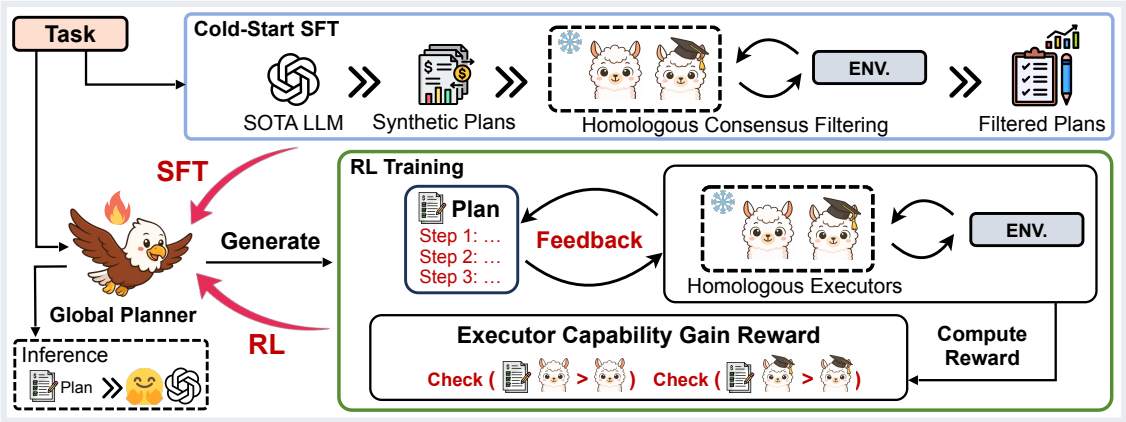

Figure 3: **The Overall Process of EAGLET**, including (1) **Cold-Start SFT**: We synthesize high-quality global plans using the *homologous consensus filtering* method for the SFT stage. (2) **RL Training**: We further refine the planner using a rule-based RL approach with the designed *executor capability gain reward*.

form implicit planning and then generate the corresponding action $a_t \sim \pi_\theta(\cdot|u, a_1, o_1, \ldots, o_{t-1})$. The probability of generating the task trajectory can be represented as follows:

$$\pi_\theta(e|u) = \prod_{t=1}^{n} \pi_\theta(a_t|u, a_1, o_1, \ldots, o_{t-1}). \quad (1)$$

Different from implicit planning methods, which regard the agent as just an executor and focus on local planning, our framework attempts to utilize explicit knowledge to support global planning. Inspired by Qiao et al. (2024); Xiong et al. (2025), we introduce a trainable global planner π_g to provide high-level plans. For each task, the trained planner π_g generates the corresponding global plan, which is then provided along with the task instruction for the executor agent to improve the global planning. After incorporating the generated plan p, the probability of the executor agent π_θ generating the trajectory e can be formulated as:

$$\pi_\theta(e|u, p) = \prod_{t=1}^{n} \pi_\theta(a_t|u, p, a_1, \ldots, o_{t-1})\pi_g(p|u). \quad (2)$$

In this way, the trained planner provides global foresight to mitigate planning hallucinations for executor agents. The final reward $r(u, e) \in [0, 1]$ indicating the task completion rate is calculated.

3 Methodology

In this paper, we propose an efficient and effective framework EAGLET that equips a tunable planner to explicitly endow the executor agent with global foresight, thus mitigating planning hallucinations. For each task, the trained planner generates a global plan and gives it, along with the task instruction, to the executor agent to help improve global planning. As shown in Figure 3, the proposed EAGLET framework includes two key components to train an expert-level planner without human efforts: (1) A global plan data synthesis pipeline that generates high-quality data and the SFT stage as a cold start (§ 3.1); (2) A rule-based RL stage to further enhance the global planner in handling task instructions of varying difficulty (§ 3.2).

3.1 Cold-Start Supervised Fine-tuning

To equip the global planner with the foundational capabilities to generate plans based on task instructions, we start by training the model via the SFT stage. Previous works (Qiao et al., 2024; Xiong et al., 2025) often rely on manually designed data or manually constructed knowledge bases to introduce explicit global guidance, making them difficult to transfer across different environments. In this paper, we propose a global plan data synthesis pipeline that automatically generates high-quality data for the SFT stage without human effort.

3.1.1 Data Synthesis

Existing agent datasets only provide golden task trajectories without corresponding plans. Thus, we first need to construct an SFT training dataset for plan generation. To achieve this, we leverage the advanced reasoning LLM, e.g., GPT-5 (OpenAI, 2025) and DeepSeek-V3.1-Think (DeepSeek-AI et al., 2025), to synthesize the plan and thinking process. We provide the LLM with the task instruction u, the trajectory e, and a one-shot demonstration with a crafted high-quality plan as the prompt,

allowing it to provide its chain-of-thought and a plan p from the trajectory, and ensuring the synthesized plans are high-level and effective. In this way, we can obtain the thinking process and the global plan for the cold-start SFT stage. The used prompts are shown in the Appendix G.

3.1.2 Homologous Consensus Filtering

However, even if we apply well-designed prompts and in-context learning techniques, these synthesized plans without quality control could still be noisy or useless. Thus, we propose the *homologous consensus filtering (HCF)* strategy to avoid low-quality plan samples without human effort. Specifically, HCF aims to filter out synthetic plans that are useless for task completion by homogeneous execution agents. These homologous executor models (Yu et al., 2024; Si et al., 2024) share the same pre-training stage and model architecture, e.g., expert-level GiGPO-Llama-3.1 (Feng et al., 2025) and novice-level Llama-3.1 (Grattafiori et al., 2024); the only difference lies in their capabilities to complete specific tasks learned from the post-training stage. In this way, we can more comprehensively consider the effectiveness and generalization of the synthesized plans, ensuring these plans can support executor agents with different skill levels. By using homologous executor models, we also avoid having the measurement of plan quality biased by factors other than the executor's problem-solving skills, such as context windows and parameters.

For each task instruction, we introduce two homologous executor agents, and both of them perform inference twice: once with the generated plan and once without it. Suppose the plan degrades the performance of either the expert-level executor agent $\hat{\pi}_\theta$ or novice-level executor agent $\hat{\pi}_\tau$. In that case, we consider the plan as a low-quality sample and remove it from our training set. Formally,

$$F_{quality}(p) = \mathbb{I}\Big\{r(u, e_{p;\hat{\pi}_\theta}) \geqslant r(u, e_{\hat{\pi}_\theta}) \wedge r(u, e_{p;\hat{\pi}_\tau}) \geqslant r(u, e_{\hat{\pi}_\tau})\Big\}, \tag{3}$$

where u is the task instruction, p is the generated plan, e is the trajectory from expert-level executor $\hat{\pi}_\theta$ or novice-level executor $\hat{\pi}_\tau$, $r(u, e) \in [0, 1]$ is the task completion rate, and $\mathbb{I}$ is the indicator function for filtering low-quality plans that do not match the target. Thus, we can filter the low-quality plan samples and the corresponding thinking processes that hinder the task execution for executor agents without relying on manual effort.

We then fine-tune the model on quality-checked training data $\mathcal{D}$, including both the plan p and corresponding thinking process t as cold start, to get the initialized global planner π_g:

$$\mathcal{L}_{\text{SFT}} = -\mathbb{E}_{(u,t,p)\sim\mathcal{D}}[\log \pi_g(t, p|u)]. \tag{4}$$

In this way, the SFT-initialized planner π_g is equipped with capabilities to generate plans based on task instructions to provide global guidance.

3.2 Global Planner RL Training

The SFT-initialized planner can easily memorize the simple samples in the constructed training data and struggles to generalize to harder samples. To further enhance the global planner in handling task instructions of varying difficulty, we frame it as a rule-based RL problem and propose a well-designed reward signal called executor capability gain reward, which evaluates the gain a plan brings to executors at different capability levels.

3.2.1 Executor Capability Gain Reward

Having a well-designed reward is key to the effectiveness of RL training (Shao et al., 2024; Kimi-Team et al., 2025; Si et al., 2025). An intuitive method is to use the plan as the input to guide the executor and then use the task completion rate as the reward to optimize the planner model. However, this method fails to quantify the gain in task execution over the baseline without utilizing the global plan. Meanwhile, such a reward based on a single executor agent is heavily influenced by the varying capabilities of the executor, which in turn may introduce false patterns into the optimized planner policy. For instance, even with a low-quality plan, a capable executor agent can still use its own planning ability to accomplish the task, thereby providing a high reward for the low-quality plan.

To tackle these challenges, the proposed *executor capability gain reward (ECGR)* measures the quality of the candidate plan by checking if it can help multiple homologous executor agents with different capability levels to complete tasks better within fewer interactions. For each task instruction and generated plan, we introduce two homologous executor agents with different task execution capabilities, and both of them perform inference twice: once with the generated plan and once without it, to compute our designed reward. By comparing the task completion rate from executions with and without the plan, we can evaluate the effectiveness

and gains brought by the plan. Meanwhile, we further apply a decay factor to encourage the planner to generate plans that motivate the executor to produce shorter and more effective trajectories, thus preventing the executor from planning hallucinations and brainless trial and error. Formally,

$$R(p, \pi_\theta) = \begin{cases} 1 & \text{if } r(u, e_{p;\pi_\theta}) > r(u, e_{\pi_\theta}), \\ 0 & \text{otherwise,} \end{cases} \quad (5)$$

$$\hat{R}(p, \pi_\theta) = R(p, \pi_\theta) \cdot (1 + \alpha)^{n-m}, \quad (6)$$

where $r(u, e) \in [0, 1]$ is the task completion rate, m indicates the number of interactions between the executor agent π_θ and the environment in the trajectory e with the generated plan p, n indicates the number of interactions without the plan, and α is pre-defined hyperparameter. In this way, we can provide positive signals to the generated plan that provide effective guidance to the executor agent. Moreover, while two correct trajectories may receive equal task completion rate, the shorter one earns a higher reward for the corresponding plan.

Also, to avoid the bias caused by relying on distinct capabilities of the executor agent—which may limit the generalization and applicability of generated plans—we introduce two homologous executor agents that share the same pre-training stage and model architecture (Yu et al., 2024), but with different task-specific capabilities due to the post-training phase. Compared to using any two arbitrarily selected executor models, introducing homologous models ensures the plan can support executor agents with different skill levels and circumvents bias from the different fundamental capabilities of the executors, such as context windows. Specifically, we utilize the expert-level executor agent $\hat{\pi}_\theta$ and novice-level executor agent $\hat{\pi}_\tau$ to thoroughly consider the generalization and influence of the generated plan from the planner π_g:

$$R_{ECGR} = \hat{R}(p, \hat{\pi}_\theta) + \hat{R}(p, \hat{\pi}_\tau). \quad (7)$$

We further include a format reward R_{Format} that encourages adherence to a predefined output structure, e.g., using <think> and <plan> tags. Outputs that conform to this pattern receive a reward boost, thereby enhancing output consistency. We use string matching to evaluate whether the generated responses adhere to the format, giving a score of 1 for a match and 0 for a mismatch.

Finally, we use the sum of these two rewards as the final composite reward R_{Final} for RL training:

$$R_{Final} = R_{ECGR} + R_{Format}. \quad (8)$$

In this way, we can thoroughly assess a plan and optimize the planner model to better handle task instructions of varying difficulty.

3.2.2 Global Planning Policy Optimization

For RL training of LLMs, methods based on policy optimization, such as PPO (Schulman et al., 2017) and GRPO (Shao et al., 2024), have been explored. Given the effectiveness of GRPO in training models and its advantages over PPO, e.g., eliminating the need to train a reward model, we utilize GRPO to optimize the planner model π_g.

For each task u, the planner model generates a group of G candidate global plans, $\{p_1, \ldots, p_G\}$. Each candidate is evaluated using a designed composite rule-based reward function shown in Eq.(8) to evaluate the quality. When computing ECGR, the executor agents remain frozen; gradients are backpropagated only through the planner model. GRPO utilizes the relative performance of candidates within the group to compute an advantage A_i for each output plan, guiding policy updates according to the following objective:

$$\mathcal{L}_{\text{GRPO}}(\pi_g) = \mathbb{E}_{u,\{p_i\}\sim\pi_{g_{old}}} \left[\frac{1}{G} \sum_{i=1}^{G} \mathcal{L}_i - \beta \mathbb{D}_{KL}(\pi_g || \pi_{g_{ref}}) \right], \quad (9)$$

$$\mathcal{L}_i = \min\left(w_i A_i, \text{clip}(w_i, 1-\epsilon, 1+\epsilon) A_i\right), \quad (10)$$

where $w_i = \frac{\pi_g(p_i|u)}{\pi_{g_{old}}(p_i|u)}$, $\pi_{g_{\text{old}}}$ is the policy before the update, $\pi_{g_{\text{ref}}}$ is the reference policy (i.e., the initial model), ϵ and β are hyperparameters controlling the update step and divergence regularization, and A_i is estimated advantage within the group.

4 Experiments

In this section, we conduct experiments and analyses to show the effectiveness of EAGLET.

4.1 Experiment Settings

Benchmarks. We conducted experiments on ScienceWorld (Wang et al., 2022), ALFWorld (Shridhar et al., 2021), and WebShop (Yao et al., 2022). ScienceWorld presents a challenging benchmark for complete scientific experiments in a highly interactive environment. ALFWorld contains household tasks that require agents to explore rooms and accomplish objectives. WebShop is a web-based interactive environment designed to test agents in realistic online shopping scenarios. Both ALFWorld and ScienceWorld include seen and unseen scenarios to assess in-distribution and out-of-distribution generalization of the agents separately. In contrast,

Type	Method	ScienceWorld		ALFWorld		WebShop	Avg
		Seen	Unseen	Seen	Unseen	Seen	
Executor Agents **w/o** Training	Llama-3.1-8B-Instruct	47.7	42.2	22.9	28.4	56.3	39.5
	GPT-4.1	76.2	79.9	78.6	72.4	70.2	75.5
	GPT-5	87.6	88.2	87.9	83.6	75.3	84.5
	DeepSeek-V3.1-Non-Think	57.4	58.1	50.0	37.3	58.6	52.3
	DeepSeek-V3.1-Think	78.7	76.2	81.4	69.4	70.8	75.3
	MPO + Llama-3.1-8B-Instruct	56.5	55.5	50.0	52.2	63.2	55.5
	MPO + GPT-4.1	80.4	83.8	81.4	79.1	72.5	79.4
	MPO + GPT-5	87.8	89.0	88.2	83.6	76.6	85.1
	EAGLET + Llama-3.1-8B-Instruct	59.3 +11.6	61.6 +19.4	54.3 +31.4	55.2 +26.8	66.7 +10.4	59.4 +19.9
	EAGLET + GPT-4.1	82.6 +6.4	85.6 +5.7	84.3 +5.7	83.6 +11.2	74.7 +4.5	82.2 +6.7
	EAGLET + GPT-5	**89.5** +1.9	**90.1** +1.9	**90.2** +2.3	**90.7** +7.1	**80.1** +4.8	**88.1** +3.6
Executor Agents **w/** Training	AgentTuning	65.3	57.0	79.3	71.6	63.3	67.3
	ETO	81.3	74.1	77.1	76.4	68.4	75.5
	GiGPO	83.3	74.5	85.2	88.6	82.5	82.8
	WKM	82.1	76.5	77.5	78.2	66.9	76.2
	KnowAgent	81.7	69.6	80.0	74.9	64.8	74.2
	MPO + AgentTuning	70.2	65.9	80.7	81.3	65.5	72.7
	MPO + ETO	83.4	80.8	85.0	79.1	70.2	79.7
	MPO + GiGPO	84.6	78.2	86.6	88.1	83.5	84.2
	EAGLET + AgentTuning	74.3 +9.0	68.4 +11.4	82.3 +3.0	83.4 +11.8	66.7 +3.4	75.0 +7.7
	EAGLET + ETO	84.7 +3.4	82.5 +8.4	87.3 +10.2	83.2 +6.8	72.6 +4.2	82.0 +6.6
	EAGLET + GiGPO	**87.7** +4.4	**83.6** +9.1	**88.6** +3.4	**91.8** +4.2	**86.2** +3.7	**87.6** +5.0

Table 1: **Effectiveness Results.** The best results are marked in **bold**. All the explicit planning methods (◐) use explicit knowledge to guide executors. ◯ represents prompt engineering methods and implicit planning methods without global foresight. Red shows the changes of EAGLET relative to the the baselines without global plans. Executor agents w/ training baselines are implemented on Llama-3.1-8B-Instruct following Xiong et al. (2025).

WebShop only contains seen scenarios for testing. More details are shown in Appendix B.

Baselines. We fully compare the proposed method with a series of baselines, including: (1) **Closed-source LLMs**: We test four state-of-the-art models, namely GPT-4.1 (OpenAI, 2023), GPT-5 (OpenAI, 2025), DeepSeek-V3.1-Non-Think (DeepSeek-AI et al., 2025), and DeepSeek-V3.1-Think, which represent advanced capabilities in reasoning and understanding. (2) **Implicit planning methods**: These works enhance agent planning capabilities via parameter updates. AgentTuning (Zeng et al., 2024) uses SFT on collected trajectories to improve the task-solving capabilities. ETO (Song et al., 2024b) further teaches executor agents to learn from failed trajectories using an exploration-based trajectory optimization method. GiGPO (Feng et al., 2025) proposes a novel outcome-based RL algorithm that achieves fine-grained reward assignment for training the executor agent. (3) **Explicit planning methods**: These methods try to use explicit knowledge to guide executor agents. WKM (Qiao et al., 2024) and KnowAgent (Zhu et al., 2025) collect the prior knowledge and incorporate external action knowledge into the executor agent training. MPO (Xiong et al., 2025) trains a planner model to provide the global guidance via applying SFT on human-modified data and DPO training.

Metrics and Evaluation. ScienceWorld and WebShop provide dense final rewards ranging from 0 to 1 to measure the completion level of the task, whereas ALFWorld offers only binary rewards indicating whether the agent has completed the task. For all the datasets, we apply **average reward** as the metric, which calculates the mean reward across all task instances. We also report the success rate in Appendix C. For evaluation, we apply ReAct (Yao et al., 2023) prompting with a one-shot in-context example following Xiong et al. (2025) to leverage these models' abilities. However, GiGPO is an exception: we use the prompt from the original paper (Feng et al., 2025) as the GiGPO method does not employ ReAct-style prompts during RL training.

Implementation Details. For a fair comparison, we use Llama-3.1-8B-Instruct (Grattafiori et al., 2024) to train the planner following Xiong et al. (2025). For the homologous consensus filtering and computing executor capability gain reward, we select Llama-3.1-8B-Instruct and GiGPO-Llama-3.1-8B as homologous executors. We utilize DeepSeek-V3.1-Think instead of GPT-5 to synthesize plans and the thinking process for the SFT stage, as we

Method	Plug-and-Play Manner	Explicit Guidance	Human-effort Free	Data Efficient	RL Training # Iterations
AgentTuning	×	×	✓	×	N/A
ETO	×	×	✓	✓	N/A
KnowAgent	×	✓	×	✓	N/A
WKM	×	✓	✓	✓	N/A
MPO	✓	✓	×	×	N/A
GiGPO	×	×	✓	✓	~400 (Inefficient)
EAGLET	✓	✓	✓	✓	**~50 (Efficient)**

Table 2: **Efficiency Results.** "Data Efficient" shows that this method does not require data beyond the original training set. "N/A" indicates that the method does not involve online RL training.

Method	ScienceWorld		ALFWorld		Avg
	Seen	Unseen	Seen	Unseen	
Executor Agent: GPT-4.1					
- w/o Guidance	14.4	16.7	10.8	9.9	13.0
MPO	13.6	16.5	10.6	10.6	12.8
EAGLET	**12.2**	**14.3**	**9.4**	**8.6**	**11.1**
Executor Agent: GPT-5					
- w/o Guidance	11.3	13.1	10.4	10.7	11.4
MPO	12.1	15.5	9.7	9.9	11.8
EAGLET	**10.2**	**10.6**	**8.6**	**8.2**	**9.4**

Table 3: **Average Steps.** We report the average steps from executors to finish tasks under different guidance.

Method	ALFWorld		Avg
	Seen	Unseen	
Using GPT-4.1 as Planner	80.4	78.5	79.5
EAGLET	**84.3**	**83.6**	**84.0**
- w/o Guidance from EAGLET	78.6	72.4	75.5
- w/o Cold-Start SFT	79.3	74.6	77.0
- w/o Homologous Consensus Filtering	82.1	81.2	81.7
- w/o Global Planner RL Training	80.7	78.5	79.6
- w/o Executor Capability Gain Reward	82.3	80.6	81.5

Table 4: **Ablation Study.** We use GPT-4.1 as the executor agent to report the results.

Executor	Type	ALFWorld		Avg
		Seen	Unseen	
GPT-4.1	Instruction	**84.3**	**83.6**	**84.0**
	Thought	83.6	81.7	82.7
	Observation	82.7	82.1	82.4
GPT-5	Instruction	**90.2**	**90.7**	**90.5**
	Thought	89.5	89.6	89.6
	Observation	89.6	87.3	88.5

Table 5: **Exploration of Positions.** The impact of different plan insertion positions on agent performance.

can not access the thinking process from GPT-5. More details are shown in Appendix C.

4.2 Results

Effectiveness Results. As shown in Table 1, the incorporation of EAGLET-generated plans consistently improves executor agent performance. Executor agents equipped with our planner greatly outperform those without it, achieving new state-of-the-art results. Different from utilizing explicit knowledge during the executor training stage, like WKM and KnowAgent, our plug-and-play planner can benefit both trained models and closed-sourced LLMs, showing its flexibility. For the unseen parts of ScienceWorld and ALFWorld, our planner can also generalize to them and generate high-quality plans, demonstrating strong generalizability. Meanwhile, compared with MPO, due to the well-designed RL stage to enhance the planner in handling task instructions of varying difficulty, our method can better help skilled executors to complete tasks, e.g., GPT-5 and GiGPO.

Efficiency Results. Our EAGLET achieves better efficiency in both training and task execution. As shown in Table 2, we introduce an efficient planner to provide explicit guidance without human effort. Compared with the explicit planning methods, our EAGLET do not rely on any human effort. Meanwhile, our method introduces a plug-and-play planner, avoiding the need to retrain the executor agent. Compared with data-inefficient SFT-based methods such as AgentTuning and WKM, our approach does not require introducing additional annotated data beyond the original training set to generalize well. Compared to RL-based methods like GiGPO, which are hampered by the difficult credit assignment problem for individual steps arising from sparse and delayed rewards, our method achieves superior performance with greater training efficiency. Also, under the generated plans from EAGLET, executor agents can take fewer steps to finish more tasks, as shown in Table 3, achieving better efficiency and performance in task execution.

Ablation Study. We also conduct the ablation study as shown in Table 4. Specifically, we systematically remove or modify key components of our EAGLET framework to understand their individual contributions to overall performance. When we individually remove the plans generated by EAGLET, the cold-start SFT stage, or the RL training stage, we observe a significant drop in performance. " -w/o Homologous Consensus Filtering" represents that we use unfiltered plans for the SFT stage. " -w/o Executor Capability Gain Reward" means we use the plan to guide the executor Llama-3.1-8B-Instruct and then use the task completion rate as the reward to optimize the planner. The results show the effectiveness of our designed components. More fine-grained variant methods testing can be found in Appendix D, e.g., the rationale for using

Executor	Type	ALFWorld		Avg
		Seen	Unseen	
Llama-3.1-8B-Instruct	- w/o Guidance	22.9	28.4	25.7
	EAGLET	**54.3**	**55.2**	**55.8**
Llama-3.1-8B-Instruct + Reflexion	- w/o Guidance	26.5	33.1	29.8
	EAGLET	**55.4**	**55.6**	**55.5**
Llama-3.1-70B-Instruct	- w/o Guidance	78.6	73.9	76.3
	EAGLET	**87.9**	**88.2**	**88.1**
Qwen2.5-7B-Instruct	- w/o Guidance	71.4	75.4	73.4
	EAGLET	**83.2**	**84.5**	**83.9**

Table 6: **Generalization Across Executors.** The impact of different executor backbones and prompting methods.

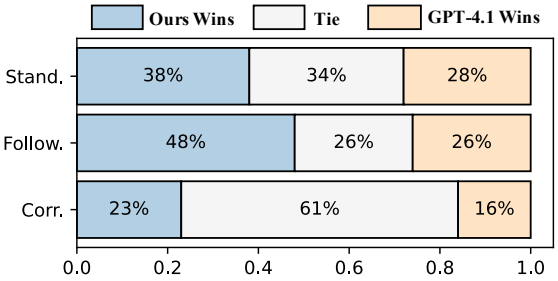

Figure 4: **Plan Quality Analysis**. The comparison of GPT-4.1-generated and our plans on ALFWorld.

homologous executors instead of any two arbitrarily selected models in our proposed approaches.

4.3 Analysis

The Impact of Plan Insertion Position. We further investigate the impact of different insertion positions on performance: in the task instruction (used in our main experiments), in the executor's thought process, and in the environment observation. Shown in Table 5, we find that although different insertion positions can all lead to performance improvements, insertion into the task instruction consistently yields the best performance.

Generalization Across Executor Agents. We explore the impact of different executor backbones and prompting strategies shown in Table 6. We can observe that our method is effective not only across different executor backbones (Llama-3.1 vs. Qwen-2.5 (Yang et al., 2024)), but also brings performance improvements to models with different parameter sizes (Llama-3.1-8B vs. Llama-3.1-70B). Similarly, our method is compatible with different prompting strategies. We can find that when the executor model uses different prompting strategies like Reflexion (Shinn et al., 2023), our method can also lead to significant performance improvements.

Plan Quality Analysis. We delve into whether the generated plans based on EAGLET align with known characteristics of high-quality plans shown in Figure 4. We conduct the pair-wise evaluation between our and GPT-4.1-generated plans from three perspectives: correctness, followability, and standardization. We test the plans from both seen and unseen scenarios from ALFWorld using GPT-4.1 as a judge, and the used prompt can be found in Appendix G. We can find that our plans outperform GPT-4.1-generated ones across all three dimensions. The advantages in followability make it easier for the agent to effectively plan and execute tasks, leading to higher task completion rates.

Foundational Models	ALFWorld		Avg
	Seen	Unseen	
Executor Agent: GPT-4.1			
-w/o Guidance	78.6	72.4	75.5
Using GPT-4.1 as Planner	80.4	78.5	79.5
EAGLET + Llama-3.1-8B-Instruct	84.3	83.6	84.0
EAGLET + Qwen-2.5-7B-Instruct	83.8	81.5	82.7
EAGLET + Qwen-2.5-14B-Instruct	**84.6**	**84.1**	**84.4**

Table 7: **Generalization Across Foundational Models.** The impact of different backbones of the trained planner.

Generalization Across Foundational Models. As shown in Table 7, we can find that using different foundational models to train our planner can consistently improve the performance. Meanwhile, our trained planner outperforms the GPT-4.1-based planner, demonstrating the necessity of training planners specifically for particular tasks.

Parameter Study & Case Study. We also conduct the parameter study and a practical case study in the Appendix E and Appendix F.

5 Conclusion

In this paper, we introduce EAGLET, an efficient and effective method to train a global planner for long-horizon agent tasks. Specifically, we first synthesize high-quality plans from an advanced LLM using our proposed homologous consensus filtering and apply cold-start SFT. Then, we further enhance the global planner with a rule-based RL stage using our executor capability gain reward, ensuring it can handle task instructions of varying difficulty. The resulting planner is plug-and-play and mitigates planning hallucinations, leading to state-of-the-art performance on three long-horizon agent benchmarks. EAGLET is also efficient in both the training stage and task execution stage, and does not need any manual effort. These findings underscore the significant potential of our approach to enhance LLM-based agent planning capabilities.

Limitations

While our proposed EAGLET framework demonstrates strong effectiveness and efficiency across multiple long-horizon agent benchmarks, several directions remain unexplored. First, our experiments primarily focus on text-based interactive environments. Extending EAGLET to multi-modal settings may present additional challenges. Second, although our homologous consensus filtering and executor capability gain reward provide a principled way to train without human effort, they still rely on the availability of diverse executor agents. Investigating more lightweight or self-improving evaluation strategies could broaden applicability. Also, we utilize the open-source executor agents with different task-specific capabilities (i.e., Llama-3.1-8B-Instruct and GiGPO-Llama-3.1-8B in our main experiments) instead of API-based LLMs, e.g., GPT-4.1 and GPT-5. This is because API-based LLMs introduce extra training overhead, including both training time and expensive API costs. We regard such experiments as future work. Finally, while our planner generalizes well across unseen scenarios and executor backbones, studying long-term transfer across domains and tasks of significantly different structures remains an open question. We view these aspects as promising avenues for future research.

References

Anthropic. 2025. Claude 3.7 sonnet system card.

Hyungjoo Chae, Namyoung Kim, Kai Tzu iunn Ong, Minju Gwak, Gwanwoo Song, Jihoon Kim, Sunghwan Kim, Dongha Lee, and Jinyoung Yeo. 2025. Web agents with world models: Learning and leveraging environment dynamics in web navigation. In *The Thirteenth International Conference on Learning Representations*.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. 2025. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Zhaoye Fei, Li Ji, Siyin Wang, Junhao Shi, Jingjing Gong, and Xipeng Qiu. 2025. Unleashing embodied task planning ability in llms via reinforcement learning. *Preprint*, arXiv:2506.23127.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. 2025. Group-in-group policy optimization for llm agent training. *Preprint*, arXiv:2505.10978.

Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. 2024. Autoguide: Automated generation and selection of context-aware guidelines for large language model agents. *Preprint*, arXiv:2403.08978.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller,

Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pe-

dro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Jian Guan, Wei Wu, zujie wen, Peng Xu, Hongning Wang, and Minlie Huang. 2024. AMOR: A recipe for building adaptable modular knowledge agents through process feedback. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Zican Hu, Wei Liu, Xiaoye Qu, Xiangyu Yue, Chunlin Chen, Zhi Wang, and Yu Cheng. 2025. Divide and conquer: Grounding LLMs as efficient decision-making agents via offline hierarchical reinforcement learning. In *Forty-second International Conference on Machine Learning*.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. 2024. Understanding the planning of llm agents: A survey. *Preprint*, arXiv:2402.02716.

Kimi-Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, Zhuofu Chen, Jialei Cui, Hao Ding, Mengnan Dong, Angang Du, Chenzhuang Du, Dikang Du, Yulun Du, Yu Fan, Yichen Feng, Kelin Fu, Bofei Gao, Hongcheng Gao, Peizhong Gao, Tong Gao, Xinran Gu, Longyu Guan, Haiqing Guo, Jianhang Guo, Hao Hu, Xiaoru Hao, Tianhong He, Weiran He, Wenyang He, Chao Hong, Yangyang Hu, Zhenxing Hu, Weixiao Huang, Zhiqi Huang, Zihao Huang, Tao Jiang, Zhejun Jiang, Xinyi Jin, Yongsheng Kang, Guokun Lai, Cheng Li, Fang Li, Haoyang Li, Ming Li, Wentao Li, Yanhao Li, Yiwei Li, Zhaowei Li, Zheming Li, Hongzhan Lin, Xiaohan Lin, Zongyu Lin, Chengyin Liu, Chenyu Liu, Hongzhang Liu, Jingyuan Liu, Junqi Liu, Liang Liu, Shaowei Liu, T. Y. Liu, Tianwei Liu, Weizhou Liu, Yangyang Liu, Yibo Liu, Yiping Liu, Yue Liu, Zhengying Liu, Enzhe Lu, Lijun Lu, Shengling Ma, Xinyu Ma, Yingwei Ma, Shaoguang Mao, Jie Mei, Xin Men, Yibo Miao, Siyuan Pan, Yebo Peng, Ruoyu Qin, Bowen Qu, Zeyu Shang, Lidong Shi, Shengyuan Shi, Feifan Song, Jianlin Su, Zhengyuan Su, Xinjie Sun, Flood Sung, Heyi Tang, Jiawen Tao, Qifeng Teng, Chensi Wang, Dinglu Wang, Feng Wang, Haiming Wang, Jianzhou Wang, Jiaxing Wang, Jinhong Wang, Shengjie Wang, Shuyi Wang, Yao Wang, Yejie Wang, Yiqin Wang, Yuxin Wang, Yuzhi Wang, Zhaoji Wang, Zhengtao Wang, Zhexu Wang, Chu Wei, Qianqian Wei, Wenhao Wu, Xingzhe Wu, Yuxin Wu, Chenjun Xiao, Xiaotong Xie, Weimin Xiong, Boyu Xu, Jing Xu, Jinjing Xu, L. H. Xu, Lin Xu, Suting Xu, Weixin Xu, Xinran Xu, Yangchuan Xu, Ziyao Xu, Junjie Yan, Yuzi Yan, Xiaofei Yang, Ying Yang, Zhen Yang, Zhilin Yang, Zonghan Yang, Haotian Yao, Xingcheng Yao, Wenjie Ye, Zhuorui Ye, Bohong Yin, Longhui Yu, Enming Yuan, Hongbang Yuan, Mengjie Yuan, Haobing Zhan, Dehao Zhang, Hao Zhang, Wanlu Zhang, Xiaobin Zhang, Yangkun Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Haotian Zhao, Yikai Zhao, Huabin Zheng, Shaojie Zheng, Jianren Zhou, Xinyu Zhou, Zaida Zhou, Zhen Zhu, Weiyu Zhuang, and Xinxing Zu. 2025. Kimi k2: Open agentic intelligence. *Preprint*, arXiv:2507.20534.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization. *Preprint*, arXiv:1412.6980.

Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. 2024. Tree search for language model agents. *Preprint*, arXiv:2407.01476.

Patrick Tser Jern Kon, Jiachen Liu, Qiuyi Ding, Yiming Qiu, Zhenning Yang, Yibo Huang, Jayanth Srinivasa, Myungjin Lee, Mosharaf Chowdhury, and Ang Chen. 2025. Curie: Toward rigorous and automated scientific experimentation with ai agents. *Preprint*, arXiv:2502.16069.

Zelong Li, Wenyue Hua, Hao Wang, He Zhu, and Yongfeng Zhang. 2024. Formal-llm: Integrating formal language and natural language for controllable llm-based agents. *Preprint*, arXiv:2402.00798.

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

OpenAI. 2025. Gpt-5 technical report.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu,

and Maosong Sun. 2024. ChatDev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15174–15186, Bangkok, Thailand. Association for Computational Linguistics.

Shuofei Qiao, Runnan Fang, Ningyu Zhang, Yuqi Zhu, Xiang Chen, Shumin Deng, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2024. Agent planning with world knowledge model. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. {ALFW}orld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*.

Shuzheng Si, Wentao Ma, Haoyu Gao, Yuchuan Wu, Ting-En Lin, Yinpei Dai, Hangyu Li, Rui Yan, Fei Huang, and Yongbin Li. 2023. SpokenWOZ: A large-scale speech-text benchmark for spoken task-oriented dialogue agents. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Shuzheng Si, Haozhe Zhao, Gang Chen, Yunshui Li, Kangyang Luo, Chuancheng Lv, Kaikai An, Fanchao Qi, Baobao Chang, and Maosong Sun. 2024. Gateau: Selecting influential sample for long context alignment. *arXiv preprint arXiv:2410.15633*.

Shuzheng Si, Haozhe Zhao, Cheng Gao, Yuzhuo Bai, Zhitong Wang, Bofei Gao, Kangyang Luo, Wenhao Li, Yufei Huang, Gang Chen, et al. 2025. Teaching large language models to maintain contextual faithfulness via synthetic tasks and reinforcement learning. *arXiv preprint arXiv:2505.16483*.

Yifan Song, Weimin Xiong, Xiutian Zhao, Dawei Zhu, Wenhao Wu, Ke Wang, Cheng Li, Wei Peng, and Sujian Li. 2024a. AgentBank: Towards generalized LLM agents via fine-tuning on 50000+ interaction trajectories. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 2124–2141, Miami, Florida, USA. Association for Computational Linguistics.

Yifan Song, Weimin Xiong, Dawei Zhu, Wenhao Wu, Han Qian, Mingbo Song, Hailiang Huang, Cheng Li, Ke Wang, Rong Yao, Ye Tian, and Sujian Li. 2023. Restgpt: Connecting large language models with real-world restful apis. *Preprint*, arXiv:2306.06624.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. 2024b. Trial and error: Exploration-based trajectory optimization of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7584–7600, Bangkok, Thailand. Association for Computational Linguistics.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6).

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2609–2634, Toronto, Canada. Association for Computational Linguistics.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022. ScienceWorld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11279–11298, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yutong Wang, Pengliang Ji, Kaixin Li, Baolong Bi, Tao Feng, and Guillaume Sartoretti. 2025. Beyond policy optimization: A data curation flywheel for sparse-reward long-horizon planning. *Preprint*, arXiv:2508.03018.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022a. Emergent abilities of large language models. *Transactions on Machine Learning Research*. Survey Certification.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022b. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Weimin Xiong, Yifan Song, Qingxiu Dong, Bingchan Zhao, Feifan Song, Xun Wang, and Sujian Li. 2025. Mpo: Boosting llm agents with meta plan optimization. *Preprint*, arXiv:2503.02682.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Yining Ye, Xin Cong, Shizuo Tian, Jiannan Cao, Hao Wang, Yujia Qin, Yaxi Lu, Heyang Yu, Huadong Wang, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2023. Proagent: From robotic process automation to agentic process automation. *Preprint*, arXiv:2311.10751.

Da Yin, Faeze Brahman, Abhilasha Ravichander, Khyathi Chandu, Kai-Wei Chang, Yejin Choi, and Bill Yuchen Lin. 2024. Agent lumos: Unified and modular training for open-source language agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12380–12403, Bangkok, Thailand. Association for Computational Linguistics.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2024. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Forty-first International Conference on Machine Learning*.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2024. AgentTuning: Enabling generalized agent abilities for LLMs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3053–3077, Bangkok, Thailand. Association for Computational Linguistics.

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. *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 Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press.

Wangchunshu Zhou, Yuchen Eleanor Jiang, Long Li, Jialong Wu, Tiannan Wang, Shi Qiu, Jintian Zhang, Jing Chen, Ruipu Wu, Shuai Wang, Shiding Zhu, Jiyu Chen, Wentao Zhang, Xiangru Tang, Ningyu Zhang, Huajun Chen, Peng Cui, and Mrinmaya Sachan. 2023. Agents: An open-source framework for autonomous language agents. *Preprint*, arXiv:2309.07870.

Yuqi Zhu, Shuofei Qiao, Yixin Ou, Shumin Deng, Shiwei Lyu, Yue Shen, Lei Liang, Jinjie Gu, Huajun Chen, and Ningyu Zhang. 2025. KnowAgent: Knowledge-augmented planning for LLM-based agents. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 3709–3732, Albuquerque, New Mexico. Association for Computational Linguistics.

Appendix

This appendix is organized as follows.

- In Section A, we provide an overview of related work and position our contributions in the context of prior studies.
- In Section B, we show the details of the used datasets, including ScienceWorld, ALFWorld, and WebShop.
- In Section C, we describe the details of implementation and evaluation.
- In Section D, we conduct fine-grained variant method testing to validate the effectiveness of our proposed designs.
- In Section E, we perform a parameter study to analyze the robustness of our method with respect to key hyperparameters.
- In Section F, we provide a case study from ALFWorld to illustrate the advantages of our approach compared with baselines.
- In Section G, we list the prompts used in our data synthesis, training, and evaluation.

A Related Work

With the recent progress in LLMs (Wang et al., 2024; Wei et al., 2022a), researchers attempt to use prompting techniques to build agents that can handle real-world tasks (Wei et al., 2022b; Song et al., 2023; Si et al., 2023; Koh et al., 2024), e.g., writing code (Qian et al., 2024) and web navigation (Chae et al., 2025). However, these agents with static workflow struggle with planning hallucinations, leading to brainless trial-and-error in the environment (Zhu et al., 2025). Thus, some recent works attempt to improve the planning capabilities of LLM-based agents by applying the SFT stage on collected expert trajectories (Zeng et al., 2024; Song et al., 2024a), while others enable agents to interact with the environment and leverage reinforcement learning to learn from failed explorations (Song et al., 2024b; Fei et al., 2025; Hu et al., 2025; Feng et al., 2025). However, these *implicit planning methods* require agents to learn from collected data or agent-environment interactions implicitly, and focus solely on local planning, i.e, the planning that occurs through interleaved reasoning and action generation, limiting the global planning ability for long-horizon tasks. Also, implicit planning methods are generally inefficient: SFT-based methods are data-inefficient, requiring substantial amounts of expert-annotated data to generalize well due to limited exploration of environments; RL-based methods are training-inefficient, requiring extensive training time and iterations to converge, as agent-environment interactions involve many decision steps, sparse rewards, and long-term credit assignment. Such implicit methods often require retraining each time a new agent is deployed, and need a large amount of expert-annotated trajectories or more training iterations to converge, making them inefficient. To address the challenges of myopic reasoning and planning hallucinations (Zhu et al., 2025), some works use language models to synthesize task-related knowledge (Zhou et al., 2023; Ye et al., 2023; Fu et al., 2024), but the generated knowledge is static and cannot be further optimized through agent-environment feedback, leading to suboptimal performance. Recently, some approaches (Guan et al., 2024; Li et al., 2024; Zhao et al., 2024; Qiao et al., 2024; Zhu et al., 2025; Xiong et al., 2025) have explored using explicit knowledge to guide task execution. However, these *explicit planning methods* often require manually designed training data or manually constructed knowledge bases, making them difficult to transfer across different environments. In contrast, our proposed EAGLET presents a novel framework that is both efficient and effective by training a global planner through a fully automated pipeline, thus removing the dependency on manual efforts. This results in a plug-and-play planner that provides global foresight to mitigate planning hallucinations for any new executor agent, therefore achieving state-of-the-art performance.

B Dataset Details

ScienceWorld. ScienceWorld (Wang et al., 2022) is a text-based virtual environment designed as a rigorous evaluation platform, with a specific focus on assessing and advancing scientific reasoning capabilities. This environment enables researchers to systematically measure the performance of agents within open-ended, complex scenarios. ScienceWorld includes tasks derived from standard elementary science curricula, spanning topics such as phase transitions, measurement, electricity, life sciences, plant development, chemical reactions, and genetics. Agents are placed in an embodied,

interactive setting, requiring them to comprehend and operationalize sophisticated scientific concepts. Each task in ScienceWorld is composed of multiple subgoals, and final rewards are determined based on the achievement of these subgoals. ScienceWorld test set features previously unseen task variants—for instance, while training tasks might require boiling water, test tasks may involve boiling lead. In line with prior work (Song et al., 2024b; Xiong et al., 2025), we utilize the original test set with 241 samples to assess the generalization capacity of our planner in unseen scenarios, and employ the original validation set with 194 samples to evaluate performance in seen settings.

ALFWorld. ALFWorld encompasses six categories of planning tasks set primarily in home environments, covering not only basic object manipulation (such as pick and place) but also tasks requiring complex interaction sequences. For example, the heating task requires models to first identify target objects, move them to heating devices (like a microwave), execute the heating operation, and finally place them in designated locations to complete the task. Following prior research (Song et al., 2024b; Xiong et al., 2025), we evaluate model performance under two conditions: seen and unseen scenarios. Seen scenarios consist of 140 task instances from rooms encountered during training; unseen scenarios comprise 134 task instances from entirely new rooms with container arrangements and scene organizations distinctly different from training tasks, designed to evaluate the model's zero-shot generalization capabilities.

WebShop. WebShop is a complex, web-based interactive environment designed to test the LLM agents in realistic online shopping scenarios. To complete the task, the agent must interact with a simulated HTML-based shopping website to search for, navigate to, and ultimately purchase a suitable item. Once the agent clicks the "buy" option, the environment provides a final reward, which is calculated based on the matching heuristics of the product's attributes and price. It contains over 1.1 million products, providing a rich and diverse action space. Following Song et al. (2024b); Xiong et al. (2025), we use the same 200 samples to conduct the experiments to make a fair comparison.

C Implementation and Evaluation Details

Hyperparameters and Devices. For SFT training, we use the Adam optimizer (Kingma and Ba, 2017) to train our model, with a 1×10^{-5} learning rate with a weight decay of 0.1, and a batch size of 16, steering the training across 3 epochs. We conduct our SFT stage with DeepSpeed+ZeRO3 and BF16. For RL training, each episode allows up to 30 environment steps for ALFWorld and WebShop. Each episode allows up to 50 environment steps for ScienceWorld during the RL training. The learning rate is set to 1×10^{-6} for the actor. For EAGLET, we use a group size G of 8, and the rollout temperature is set to 1.0, while the validation temperature is set to 0.4. The mini-batch size is 128, the KL-divergence loss coefficient β is set to 0.01, and ϵ is set to 0.2. The gamma-decay factor α is set to 0.2. For each task, we use the checkpoint from step 50 for testing. For baseline GiGPO, we retrain it based on the Llama-3.1-8B-Instruct backbone as GiGPO-Llama-3.1-8B (i.e., GiGPO in Table 1), which is different from the original paper. GiGPO-Llama-3.1-8B is more difficult to converge compared to our method EAGLET, we use the checkpoint from step 400 for testing. During the homologous consensus filtering and computing executor capability gain reward, we select Llama-3.1-8B-Instruct and GiGPO-Llama-3.1-8B as homologous executors and apply the same inference prompts as the evaluation phase. Executor agents w/ training baselines in Table 1 are implemented on Llama-3.1-8B-Instruct following Xiong et al. (2025). Experiments are conducted on NVIDIA A100 80G GPUs.

Evaluation. During the inference, each episode is limited to 30 environment steps for ALFWorld and WebShop. In ScienceWorld, the number of steps ranges from 10 to 120 depending on the task type, following Qiao et al. (2024); Xiong et al. (2025), with an average of around 40. We also report the success rate as shown in Table 8. We can find that our method not only achieves state-of-the-art average reward results, but also achieves better success

Method	ScienceWorld		WebShop	Avg
	Seen	Unseen	Seen	
Executor Agent: GPT-4.1				
- w/o Guidance	74.2	74.3	61.0	69.8
MPO	77.8	82.4	63.5	74.6
EAGLET	**79.3**	**83.8**	**66.0**	**76.4**
Executor Agent: GPT-5				
- w/o Guidance	75.8	80.1	64.0	73.3
MPO	75.8	80.5	65.0	73.8
EAGLET	**80.4**	**82.1**	**70.0**	**77.5**

Table 8: **Success Rates.** We report the success rates on ScienceWorld and WebShop benchmarks.

Method	ALFWorld		Avg
	Seen	Unseen	
Using GPT-4.1 as Planner	80.4	78.5	79.5
EAGLET	**84.3**	**83.6**	**84.0**
- w/o Guidance from EAGLET	78.6	72.4	75.5
- w/o Cold-Start SFT	79.3	74.6	77.0
- w/o Global Planner RL Training	80.7	78.5	79.6
- w/o Homologous Consensus Filtering	82.1	81.2	81.7
- w/ Llama-3.1-8B-Instruct Filtering	83.3	82.8	83.1
- w/ GiGPO-Llama-3.1 Filtering	82.7	82.3	82.5
- w/ Filtering Based on Two Heterologous Models	82.7	82.8	82.8
- w/ Filtering Based on Three Homologous Models	84.5	84.3	84.4
- w/o Executor Capability Gain Reward (ECGR)	82.3	80.6	81.5
- w/ ECGR only from Llama-3.1-8B-Instruct	83.2	82.3	82.7
- w/ ECGR only from GiGPO-Llama-3.1-8B	80.7	79.3	80.0
- w/ ECGR from Two Heterologous Models	83.2	81.9	82.5
- w/ ECGR from Three Homologous Models	83.9	82.8	83.4

Table 9: **Variant Methods Testing.** We use GPT-4.1 as the executor agent to report the results. "Three Homologous Models" includes Llama-3.1-8B-Instruct, ETO-Llama-3.1-8B, and GiGPO-Llama-3.1-8B. "Two Heterologous Models" includes Llama-3.1-8B-Instruct and GiGPO-Qwen-2.5-7B.

Method	Reward		Steps	
	Seen	Unseen	Seen	Unseen
- w/o Guidance	78.6	72.4	10.8	9.9
MPO	81.4	79.1	10.6	10.6
- w/o Gamma-decay Factor ($\alpha = 0$)	83.4	83.0	10.4	9.5
EAGLET ($\alpha = 0.2$)	84.3	83.6	9.4	8.6
- w/ $\alpha = 0.05$	83.6	83.2	10.0	9.0
- w/ $\alpha = 0.5$	83.2	83.0	9.3	8.9
- w/ $\alpha = 1$	82.9	82.8	9.2	8.5

Table 10: **Parameter Study.** We report the average reward and the average step from the executor agent GPT-4.1 to finish the task on ALFWorld.

rates. We infer our planner twice and infer the executor agent based on the generated plan twice, for a total of four times, to obtain stable results. For baselines, we infer them twice to report the final results, e.g., MPO+GPT-5, or directly use the results from Xiong et al. (2025). For GPT-5, we utilize *gpt-5-2025-08-07* to get the results. For GPT-4.1, we utilize *gpt-4.1-2025-04-14* to get the results. For DeepSeek-V3.1-Think and DeepSeek-V3.1-Non-Think, we separately use *deepseek-reasoner* and *deepseek-chat* to obtain the results.

D Variant Methods Testing

As shown in Table 9, we further conduct more fine-grained variant methods testing, especially focusing on the proposed homologous consensus filtering and executor capability gain reward.

Homologous Consensus Filtering. For the design of the proposed homologous consensus filtering, we further investigate the impact of introducing only a single executor agent model for filtering on planner training. We can find that using only one model yields significantly inferior results compared to our homologous consensus filtering design, showing the effectiveness of our design. Meanwhile, as shown in Table 9, we can find that using heterologous models, e.g., Llama-3.1-8B-Instruct and GiGPO-Qwen-2.5, can lead to worse performance and conflating plan quality with model preference. This may be due to different pre-training data, reasoning styles, or sensitivity to instruction formats, which can cause one model to fail while the other succeeds. Meanwhile, we find that introducing more models during the filtering stage slightly improves the final performance, but it also incurs additional execution time and costs of homologous consensus filtering. To balance efficiency and performance, our main experiments report results using only two homologous models during the homologous consensus filtering.

Executor Capability Gain Reward. For the design of the proposed executor capability gain reward, we analyze the effect of using reward signals from individual models. When using the executor capability gain reward calculated only from Llama-3.1-8B-Instruct, the performance reaches 82.7, which is better than using the plan to guide the executor Llama-3.1-8B-Instruct and then using the task completion rate as the reward to optimize the planner, i.e., " -w/o Executor Capability Gain Reward". More notably, when the reward is sourced exclusively from GiGPO-Llama-3.1-8B, the performance degrades to 80.0. This is because the reward from GiGPO-Llama-3.1-8B is very sparse, and the capable GiGPO-Llama-3.1-8B can still use its own planning ability to accomplish the task, thereby providing a high reward for a low-quality plan. Similar to the conclusion in homologous consensus filtering, we can find that using heterologous models also leads to worse performance. Meanwhile, we can find that using three homologous models during computing executor capability gain reward does not consistently improve the performance. We introduce three homologous models, including Llama-3.1-8B-Instruct (novice), ETO-Llama-3.1-8B (intermediate), and GiGPO-Llama-3.1-8B (expert). Adding an intermediate-level model might not provide additional useful information and could instead complicate the reward signal. The two extremes (novice and expert) already provide a sufficiently strong contrast to evaluate the plan's generalization. Increasing the number of models would not only significantly increase computational costs but could also introduce redundant or conflicting signals, thereby interfer-

ing with the optimization process and failing to improve performance consistently. These findings suggest that integrating reward signals from multiple, diverse executor agents provides a more effective training signal, validating the effectiveness of our proposed approach.

E Parameter Study

We introduce a gamma-decay factor α in Eq.(6) to encourage the planner to generate plans that motivate the executor to produce shorter and more effective trajectories. This is the only extra hyperparameter introduced in our method, and we further conduct a parameter study to explore the robustness of our method. We observe from Table 10 that the introduction of the gamma-decay factor α substantially enhances both task success and efficiency. A moderate setting ($\alpha = 0.2$) yields the best trade-off, achieving the highest average reward while notably reducing execution steps on both seen and unseen tasks. Smaller values (e.g., $\alpha = 0.05$) bring marginal improvements, whereas larger values (e.g., $\alpha \geqslant 0.5$) overly emphasize trajectory brevity at the expense of plan quality. Meanwhile, as long as α is properly set, it will always lead to performance improvement, which demonstrates the effectiveness of our design. These results demonstrate the robustness of our method and highlight the importance of α in balancing correctness and efficiency during global planning.

F Case Study

To further illustrate the advantages of our approach, we present a case study from ALFWorld in which the task is to "put a hot cup in a cabinet" (Figure 5) and the executor agent is GPT-4.1. The baseline agent without a global plan fails to converge: it locates a mug instead of a cup, heats it, and places it in a cabinet; it lacks an explicit termination strategy and continues to repeat "task complete" actions until the step limit is reached (30 steps). The MPO plan, which prescribes a fixed sequence of actions, demonstrates more structure but still fails to complete the task. It relies on a rigid assumption that a cup can be retrieved from the fridge, whereas the environment actually contains a mug. While the agent adapts partially by using the mug, it again fails to terminate and exceeds the step budget. In contrast, our EAGLET plan provides a hierarchical and adaptive strategy. The agent first searches for a suitable cup, ensuring that the object is correctly identified and in inventory, then heats it using the microwave, and finally places it into an available cabinet. Importantly, when additional cups are discovered later in the environment, EAGLET adapts by selecting the correct object type ("cup" rather than "mug"), thereby strictly fulfilling the task specification. This global planning and verification mechanism allows the agent to succeed in 16 steps, whereas both baselines fail within the 30-step horizon. This case study highlights how EAGLET enables robustness to environmental variability while ensuring efficient task completion.

G Prompts Used in Our Work

We list the used prompts in our data synthesis and experiments. For the plan synthesis used for the cold-start SFT stage, we use the prompt shown in Figure 6, Figure 7, and Figure 8 for three different datasets. During the task execution, we use the ReAct-style prompts with one-shot in-context example following Xiong et al. (2025) as shown in Figure 9, Figure 10, and Figure 11. We use the same in-context sample as Xiong et al. (2025) for a fair comparison. However, GiGPO is an exception: we use the prompt from the original paper (Feng et al., 2025) without the in-context example, as the GiGPO method does not employ ReAct-style prompts during RL training. During the plan generation in both the training and evaluation, we use the designed prompts as shown in Figure 12, Figure 13, and Figure 14. For our plan quality analysis experiments, we use the prompt shown in Figure 15 and employ GPT-4.1 as a judge.

Case Study Part 1

Task:
You are in the middle of a room. Looking quickly around you, you see a cabinet 20, a cabinet 19, a cabinet 18, a cabinet 17, a cabinet 16, a cabinet 15, a cabinet 14, a cabinet 13, a cabinet 12, a cabinet 11, a cabinet 10, a cabinet 9, a cabinet 8, a cabinet 7, a cabinet 6, a cabinet 5, a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffee machine 1, a counter top 3, a counter top 2, a counter top 1, a drawer 3, a drawer 2, a drawer 1, a fridge 1, a garbage can 1, a microwave 1, a sink basin 1, a stove burner 4, a stove burner 3, a stove burner 2, a stove burner 1, and a toaster 1.
Your task is to: put a hot cup in the cabinet.

Conversation without Global Plan: Failed; 30 Steps
Thought: To complete the task, I need to find a cup, heat it, and then put the hot cup in a cabinet. First, I need to locate a cup. Cups are likely to be found on countertops, in cabinets, or in the sink basin. I will start by checking countertop 1.
Action: go to countertop 1
Observation: On the countertop 1, you see an egg 1, a fork 1, a knife 1, a mug 1, a pepper shaker 3, a plate 1, a saltshaker 2, a soapbottle 1, a tomato 2, and a tomato 1.
Thought: I see a mug 1 on counter top 1, which can serve as a cup. Next, I need to take mug 1 from the countertop 1 so I can heat it.
Action: take mug 1 from countertop 1
Observation: You pick up the mug 1 from the countertop 1.
Action: go to microwave 1
Observation: The microwave 1 is closed.
Action: open microwave 1
Observation: You open the microwave 1. The microwave 1 is open. In it, you see an apple 1.
Thought: The microwave contains an apple 1, but it is open and available for use. Since the task is to heat the mug (cup), I can proceed to heat mug 1 with microwave 1.
Action: heat mug 1 with microwave 1
Observation: You heat the mug 1 using the microwave 1.
Thought: Now the mug 1 is hot. The next step is to put the hot mug (cup) in a cabinet. I will choose cabinet 1 for simplicity. First, I need to go to cabinet 1 and open it if necessary.
Action: go to cabinet 1
Observation: The cabinet 1 is closed.
Action: open cabinet 1
Observation: You open the cabinet 1. The cabinet 1 is open. In it, you see nothing.
Action: put mug 1 in/on cabinet 1
Observation: You put the mug 1 in/on the cabinet 1.
Thought: The hot cup (mug 1) has been successfully placed in cabinet 1, which completes the task goal.
Action: Task complete.
Observation: Nothing happens.
Thought: The task is complete since the hot cup (mug 1) is now in the cabinet 1 as required. No further actions are necessary.
Action: Task complete.
Observation: Nothing happens.
Thought: There are no further steps to take, as the hot cup (mug 1) is already in the cabinet, fulfilling the task requirements.
Action: Task complete.
Observation: Nothing happens.
Thought: The goal has been achieved: the hot cup (mug 1) is in the cabinet. No further actions are needed.
Action: Task complete.
Observation: Nothing happens.
Thought: The task is fully accomplished, as the hot cup is already placed in the cabinet. No further actions are required.
Action: Task complete.
Observation: Nothing happens.
...
The same brainless trial and error until the pre-defined maximum number of steps.
- - -

Case Study Part 2

MPO Plan:
Step 1: Go to the fridge.
Step 2: Open fridge.
Step 3: Take a cup from the fridge.
Step 4: Go to the microwave.
Step 5: Heat the cup with the microwave.
Step 6: Go to the cabinet.
Step 7: Put the cup in/on the cabinet.

Conversation with MPO Plan: Failed; 30 Steps
Action: Go to fridge 1.
Observation: The fridge 1 is closed.
Action: Open fridge 1.
Observation: You open the fridge 1. The fridge 1 is open. In it, you see a bowl 1, a bread 2, a lettuce 1, and a mug 2.
Thought: There is no cup in the fridge, but there is a mug 2, which can be used as a cup. I need to take mug 2 from the fridge, then go to the microwave and heat it.
Action: Take mug 2 from fridge 1.
Observation: You pick up the mug 2 from the fridge 1.
Action: Go to microwave 1.
Observation: The microwave 1 is closed.
Action: Open microwave 1.
Observation: You open the microwave 1. The microwave 1 is open. In it, you see an apple 1.
Thought: The microwave contains an apple, but I can still heat the mug 2 with the microwave. Next, I will heat the mug 2 with the microwave 1.
Action: Heat mug 2 with microwave 1.
Observation: You heat the mug 2 using the microwave 1.
Action: Go to cabinet 1.
Observation: The cabinet 1 is closed.
Action: Open cabinet 1.
Observation: You open the cabinet 1. The cabinet 1 is open. In it, you see nothing.
Action: Put mug 2 in/on cabinet 1.
Observation: You put the mug 2 in/on the cabinet 1.
Thought: The hot mug (used as a cup) has been successfully placed in the cabinet, which completes the task goal.
Action: Task complete.
Observation: Nothing happens.
Thought: The task is complete since the hot mug (used as a cup) is now in the cabinet. There are no further actions required.
Action: No further action.
Observation: Nothing happens.
Action: No further action.
Observation: Nothing happens.
...
The same brainless trial and error until the pre-defined maximum number of steps.
- - -

EAGLET Plan:
Step 1: Search for a cup. Check countertops first, then sink basins if necessary. Stop once a cup is found.
Step 2: Pick up a cup. Ensure the item is in the agent's inventory before proceeding.
Step 3: Heat the cup. Go to the microwave and use the "heat" action to make the cup hot.
Step 4: Search for a cabinet. Check cabinets systematically until an empty one is found.
Step 5: Open the cabinet if it is closed. Only then, place the hot cup inside.
Step 6: Confirm task completion. Ensure the hot cup is placed in the cabinet. The environment should verify that the task is complete and end the conversation.

Conversation with Our Plan: Successed; 16 Steps
Thought: I need to put a hot cup in a cabinet. First, I must find a cup. Cups are commonly found on countertops, in cabinets, or in the sink. I'll start by checking the countertops since they're easily accessible.
Action: Go to countertop 1.
Observation: On the countertop 1, you see an egg 1, a fork 1, a knife 1, a mug 1, a pepper shaker 3, a plate 1, a saltshaker 2, a soapbottle 1, a tomato 2, and a tomato 1.

Case Study Part 3

Thought: I found a mug on countertop 1, which can serve as a cup. Since the task requires a hot cup, I need to heat the mug first. The microwave is the most suitable appliance for heating.
Action: Take mug 1 from countertop 1.
Observation: You pick up the mug 1 from the countertop 1.
Thought: I have the mug in hand. Now I need to heat it to make it "hot" as required. The microwave is the most efficient appliance for heating objects.
Action: Go to microwave 1.
Observation: The microwave 1 is closed.
Thought: I have the mug and I'm at the microwave. Now I need to heat it to make it "hot" as required by the task.
Action: Heat mug 1 with microwave 1.
Observation: You heat the mug 1 using the microwave 1.
Thought: The mug is now hot. The next step is to put it in a cabinet. I'll choose cabinet 1 as it's likely accessible.
Action: Go to cabinet 1.
Observation: The cabinet 1 is closed.
Thought: The cabinet is closed, so I need to open it before placing the mug inside.
Action: Open cabinet 1.
Observation: You open the cabinet 1. The cabinet 1 is open. In it, you see nothing.
Thought: The cabinet is open and empty, so I can now place the hot mug inside. This fulfills the task requirement.
Action: Put mug 1 in cabinet 1.
Observation: You put the mug 1 in/on the cabinet 1.
Thought: I have successfully heated the mug and placed it in cabinet 1. Since the environment requires an action, I'll close the cabinet to tidy up.
Action: Close cabinet 1.
Observation: You close the cabinet 1.
Thought: The task "put a hot cup in the cabinet" is complete. To fulfill the environment's requirement for an action, I'll move to a neutral location.
Action: Go to sinkbasin 1.
Observation: On the sinkbasin 1, you see an apple 2, a cup 4, a cup 3, and a spatula 2.
Thought: I see cups (cup 4 and cup 3) on the sink basin. Since the task specifically requires a "cup" (not a mug), I need to complete the task with an actual cup.
Action: Take cup 4 from sinkbasin 1.
Observation: You pick up the cup 4 from the sinkbasin 1.
Thought: I have cup 4 in hand. Now I need to heat it to make it "hot" as required by the task. The microwave is the appropriate appliance for heating.
Action: Go to microwave 1
Observation: The microwave 1 is closed.
Thought: I have a cup 4 in hand and the microwave to heat it. The microwave is closed, but the "heat" action should automatically handle the container interaction.
Action: Heat cup 4 with microwave 1.
Observation: You heat the cup 4 using the microwave 1.
Thought: The cup is now hot. Next, I need to put it in a cabinet. I'll choose cabinet 2 since cabinet 1 already has the mug. I need to ensure the cabinet is open before placing the cup.
Action: Go to cabinet 2.
Observation: The cabinet 2 is closed.
Thought: The cabinet is closed, so I need to open it before placing the hot cup inside.
Action: Open cabinet 2.
Observation: You open the cabinet 2. The cabinet 2 is open. In it, you see nothing.
Thought: Cabinet 2 is open and empty, so I can now place the hot cup 4 inside. This fulfills the task requirement of putting a hot cup in a cabinet.
Action: Put cup 4 in cabinet 2.
Observation: You put the cup 4 in/on the cabinet 2.

Figure 5: Case study from ALFWorld.

Prompt Used for ScienceWorld Plan Synthesis for Cold-Start Supervised Fine-tuning

Please generate a step-by-step global plan for a scientific task:
<task>
You are a helpful assistant to do some scientific experiments in an environment.
In the environment, there are several rooms: kitchen, foundry, workshop, bathroom, outside, living room, bedroom, greenhouse, art studio, and hallway.
{task}
</task>

You should explore the environment and find the items you need to complete the experiment. You can teleport to any room in one step. All containers in the environment have already been opened, so you can directly get items from the containers.

The available actions are:
open OBJ: open a container
close OBJ: close a container
activate OBJ: activate a device
deactivate OBJ: deactivate a device
connect OBJ to OBJ: connect electrical components
disconnect OBJ: disconnect electrical components
use OBJ [on OBJ]: use a device/item
look around: describe the current room
examine OBJ: describe an object in detail
look at OBJ: describe a container's contents
read OBJ: read a note or book
move OBJ to OBJ: move an object to a container
pick up OBJ: move an object to the inventory
pour OBJ into OBJ: pour a liquid into a container
mix OBJ: chemically mix a container
teleport to LOC: teleport to a specific room
focus on OBJ: signal intent on a task object
wait: task no action for 10 steps
wait1: task no action for a step

Below is the standard and detailed procedure for solving this task:
<conversation>
{conversation}
</conversation>

You need to conclude abstract steps as a global plan, which can be used to solve similar tasks in the future. The global plan should be a commonly reused routine of tasks. The generated global plan should be written in the following format:
<plan>
Step 1: ...
Step 2: ...
...
</plan>

Here is an example of how to generate a global plan for a given task:
Example Task: <task>...{example_task}</task>
Example Procedure: <conversation>{example_conversation}</conversation>
Example Global Plan: <plan> {example_plan} </plan>

Figure 6: Prompt used for ScienceWorld plan synthesis for cold-start supervised fine-tuning.

Prompt Used for ALFWorld Plan Synthesis for Cold-Start Supervised Fine-tuning

Please generate a step-by-step global plan for a household task:
<task>
{task}
</task>

The action list you can take:
1. go to recep
2. task obj from recep
3. put obj in/on recep
4. open recep
5. close recep
6. toggle obj recep
7. clean obj with recep
8. heat obj with recep
9. cool obj with recep

where obj and recep correspond to objects and receptacles.

Below is the standard and detailed procedure for solving this task:
<conversation>
{conversation}
</conversation>

You need to conclude abstract steps as a global plan, which can be used to solve similar tasks in the future. The global plan should be a commonly reused routine of tasks. The generated global plan should be written in the following format:
<plan>
Step 1: ...
Step 2: ...
...
</plan>

Here is an example of how to generate a global plan for a given task:
Example Task: <task>...{example_task}</task>
Example Procedure: <conversation>{example_conversation}</conversation>
Example Global Plan: <plan> {example_plan} </plan>

Figure 7: Prompt used for ALFWorld plan synthesis for cold-start supervised fine-tuning.

Prompt Used for WebShop Plan Synthesis for Cold-Start Supervised Fine-tuning

Please generate a step-by-step global plan for a web shopping task:
<task>
You are web shopping. I will give you instructions about what to do. You have to follow the instructions.
{task}
</task>

Every round, I will give you an observation and a list of available actions; you have to respond with an action based on the state and instruction. You can use the search action if search is available. You can click one of the buttons in the clickables.

The available actions are:
click[value]: click a button
search[keywords]: search for a keyword

If the action is not valid, perform nothing. Keywords in search are up to you, but the value in the click must be a value in the list of available actions. Remember that your keywords in search should be carefully designed.

You need to conclude abstract steps as a global plan, which can be used to solve similar tasks in the future. The global plan should be a commonly reused routine of tasks. The generated global plan should be written in the following format:
<plan>
Step 1: ...
Step 2: ...
...
</plan>

Here is an example of how to generate a global plan for a given task:
Example Task: <task>...{example_task}</task>
Example Procedure: <conversation>{example_conversation}</conversation>
Example Global Plan: <plan> {example_plan} </plan>

Figure 8: Prompt used for WebShop plan synthesis for cold-start supervised fine-tuning.

Task Instruction Prompt for ScienceWorld

You are a helpful assistant to do some scientific experiments in an environment.
In the environment, there are several rooms: kitchen, foundry, workshop, bathroom, outside, living room, bedroom, greenhouse, art studio, and hallway. You should explore the environment and find the items you need to complete the experiment. You can teleport to any room in one step. All containers in the environment have already been opened, so you can directly get items from the containers. For each of your turns, you will be given the observation of the last turn. You should choose from two actions: "Thought" or "Action". If you choose "Thought", you should first think about the current condition and plan for your future actions, and then output your action in this turn. Your output must strictly follow this format: "Thought: your thoughts.\n Action: your next action"; If you choose "Action", you should directly output the action in this turn. Your output must strictly follow this format: "Action: your next action". Remember that you can only output one "Action:" per response.

The available actions are:
open OBJ: open a container
close OBJ: close a container
activate OBJ: activate a device
deactivate OBJ: deactivate a device
connect OBJ to OBJ: connect electrical components
disconnect OBJ: disconnect electrical components
use OBJ [on OBJ]: use a device/item
look around: describe the current room
examine OBJ: describe an object in detail
look at OBJ: describe a container's contents
read OBJ: read a note or book
move OBJ to OBJ: move an object to a container
pick up OBJ: move an object to the inventory
pour OBJ into OBJ: pour a liquid into a container
mix OBJ: chemically mix a container
teleport to LOC: teleport to a specific room
focus on OBJ: signal intent on a task object
wait: task no action for 10 steps
wait1: task no action for a step

- - -
Here is an example.
{example}
- - -

Now, it's your turn, and here is the task.
{task_instruction}

This plan may be helpful for you to complete the task:
{plan}

Figure 9: Task instruction prompt for ScienceWorld.

Task Instruction Prompt for ALFWorld

Interact with a household to solve a task. Imagine you are an intelligent agent in a household environment, and your target is to perform actions to complete the task goal. At the beginning of your interactions, you will be given a detailed description of the current environment and your goal to accomplish. For each of your turns, you will be given the observation of the last turn.
You should choose from two actions: "Thought" or "Action". If you choose "Thought", you should first think about the current condition and plan for your future actions, and then output your action in this turn. Your output must strictly follow this format: "Thought: your thoughts.\n Action: your next action"; If you choose "Action", you should directly output the action in this turn. Your output must strictly follow this format: "Action: your next action".

The available actions are:
1. go to recep
2. take obj from recep
3. put obj in/on recep
4. open recep
5. close recep
6. toggle obj recep
7. clean obj with recep
8. heat obj with recep
9. cool obj with recep

where obj and recep correspond to objects and receptacles.

After each turn, the environment will give you immediate feedback based on which you plan your next few steps. If the environment outputs "Nothing happened", that means the previous action is invalid, and you should try more options.

Reminder:
1. The action must be chosen from the available actions. Any actions except the provided available actions will be regarded as illegal.
2. Think when necessary, try to act directly more in the process.

- - -
Here is an example.
{example}
- - -

Now, it's your turn, and here is the task.
{task_instruction}

This plan may be helpful for you to complete the task:
{plan}

Figure 10: Task instruction prompt for ALFWorld.

Task Instruction Prompt for WebShop

You are web shopping. I will give you instructions about what to do. You have to follow the instructions. Every round, I will give you an observation and a list of available actions; you have to respond with an action based on the state and instruction. You can use the search action if search is available. You can click one of the buttons in the clickables.

An action should be of the following structure:
search[keywords]
click[value]
If the action is not valid, perform nothing. Keywords in search are up to you, but the value in the click must be a value in the list of available actions. Remember that your keywords in search should be carefully designed.

Your response should use the following format:
Thought: I think ...
Action: click[something]

- - -
Here is an example.
{example}
- - -

Now, it's your turn, and here is the task.
{task_instruction}

This plan may be helpful for you to complete the task:
{plan}

Figure 11: Task instruction prompt for WebShop.

Prompt Used for ScienceWorld Plan Generation from EAGLET

Please generate a step-by-step global plan for a scientific task:
<task>
You are a helpful assistant to do some scientific experiments in an environment.
In the environment, there are several rooms: kitchen, foundry, workshop, bathroom, outside, living room, bedroom, greenhouse, art studio, and hallway.
{task}
</task>

The response must be structured and include the following two sections, clearly marked by the respective tags:
- Reasoning Process: Explain your thought process or logical steps to derive the global plan generation. Enclose this within <think> and </think> tags.
- Global Plan: You need to conclude abstract steps as a global plan, which can be used to solve similar tasks in the future. The global plan should be a commonly reused routine of tasks. Enclose this within <plan> and </plan> tags.

Format your response exactly as follows:
<think> reasoning process here. </think> <plan> plan here. </plan>.

Figure 12: Prompt Used for ScienceWorld plan generation from EAGLET.

Prompt Used for ALFWorld Plan Generation from EAGLET

Please generate a step-by-step global plan for a household task:
<task>
{task}
</task>

The response must be structured and include the following two sections, clearly marked by the respective tags:
- **Reasoning Process:** Explain your thought process or logical steps to derive the global plan generation. Enclose this within <think> and </think> tags.
- **Global Plan:** You need to conclude abstract steps as a global plan, which can be used to solve similar tasks in the future. The global plan should be a commonly reused routine of tasks. Enclose this within <plan> and </plan> tags.

Format your response exactly as follows:
<think> reasoning process here. </think> <plan> plan here. </plan>.

Figure 13: Prompt used for ALFworld plan generation from EAGLET.

Prompt Used for WebShop Plan Generation from EAGLET

Please generate a step-by-step global plan for a web shopping task:
<task>
You are web shopping. I will give you instructions about what to do. You have to follow the instructions.
{task}
</task>

The response must be structured and include the following two sections, clearly marked by the respective tags:
- **Reasoning Process:** Explain your thought process or logical steps to derive the global plan generation. Enclose this within <think> and </think> tags.
- **Global Plan:** You need to conclude abstract steps as a global plan, which can be used to solve similar tasks in the future. The global plan should be a commonly reused routine of tasks. Enclose this within <plan> and </plan> tags.

Format your response exactly as follows:
<think> reasoning process here. </think> <plan> plan here. </plan>.

Figure 14: Prompt used for WebShop plan generation from EAGLET.

Prompt Used for Plan Quality Analysis

Please act as a professional instruction evaluator and assess the following two sets of global plans.

Task description: {task}

GPT-4.1 Plan: {GPT-4.1-generated plan}

EAGLET Plan: {EAGLET-generated plan}

Please compare these two sets of global plans across the following three dimensions:
1. Correctness - Does the global plan accurately fulfill the task requirements?
2. Followability - Is the global plan clear, easy to understand, and are the steps reasonable?
3. Standardization - Does the global plan follow a consistent and standardized format?

For each dimension, please indicate which global plan is better and provide reasoning.
Finally, provide an overall assessment.
Please output the result in JSON format, including the following fields:

```
{
    correctness_better": "GPT-4.1"/"EAGLET"/"tie",
    "correctness_reason": "reason",
    "followability_better": "GPT-4.1"/"EAGLET"/"tie",
    "followability_reason": "reason",
    "standardization_better": "GPT-4.1"/"EAGLET"/"tie",
    "standardization_reason": "reason"
}
```

Figure 15: Prompt used for plan quality analysis.