MPO: Boosting LLM Agents with Meta Plan Optimization

Weimin Xiong[♡], **Yifan Song**[♡], **Qingxiu Dong**[♡], **Bingchan Zhao**[♣]
Feifan Song[♡], **Xun Wang**[♠], **Sujian Li**[♡*]
[♡]National Key Laboratory for Multimedia Information Processing,
School of Computer Science, Peking University
[♣]University of Washington [♠]Peking University
{wmxiong, lisujian}@pku.edu.cn

https://github.com/WeiminXiong/MPO

Abstract

Recent advancements in large language models (LLMs) have enabled LLM-based agents to successfully tackle interactive planning tasks. However, despite their successes, existing approaches often suffer from planning hallucinations and require retraining for each new agent. To address these challenges, we propose the **Meta Plan Optimization (MPO)** framework, which enhances agent planning capabilities by directly incorporating explicit guidance. Unlike previous methods that rely on complex knowledge, which either require significant human effort or lack quality assurance, MPO leverages high-level general guidance through meta plans to assist agent planning and enables continuous optimization of the meta plans based on feedback from the agent's task execution. Our experiments conducted on two representative tasks demonstrate that MPO significantly outperforms existing baselines. Moreover, our analysis indicates that MPO provides a plug-and-play solution that enhances both task completion efficiency and generalization capabilities in previous unseen scenarios.

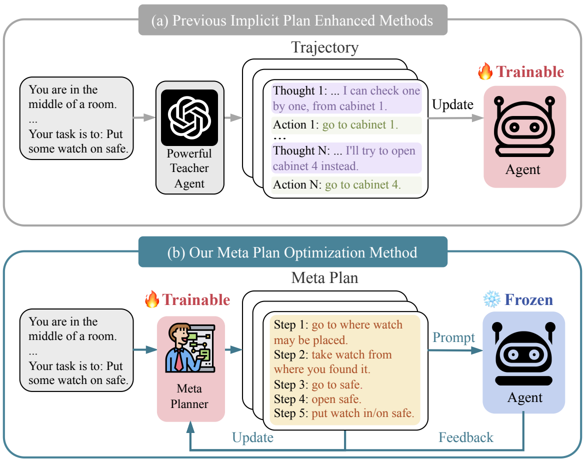
(a) Previous Implicit Plan Enhanced Methods
Trajectory
You are in the middle of a room. ... Your task is to: Put some watch on safe.
Powerful Teacher Agent
Thought 1: ... I can check one by one, from cabinet 1.
Action 1: go to cabinet 1.
...
Thought N: ... I'll try to open cabinet 4 instead.
Action N: go to cabinet 4.
Update
🔥Trainable
Agent
(b) Our Meta Plan Optimization Method
Meta Plan
🔥Trainable
You are in the middle of a room. ... Your task is to: Put some watch on safe.
Meta Planner
Step 1: go to where watch may be placed.
Step 2: take watch from where you found it.
Step 3: go to safe.
Step 4: open safe.
Step 5: put watch in/on safe.
Prompt
❄️ Frozen
Agent
Update
Feedback

Figure 1: Unlike previous implicit plan enhancing methods that require agent parameter updates, our method incorporates meta plans into prompts for direct planning guidance and can improve them based on feedback.

1 Introduction

Recent advancements in large language models (LLMs) (Achiam et al., 2023; Liu et al., 2024; Yang et al., 2024a) have enabled LLM-based agents to tackle complex multi-step tasks, including embodied housework (Shridhar et al., 2020) and science experiments (Wang et al., 2022). These tasks require sophisticated *planning* abilities, as agents need to understand long-term dependencies (Zhang et al., 2024), reason about sequential actions, and adapt to dynamic environments (Yao et al., 2022). The planning quality of these agents plays a crucial role in determining their overall performance.

Current mainstream LLM-based agents primarily develop their planning capabilities through implicit methods, either directly leveraging the model's inner ability or fine-tuning from expert trajectories. For example, ReAct (Yao et al., 2022) and Reflexion (Shinn et al., 2024) perform planning *on-the-fly* during task execution and are prone to getting lost due to planning hallucination (Zhu et al., 2024). The works including AgentTuning (Zeng et al., 2023), Lumos (Yin et al., 2023), and ETO (Song et al., 2024b) employ trajectory tuning to enhance implicit planning capabilities and require retraining for each new agent, resulting in huge computational cost (Figure 1(a)).

Beyond implicit planning, a few studies have begun exploring the use of explicit knowledge to guide agents in task execution (Zhu et al., 2024; Qiao et al., 2024). While these works bring in advantages such as explicit guidance, interpretability, and low integration costs for agents, they either require substantial manual design efforts or lack quality assurance in the process of complex knowledge acquisition, resulting in inconsistent improvements for agents (Wang et al., 2024). Building on

[*]Corresponding Authors.

these efforts, we introduce the concept of Meta Plan, which provides high-level, abstract guidance to assist in agent planning. As shown in Figure 1(b), for the task "put some watch on safe," the meta plan outlines an abstract strategy for general task completion. Unlike previous implicit plans obtained in the execution process, the meta plan is decoupled from specific environmental details (e.g. “cabinet 1”, “cabinet 4”) and complex agent trajectories, making it more amenable to optimization.

Furthermore, to automatically improve the quality of the meta plan, we propose an optimization framework, **Meta Plan Optimization (MPO)**, which consists of a meta planner and an agent. The meta planner is responsible for generating high-level meta plans, while the agent provides feedback on the execution to assess the quality of the inputted meta plans and help refine the meta planner. Initially, we collect meta plans from expert trajectories and perform a cold start on the meta planner through supervised fine-tuning. To further optimize the meta planner, we use Monte Carlo (MC) sampling to estimate the task completion rate of the agent as feedback. Specifically, given a task, the planner generates multiple meta plans through sampling. Then for each meta plan, the agent is also sampled to produce multiple execution trajectories, and the task completion rate is estimated accordingly. After identifying contrastive meta plan pairs—those leading to the highest and lowest task completion rates—we apply DPO (Rafailov et al., 2024) to refine the meta planner on these plan pairs. Finally, the trained meta planner can be detached from the MPO framework and function as a plug-and-play component, capable of generating high-quality meta plans for tasks in the target environment. This facilitates task completion for any new agent without incurring additional training costs.

We evaluate our approach on two representative benchmarks: ALFWorld (Shridhar et al., 2020) for embodied household task and ScienceWorld (Wang et al., 2022) for textual science experiment task. Across all test tasks, agents equipped with our meta planner significantly outperform those without it, achieving performance improvements of up to 100%. Additionally, the meta planner is compatible with various existing agent frameworks. When combined with these methods, out approach delivers even greater performance gains, which demonstrates its effectiveness in a larger application scope. Further analysis reveals that our generated meta plans significantly increase the agent’s average reward per action, thereby improving task completion efficiency.

In summary, our contributions are as follows:

- We introduce the MPO, which leverages meta plan optimization to improve the performance of LLM agents. This progress offers an innovative approach to enhance agents’ planning capabilities in a plug-and-play manner, while remaining its compatibility with existing frameworks.
- Extensive experiments conducted on two representative benchmarks demonstrate that our method has significantly improved the performance of existing LLM agents.
- Further analysis indicates that: (1) Our proposed method substantially boosts the agent’s task completion efficiency; (2) A lightweight meta planner can guide more powerful agents in their planning. and (3) MPO increases the correctness, followability, and standardization of the meta plan.

2 Task Formulation

LLM Agent Planning The primary scope of this study is the planning of LLM agents interacting with the environment and receiving feedback for task solution. Following Song et al. (2024b), the agent’s task planning trajectory can be represented as $e = (u, a_1, o_1, \ldots, a_n)$, where $u \in \mathcal{U}$ is the task instruction, $a \in \mathcal{A}$ the agent actions, and $o \in \mathcal{O}$ the observation from the environment. At each time step t, the agent performs implicit planning and generates the corresponding action $a_t \sim \pi_\theta(\cdot|u, a_1, o_1, \ldots, o_{t-1})$. The probability of generating the task planning trajectory is given by:

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

Finally, the final reward $r(u, e) \in [0, 1]$ representing the task completion rate is calculated.

Meta Plan The meta plan serves as high-level, natural guidance to assist in agent planning. It outlines an abstract, general strategy for task completion that is decoupled from specific environmental details, indicating its potential to generalize across various agents. For instance, given the instruction "look at the CD under the desklamp", the meta plan could be: "1. Go to where the CD may be placed. 2. Take the CD from where you found it. 3. Go to where the desklamp is located. 4. Use

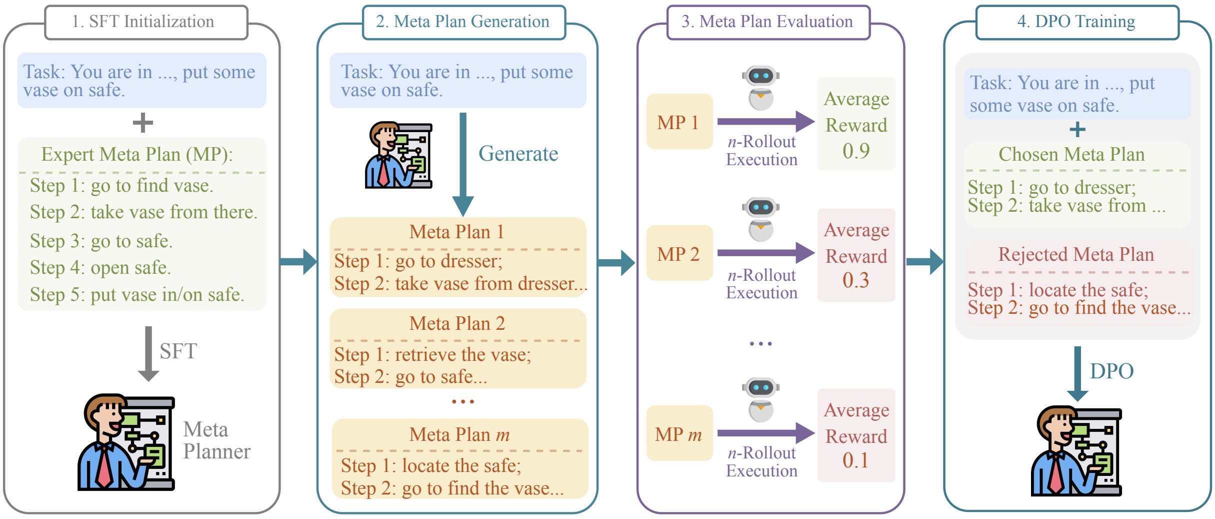

Figure 2: The overall architecture of MPO. The meta planner is first supervised fine-tuned on the seed meta plan (MP) set. Then we optimize the meta planner through preference learning on contrastive meta plan pairs.

the desklamp to look at the CD." A low-quality meta plan might mislead the agent's planning process. To ensure meta plan quality, MPO develops a lightweight parameterized meta planner π_g to generate meta plans, which can be further optimized to produce better results. After incorporating the meta plan, the probability of the agent generating trajectory e is formulated as:

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

3 Method

The overall framework of our method is illustrated in Figure 2. First, we construct a seed meta plan training set to initialize a basic meta planner (§ 3.1). Then, we develop the MC method to assess the quality of the meta plan through exploration (§ 3.2). Finally, we further enhance the meta planner via preference-based optimization using contrastive meta plan pairs (§ 3.3).

3.1 Supervised Fine-tuning Initialization

To equip the meta planner with the foundational capabilities to generate meta plans based on task instructions and the environmental state, we initialize the model using supervised fine-tuning. However, existing agent datasets only provide golden task completion trajectories without corresponding meta plans. Therefore, we first need to construct a training dataset for meta plan generation. To achieve this, we leverage GPT-4o to assist in creating the dataset. We provide the model with the original task instruction u and the corresponding golden trajectory e as the prompt, allowing it to summarize a generalizable plan p from the trajectory. The specific prompt template can be found in Appendix E.1. To ensure the quality of the meta plan p, we manually review the results generated by GPT-4o and refine any meta plans that are incorrect, overly complex, or non-standard. This quality control process ensures that each meta plan p represents a reusable planning strategy that effectively assists agents in task completion. The detailed process for controlling the quality of the seed meta plan set can be found in Appendix C. Since the meta planner needs to generate plans without access to golden trajectories during inference, we remove them from the training data, thus obtaining the initialization dataset for the meta planner:

$$\mathcal{D}_s = \left\{(u,p)^{(i)}\right\}_{i=1}^{|\mathcal{D}_s|} \tag{3}$$

We then fine-tune the model on the auto-regressive loss the get the initialized meta planner π_g:

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

3.2 Meta Plan Quality Evaluation

To further enhance the meta planner, we need to evaluate the quality of its generated meta plans. While prior studies typically rely on reward models trained on human preference annotations (Bai et al., 2022a; Ouyang et al., 2022; Dubey et al., 2024) or advanced AI (Bai et al., 2022b; Lee et al., 2023) models to assess model outputs, these approaches have limitations. They not only incur additional costs for human labeling or API calls, but may also be less applicable to LLM agents, as their preferences for meta plans are not aligned with the agent or task environment. To circumvent these challenges, we adopt an exploration-based approach to evaluate the quality of meta plans.

Intuitively, a higher-quality meta plan should enable the agent to more easily succeed in the task. Therefore, for a give meta plan p, we insert it into the prompt of the agent and have the agent attempt to complete the task N times. This results in N task completion trajectories generated by the agent:

$$\{e^{(i)}|i = 1, ..., N\} \sim \pi_\theta(e|u, p) \tag{5}$$

For each trajectory $e^{(i)}$, the environment returns the task completion rate $r(u, e^{(i)})$. Thus, the quality of the meta plan p is determined by the agent success rate in completing the task based on it, which can be represented as:

$$Q(p) = \frac{1}{N}\sum_{i=1}^{N} r(u, e^{(i)}) \tag{6}$$

In this paper, we use Llama-3.1-8B-Instruct (Dubey et al., 2024) as the agent to evaluate the quality of the meta plans. This model demonstrates strong instructing-following capabilities and is already effective at completing agent tasks. Moreover, the meta plans evaluated with this model can be generalized to agents based on other models, which we verify in the experiments later.

3.3 Meta Planner DPO Training

After we are able to automatically evaluate the quality of meta plans, we can further optimize the SFT-initialized meta planner through reinforcement learning. We choose DPO (Rafailov et al., 2024) as our optimization algorithm due to its training stability and low resource consumption. The DPO algorithm requires paired preference data to optimize the meta planner, specifically pairs of high- and low-quality meta plans. We construct the DPO preference dataset $\mathcal{D}_c$ from the task training set, where the SFT-initialized meta planner generates M meta plans $\{p_i|i = 1, ..., M\} \sim \pi_g(p|u)$. We then compute scores for each meta plan using the MC method described in Section 3.2. The highest and lowest quality meta plans are selected as the chosen and rejected pairs p_w and p_l. If all meta plans are of the same quality, we skip this sample. This forms our preference training dataset:

$$\mathcal{D}_c = \left\{(u, p_w, p_l)^{(i)}\right\}_{i=1}^{|\mathcal{D}_c|} \tag{7}$$

Dataset	Train	Test Seen	Test Unseen	Action Space
ScienceWorld	1483	194	211	19
ALFWorld	3321	140	134	13

Table 1: Statistics overview of test datasets. “Test Seen” and “Test Unseen” are test set with seen and unseen scenarios respectively.

Given the preference dataset D_c, DPO optimizes the model to increase the likelihood of the chosen meta plan p_w over the rejected one p_l. We fine-tune the meta planner by minimizing the DPO loss:

$$\mathcal{L}_{DPO}(\pi_\theta; \pi_{ref}) = -\mathbb{E}_{(u,p_w,p_l)\sim\mathcal{D}_c}\Bigg[\log\sigma(\beta\log\frac{\pi_\theta(p_w|u)}{\pi_{ref}(p_w|u)} - \beta\log\frac{\pi_\theta(p_l|u)}{\pi_{ref}(p_l|u)})\Bigg], \tag{8}$$

This equation reflects the goal of maximizing the probability of generating the higher-quality meta plan p_w over the lower-quality meta plan p_l for a given task instruction u. By constructing the preference dataset and applying DPO optimization, the meta planner becomes more effective at generating high-quality meta plans, therefore better guiding the agent planning process.

4 Experiments

4.1 Experiment Settings

Datasets We conducted experiments on two representative agent datasets: ScienceWorld (Wang et al., 2022) for textual science experiment tasks and ALFWorld (Shridhar et al., 2020) for embodied household tasks. ScienceWorld provides dense final rewards ranging from 0 to 1, whereas ALFWorld offers only binary rewards, indicating whether a task has been completed. For details of the datasets, please refer to Appendix A.

The statistical information of our datasets is presented in Table 1. It is important to note that in addition to the in-distribution test sets, both ALFWorld and ScienceWorld include test sets that include out-of-distribution unseen variations. These additional test sets enable us to evaluate the generalization capabilities of the meta planner.

Implementation Details We use Llama-3.1-8B-Instruct (Dubey et al., 2024) as the base model to construct the meta planner. For SFT initialization, we set the batch size to 32, the learning rate to 1e-5 and employ a cosine scheduler with 3 training epochs. For DPO (Rafailov et al., 2024) training, we configure the meta planner to generate $M = 5$

Model	w/o Exp. Guid.	ScienceWorld		ALFWorld		Average
		Seen	Unseen	Seen	Unseen	
Agents w/o Training						
GPT-4o (Achiam et al., 2023)	✗	60.0	56.0	78.6	83.6	69.6
GPT-4o-mini (Achiam et al., 2023)	✗	49.1	42.7	32.1	41.0	41.2
Llama-3.1-8B-Instruct (Dubey et al., 2024)	✗	47.7	42.2	22.9	28.4	35.3
Qwen2.5-7B-Instruct (Yang et al., 2024a)	✗	38.5	38.8	71.4	75.4	56.0
Llama-3.1-70B-Instruct (Dubey et al., 2024)	✗	72.6	70.2	78.6	73.9	73.8
Llama-3.1-8B-Instruct + MPO	✓	56.5	55.5	50.0	52.2	53.6
GPT-4o + MPO	✓	67.3	67.8	**89.3**	**93.3**	79.4
Llama-3.1-70B-Instruct + MPO	✓	**80.4**	**79.5**	85.7	86.6	**83.1**
Agents w/ Training						
Llama-3.1-8B-Instruct + SFT (Zeng et al., 2023)	✗	65.3	57.0	79.3	71.6	68.3
Llama-3.1-8B-Instruct + ETO (Song et al., 2024b)	✗	81.3	74.1	77.1	76.4	77.2
Llama-3.1-8B-Instruct + KnowAgent (Zhu et al., 2024)	✓	81.7	69.6	80.0	74.9	76.6
Llama-3.1-8B-Instruct + WKM (Qiao et al., 2024)	✓	82.1	76.5	77.1	78.2	78.5
Llama-3.1-8B-Instruct-SFT + MPO	✓	70.2	65.9	80.7	**81.3**	74.5
Llama-3.1-8B-Instruct-ETO + MPO	✓	**83.4**	**80.8**	**85.0**	79.1	**82.1**

Table 2: Performance of different methods on two datasets. MPO-optimized meta plans significantly improve performance across various models or agent frameworks, surpassing other explicit guidance (Exp. Guid.) methods.

meta plans per task with a generation temperature of 0.7. To evaluate meta plan quality, we set the agents to generate $N = 5$ task completion trajectories for each meta plan, also using a temperature of 0.7. We utilize vLLM (Kwon et al., 2023) to accelerate the generation process. For DPO training, the batch size is 32, and the learning rate is 1e-5 with a 3% warm-up phase, and a cosine scheduler is used. The β parameter in the DPO loss function is set to 0.1 for both the ALFWorld and ScienceWorld datasets, with training conducted over 3 epochs. All training procedures are implemented using Llama-Factory (Zheng et al., 2024) with full parameter fine-tuning. The experiments are conducted on 8 NVIDIA A100 80GB GPUs.

Base Agents We evaluate our method on two types of agents, guided by MPO-optimized meta plans: (1) Agents without training, which deploy the ReAct framework using foundation models without additional training. We test two proprietary models, including GPT-4o and GPT-4o-mini (Achiam et al., 2023) as well as several open-source models, including Llama-3.1-8B-Instruct, Llama-3.1-70B-Instruct (Dubey et al., 2024), and Qwen2.5-7B-Instruct (Yang et al., 2024a). (2) Agents with training, which enhance agent planning capabilities via parameter updates to foundation models. We examine two agent frameworks: AgentTuning (Zeng et al., 2023), which uses Supervised Fine-Tuning from expert trajectories to improve the agent capabilities of the base model, and ETO (Song et al., 2024b), which learns from failed trajectories and proposes an exploration-based trajectory optimization method to enhance the task-solving process. We also compare with KnowAgent (Zhu et al., 2024) and WKM (Qiao et al., 2024), which also inject explicit guidance into the agent planning process. These two methods require fine-tuning the base models, making them incompatible with other agent frameworks.

Evaluation To ensure experimental reproducibility, we set the decoding temperature to 0 for both meta plan generation by the meta planner and task trajectory generation by the agent. For meta plan generation, we employ a zero-shot prompting approach. When generating task completion trajectories, we include a 1-shot in-context example for each task. The detailed prompts are provided in Appendix E.2. Note that once the meta plans for the test set tasks are generated by the meta planner, we use them across all agents without further modification. Our primary evaluation metric is the **Average Reward**, which calculates the mean reward across all test set task instances. We also report the **Success Rate** in Appendix B. We will release the generated meta plans and parameters of the optimized meta planner upon acceptance.

Base LLM	Setting	SciWorld	ALFWorld
GPT-4o	-	56.0	83.6
	SFT	59.5	91.0
	RFT	61.8	89.6
	MPO	**67.8**	**93.3**
Llama-3.1-8B-Ins	-	42.2	28.4
	SFT	45.6	43.3
	RFT	50.5	47.8
	MPO	**55.5**	**52.2**
Qwen2.5-7B-Ins	-	38.8	75.4
	SFT	37.4	73.9
	RFT	41.9	78.3
	MPO	**43.7**	**82.8**

Table 3: Ablation study on meta planner optimization methods.

4.2 Results

As shown in Table 2, the incorporation of MPO-optimized meta plans consistently improves agent performance across all tasks and frameworks, with the average performance increasing by up to 51.8% for the Llama-3.1-8B-Instruct based agent. Moreover, our meta planner is compatible with other agent training frameworks. The MPO-enhanced Llama-3.1-8B-Instruct-ETO achieves an average reward 3.6 higher the the current SOTA explicit guidance method, WKM. These results demonstrate that our general high-level meta plan, optimized through agent feedback, outperforms complex knowledge-based guidance that relies heavily on manual efforts, lacks generalization ability, and offers no quality assurance. These results highlight the effectiveness of our method in enhancing agent performance. Furthermore, our method demonstrates strong effectiveness in unseen scenarios. For the unseen parts of ScienceWorld and ALFWorld, despite never having encountered these tasks, the meta planner is able to generalize to them and generate high-quality meta plans. This improves the success rate of GPT-4o on the unseen part of ALFWorld by 11.1, achieving a success rate of 93.3. These results underscore that MPO can further enhance the agent's generalization capabilities, particularly in out-of-distribution scenarios.

5 Analysis

5.1 Ablation Study

We conduct ablation experiments on the training methods of the meta planner. For ScienceWorld and ALFWorld, we evaluate on the unseen test set.

Base LLM	Type	SciWorld	ALFWorld
GPT-4o	Inst.	**67.8**	**93.3**
	Thou.	65.3	85.1
	Obs.	67.6	91.8
Llama-3.1-8B	Inst.	**55.5**	**52.2**
	Thou.	38.0	34.3
	Obs.	53.3	50.8
Llama-3.1-8B-SFT	Inst.	**65.9**	**81.3**
	Thou.	47.9	25.4
	Obs.	60.6	67.2

Table 4: The impact of different meta plan insertion positions on agent performance.

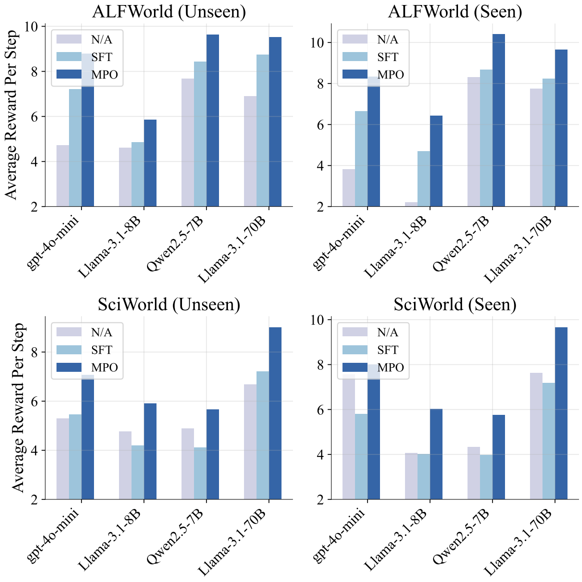

Figure 3: The average reward per step.

As shown in Table 3, the meta planner optimized by MPO leads to greater improvements in agent performance compared to other training methods, suggesting that exploring the environment and learning from comparisons help the meta planner generate higher-quality meta plans. Additionally, when using SFT-initialized meta plans, the performance of the Qwen2.5-7B-Instruct model decreases on both evaluation datasets, indicating that a low-quality meta plan may mislead the agent planning process.

5.2 How to Use Meta Plan?

In our main experiments, the meta plan is incorporated into the task instructions to guide the agent planning process. Here, we investigate the impact of different insertion positions on agent performance: in the task instruction, in the agent's thought process and in the environment observation. As shown in Table 4, we find that insertion into the

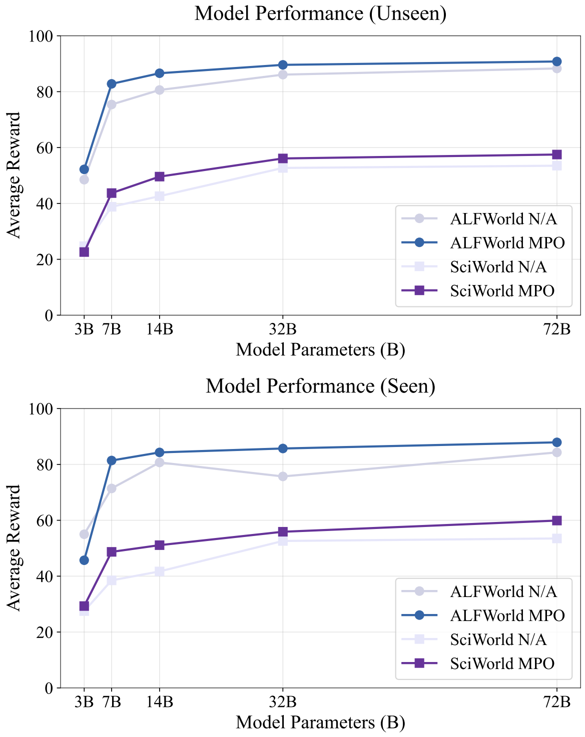

Figure 4: The effectiveness of MPO across agents with different parameter sizes.

task instruction consistently yields the best performance across all agents and tasks, while insertion into the thought process leads to the worst performance. This suggests that disrupting the agent's normal reasoning process negatively affects planning accuracy. Additionally, we observe that inserting the meta plan at other positions causes greater performance drops in agent frameworks with training, likely because the training data does not involve meta plans. In contrast, insertion into the instruction causes minimal disruption to the original task completion process. These results suggest that inserting the meta plan in the task instruction ensures optimal performance.

5.3 Efficiency Analysis

Another advantage of incorporating high-quality meta plans is that it prevents agents from unnecessary exploration, thus improving their task completion efficiency. Following Xiong et al. (2024), we evaluate action efficiency using the average reward per step, calculated for each task as the ratio of the final reward to the number of steps required to complete the task, and then averaging these values across the entire test set. Figure 3 shows the significant improvements in average step rewards achieved by our MPO compared to both the no-meta-plan (N/A) and SFT-initialized meta plans. It is also clear that for the unseen test tasks, MPO leads to an even greater increase in average reward per step, demonstrating its strong generalization to out-of-distribution tasks. These results underscore the superior performance of MPO, confirming its effectiveness in enhancing agent action efficiency.

5.4 Effect on Agents with Scaling Parameters

To further validate the effectiveness of our method, we conduct experiments on models with different parameter sizes. We choose the Qwen2.5-Instruct family as the test models, selecting a range of parameter sizes from small to large: 3B, 7B, 14B, 32B, and 72B, and evaluate their performance on both the seen and unseen parts of ScienceWorld and ALFWorld. The effectiveness of MPO across agents with different parameter sizes is shown in Figure 4. We observe that as the parameter size of the agents increases, the performance improvement from MPO initially increases and then decreases. This may be because the 72B model already has strong planning capabilities, making the improvement from MPO relatively limited. Additionally, for the 3B model, due to its limited instruction-following ability, the model struggles to effectively utilize the meta plan. As a result, inserting the meta plan into the prompt actually leads to a performance decrease. These results suggest that MPO can enhance agent performance across a wide range of parameter sizes, with the most significant improvements observed in agents with medium-sized parameters. Moreover, as a lightweight model, the meta planner has the potential to enhance more powerful agents in a plug-and-play manner without the need for retraining the agents.

5.5 What Makes a Good Meta Plan?

We further investigate why the meta plans optimized through exploration in MPO outperform those obtained soly through SFT initialization. We evaluate the meta plans from three perspectives: correctness, followability, and standardization, using GPT-4o for automated assessment. The evaluation details and prompts can be found in Appendix E.3. As shown in Figure 5, MPO-optimized meta plans consistently outperform SFT-initialized ones across all three dimensions. The advantages in correctness and followability make it easier for the agent to effectively plan and execute tasks, leading to higher task completion rates. Please refer to Appendix D for a more detailed case study.

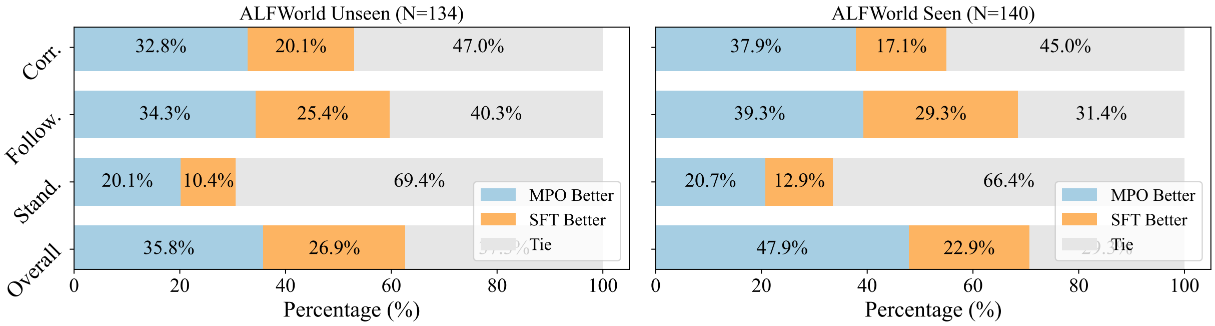

Figure 5: The comparison of SFT-initialized and MPO-optimized meta plans on ALFWorld.

Method	Plug-and-Play	Explicit Guidance	Environmental Adaption	Human-Labor Free
ReAct	✓	✗	✗	✓
AgentTuning	✗	✗	✗	✓
ETO	✗	✗	✓	✓
KnowAgent	✗	✓	✗	✗
WKM	✗	✓	✗	✓
MPO	✓	✓	✓	✓

Table 5: MPO vs alternatives: MPO offers explicit guidance to agent planning in a plug-and-play manner and leverages environmental feedback for optimization.

6 Related Work

LLM as Agents With advancements in reasoning and instruction-following capabilities of large language models (Wei et al., 2022a), researchers have begun using prompting methods (Wei et al., 2022b; Song et al., 2023) or more complex strategies (Koh et al., 2024) to build agents capable of leveraging tools (Qin et al., 2023), solving problems, writing code (Qian et al., 2023), and completing real-world tasks (Patil et al., 2023; Gur et al., 2023; Yang et al., 2024b). To enhance the capabilities of open-source models as agents, some works (Zeng et al., 2023; Song et al., 2024a) have begun using expert trajectories for supervised fine-tuning LLMs, while others (Song et al., 2024b; Xiong et al., 2024) enable agents to explore the environment autonomously and leverage reinforcement learning to learn from failed experiences. However, these methods require retraining each time a new agent is deployed, leading to significant computational overhead.

Planning in LLM Agents Planning (Huang et al., 2024) is essential for intelligent agents to complete real-world tasks, involving the decomposition of complex instructions into sub-tasks and acting on them sequentially. Previous works (Yao et al., 2022; Shinn et al., 2024) primarily focus on implicit planning, where planning occurs through interleaved reasoning and action generation. To address the challenges of myopic reasoning and planning hallucination in implicit planning (Zhu et al., 2024), some approaches (Guan et al., 2024; Li et al., 2024; Zhao et al., 2024; Zhu et al., 2024) have explored using explicit knowledge to guide task execution. However, these methods often require manually designed prompt templates or task procedures, making it difficult to transfer across different environments. Some works (Zhou et al., 2023; Ye et al., 2023; Fu et al., 2024) use language models to automate task knowledge synthesis, but the generated knowledge cannot be further optimized through exploration and environmental feedback, leading to suboptimal performance. In contrast, our MPO introduces an automatically generated meta plan that provides high-level, abstract guidance to assist in agent planning, while also allowing for further quality enhancement based on feedback from the agent task completion process.

A comparison of MPO with several alternatives in Table 5 highlights the advantages of our method in enhancing LLM agents planning capabilities.

7 Conclusion

In this paper, we present MPO, a novel framework for enhancing the planning capabilities of LLM-based agents. MPO integrates abstract, high-level guidance through meta plans, providing a plug-and-play solution to efficiently improve agent performance. By utilizing feedback from the agent's task execution, MPO enables continuous enhancement of the meta plan quality. Extensive experiments on two benchmarks demonstrate that our framework consistently outperforms existing baselines and is applicable to agents across a wide range of parameter sizes. These findings highlight the potential of our approach to advance agent planning capabilities, paving the way for future developments in artificial general intelligence.

Limitations

Despite achieving superior performance compared to other baselines, it is important to acknowledge several limitations of this work. 1) Our approach uses Llama-3.1-8B-Instruct as the base model to construct the meta planner. However, it is worth exploring the potential differences when utilizing other base models or models with varying parameter sizes for the meta planner. Future work could investigate the use of more lightweight models, such as those with as few as 1B parameters, to enhance computational efficiency. 2) Our method only focuses on constructing a separate meta planner for each individual task. However, building a meta planner that incorporates data from multiple tasks may allow it to learn from diverse knowledge sources, resulting in higher-quality meta plans. Future research could develop a unified meta planner that is applicable to various tasks. 3) In the meta planner DPO training, we employ simple sampling and Monte Carlo methods to construct contrastive meta plan pairs. Future work could explore the application of MCTS methods to improve the efficiency of the sampling process.

Ethics Statement

This work fully complies with the ACL Ethics Policy. We declare that there are no ethical issues in this paper, to the best of our knowledge.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022a. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. 2022b. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

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. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

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. *arXiv preprint arXiv:2402.01469*.

Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. 2023. A real-world webagent with planning, long context understanding, and program synthesis. *arXiv preprint arXiv:2307.12856*.

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. *arXiv preprint arXiv:2402.02716*.

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

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Ren Lu, Thomas Mesnard, Johan Ferret, Colton Bishop, Ethan Hall, Victor Carbune, and Abhinav Rastogi. 2023. Rlaif: Scaling reinforcement learning from human feedback with ai feedback.

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. *arXiv preprint arXiv:2402.00798*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2023. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*.

Chen Qian, Xin Cong, Cheng Yang, Weize Chen, Yusheng Su, Juyuan Xu, Zhiyuan Liu, and Maosong Sun. 2023. Communicative agents for software development. *arXiv preprint arXiv:2307.07924*, 6(3).

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. *CoRR*, abs/2405.14205.

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

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

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

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

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. *arXiv preprint arXiv:2410.07706*.

Yifan Song, Weimin Xiong, Dawei Zhu, Wenhao Wu, Han Qian, Mingbo Song, Hailiang Huang, Cheng Li, Ke Wang, Rong Yao, et al. 2023. Restgpt: Connecting large language models with real-world restful apis. *arXiv 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 for llm agents. *arXiv preprint arXiv:2403.02502*.

Fei Wang, Xingchen Wan, Ruoxi Sun, Jiefeng Chen, and Sercan Ö. Arık. 2024. Astute rag: Overcoming imperfect retrieval augmentation and knowledge conflicts for large language models. *Preprint*, arXiv:2410.07176.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022. Scienceworld: Is your agent smarter than a 5th grader? *arXiv preprint arXiv:2203.07540*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022a. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

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

Weimin Xiong, Yifan Song, Xiutian Zhao, Wenhao Wu, Xun Wang, Ke Wang, Cheng Li, Wei Peng, and Sujian Li. 2024. Watch every step! llm agent learning via iterative step-level process refinement. *arXiv preprint arXiv:2406.11176*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024a. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024b. Swe-agent: Agent-computer interfaces enable automated software engineering. *arXiv preprint arXiv:2405.15793*.

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

Yining Ye, Xin Cong, Shizuo Tian, Jiannan Cao, Hao Wang, Yujia Qin, Yaxi Lu, Heyang Yu, Huadong Wang, Yankai Lin, et al. 2023. Proagent: From robotic process automation to agentic process automation. *arXiv preprint arXiv:2311.10751*.

Da Yin, Faeze Brahman, Abhilasha Ravichander, Khyathi Chandu, Kai-Wei Chang, Yejin Choi, and Bill Yuchen Lin. 2023. Lumos: Learning agents with unified data, modular design, and open-source llms. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2023. Agenttuning: Enabling generalized agent abilities for llms. *arXiv preprint arXiv:2310.12823*.

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

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

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

Wangchunshu Zhou, Yuchen Eleanor Jiang, Long Li, Jialong Wu, Tiannan Wang, Shi Qiu, Jintian Zhang, Jing Chen, Ruipu Wu, Shuai Wang, et al. 2023. Agents: An open-source framework for autonomous language agents. *arXiv preprint arXiv:2309.07870*.

Yuqi Zhu, Shuofei Qiao, Yixin Ou, Shumin Deng, Ningyu Zhang, Shiwei Lyu, Yue Shen, Lei Liang, Jinjie Gu, and Huajun Chen. 2024. Knowagent: Knowledge-augmented planning for llm-based agents. *arXiv preprint arXiv:2403.03101*.

A Dataset Details

ScienceWorld ScienceWorld (Wang et al., 2022) is a text-based virtual environment that provides a testing platform for AI research, specifically designed to evaluate and improve AI systems' scientific reasoning abilities. Researchers can use this platform to assess the performance of AI agents in open, complex environments. ScienceWorld simulates tasks from standard elementary school science curricula, covering areas such as state changes of matter, measurement, electricity, life sciences, plant growth, chemical reactions, and genetics. Agents are deployed in an embodied interactive environment to understand and apply complex scientific concepts. Tasks in ScienceWorld involve several subgoals, and the overall final reward is calculated based on the completion of these subgoals.

The original test set of ScienceWorld includes unseen task variations. For example, in the training set, a task may involve boiling water, while in the test set, the task may be boiling lead. Following Song et al. (2024b), we use the original test set to evaluate the generalization ability of our meta planner in unseen scenarios, and the original validation set serves as our test set for seen scenarios.

ALFWorld ALFWorld (Shridhar et al., 2020) are household tasks that require agents to explore rooms and use commonsense reasoning to perform tasks, such as "put a pencil on the desk". The environment provides the outcome on whether the agent successfully completes the task within given steps. The original ALFWorld dataset comprises both seen and unseen evaluation sets. The seen set is designed to assess in-distribution generalization, whereas the unseen set with new tasks measures out-of-distribution generalization of the agents.

B Success Rate

We report the success rate of our experiments in Table 7. Note that the definition of success rate differs between the two tasks. For ScienceWorld, the original paper does not provide a specific definition for success rate. However, based on the official environment, a trajectory is considered successful if the agent reaches a predefined latent state, even if the reward is not exactly 1.0. For ALFWorld, since it only provides binary final rewards, the success rate is equivalent to the average final reward. After inserting the MPO-optimized meta plan, all agents show consistent and significant success rate improvements across both tasks.

C Seed Meta Plans Quality Control

A high-quality seed meta plan training set is crucial for initializing a more effective meta planner. As such, we carefully control the quality of the meta plans generated by GPT-4o. We have identified several key issues with the meta plans it produces: (1) they often include excessively detailed steps or environmental information, which makes them difficult to generalize and optimize; (2) they sometimes feature manipulation types that are not applicable to the environment; (3) they fail to adhere to the predefined meta plan format. To address the first two issues, we adjust the temperature during GPT-4o's generation and re-summarize the meta plan. For the third issue, we additionally prompt GPT-4o to extract correctly formatted meta plans from the response. Although manual verification is required to ensure quality, the human effort involved in this process is negligible compared to the manual construction of knowledge in Zhu et al. (2024).

D Case Study

Here we provide a detailed comparison of agent trajectories on the same task within ALFWorld, after inserting meta plans optimized by two different methods: SFT and MPO. This comparison demonstrates how MPO provides higher-quality plan guidance. The case is shown in Figure 11. The agent used in this case study is Llama-3.1-8B-Instruct.

In the ALFWorld scenario, the meta plan generated by the SFT-initialized meta planner mistakenly includes the instruction "go to sidetable", which misleads the agent into repeatedly executing the erroneous plan "I can try to go to sidetable first," resulting in plan hallucination. In contrast, the MPO-optimized meta planner generates a higher-quality meta plan: "go to where the first pillow may be located." This plan outlines an abstract, general task completion strategy, decoupled from specific environmental details, and correctly guides the agent in planning to locate the pillow in the environment with "I can check one by one, starting from armchair 1."

E Prompts Used in Our Work

E.1 Prompt for Seed Meta Plans Collection

We show the prompt for GPT-4o to generate the seed meta plan dataset based on the task instruc-

Model	w/o Meta Plan	ScienceWorld		ALFWorld		Average
		Seen	Unseen	Seen	Unseen	
		Agents w/o Training				
GPT-4o (Achiam et al., 2023)	✗	60.0	56.0	78.6	83.6	69.6
	✓	67.3	67.8	89.3	93.3	79.4
GPT-4o-mini (Achiam et al., 2023)	✗	49.1	42.7	32.1	41.0	41.2
	✓	55.7	52.8	64.3	79.9	63.2
Llama-3.1-8B-Instruct (Dubey et al., 2024)	✗	47.7	42.2	22.9	28.4	35.3
	✓	56.5	55.5	50.0	52.2	53.6
Qwen2.5-7B-Instruct (Yang et al., 2024a)	✗	38.5	38.8	71.4	75.4	56.0
	✓	41.7	43.7	81.4	82.8	62.4
Llama-3.1-70B-Instruct (Dubey et al., 2024)	✗	72.6	70.2	78.6	73.9	73.8
	✓	80.4	79.5	85.7	86.6	83.1
		Agents w/ Training				
Llama-3.1-8B-Instruct + SFT (Zeng et al., 2023)	✗	65.3	57.0	79.3	71.6	68.3
	✓	70.2	65.9	80.7	81.3	74.5
Llama-3.1-8B-Instruct + ETO (Song et al., 2024b)	✗	81.3	74.1	77.1	76.4	77.2
	✓	83.4	80.8	85.0	79.1	82.1

Table 6: The average reward comparison of different agents after incorporating MPO-optimized meta plans on two datasets.

Model	w/o Meta Plan	ScienceWorld		ALFWorld		Average
		Seen	Unseen	Seen	Unseen	
		Agents w/o Training				
GPT-4o (Achiam et al., 2023)	✗	59.8	57.8	78.6	83.6	70.0
	✓	61.3	65.9	89.3	93.3	77.5
GPT-4o-mini (Achiam et al., 2023)	✗	38.7	28.9	32.1	41.0	35.2
	✓	41.2	41.2	64.3	79.9	56.7
Llama-3.1-8B-Instruct (Dubey et al., 2024)	✗	25.8	25.6	22.9	28.4	25.7
	✓	47.9	53.6	50.0	52.2	50.9
Qwen2.5-7B-Instruct (Yang et al., 2024a)	✗	22.7	30.8	71.4	75.4	50.1
	✓	32.0	33.2	81.4	82.8	57.4
Llama-3.1-70B-Instruct (Dubey et al., 2024)	✗	67.5	64.9	78.6	73.9	71.2
	✓	71.7	69.7	85.7	86.6	78.4
		Agents w/ Training				
Llama-3.1-8B-Instruct + SFT (Zeng et al., 2023)	✗	59.3	64.9	79.3	71.6	68.8
	✓	68.6	72.0	80.7	81.3	75.7
Llama-3.1-8B-Instruct + ETO (Song et al., 2024b)	✗	75.8	77.7	77.1	78.4	77.3
	✓	80.9	78.7	85.0	79.1	80.9

Table 7: The success rate comparison of different agents after incorporating MPO-optimized meta plans on two datasets. For ALFWorld, the success rate is equivalent to the average final reward.

tions. We provide the task instruction, environmental information, and the current task completion trajectory, then prompt GPT-4o to extract a meta plan that includes environmental priors and can guide the task completion process. The prompt is shown in Figure 6 and Figure 7.

E.2 Prompt for Evaluation

We show the instruction prompts for ScienceWorld and ALFWorld in Figure 8 and 9, respectively.

E.3 Prompt for GPT Automated Assessment

We show the prompt in Figure 10 that enables GPT-4o to automatically evaluate the quality of the MPO-optimized meta plan from three aspects: correctness, followability, and standardization. Correctness assesses whether the plan accurately fulfills the task requirements, followability evaluates whether the plan is clear, easy to understand, and whether the steps are reasonable, while standardization checks if the meta plan follows a consistent and standardized format. For each dimension, GPT-4o is asked to first identify which set of plans is better and provide the reasoning procedure. Finally, an overall assessment is given.

Prompt for ScienceWorld Meta Plan Collection

Please generate a step-by-step meta plan for a scientific task:
<task>
You are a helpful assistant to do some scientific experiment in an environment.
In the environment, there are several rooms: kitchen, foundry, workshop, bathroom, outside, living room, bedroom, greenhouse, art studio, 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, 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 meta plan, which can be used to solve similar tasks in the future.
The meta plan should be a commonly-reused routine of the tasks.
The generated meta plan should be written in the following format:
<meta_plan>
Step 1: ...
Step 2: ...
...
</meta_plan>

Figure 6: Prompt for ScienceWorld Meta Plan Collection.

Prompt for ALFWorld Meta Plan Collection

Please generate a step-by-step meta plan for a house holding 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>

The generated meta plan should be written in the following format:
<meta_plan>
Step 1: ...
Step 2: ...
...
</meta_plan>

Figure 7: Prompt for ALFWorld Meta Plan Collection.

Instruction Prompt for ScienceWorld

You are a helpful assistant to do some scientific experiment in an environment.
In the environment, there are several rooms: kitchen, foundry, workshop, bathroom, outside, living room, bedroom, greenhouse, art studio, 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, you can directly get items from the containers.
For each of your turn, 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:" in 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 meta plan maybe helpful for you to complete the task:
{meta_plan}

Figure 8: Instruction prompt for ScienceWorld.

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 the detailed description of the current environment and your goal to accomplish.
For each of your turn, 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 your each turn, the environment will give you immediate feedback based on which you plan your next few steps. if the envrionment output "Nothing happened", that means the previous action is invalid and you should try more options.
Reminder:
1. The action must be chosen from the given available actions. Any actions except 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 meta plan maybe helpful for you to complete the task:
{meta_plan}

Figure 9: Instruction prompt for ALFWorld.

Instruction Prompt for GPT Automated Assessment

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

Task description: {task}

DPO Plan:
{dpo}

SFT Plan:
{sft}

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

For each dimension, please indicate which meta plan is better and provide reasoning. Finally, provide an overall assessment.

Please output the result in JSON format, including the following fields:

```
{
    "correctness_better": "dpo"/"sft"/"tie",
    "correctness_reason": "reason",
    "followability_better": "dpo"/"sft"/"tie",
    "followability_reason": "reason",
    "standardization_better": "dpo"/"sft"/"tie",
    "standardization_reason": "reason",
    "overall_better": "dpo"/"sft"/"tie"
}
```

Figure 10: Instruction prompt for GPT automated assessment.

Case Study of ALFWorld

SFT

Task Instruction: You are in the middle of a room. Looking quickly around you, you see a armchair 1, a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a drawer 5, a drawer 4, a drawer 3, a drawer 2, a drawer 1, a dresser 1, a garbagecan 1, a safe 1, a shelf 12, a shelf 11, a shelf 10, a shelf 9, a shelf 8, a shelf 7, a shelf 6, a shelf 5, a shelf 4, a shelf 3, a shelf 2, a shelf 1, a sidetable 1, and a sofa 1. Your task is to: find two pillow and put them in sofa.

# Bad Meta Plan

Meta Plan: Step 1: go to sofa. Step 2: go to sidetable. Step 3: take pillow from sidetable. Step 4: go to sofa. Step 5: put pillow in/on sofa. Step 6: go to sidetable. Step 7: take another pillow from sidetable. Step 8: go to sofa. Step 9: put second pillow in/on sofa.

Thought: To solve the task, I need to find two pillow and put them in sofa. The meta plan suggests that I should go to sofa first, then go to sidetable.

Action: go to sofa

Observation: Nothing happens.

Thought: It seems that the action "go to sofa" is invalid. I can try to go to sofa again, or I can try to go to sidetable first.

Action: go to sidetable

Observation: Nothing happens.

Thought: It seems that the action "go to sidetable" is also invalid. I can try to go to sofa again, or I can try to go to a cabinet or a drawer instead.

Action: go to cabinet 1

Observation: The cabinet 1 is closed.

Thought: Now that I am in front of the cabinet 1, I can try to open it to see if there are any pillow inside.

Action: open cabinet 1

Observation: You open the cabinet 1. The cabinet 1 is open. In it, you see a tissuebox 1.

Thought: It seems that there is no pillow in the cabinet 1. I can try to go to sidetable again to see if there is a pillow.

Action: go to sidetable

Observation: Nothing happens.

MPO

Task Instruction: You are in the middle of a room. Looking quickly around you, you see a armchair 1, a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a drawer 5, a drawer 4, a drawer 3, a drawer 2, a drawer 1, a dresser 1, a garbagecan 1, a safe 1, a shelf 12, a shelf 11, a shelf 10, a shelf 9, a shelf 8, a shelf 7, a shelf 6, a shelf 5, a shelf 4, a shelf 3, a shelf 2, a shelf 1, a sidetable 1, and a sofa 1. Your task is to: find two pillow and put them in sofa.

# Good Meta Plan

Meta Plan: Step 1: go to where the first pillow may be located. Step 2: take first pillow. Step 3: go to where sofa is. Step 4: put first pillow in/on sofa. Step 5: go to where the second pillow may be located. Step 6: take second pillow. Step 7: go back to sofa. Step 8: put second pillow in/on sofa.

Thought: A pillow is more likely to appear in armchair (1), cabinet (1-4), drawer (1-5), dresser (1), shelf (1-12), sofa (1). I can check one by one, starting with armchair 1.

Action: go to armchair 1

Observation: On the armchair 1, you see a pillow 2, a pillow 1, and a remotecontrol 1.

Thought: Now I find the first pillow (1). Next, I need to take it. I can find the second pillow (2) later in armchair 1.

Action: take pillow 1 from armchair 1

Observation: You pick up the pillow 1 from the armchair 1.

Thought: Now I take the first pillow (1). Next, I need to put it in/on sofa. I can directly go to sofa.

Action: go to sofa 1

Observation: On the sofa 1, you see nothing.

Thought: Now I go to sofa. Next, I need to put the first pillow in/on sofa. I can directly put the first pillow in/on sofa.

Action: put pillow 1 in/on sofa 1

Observation: You put the pillow 1 in/on the sofa 1.

Thought: Now I put the first pillow in sofa. Next, I need to find the second pillow. I can directly go to armchair 1.

Thought: It seems that the action "go to sidetable" is still invalid. I can try to go to sofa again to start the workflow.
Action: go to sofa
Observation: Nothing happens.
Thought: It seems that the action "go to sofa" is still invalid. I can try to go to a drawer instead, as the workflow does not specify which drawer to go to.
Action: go to drawer 1
Observation: The drawer 1 is closed.

....

....

....

Thought: It seems that there is no pillow in the cabinet 2. I can try to go to a different cabinet or location.
Action: go to cabinet 3
Observation: The cabinet 3 is closed.
Thought: Now that I am in front of the cabinet 3, I can try to open it to see if there are any pillow inside.
Action: open cabinet 3
Observation: You open the cabinet 3. The cabinet 3 is open. In it, you see a box 3, and a tissuebox 2.
Reward: 0.0

Action: go to armchair 1
Observation: On the armchair 1, you see a pillow 2, and a remotecontrol 1.
Thought: Now I find the second pillow (2). Next, I need to take it.
Action: take pillow 2 from armchair 1
Observation: You pick up the pillow 2 from the armchair 1.
Thought: Now I take the second pillow (2). Next, I need to put it in/on sofa.
Action: go to sofa 1
Observation: On the sofa 1, you see a pillow 1.
Thought: Now I go to sofa. Next, I need to put the second pillow in/on sofa. I can directly put the second pillow in/on sofa.
Action: put pillow 2 in/on sofa 1
Observation: You put the pillow 2 in/on the sofa 1.
Reward: 1.0

Figure 11: Case study of ALFWorld.