DPEPO: Diverse Parallel Exploration Policy Optimization for LLM-based Agents

Anonymous ACL submission

Abstract

Large language model (LLM) agents that follow the sequential "reason-then-act" paradigm have achieved superior performance in many complex tasks. However, these methods suffer from limited exploration and incomplete environmental understanding, as they interact with only a single environment per step. In this paper, we first introduce a novel paradigm that enables an agent to interact with multiple environments simultaneously and share cross-trajectory experiences. Build upon this paradigm, we further propose Diverse Parallel Exploration Policy Optimization (DPEPO), a reinforcement learning (RL) algorithm that encourages the agent to perform diverse parallel exploration. There are two stages in DPEPO: initial supervised fine-tuning (SFT) imparts basic parallel reasoning and action generation, followed by reinforcement learning stage with a hierarchical reward scheme. We design a parallel trajectory-level success reward and two step-level rewards: Diverse Action Reward and Diverse State Transition Reward, which actively penalize behavioral redundancy and promote broad exploration. Extensive experiments on ALFWorld and ScienceWorld show that DPEPO achieves state-of-the-art (SOTA) success rates, while maintaining comparable efficiency to strong sequential baselines.[1]

1 Introduction

LLM-based autonomous agents commonly adopt the ReAct paradigm (Yao et al., 2023), where the agent alternates between reasoning and acting: it selects a single action, observes the environmental feedback, and then uses that observation to inform its next step. This "reason, then act" framework has shown notable success in complex, long-horizon tasks (Shridhar et al., 2021; Yao et al., 2022; Wang et al., 2022) by enabling structured, step-by-step reasoning. Recent work (Richens et al., 2025) demonstrates that a comprehensive understanding of the environment is crucial for effective and reliable agents. However, ReAct imposes a fundamental limitation: at each step, the agent can perform only one action and observe a single observation. This restricts the breadth of environmental exploration and forces the agent to construct its understanding from a narrow, linear view. As a result, agents often develop an incomplete and biased cognition of the environment, leading to suboptimal decisions and failures in complex tasks.

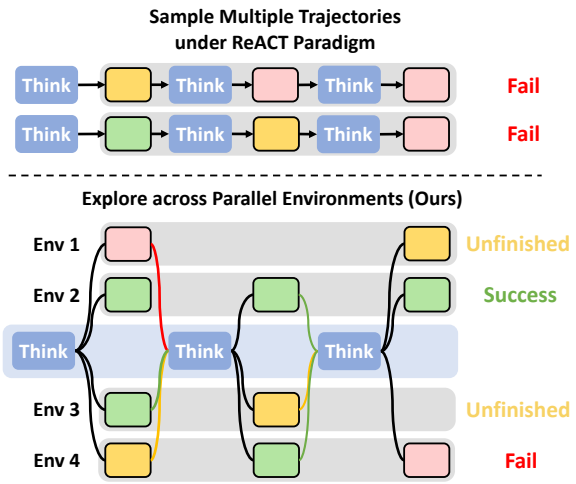

Figure 1: ReAct-based agent constructs environmental understandings from a narrow, linear view. Our DPEPO allows agent to build comprehensive cognition of environments from diverse parallel exploration.

A straightforward approach to broaden exploration is to sample multiple independent trajectories for the same task. However, this naive multi-sampling strategy faces two critical shortcomings: (1) **Lack of diversity**: Despite increased output entropy, actions sampled across multiple trials tend to converge to similar choices, failing to promote truly diverse exploration and preventing the agent from

[1]Code is available at https://anonymous.4open.science/r/Code-for-DPEPO-C6C5/

developing a comprehensive grasp of the environment. (2) **Inefficiency and isolation**: Experiences from different trajectories are neither shared nor leveraged, meaning the agent cannot learn from cross-trajectory insights. Moreover, sampling multiple trials sequentially inflates token consumption and running time, making the approach prohibitively costly for practical deployment.

To overcome these limitations, we introduce parallel exploration, a paradigm in which an agent interacts with multiple environments simultaneously. This allows the agent to gather diverse observations in parallel, share experiences across environments, and build a more holistic understanding of the environment dynamics. Building on this exploration paradigm, we propose Diverse Parallel Exploration Policy Optimization (DPEPO), a novel reinforcement learning framework that endows agents with structured parallel exploration capabilities. After an initial SFT phase to acquire basic parallel reasoning and action generation skills, the agent is optimized via a hierarchical reward scheme that encourages diverse exploration at both the trajectory and step levels. Specifically, we design a parallel trajectory-level success reward and two novel step-level rewards: Diverse Action Reward and Diverse State Transition Reward, that actively penalize redundant behaviors. These rewards are combined through a group-relative advantage computation to provide multi-granularity learning signals, enabling the agent to explore more broadly while staying focused on task completion. Extensive experiments on ALFWorld and ScienceWorld, demonstrate that DPEPO achieves state-of-the-art success rates across all task splits. Notably, our analysis of trajectory length and token usage reveals that DPEPO achieves efficiency comparable to strong sequential baselines, despite its parallel nature.

Our contributions are summarized as follows:

- We propose Diverse Parallel Exploration Policy Optimization (DPEPO), a novel RL framework that pioneers parallel exploration for agents to build holistic environment cognition.
- We design a Parallel Trajectory-level Success Reward to optimize task completion in concurrent multi-environment settings.
- We propose Parallel Step-level Rewards (Diverse Action & State Transition) to penalize redundancy and foster distinct explorations.
- Extensive experiments on ALFWorld and ScienceWorld show DPEPO achieves SOTA performance in terms of success rates while maintaining comparable efficiency with single exploration methods.

2 Related Work

LLM as Autonomous Agents LLMs are increasingly being deployed as autonomous agents across a wide range of domains, including code generation (Zhang et al., 2024), web navigation (Abuelsaad et al., 2024), smart device control (Zhang and Zhang, 2024; Gur et al., 2024; Liu et al., 2025b; Ye et al., 2025), external tool use (Gao et al., 2024; Shi et al., 2024, 2025; Huang et al., 2025) and interactive gaming (Shridhar et al., 2021; Yao et al., 2022; Wang et al., 2022, 2023). Early LLM-based agents typically relied on engineered prompting strategies (Wei et al., 2022) to achieve strong performance, such as ReAct (Yao et al., 2023) and Reflexion (Shinn et al., 2023). Recent studies has shifted their focus toward fine-tuning LLMs for agent-related tasks using either SFT (Song et al., 2024; Xi et al., 2024) or RL (Feng et al., 2025; Zhang et al., 2025), aiming to develop agents with greater generalization ability and robustness, ultimately resulting in more efficient and resilient behaviors compared to earlier prompt-based methods.

Reinforcement Learning for LLM-based Agents Reinforcement learning has been recognized as a foundational approach for enabling LLMs to operate as autonomous agents in complex environments. Early work in this domain primarily employed PPO (Schulman et al., 2017), which allowed agents to learn diverse interactive tasks while avoiding the need for expensive imitation data. With the emergence of reinforcement learning with verifiable reward (RLVR) (Guo et al., 2025; Lambert et al., 2025) and group-based policy optimization (Shao et al., 2024; Zheng et al., 2025; Kool et al., 2019; Liu et al., 2025c), a series of studies have eliminated complex reward model and critic model, instead adopting rule-based reward signals: GiGPO (Feng et al., 2025) proposes anchor state groups to estimate the quality of specific steps; RLVMR (Zhang et al., 2025) introduces meta-reasoning and corresponding step-level rewards to address inefficient exploration; iStar (Liu et al., 2025a) estimates step rewards via the LLM's intrinsic implicit rewards. However, these methods remain constrained by the sequential ReAct paradigm, which limits their exploration space and maintains a narrow fo-

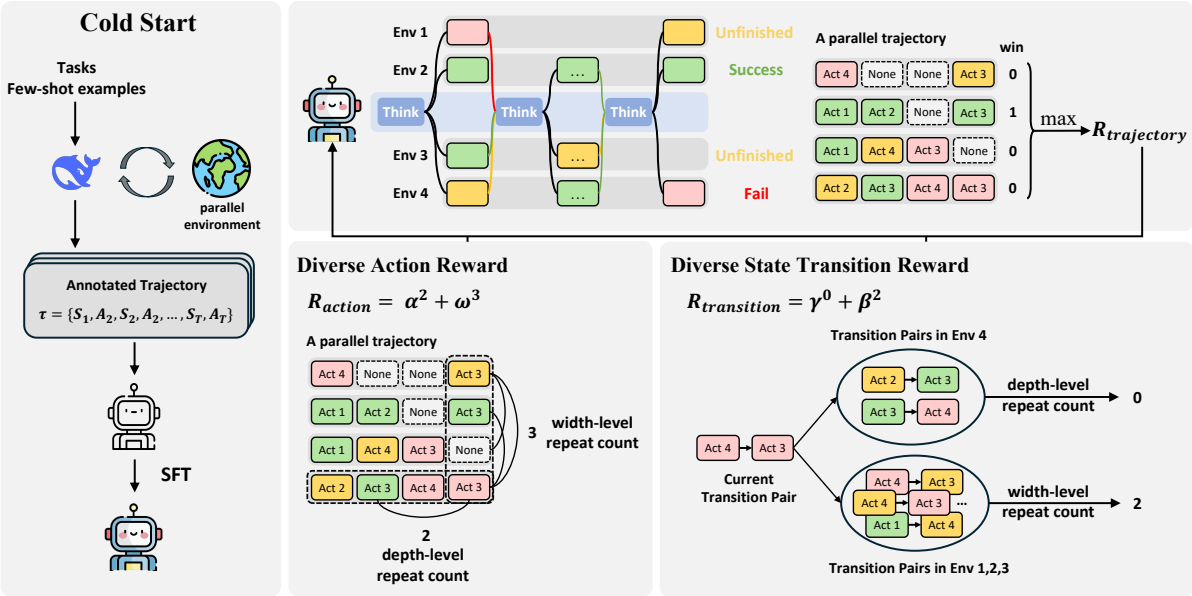

Figure 2: Training framework of our proposed Diverse Parallel Exploration Policy Optimization (DPEPO).

cus on single-path reasoning. In this work, we introduce a novel reinforcement learning algorithm for training agents with more fine-grained reward shaping, drawing from the insight that more diverse and comprehensive exploration, helping agents build a better understanding of environments, thereby enabling them to perform tasks more effectively.

3 Methodology

In this section, we first introduce the parallel agent exploration framework, and then we detail the framework for training the LLM-based agent to perform parallel exploration. The overall architecture is shown in Figure 2.

3.1 Parallel Agent Exploration Formulation

Given a task specified by a prompt and a set of K parallel environments $\mathcal{E} = \{E_1, E_2, \ldots, E_K\}$ sharing the same initial state, an LLM-based agent interacts with $\mathcal{E}$ over multiple turns until the task is completed or the maximum allowed step T is reached. Specifically, at step t, the agent selects a subset of all environments $\mathcal{E}'_t$. Let $|\mathcal{E}'_t|$ denote the number of selected subset, due to the agent dynamically selects environments to interact, $|\mathcal{E}'_t|$ can vary at each step. For the selected environments, the agent generates a set of parallel actions $A_t = \{(E_i, a_t) \mid E_i \in \mathcal{E}'_t\}$, which are executed simultaneously in their respective environments. The agent then observes the resulting states $S_t = \{(E_i, s_t) \mid E_i \in \mathcal{E}'_t\}$. Consequently, the agent's full trajectory can be denoted as:

$$\tau = \{(S_t, A_t)\}_{t=1}^{T}, \tag{1}$$

where the trajectory τ consists a sequence of parallel states and parallel actions.

3.2 Parallel Exploration Capability Cold Start

We begin with an SFT-based cold start stage to endow the agent with foundational skills for structured output and parallel exploration. For the SFT training data, we introduce human prior knowledge into its synthetic pipeline. The data synthesis pipeline is two steps: (1) We first manually annotated five ground-truth trajectories. Each trajectory consists of multiple steps, with some key steps containing two or more parallel exploration actions. (2) Based on these trajectories, we employ a powerful teacher LLM to annotate trajectories containing parallel exploration behaviors. Finally, the agent is fine-tuned on these annotated trajectories, learning to imitate the parallel exploration patterns.

Next, we employ a reinforcement learning method to train the LLM to perform better parallel exploration actions by using dual-level rewards as the training signal: trajectory-level and step-level.

3.3 Parallel Trajectory-level Success Reward

In this section, we extend the rule-based success reward to parallel setting. Formally, given a task, parallel environments $\mathcal{E}$, agent interaction trajectory

τ and task success state $\mathcal{G}$, the parallel trajectory-level success reward is calculated as follows:

$$R_{traj}(\tau) = \begin{cases} 1 & \text{if } S_T \cap \mathcal{G} \neq \emptyset, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

This parallel trajectory-level success reward signifies that when any state in S_T reaches a goal state, the overall trajectory is considered successful. This reward provides a important training signal that guides the agent to complete the task.

3.4 Parallel Step-level Rewards

The trajectory-level reward provides an important yet coarse-grained signal, failing to assess individual steps with parallel actions. To provide fine-grained guidance for diverse exploration and better information acquisition, we introduce two novel step-level and diversity-driven rewards: **Diverse Action Reward** and **Diverse State Transition Reward**. These rewards discourage redundant exploration behavior, forcing the agent to gather more information for making better decisions. These dense, step-level reward signals complement the sparse trajectory-level reward, together providing a multi-granularity learning signal for parallel agent.

Diverse Action Reward To incentivize diverse exploration behavior, we define a composite reward that incorporates both depth-wise (within a single environment) and width-wise (across different environments at the same step) considerations.

Formally, at step t, given the selected environments $\mathcal{E}'_t$ and generated parallel actions A_t. We compute this reward as follows:

$$R_{\text{action}}(A_t) = \frac{1}{|\mathcal{E}'_t|} \sum_{E_i \in \mathcal{E}t'} \alpha^{C_{\text{depth}}(E_i, a_t)} + \omega^{C_{\text{width}}(A_t)}, \tag{3}$$

where $C_{\text{depth}}(E_i, a_t)$ is the number of repetitions of action a_t in environment E_i, $\omega^{C_{\text{width}}(A_t)}$ is the number of repeated actions in $\mathcal{A}_t$. The $\alpha, \omega \in (0, 1]$ are discount factors corresponding to depth-wise and width-wise rewards. The Diverse Action Reward encourages diversity within each environment and between parallel environments.

Diverse State Transition Reward Beyond individual actions, we further propose the diverse state transition reward to discourage the agent from repeating the same state-action transitions, this reward also operates at both depth and width level.

At step t, for each environment in selected $\mathcal{E}'_t$, the agent observes state s_t, takes an action a_t, define this state-action transition as $p_t = s_t \rightarrow a_t$. The parallel state S_t and parallel actions A_t result in parallel state-action transition $\{(E_i, p_t) | E_i \in \mathcal{E}'_t\}$. We compute this reward as follows:

$$\begin{aligned} R_{\text{transition}}(A_t) = & \frac{1}{|\mathcal{E}'_t|} \sum_{E_i \in \mathcal{E}'_t} \gamma^{M_{\text{depth}}(E_i, p_t)} \\ & + \frac{1}{|\mathcal{E}'_t|} \sum_{E_i \in \mathcal{E}'_t} \beta^{M_{\text{width}}(E_i, p_t)}, \end{aligned} \tag{4}$$

where $M_{\text{depth}}(E_i, p_t)$ measures the recurrence of the transition p_t within the environment E_i, and $M_{\text{width}}(E_i, p_t)$ counts how many other selected environments have previously observed the same transition. The $\gamma, \beta \in (0, 1]$ are discount factors for the depth-wise and width-wise terms, respectively.

Step-Level Composite Reward The final reward for step A_t is the average of **Diverse Action Reward** and **Diverse State Transition Reward**:

$$R_{step}(A_t) = \frac{R_{action}(A_t) + R_{transition}(A_t)}{2}. \tag{5}$$

This reward provides a dense, multi-granularity policy optimizing signal that actively promotes diverse and efficient exploration.

3.5 Policy Optimization

Similar to GRPO (Shao et al., 2024), DPEPO eliminates the critic model and instead estimates the advantages of trajectories through group sampling and relative advantage computation. For a given task, we sample a group G containing N trajectories $\{\tau\}_{i=1}^{N}$, each trajectory paired with its corresponding trajectory reward $R_{traj}(\tau_i)$:

$$G = \left\{(\tau_i, R_{\text{traj}}(\tau_i))\right\}_{i=1}^{N}. \tag{6}$$

We then access the advantage of each trajectory by normalizing its reward using group-level statistics:

$$\Phi_{traj}(\tau_i) = \frac{R(\tau_i) - \text{mean}\left(\{R(\tau_j)\}_{j=1}^{N}\right)}{\text{std}\left(\{R(\tau_j)\}_{j=1}^{N}\right)}. \tag{7}$$

The trajectory-level advantage $\Phi_{traj}(\tau_i)$ reflects the agent's overall success in completing the task. Similar to standard GRPO, it performs on outcome-level credit assignment, encouraging policy to develop long-horizon behaviors to complete the task. We use this trajectory-level advantage as the base advantage for more fine-grained, step-level advantage computation. More specifically, at step t,

given a step $A_{i,t}$ in trajectories τ_i with its associated step reward $R_{\text{step}}(A_{i,t})$, we compute the step-level advantage by combining this immediate reward with the global trajectory advantage:

$$\Phi_{step}(A_{i,t}) = \begin{cases} R_{\text{step}}(A_{i,t}), & \text{if } \Phi_{traj}(\tau_i) > 0, \\ 2 - R_{\text{step}}(A_{i,t}), & \text{otherwise.} \end{cases} \tag{8}$$

$$\Phi(A_{i,t}) = \Phi_{step}(A_{i,t}) \cdot \Phi_{traj}(\tau_i). \tag{9}$$

This combined advantage provides hierarchical supervision by assigning credit at both trajectory and step scales, thereby enabling the learned policies to exhibit more diverse behaviors across parallel environments. We then optimize the policy for each step using the same objective as GRPO. The pseudo code for our algorithm can be found in Appendix D.

4 Experimental Setup

4.1 Datasets

We train and evaluate the LLM-based agents on two challenging benchmarks: ALFWorld (Shridhar et al., 2021) and ScienceWorld (Wang et al., 2022). ALFWorld is a text-based interactive environment where agents perform everyday household tasks through multi-step reasoning and interactions. It presents goals like "put the pencil on the desk", requiring agents to explore simulated rooms based on common sense. The environment is fully text-based and only returns a reward signal upon final success or failure. ScienceWorld is a simulated benchmark designed to evaluate scientific agent's reasoning capabilities across diverse elementary science topics. It includes tasks from 10 domains, including electricity, life cycles, and chemical reactions. Each task consists of multiple intermediate sub-goals that agents must complete sequentially.

4.2 Baselines

For ALFWorld and ScienceWorld, we compare our approach with a range of competitive baselines under the ReAct paradigm: (1) Closed-source LLMs: GPT-4o (Hurst et al., 2024), DeepSeek-V3 (Liu et al., 2024), and DeepSeek-R1 (Guo et al., 2025), which show superior performance on various natural language tasks. (2) A series of RL methods: GRPO (Shao et al., 2024), which performs advantage estimation over trajectory groups; GiGPO (Feng et al., 2025), introducing step-level advantage through anchor state grouping; RLVMR (Zhang et al., 2025), which proposes meta-reasoning rewards to address inefficient exploration; and SPEAR (Qin et al., 2025), leveraging self-imitation and intrinsic rewards to improve the exploration–exploitation balance.

4.3 Implementation Details

We conduct experiments on various model sizes of Qwen2.5-Instruct (Yang et al., 2025b) and Qwen3 (Yang et al., 2025a). For cold start training, we annotate approximately 500 and 1,000 for ALFWorld and ScienceWorld, respectively. For reinforcement learning stage, the number of training sample is 500, the temperature for rollout is 0.4, the rollout group size N is set to 8 for ALFWorld and 4 for ScienceWorld, with 1 epoch and a total of 125 training steps. The max step is 25, and the number of parallel environments is 4. Full hyperparameter details and prompts used for training are provided in Appendix B and Appendix C.

5 Experimental Results

5.1 Overall Performance

Table 1 demonstrates the performance of DPEPO and baselines on ALFWorld and ScienceWorld datasets. As illustrated, closed-source LLMs exhibit only limited performance: On ALFWorld, DeepSeek-R1 achieves a success rate of 80.1%, and on ScienceWorld, GPT-4o reaches 45.2%. Under the ReACT paradigm, RL training yields substantial improvements: GRPO raises the average success rate on ALFWorld to 72.0% for the 1.5B model and 77.5% for the 7B model, demonstrating the effectiveness of rule-based success reward. However, the coarse-grained nature of this reward limits further performance gains. In contrast, GiGPO, SPEAR and RLVMR address this issue by incorporating step-level rewards that provide more fine-grained optimization signals for intermediate reasoning steps. Among these methods, RLVMR achieves the strongest performance, with average success rates of 91.6% on ALFWorld and 47.5% on ScienceWorld. Despite the demonstrated effectiveness of these RL methods, they remain constrained by the ReAct paradigm, which restricts the breadth of environmental exploration and forces the agent to construct its understanding from a narrow, linear view. Our proposed DPEPO addresses this limitation through parallel exploration and diversity-driven rewards, achieving the best results: 98.2% on ALFWorld and 61.4% on ScienceWorld. It also demonstrates strong generalization on unseen ALFWorld tasks and on the L1 and L2 splits of Sci-

Method	ALFWorld									ScienceWorld			
	Pick	Cool	Heat	Look	Pick2	Clean	All	OOD	Avg.	L0	L1	L2	Avg.
Frontier LLMs													
GPT-4o	75.3	21.6	56.7	60.8	49.8	31.2	48.0	66.0	57.0	45.4	49.2	41.0	45.2
DeepSeek-V3	71.4	76.0	81.3	61.5	75.0	77.8	74.3	80.6	77.5	27.3	35.2	26.5	29.7
DeepSeek-R1	80.0	80.0	56.3	46.2	87.5	77.8	75.0	85.1	80.1	22.2	31.4	29.1	27.6
Qwen2.5-1.5B-Instruct													
ETO	-	-	-	-	-	-	64.1	66.4	65.3	39.1	22.7	15.6	25.8
GRPO	85.3	59.7	78.2	53.7	53.5	84.5	72.8	71.1	72.0	21.1	13.7	10.9	15.2
GiGPO	94.4	79.8	94.4	67.5	76.4	94.8	86.7	83.2	85.0	25.8	15.2	4.7	15.2
SPEAR	96.4	87.6	**99.0**	86.5	**91.6**	96.1	93.2	-	-	-	-	-	-
RLVMR	-	-	-	-	-	-	89.1	87.9	88.5	46.9	34.3	26.5	35.9
DPEPO	**100.0**	**96.0**	93.8	**100.0**	87.5	**96.3**	**95.7**	**92.5**	**94.1**	**59.8**	**58.1**	**34.2**	**50.7**
Qwen2.5-7B-Instruct													
ETO	-	-	-	-	-	-	70.3	74.2	72.3	62.5	40.6	28.1	43.7
GRPO	90.8	72.5	74.7	66.1	77.6	89.3	77.6	77.3	77.5	49.1	30.1	26.6	35.3
GiGPO	97.7	89.3	83.7	82.7	79.2	**98.8**	90.8	90.2	90.5	53.4	25.2	25.8	34.8
SPEAR	99.9	89.9	**96.4**	85.1	95.1	95.6	94.7	-	-	-	-	-	-
RLVMR	-	-	-	-	-	-	91.4	91.8	91.6	**67.2**	43.0	32.2	47.5
DPEPO	**100.0**	**100.0**	93.8	**100.0**	**100.0**	96.3	**98.6**	**97.8**	**98.2**	66.6	**66.5**	**51.0**	**61.4**

Table 1: Performance comparison on ALFWorld and ScienceWorld. For ALFWorld, we report success rate (%) on seen (in-domain) and unseen (out-of-domain) tasks. For ScienceWorld, we report results on three splits: L0 (seen variants and categories), L1 (unseen variants, seen categories), and L2 (unseen variants and categories). DPEPO achieves superior in-domain performance and strong generalization ability. **Bold** is best, underlined is second-best.

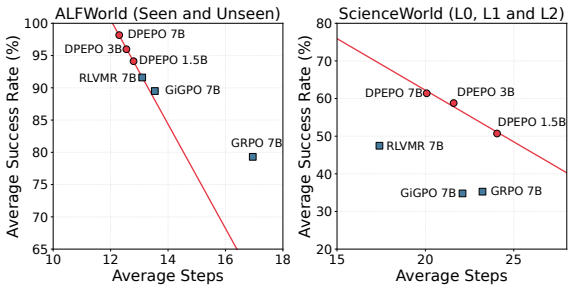

Figure 3: Scaling experiments of DPEPO with varying model sizes. The x-axis and y-axis show the average steps and average success rate across all splits.

enceWorld. This significant improvement can be attributed to diverse parallel exploration, which allows agents to gain a comprehensive understanding of the environment, thereby supporting more informed decision-making and ultimately leading to more successful task completion.

5.2 Scaling Abilities of DPEPO

In this section, to demonstrate the scaling potential of DPEPO, we present its performance across models with varying model sizes. As shown in Figure 3, DPEPO exhibits consistent performance improvements and higher efficiency as the model size increases, underscoring its generalization and scaling abilities. Moreover, on the ALFWorld benchmark, even our smallest DPEPO variant (based on Qwen2.5-1.5B-Instruct) outperforms the strong baseline RLVMR by 2.53% in success rate and completes tasks in fewer steps. On the more challenging ScienceWorld benchmark, DPEPO further achieves higher performance with competitive efficiency. This highlights that diverse parallel exploration enables more effective and efficient task completion. More scaling experiments conducted on Qwen3 series can be found in Appendix A.1.

5.3 Analysis of Environment Numbers

Furthermore, we analyze the impact of varying the number K of parallel environments during inference. For this analysis, we evaluate on both seen and unseen tasks in ALFWorld and report the average success rate. During DPEPO inference, we introduce a limiting prompt that specifies the maximum number of parallel environments the agent is allowed to explore; any trial that exceeds K is treated as a failure. We use GiGPO with multiple sampling as our baseline; in GiGPO, K denotes the number of sampled trajectories.

As shown in Figure 4, both DPEPO and GiGPO exhibit stable performance improvement trends as K increases, but DPEPO consistently outper-

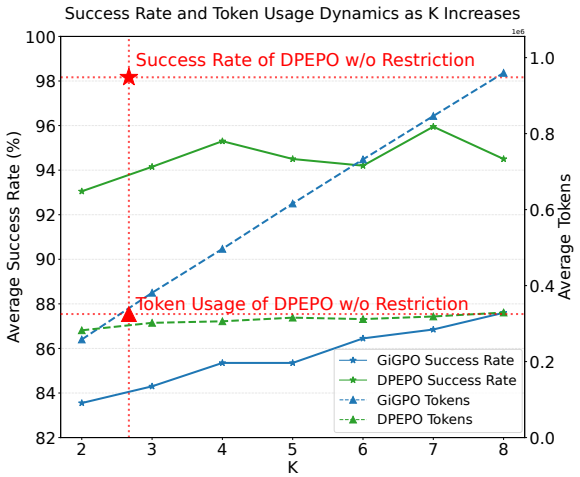

Figure 4: Average success rate and token budgets on ALFWorld's seen and unseen splits, under two scaling strategies: (1) increasing sample count for GiGPO, and (2) increasing the number of environments for DPEPO.

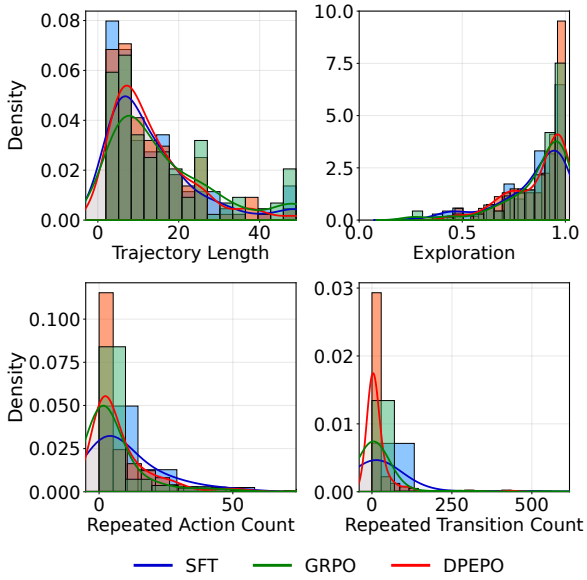

Figure 5: Analysis of exploration dynamics across ablation variants on the ALFWorld seen-task split.

forms GiGPO across all values of K. Moreover, DPEPO achieves strong performance even at small K, demonstrating high sample efficiency. In terms of token usage, the token budget of DPEPO is only slightly higher than that of GiGPO when K is small. As K increases, GiGPO's token consumption increases approximately linearly, while the token budget of DPEPO grows much more slowly. Notably, DPEPO without any restriction on exploration environments not only achieves the best performance, but also explores fewer environments on average. The reason for this phenomenon is that the environmental limit prompt may introduce additional inductive bias, causing the model to explore all environments exhaustively rather than selectively.

5.4 Ablation Study

Next, we conducted an ablation study to evaluate the contribution of each proposed reward component. We compare DPEPO against four variants: the variant only performed the cold start stage (ColdStart), DPEPO without Diverse Action Reward (DPEPO w/o DAR), DPEPO without Diverse State Transition Reward (DPEPO w/o DTR), and DPEPO with neither reward (DPEPO w/o DAR & DTR). As shown in Table 2, the ColdStart variant yields only moderate performance. Removing either DAR or DTR leads to a significant performance degradation, and the variant without both rewards shows an even greater decline. This suggests that DAR and DTR jointly provide training signals for the policy optimization process, thus contributing to the superior performance of DPEPO. More detailed results of specific tasks can be found in Appendix A.2.

Method	ALFWorld		ScienceWorld		
	In Domain	OOD	L0	L1	L2
ColdStart	93.6	97.8	66.5	62.2	48.1
DPEPO	**98.6**	97.8	**66.6**	**66.5**	**51.0**
w/o DAR	97.1	97.0	65.1	62.8	49.0
w/o DTR	96.4	98.5	64.4	63.6	49.7
w/o DAR & DTR	96.4	**98.5**	66.3	65.7	49.9

Table 2: Ablation results on ALFWorld and ScienceWorld. **Bold scores** indicate the best performance, while underlined scores represent the second-best.

To further understand how these reward components influence agent's behavior, we visualize several exploration metrics across aforementioned variants: (1) exploration diversity, (2) state transition repeat count, (3) action repeat count, and (4) trajectory length. The repetition counts are computed using Equations 3 and 4, exploration diversity is the fraction of unique actions among all actions in parallel trajectories. As illustrated in Figure 5, agents trained with DPEPO exhibit significantly lower action and transition repetition and higher exploration diversity compared to ablated variants. This indicates that the DAR and DTR shape the agent's policy to generate trajectories that are not only more goal-directed but also more diverse and efficient, thereby achieving a better trade-off between exploration and task success.

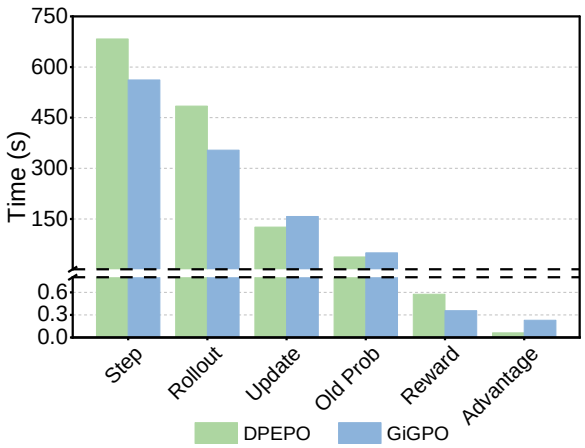

Figure 6: Time consumption details of the main components at each step during DPEPO training process.

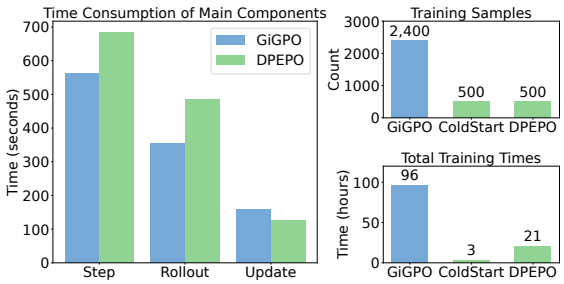

Figure 7: Comparison of training efficiency.

Method	Tokens	Steps	Time (s)
DeepSeek-V3	950.0	20.5	62.4
DeepSeek-R1	1667.9	24.8	237.0
GiGPO	1115.1	15.2	70.8
DPEPO	2283.4	12.3	44.7

Table 3: Inference time comparison between DPEPO and other methods under identical experimental settings.

5.5 Efficiency Analysis

5.5.1 Inference-time Efficiency Analysis

Having established the effectiveness of our approach in terms of success rate, we now turn to its efficiency: while diverse parallel exploration behavior broadens the search space, it may also lead to a higher computational budget or inference time.

To quantitatively assess this trade-off, We compare the token budget and inference time of DPEPO with those of various methods on ALFWorld. As shown in Table 3, although understanding parallel environments brings higher token budgets, DPEPO completes tasks with significantly shorter trajectories and faster inference time compared to GiGPO and the DeepSeek series. We attribute this to two factors: (1) simultaneously executed parallel actions do not increase inference time, and (2) parallel interaction enables the agent to acquire more comprehensive environmental insights, supporting more purposeful decision-making that leads to more efficient and successful task completion.

5.5.2 Training Efficiency Analysis

We further evaluate the training efficiency of DPEPO in comparison with GiGPO, a critic-free method known for its training efficiency and strong performance. We first compare the time consumption per reasoning step. As shown in the left part of Figure 6 and Figure 7, although our method explores more environments in parallel, the time consumption per step does not increase significantly. Despite the longer time per step, the overall training efficiency of DPEPO is significantly higher: as shown in the right part of Figure 7, including the cold start stage, DPEPO achieves superior final performance in approximately 24 hours using only 500 RL training samples. In contrast, GiGPO requires 96 hours of training to reach its peak performance. This experiment is conducted using Qwen2.5-1.5B-Instruct. Further details and experimental setup of this experiment are provided in Appendix A.3.

6 Conclusion

In this work, we propose a novel parallel exploration paradigm and reinforcement learning algorithm to address the constrained exploration challenge inherent in ReACT paradigm. We first extend the ReACT framework to enable agents to simultaneously explore multiple parallel environments. We then introduce Diverse Parallel Exploration Policy Optimization (DPEPO), a novel diversity-driven policy gradient algorithm specifically designed for parallel agents. By incorporating fine-grained, step-level rewards into the parallel trajectory-level success reward, DPEPO encourages agents to perform more diverse exploration in parallel environments. Empirical evaluations across all test splits on ALFWorld and the more challenging ScienceWorld benchmarks demonstrate that DPEPO achieves state-of-the-art performance and high sample efficiency. Through extensive experiments and analysis, we show that the diverse parallel exploration behavior reinforced by DPEPO leads to superior task performance, higher efficiency, and more effective token usage.

Limitations

While DPEPO shows significant improvements in performance and efficiency, it still faces challenges in real-world scenarios where parallel environments are hard to build. However, we suggest a possible solution: training agents to complete different tasks in parallel, which might help mitigate this issue in real-world scenarios. We leave this exploration for future work.

Ethical Considerations

In some real-world environments that may contain biases, parallel exploration enables the agent to gather more experience, which could lead it to learn those biases, potentially resulting in unfair or unsafe behavior. Therefore, before applying our proposed method to real-world tasks, developers should conduct thorough evaluations in their target environments to ensure the agent acquires accurate and objective knowledge and performs actions safely and responsibly.

References

Tamer Abuelsaad, Deepak Akkil, Prasenjit Dey, Ashish Jagmohan, Aditya Vempaty, and Ravi Kokku. 2024. Agent-e: From autonomous web navigation to foundational design principles in agentic systems. *Preprint*, arXiv:2407.13032.

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

Shen Gao, Zhengliang Shi, Minghang Zhu, Bowen Fang, Xin Xin, Pengjie Ren, Zhumin Chen, Jun Ma, and Zhaochun Ren. 2024. Confucius: iterative tool learning from introspection feedback by easy-to-difficult curriculum. 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.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. 2024. A real-world webagent with planning, long context understanding, and program synthesis. In *International Conference on Representation Learning*, volume 2024, pages 52690–52717.

Chengrui Huang, Shen Gao, Zhengliang Shi, Dongsheng Wang, and Shuo Shang. 2025. TTPA: Token-level tool-use preference alignment training framework with fine-grained evaluation. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 16240–16255, Suzhou, China. Association for Computational Linguistics.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Wouter Kool, Herke van Hoof, and Max Welling. 2019. Buy 4 reinforce samples, get a baseline for free! In *ICLR 2019 Workshop*.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. 2025. Tulu 3: Pushing frontiers in open language model post-training. *Preprint*, arXiv:2411.15124.

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*.

Xiaoqian Liu, Ke Wang, Yuchuan Wu, Fei Huang, Yongbin Li, Junge Zhang, and Jianbin Jiao. 2025a. Agentic reinforcement learning with implicit step rewards. *arXiv preprint arXiv:2509.19199*.

Yuhang Liu, Pengxiang Li, Congkai Xie, Xavier Hu, Xiaotian Han, Shengyu Zhang, Hongxia Yang, and Fei Wu. 2025b. Infigui-r1: Advancing multimodal gui agents from reactive actors to deliberative reasoners. *Preprint*, arXiv:2504.14239.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025c. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*.

Yulei Qin, Xiaoyu Tan, Zhengbao He, Gang Li, Haojia Lin, Zongyi Li, Zihan Xu, Yuchen Shi, Siqi Cai, Renting Rui, et al. 2025. Learn the ropes, then trust the wins: Self-imitation with progressive exploration for agentic reinforcement learning. *arXiv preprint arXiv:2509.22601*.

Jonathan Richens, Tom Everitt, and David Abel. 2025. General agents need world models. In *Forty-second International Conference on Machine Learning*.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Zhengliang Shi, Shen Gao, Xiuyi Chen, Yue Feng, Lingyong Yan, Haibo Shi, Dawei Yin, Pengjie Ren, Suzan Verberne, and Zhaochun Ren. 2024. Learning to use tools via cooperative and interactive agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10642–10657, Miami, Florida, USA. Association for Computational Linguistics.

Zhengliang Shi, Shen Gao, Lingyong Yan, Yue Feng, Xiuyi Chen, Zhumin Chen, Dawei Yin, Suzan Verberne, and Zhaochun Ren. 2025. Tool learning in the wild: Empowering language models as automatic tool agents. In *Proceedings of the ACM on Web Conference 2025*, WWW '25, page 2222–2237, New York, NY, USA. Association for Computing Machinery.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. Alfworld: Aligning text and embodied environments for interactive learning. *Preprint*, arXiv:2010.03768.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. 2024. Trial and error: Exploration-based trajectory optimization for llm agents. *arXiv preprint arXiv:2403.02502*.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. *Preprint*, arXiv:2305.16291.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Dingwen Yang, Chenyang Liao, Xin Guo, Wei He, et al. 2024. Agentgym: Evolving large language model-based agents across diverse environments. *arXiv preprint arXiv:2406.04151*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. 2025a. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

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, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2025b. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. WebShop: Towards Scalable Real-World Web Interaction with Grounded Language Agents. In *Advances in Neural Information Processing Systems*, volume 35, pages 20744–20757. Curran Associates, Inc.

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*.

Jiabo Ye, Xi Zhang, Haiyang Xu, Haowei Liu, Junyang Wang, Zhaoqing Zhu, Ziwei Zheng, Feiyu Gao, Junjie Cao, Zhengxi Lu, Jitong Liao, Qi Zheng, Fei Huang, Jingren Zhou, and Ming Yan. 2025. Mobile-agent-v3: Fundamental agents for gui automation. *Preprint*, arXiv:2508.15144.

Kechi Zhang, Jia Li, Ge Li, Xianjie Shi, and Zhi Jin. 2024. CodeAgent: Enhancing code generation with tool-integrated agent systems for real-world repo-level coding challenges. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13643–13658, Bangkok, Thailand. Association for Computational Linguistics.

Zhuosheng Zhang and Aston Zhang. 2024. You only look at screens: Multimodal chain-of-action agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3132–3149, Bangkok, Thailand. Association for Computational Linguistics.

Zijing Zhang, Ziyang Chen, Mingxiao Li, Zhaopeng Tu, and Xiaolong Li. 2025. Rlvmr: Reinforcement learning with verifiable meta-reasoning rewards for robust long-horizon agents. *Preprint*, arXiv:2507.22844.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. 2025. Group sequence policy optimization. *Preprint*, arXiv:2507.18071.

A Supplementary Results of Experiments

A.1 Additional Results of Scaling Analysis

In Section 5.2, we presented the performance of Qwen2.5 variants across different model scales. Here, we provide a detailed results of Qwen3 variants with varying parameter, reporting their average success rate on both seen and unseen tasks of ALFWorld. Full results are shown in Figure 8. As illustrated, stronger models consistently achieve higher performance. Within the Qwen3 series, success rate improves steadily with model size, demonstrating clear and predictable scaling behavior.

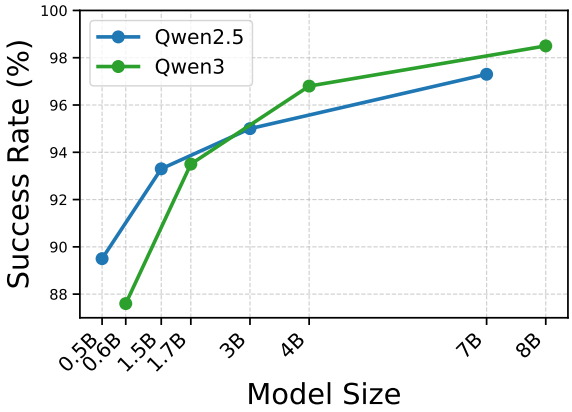

Figure 8: Scaling experiments conducted on Qwen3.

A.2 Detailed Results of Ablation Study

In Section 5.4, we have reported the performance of various DPEPO variants. In this section, we provide a more detailed breakdown of their results across individual tasks in the ALFWorld In-Domain setting. Full scores are shown in Table 4.

A.3 Details of Training Efficiency Analysis

Both our approach and GiGPO are adapted from GRPO and thus follow the same training pipeline as GRPO, which includes multi-turn rollout, old policy probability computation, reward and advantage estimation, and policy updates. We have reported the total training time in Section 5.5.2; in this section, we detail the time consumption of different training components per step.

During DPEPO training, the agent is allowed to explore up to 4 parallel environments per step, with a maximum of 25 steps per trajectory. In contrast, GiGPO explores only a single environment with no more than 50 steps. Under this setting, Figure 6 indicates that although the DPEPO rollout phase takes longer than that of GiGPO due to parallel execution, policy updates, probability recomputation, and advantage estimation are consistently faster. We attribute this to the reduced average trajectory depth under parallel exploration, which yields fewer but higher-quality samples for policy updates.

A.4 Parallel Exploration Behavior Analysis

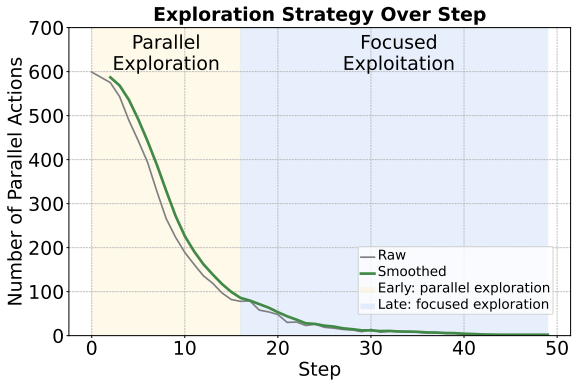

Figure 9: Agent's Exploration Strategy Dynamics

Exploration Strategy As presented in Figure 9, the agent performs more parallel actions in the early steps. At this stage, the agent is unfamiliar with the environment and relies on parallel exploration to rapidly build a preliminary understanding of task and environment. In the later steps, after acquiring a basic model of the environment and the task, the agent shifts its strategy toward focused exploration. It reduces the number of parallel actions and concentrates on a specific environment, using only a few parallel actions to support its more targeted exploration.

Efficient Search via Parallel Actions The case shown in Figure 18 illustrates how the agent efficiently searches for a candle that is not visible in the initial observation. Recognizing that candles are commonly stored on shelves, the agent prioritizes the three shelves in the room as likely candidates. Instead of checking them one by one, it leverages parallel environments to visit shelf 1, shelf 2, and shelf 3 simultaneously. This strategy reduces search time and demonstrates the agent's ability to perform goal-directed exploration by distributing candidate actions across multiple environments.

Validating Hypotheses via Parallel Exploration As the case shown in Figure 19, when the agent repeatedly fails to take book 2 from the bed, it

Method	ALFWorld									ScienceWorld			
	Pick	Cool	Heat	Look	Pick2	Clean	All	OOD	Avg.	L0	L1	L2	Avg.
Qwen2.5-7B-Instruct													
ColdStart	**100.0**	84.0	**100.0**	92.3	91.7	92.6	93.6	97.8	95.7	66.5	62.2	48.1	58.9
DPEPO	**100.0**	**100.0**	93.8	**100.0**	**100.0**	96.3	**98.6**	97.8	**98.2**	**66.6**	**65.5**	**51.0**	**61.4**
w/o DAR	**100.0**	96.0	93.8	84.6	**100.0**	**100.0**	97.1	97.0	97.1	65.1	62.8	49.0	59.0
w/o DTR	**100.0**	**100.0**	**100.0**	92.3	87.5	96.3	96.4	**98.5**	97.5	64.4	63.6	49.7	59.2
w/o DAR & DTR	97.1	92.0	**100.0**	**100.0**	91.7	**100.0**	96.4	**98.5**	97.5	66.3	65.7	49.9	60.6

Table 4: Detailed per-task success rate of DPEPO variants on ALFWorld In Domain split, alongside overall results on ScienceWorld. This table complements the summary success rate in Table 2 by providing fine-grained performance across individual ALFWorld tasks. **Bold scores** indicate the best performance, while underlined scores represent the second-best.

hypothesizes that there might be a carrying limit. To test this, it uses parallel environments to check its inventory in env_1 and env_2 while also looking around in env_3 to gather more context. This demonstrates the model's ability to perform reasoning and efficiently validate hypotheses through parallel exploration.

Keep Exploring Even After Finding a Solution The case in Figure 20 shows that even after finding the mug in environment 4, the agent keeps checking other places in parallel, just in case there's a better option or something it missed. This shows the model doesn't stop exploring too early, which helps it make better decisions in uncertain environments.

Structured Search Across Location Types The case in Figure 21 shows that after failing to find a tomato in the fridge across all environments, the agent, based on common-sense knowledge of where tomatoes might be, systematically uses each parallel environment to explore a different type of location: cabinets, drawers, and countertops. Instead of randomly trying places, it assigns each environment a distinct category to cover more ground efficiently. This demonstrates the DPEPO's ability to perform structured exploration.

B Experimental Details

B.1 SFT Training Details

Datasets For ALFWorld, we annotate 500 successful trajectories from the training split. For ScienceWorld, we adopt the same data partitioning strategy as RLVMR, which defines three difficulty levels: L0, L1, and L2. For training on L0 and L1, we annotate 1,000 trajectories from the L1 training set (used for both L0 and L1 training for simplicity). Consequently, our model faces a more challenging L0 evaluation compared to RLVMR, as it is trained with fewer L0-specific examples. For L2, we annotate 500 trajectories for training.

Experiment and Hyperparameters Details All our SFT experiments are conducted using the OpenRLHF framework. Specifically, we employ DeepSpeed ZeRO-3 with Adam optimizer offloading enabled. The model is trained with a maximum sequence length of 8,192 tokens, a learning rate of 5e-6, and bfloat16 mixed-precision, with FlashAttention-2 enabled in all experiments. The batch size is set to 4 for ALFWorld and 16 for ScienceWorld.

B.2 RL Training Details

Datasets For both ALFWorld and ScienceWorld, we select 500 samples from the training set that excludes SFT samples for RL training.

Experiment and Hyperparameters Details All RL experiments is conducted based on verl-agent framework and Ray-based dirtributed parallel environment. The maximum prompt length is set to 10,240 tokens and the maximum response length to 2,048 tokens, with each trajectory allowed up to 25 environment steps. The final actor learning rate is 1e-6, and we employ parallel trajectory-level reward that awards +1 for task success. Repetition penalties are applied consistently across both environments, with coefficients of 0.8 for depth-level action repetition and 0.95 for width-level action repetition, depth-level transition repetition, and width-level transition repetition. Notably, ScienceWorld includes an additional penalty coefficient of 0.5 for invalid actions, which is not used in ALFWorld. Both environments run with a maximum of 4 parallel exploration trajectories during training. For group-based reinforcement learning, ALFWorld adopts a group size of 8 while ScienceWorld uses a group size of 4; in both cases, 4 distinct groups

are sampled per rollout. The rollout temperature is set to 0.4, and the mini-batch size is 32. All policy updates are performed using LoRA, with the LoRA rank and scaling factor are all set to 32.

B.3 Computation Details.

For parallel environments, we cap the maximum CPU usage at 0.3. For training on ALFWorld and ScienceWorld, experiments with Qwen2.5-1.5B are conducted on 4×L40 GPUs, and those with Qwen2.5-7B on 4×H20 GPUs. Each model is trained for 1 epoch (500 samples) with 125 iterations.

C Prompt

C.1 Prompt Details for Training

System Prompt System prompt is crucial for prompt engineering. The LLM follows the instructions and assumes the role specified in the it, such as translator, programmer, etc. Therefore, as illustrated in Figure 11, we have provided a detailed description of the desired behaviors of the parallel agent within the system prompt to facilitate its parallel exploration and thereby accomplish the task better. Specifically, similar to ReAct, parallel agent is required to engage in reasoning before making a decision. For example, it needs to consider which actions can be executed in parallel to gather more information and which environments warrant further exploration. This reasoning process must be enclosed within the `<think></think>` tags. After reasoning, the agent can take actions. Unlike ReAct, the parallel agent can interact with multiple environments simultaneously. Thus, we require it to specify environment-specific actions within `<parallel></parallel>` tags, with each individual action wrapped in `<env_i><env>` tags to clearly indicate which environment the action belongs to and to enable easier parsing of the actions.

Prompt for First Step At the beginning of a task, the agent has not yet interacted with the environment. The environment provides only initial state information, a set of available actions, and the task description. We organize this information into a concise yet clear prompt (As shown in Figure 12) to help the agent understand the initial context. Additionally, for ALF-World, we do not include explanations for the available actions. However, for the ScienceWorld dataset, which involves combinations of actions and objects, we introduce brief action descriptions to avoid excessive token consumption.

Prompt for Intermediate Step As the number of parallel agent exploration steps increases, maintaining the full history of interactions becomes costly in terms of both memory and computation. To address this, at step t, for information from steps 1 to $t-2$, we retain only the actions and their corresponding observations within {history_info}. In contrast, for the immediate previous step $t-1$, we preserve all of the parallel actions, observations, and executable actions within {last_history}. The details of {history_info} and {last_history} are presented in Figure 14 and Figure 15. With this carefully designed prompt presented in Figure 13, the context length is reduced while still retaining sufficient information for the agent to perform its task more effectively.

Exploration-limiting prompt For experiments that require controlling the number of parallel environments, we designed a prompt to limit the maximum number of environments the model can explore. In this prompt, we specify the allowed number of environments using "env_num" and inform the model that if it attempts to explore more than this number, the entire task will be considered a failure. In addition to this prompt-based restriction, we also enforce behavioral constraints: if the model tries to access an environment that is not permitted, the entire task is immediately seen as failed.

Action-Explanation Prompt Since the tasks in ScienceWorld require actions to interact with objects and involves a large number of both actions and objects, it is impractical to include all possible object-action combinations in the limited context. Therefore, we adopt the same prompt as RLVMR, which provides only a list of executable actions along with explanations of their specific details and the objects they can interact with. This prompt is illustrated in Figure 17.

C.2 Effectiveness of Designed Prompts

Token Budget Analysis When tackling long-horizon tasks, traditional methods typically append all historical information and intermediate reasoning steps into a single context window. However, as the number of steps increases, the context continuously expands. In this section, we evaluate the effectiveness of our designed prompt in compressing the context window. As illustrated in Figure 10,

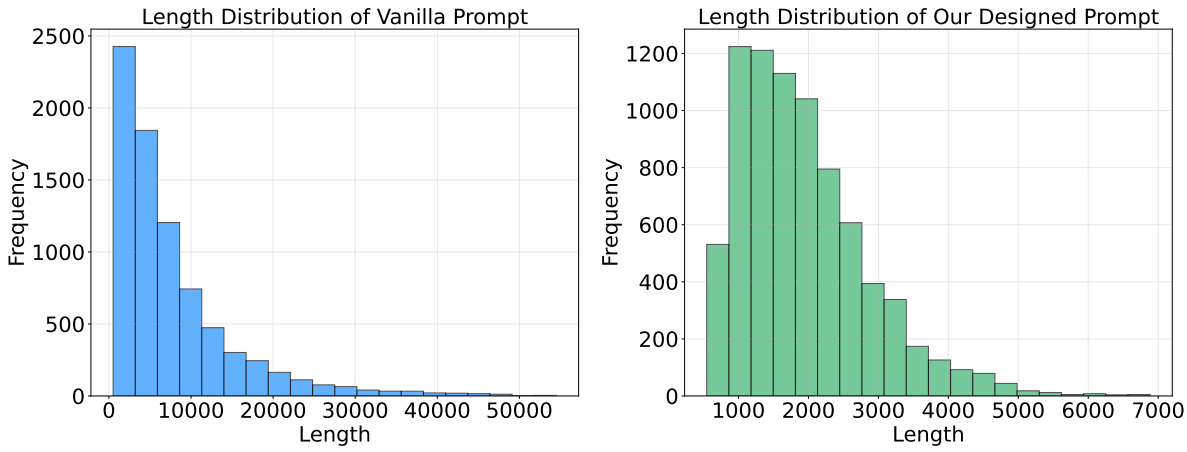

Figure 10: Context length distribution for the vanilla prompt (left) and our designed prompt (right).

our method can compress prompts that originally contain at most 50,000 tokens down to at most 7,000 tokens, significantly reducing the token budget while still maintaining the agent's strong performance.

Comparison of Context Management Strategies GiGPO employs a similar prompting strategy, but instead of retaining all historical information, it only preserves the most recent few steps within a limited context window. We evaluate GiGPO's performance with varying context window sizes, as shown in Table 5. Although the context length may be shorter, its performance does not improve as the context window expands. Furthermore, once the number of retained steps exceeds 16, the total context length will surpasses the model's maximum sequence length, leading to truncation.

In contrast, our proposed prompting design not only preserves more comprehensive historical information but also adheres to the token budget. Furthermore, as demonstrated in Section 5.2 and Section 5.5, under our prompting scheme, the performance of our method consistently improves as the number of environments increases, highlighting its efficiency, robustness and scalability.

D Pseudo Code for DPEPO

To clearly illustrate the overall pipeline of our proposed DPEPO, this section provides the detailed algorithmic pseudo code, as shown in Table 1.

Validating Hypotheses via Parallel Exploration As the case shown in Figure 19, when the agent repeatedly fails to take book 2 from the bed, it hypothesizes that there might be a carrying limit. To test this, it uses parallel environments to check its inventory in env_1 and env_2 while also looking around in env_3 to gather more context. This demonstrates the model's ability to perform reasoning and efficiently validate hypotheses through parallel exploration.

System Prompt

You are an expert agent operating in the {ALFRED / ScienceWorld} embodied environment. Given a task, you need to reason first in your mind. Your reasoning process must be enclosed within <think> </think> tags, for example: <think> reasoning process here </think>.
After thinking, you may take actions. You can either explore multiple parallel environments with multiple actions or take an action in a specific environment. At the very beginning, every environment have the same status, but each environment is independent, they do not share state changes after actions are taken. So, parallel actions are executed simultaneously across different environments. The parallel actions are not carried out sequentially. You must wrap each action in specific environment tags like `<env_i> ... </env_i>` to indicate which environment you are acting in.
To take multiple actions at the same time in different environments, use the `<parallel> ... </parallel>` tags and wrap each action within its corresponding `<env_i> ... </env_i>` tag, where i refers to the i-th environment:

```
<parallel>
<env_1> possible action 1 </env_1>
...
<env_i> possible action 2 </env_i>
</parallel>
```

Your output must follow the rules above.

Figure 11: The system prompt template.

Prompt for First Step

You are an expert agent operating in the {ALFRED / ScienceWorld} Embodied Environment.
Your task is to: `{task_description}`
Your current observation is: `{current_observation}`
Your admissible actions in the current situation are: `{admissible_actions}`.
Your output must follow the rules above.

Figure 12: The prompt template at the beginning of a task.

Prompt for Intermediate Step

You are an expert agent operating in the {ALFRED / ScienceWorld} Embodied Environment.
Your task is to: `{task_description}`.
Your initial observation is: `{initial_observation}`.
`{history_info}`.
In your last step, your actions, corresponding observations, and admissible actions are: `{last_history}`

Figure 13: Careful designed prompt for intermediate step.

history_info

You have already taken multiple actions in multiple parallel environments. Below are the most recent observations and the corresponding actions you took:
In Environment 1
Action 1: {action_1}
Observation 1: {obs_1}
Action 2: {action_2}
Observation 2: {obs_2}

In Environment 2
Action 1: {action_1}
Observation 1: {obs_1}
Action 2: {action_2}
Observation 2: {obs_2}
Action 3: {action_3}
Observation 3: {obs_3}

Figure 14: Historical information used in the "Prompt for Intermediate Step" demonstration.

last_history

In Environment 1
Action 1: {action_1}
Observation 1: {obs_1}
Action 2: {action_2}
Observation 2: {obs_2}
Next Possible Actions: {poa_2}

In Environment 2
Action 1: {action_1}
Observation 1: {obs_1}
Next Possible Actions: {poa_1}

Figure 15: Last step information used in the "Prompt for Intermediate Step" demonstration.

Exploration_limiting Prompt

You can explore up to {env_num} different environments, ranging from 1 to {env_num}. If you explore more than {env_num} parallel environments, the task seens failed.

Figure 16: Prompt for limiting the model to exploring at most `{env_num}` environments.

Action Explanation Prompt for ScienceWorld

Here the explanations of actions you may take:
{{"action": "open OBJ", "description": "open a container"}},
{{"action": "close OBJ", "description": "close a container"}},
{{"action": "activate OBJ", "description": "activate a device"}},
{{"action": "deactivate OBJ", "description": "deactivate a device"}},
{{"action": "connect OBJ to OBJ", "description": "connect electrical components"}},
{{"action": "disconnect OBJ", "description": "disconnect electrical components"}},
{{"action": "use OBJ", "description": "use a device/item"}},
{{"action": "look around", "description": "describe the current room"}},
{{"action": "look at OBJ", "description": "describe an object in detail"}},
{{"action": "look in OBJ", "description": "describe a container's contents"}},
{{"action": "read OBJ", "description": "read a note or book"}},
{{"action": "move OBJ to OBJ", "description": "move an object to a container"}},
{{"action": "pick up OBJ", "description": "move an object to the inventory"}},
{{"action": "put down OBJ", "description": "drop an inventory item"}},
{{"action": "pour OBJ into OBJ", "description": "pour a liquid into a container"}},
{{"action": "dunk OBJ into OBJ", "description": "dunk a container into a liquid"}},
{{"action": "mix OBJ", "description": "chemically mix a container"}},
{{"action": "go to LOC", "description": "move to a new location"}},
{{"action": "eat OBJ", "description": "eat a food"}},
{{"action": "flush OBJ", "description": "flush a toilet"}},
{{"action": "focus on OBJ", "description": "signal intent on a task object"}},
{{"action": "wait", "description": "take no action for 10 iterations"}},
{{"action": "wait1", "description": "take no action for 1 iteration"}},
{{"action": "task", "description": "describe current task"}},
{{"action": "inventory", "description": "list your inventory"}}

Figure 17: Action explanation prompt used in ScienceWorld, listing all valid environment interaction commands and their natural-language descriptions.

Algorithm 1 Training LLM Agents with DPEPO

Require: Initial policy $\pi_{\theta_{\text{old}}}$, task distribution $p(X)$, group size N, number of parallel environment K
for each training iteration **do**
    Update the old policy model: $\theta_{\text{old}} \leftarrow \theta$
    // Multi-turn parallel rollout stage
    Sample task $x \sim p(X)$ and initialize N parallel environment workers, each worker manages K identical environments.
    for $t = 1$ to T **do**
        Sample parallel actions $\left\{A_{i,t} \sim \pi_{\theta_{\text{old}}}\right\}_{i=1}^{N}$
        Execute parallel actions , observe parallel observations $\{S_{i,t+1}\}_{i=1}^{N}$
    end for
    // Advantage computation stage
    Compute trajectory relative advantages $\Phi_{traj}(\tau_i)$ via Equation (7)
    Compute step advantages $\Phi_{step}(A_{i,t})$ via Equation (8)
    // Policy update phase
    Combine these advantages: $\Phi(A_{i,t}) = \Phi_{step}(A_{i,t}) \cdot \Phi_{traj}(\tau_i)$
    Update policy θ by maximizing objective same as GRPO
end for

Method	Window Size	ALFWorld						
		Pick	Cool	Heat	Look	Pick2	Clean	All
GiGPO	2	80.0	72.0	87.5	84.6	79.2	74.1	78.6
	4	**91.4**	**84.0**	**93.8**	84.6	**95.8**	92.6	**90.7**
	6	88.6	**84.0**	**93.8**	**92.3**	79.2	92.6	87.9
	8	74.3	80.0	87.5	**92.3**	79.2	92.6	82.9
	10	80.0	80.0	**93.8**	76.9	75.0	92.6	82.9
	12	88.6	80.0	**93.8**	76.9	70.8	**96.3**	85.0

Table 5: Performance of GiGPO with different context window sizes on ALFWorld. **Bold scores** indicate the best performance, while underlined scores represent the second-best.

A Case for Efficient Search via Parallel Actions

User:
Your task is to: put some candle on toilet. Your current observation is: -= Welcome to TextWorld, ALFRED! =-
You are in the middle of a room. Looking quickly around you, you see a bathtubbasin 1, a garbagecan 1, a handtowelholder 1, a shelf 3, a shelf 2, a shelf 1, a sinkbasin 1, a toilet 1, a toiletpaperhanger 1, and a towelholder 1.
Your admissible actions of the current situation are: ['go to bathtubbasin 1', 'go to garbagecan 1', 'go to handtowelholder 1', 'go to shelf 1', 'go to shelf 2', 'go to shelf 3', 'go to sinkbasin 1', 'go to toilet 1', 'go to toiletpaperhanger 1', 'go to towelholder 1', 'help', 'inventory', 'look'].
Assistant:

<think>

My task is to put some candle on toilet. First, I need to find a candle. Looking at the objects in the room, I don't see a candle mentioned in the current observation. The objects are: bathtubbasin 1, garbagecan 1, handtowelholder 1, shelf 3, shelf 2, shelf 1, sinkbasin 1, toilet 1, toiletpaperhanger 1, and towelholder 1.
Candles are often found on shelves or in other storage areas. I should check the shelves first since they are the most likely places to contain items like candles. There are three shelves: shelf 1, shelf 2, and shelf 3.
Since I can explore up to 5 different environments, I can check multiple shelves simultaneously to speed up the search for the candle. I'll start by going to shelf 1, shelf 2, and shelf 3 in parallel environments.
In each environment, after going to the shelf, I'll need to look around to see if there's a candle there. But for now, I'll just navigate to the shelves first.

```
</think>
<parallel>
<env_1> go to shelf 1 </env_1>
<env_2> go to shelf 2 </env_2>
<env_3> go to shelf 3 </env_3>
</parallel>
```

Figure 18: A Case for Efficient Search via Parallel Actions.

A Case for Validating Hypotheses via Parallel Exploration

User: You are an expert agent operating in the ALFRED Embodied Environment.
Your task is to: put two book in desk.
Your initial observation is: -= Welcome to TextWorld, ALFRED! =-
You are in the middle of a room. Looking quickly around you, you see a bed 1, a desk 1, a drawer 6, a drawer 5, a drawer 4, a drawer 3, a drawer 2, a drawer 1, a garbagecan 1, a shelf 2, a shelf 1, and a sidetable 1.. You have already taken multiple actions in multiple parallel environments. Below are the most recent observaitons and the corresponding actions you took:
...
In the past few steps, the agent successfully picked up book 1 in environment 2, but has repeatedly failed to take book 2 from the same bed.
Assistant:

```
<think>
```

It seems like I am having trouble taking book 2 from bed 1 in environment 2. The action "take book 2 from bed 1" keeps resulting in "Nothing happens." This might be because I can only carry one item at a time, or there might be another issue.
Let me check the inventory in environment 2 to see what Iḿ currently carrying. I should also check if there are any other books available in the room that I could use as alternatives.
Since I already have book 1, I should first put it down somewhere, then go back to get book 2. Or maybe I need to go to the desk with book 1 first, then come back for book 2.
Let me check the inventory in environment 2 first to understand the current state.

```
</think>
<parallel>
<env_1> inventory </env_1>
<env_2> inventory </env_2>
<env_3> look </env_3>
</parallel>
```

Figure 19: A Case for Validating Hypotheses via Parallel Exploration.

A Case for Keep Exploring Even After Finding a Solution

User: You are an expert agent operating in the ALFRED Embodied Environment.
Your task is to: put a cool mug in coffeemachine.
Your initial observation is: -= Welcome to TextWorld, ALFRED! =-
You are in the middle of a room. Looking quickly around you, you see a cabinet 27, a cabinet 26, a cabinet 25, a cabinet 24, a cabinet 23, a cabinet 22, a cabinet 21, 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 coffeemachine 1, a countertop 2, a countertop 1, a diningtable 1, a drawer 12, a drawer 11, a drawer 10, a drawer 9, a drawer 8, a drawer 7, a drawer 6, a drawer 5, a drawer 4, a drawer 3, a drawer 2, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1. In your last step, your actions, corresponding observations and admissible actions are:
...
Assistant:

```
<think>
```

Perfect! Iḿ still at countertop 2 in environment 4 where I found the mug. Now I need to take the mug and then put it in the coffeemachine. The task requires a "cool mug" though, so I need to cool it down first. I should take the mug and then go to the fridge to cool it.
Let me take the mug from countertop 2 in environment 4, and continue exploring other environments in parallel.

```
</think>
<parallel>
<env_1>open cabinet 2</env_1>
<env_2>open cabinet 11</env_2>
<env_3>go to drawer 5</env_3>
<env_4>take mug 1 from countertop 2</env_4>
<env_5>open cabinet 21</env_5>
</parallel>
```

Figure 20: A Case for Keep Exploring Even After Finding a Solution.

A Case for Structured Search Across Location Types

User: You are an expert agent operating in the ALFRED Embodied Environment.
Your task is to: cool some tomato and put it in microwave.
Your initial observation is: -= Welcome to TextWorld, ALFRED! =-
You are in the middle of a room. Looking quickly around you, you see a cabinet 26, a cabinet 25, a cabinet 24, a cabinet 23, a cabinet 22, a cabinet 21, 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 coffeemachine 1, a countertop 3, a countertop 2, a countertop 1, a drawer 12, a drawer 11, a drawer 10, a drawer 9, a drawer 8, a drawer 7, a drawer 6, a drawer 5, a drawer 4, a drawer 3, a drawer 2, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1.
...
The agent went to and opened the fridge in all 5 environments, but found it empty, no tomato located yet.
Assistant:

<think>

The fridge is empty in all environments. I need to search other locations for a tomato. Tomatoes might be in cabinets, drawers, or on countertops. I should search systematically. Since I have multiple environments, I can search different locations simultaneously.
I'll use the environments to search different types of locations: - Environment 1: Search a cabinet (e.g., cabinet 1)
- Environment 2: Search a drawer (e.g., drawer 1)
- Environment 3: Search countertop 1
- Environment 4: Search countertop 2
- Environment 5: Search countertop 3
This will help me find the tomato faster by searching multiple locations at once.

```
</think>
<parallel>
<env_1> go to cabinet 1 </env_1>
<env_2> go to drawer 1 </env_2>
<env_3> go to countertop 1 </env_3>
<env_4> go to countertop 2 </env_4>
<env_5> go to countertop 3 </env_5>
</parallel>
```

Figure 21: A Case for Structured Search Across Location Types.