ProCeedRL: Process Critic with Exploratory Demonstration Reinforcement Learning for LLM Agentic Reasoning

Jingyue Gao[1,2], **Yanjiang Guo**[1,2], **Xiaoshuai Chen**[3], **Jianyu Chen**[1,2]
[1]Institute for Interdisciplinary Information Sciences, Tsinghua University,
[2]Shanghai Qizhi Institute, [3]Independent Researcher
gaojy23@mails.tsinghua.edu.cn

Abstract

Reinforcement Learning (RL) significantly enhances the reasoning abilities of large language models (LLMs), yet applying it to multi-turn agentic tasks remains challenging due to the long-horizon nature of interactions and the stochasticity of environmental feedback. We identify a structural failure mode in agentic exploration: suboptimal actions elicit noisy observations into misleading contexts, which further weaken subsequent decision-making, making recovery increasingly difficult. This cumulative feedback loop of errors renders standard exploration strategies ineffective and susceptible to the model's reasoning and the environment's randomness. To mitigate this issue, we propose **ProCeedRL**: **Proc**ess **C**ritic with **E**xplorative **D**emonstration RL, shifting exploration from passive selection to active intervention. ProCeedRL employs a process-level critic to monitor interactions in real time, incorporating reflection-based demonstrations to guide agents in stopping the accumulation of errors. We find that this approach significantly exceeds the model's saturated exploration performance, demonstrating substantial exploratory benefits. By learning from exploratory demonstrations and on-policy samples, ProCeedRL significantly improves exploration efficiency and achieves superior performance on complex deep search and embodied tasks[1].

1 Introduction

Large Language Models (LLMs) like DeepSeek-R1 (DeepSeek-AI, 2025) exhibit exceptional reasoning capabilities, driven by Reinforcement Learning with Verifiable Rewards (RLVR) (Shao et al., 2024; DeepSeek-AI, 2025). This paradigm, which optimizes LLMs with reward based on the correctness of the final output, has proven effective for single-turn problems such as mathematics (OpenAI et al., 2024; Team et al., 2025), and agentic reasoning tasks like tool-integrated reasoning recently (Wang et al., 2025a; Feng et al., 2025a; Song et al., 2025; Jin et al., 2025; Singh et al., 2025).

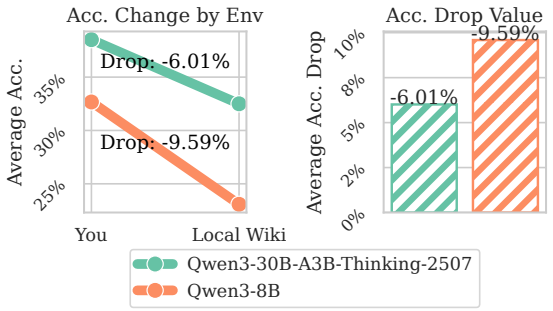

Figure 1: In multi-turn agentic tasks, we find that noisy environmental feedback in the context can degrade a model's reasoning ability (left), with weaker models being affected more severely (right). These two phenomena make recovery in multi-turn interactions extremely difficult, as accumulated contextual noise rapidly compounds and quickly weakens the model.

However, multi-turn agentic reasoning introduces additional challenges for standard RLVR exploration due to its long-horizon, stochastic nature. Agents must continuously interact with stochastic environments, in which the history of feedback is incorporated into the context for subsequent reasoning. Such a workflow creates a high-stakes dependency between agents and environments. Suboptimal actions, such as a vague search query due to limited agent capability, elicit irrelevant or misleading feedback from the environment, which is then added to the context. The accumulated context noise, in turn, influences and degrades subsequent reasoning and action quality. This results in a vicious circle in agentic exploration. Any suboptimal action or environmental stochasticity may trigger and progressively reinforce this loop of accumulated error, making recovery increasingly difficult. Therefore, the upper bound of standard RLVR exploration achieved through repeated sampling is limited by the agents and environments.

[1]Code is open-sourced at this repository.

Findings of the Association for Computational Linguistics: ACL 2026, pages 16281–16295

To analyze how noise in context affects model capability, we compare different models on search-augmented question answering under varying levels of environmental noise. Fig. 1 shows that both models are sensitive to environment feedback and experience a performance drop in the noisier environment, suggesting the impact of observation quality on agent reasoning. Notably, we find that the weaker model exhibits a greater decline in performance. The amplified degradation indicates that limited reasoning capabilities exacerbate the impact of noisy observations. The adverse effects of suboptimal actions accumulate through the feedback loop between the agent and the environment, making recovery difficult. This validates the susceptibility of vanilla agentic exploration to agents and environments, underscoring its importance for agentic reasoning.

To mitigate this problem, we propose **ProCeedRL**: **Pro**cess **C**ritic with **E**xplorative **D**emonstration **RL**.Rather than waiting to penalize negative trajectories after collecting repeated samples, ProCeedRL employs a process-level critic to detect suboptimal steps in real time. By rewinding the agent to retake the step with a refined demonstration before proceeding, ProCeedRL prunes invalid actions and breaks the vicious circle of accumulating error between actions and environments. It addresses the vicious circle in exploration and enables the model to learn beyond its inherent exploration limit. Furthermore, learning from these refined demonstrations at negative steps embeds this knowledge into the model's inherent capabilities, thereby eliminating the need for additional pipelines at test time.

Several related works also share a similar idea of providing process supervision during agentic reasoning from different viewpoints. Wang et al. (2025b); Deng et al. (2025) introduce process-level rewards based on extracted documents or fine-grained actions, which requires costly data labeling and reward design. Another line of concurrent work (Li et al., 2025; Fu et al., 2025) proposes different formulations of self-correcting search pipelines, including self-proposed sub-goals or an agent-invoking judging pipeline. However, models are trained to use these frameworks at test time, whereas our method internalizes the critique information into the model's inherent capabilities.

We conduct comprehensive experiments and analyses in agentic reasoning to validate the effectiveness of our method. Compared to baselines, ProCeedRL significantly improves exploration efficiency and surpasses the model's upper bound in standard RLVR. In summary, our contributions are as follows:

- Identifies the cumulative interaction between suboptimal actions and environmental noise, which mutually amplify into a vicious cycle that renders standard exploration inefficient;
- Proposes ProCeedRL, a framework that shifts from passive selection to active intervention by using process critics to rewind and refine errors;
- Show the effectiveness of ProCeedRL in improving exploration and surpassing the reasoning limit on deep search and embodied tasks.

2 Related Works

Reinforcement Learning in LLMs Reinforcement Learning with Verifiable Rewards (RLVR) excels in enhancing the reasoning abilities of LLMs on complex tasks like mathematical reasoning and code generation (DeepSeek-AI, 2025; Shao et al., 2024; Team, 2025b). Recent work (Wang et al., 2025a; Feng et al., 2025b,a; Chen et al., 2025) extends RLVR with outcome rewards to improve agentic reasoning, such as deep search and tool use. However, these methods rely on independently repeated generation, which limits exploration efficiency and reasoning upper bound constrained by the base model (Yue et al., 2025). ProCeedRL improves exploration by actively leveraging the critic and refining actions at the step level, thereby surpassing an agent's inherent limits.

Process Supervision for Reasoning Process supervision yields dense rewards and better credit assignment for LLM reasoning. Lightman et al. (2024); Zhang et al. (2025b); Wang et al. (2023) propose and discuss how to develop process reward models. Another line of work (Dong et al., 2025; Kazemnejad et al., 2024; Gao et al., 2025; Chen et al., 2025; Wang et al., 2025b; Deng et al., 2025) uses carefully designed rewards or pipelines to provide process supervision for reasoning without PRMs. Among them, some concurrent works (Li et al., 2025; Fu et al., 2025) design test-time reflection pipelines with specially designed rewards to encourage it. In contrast, ProCeedRL incorporates refined demonstrations during training that serve as implicit guidance, avoiding the cumbersome design of rewards or additional overhead at test time.

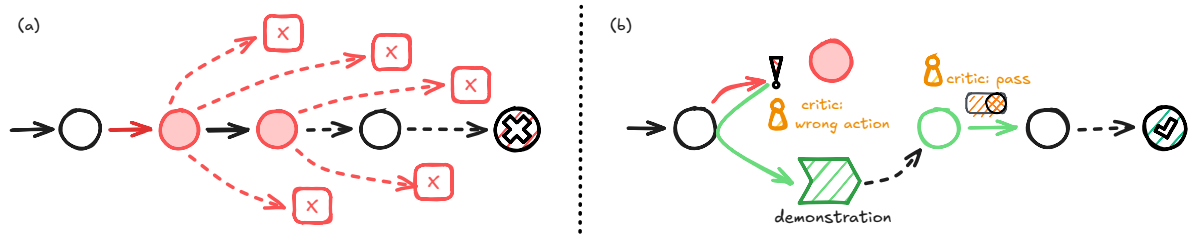

Figure 2: Comparison between standard independently repeated sampling (a) and ProCeedRL rollouts (b). **Left**: In vanilla exploration, the model's suboptimal action may result in irrelevant or misleading observations, which hinder all subsequent reasoning and the exploration of correct samples. **Right**: In ProCeedRL, a critic actively monitors the planning process. When an adverse action is detected due to faulty reasoning or low-quality returned observations, a refined demonstration replaces it to guide the agent out of the vicious circle in (a) and mitigate the exploration issue.

3 Method

In this section, we introduce our method, beginning with the formulation of the agentic reasoning problem in Sec. 3.1. We then analyze a key problem in agentic reasoning in Sec. 3.2 and present our solution with the ProCeedRL framework in Sec. 3.3. Sec. 3.4 outlines the overall training pipeline of our method, additionally with an overview and a case example in Fig. 3 and the algorithm in Alg. 1.

3.1 Preliminaries

Problem Settings Given a multi-turn reasoning task s_0 in environments, we define a trajectory as $\tau = (s_0, a_0, r_0, \cdots)$, where both s_t and a_t are textual strings representing the agent's observed state and action, respectively. At each timestep t, the LLM agent π_θ, receives the observed state s_t and produces action a_t according to policy $\pi_\theta(\cdot|\tau_t)$. The environment yields the next state s_{t+1} and a reward $r_t \in \mathbb{R}$ based on a_t. We adopt an outcome-verified reward setting: the environment assigns a terminal reward of $r(\tau) = 1$ upon successful completion (e.g., a correct answer or successful task completion), and $r_t = 0$ for all intermediate steps.

LLM RL Algorithms We optimize policy π_θ to maximize the expected reward $\mathbb{E}_{\tau\sim\pi\theta(\tau)}r(\tau)$ using RL algorithms like GRPO (Shao et al., 2024) and DAPO (Yu et al., 2025). While these methods leverage group-based relative advantage for stability, they rely on vanilla repeated sampling for exploration. Building on advances in RLVR, our work addresses the exploration challenge in self-generated rollouts by incorporating process-level critic and refinement signals into the rollout.

3.2 Vicious Circle in Agentic Reasoning

We identify a critical problem in agentic reasoning: a downward spiral between suboptimal actions and misleading environment observations. Since agent context is cumulative of actions and observations, it creates a dependency between the agent's actions and the observations it receives. Any errors in actions, due to incorrect reasoning or hallucination, introduce misleading feedback into the context. The "poisoned" context, in the meantime, derails subsequent reasoning and leads to suboptimal actions. It reinforces a positive feedback loop that progressively degrades final performance, which is inevitable given the agent's reasoning ability and the environment's randomness.

To validate this hypothesis, we conduct an illustrative experiment (Fig. 1) comparing two models with distinct reasoning capabilities, `Qwen3-8B` and `Qwen3-30B-A3B-Thinking-2507`, on search-augmented QA. We simulate varying environmental noise by using different search engines: one is a commercial search engine, You[2], while the noisier one is a local dense retriever based on the Wikipedia corpus[3]. We evaluate the average accuracy of both models on Bamboogle, Frames, and WebwalkerQA across the two environments.

The findings first validate the premise that a noisy context degrades the model's reasoning, as both models suffer performance drops with the local retriever across three benchmarks. In particular, the weaker reasoner (`Qwen3-8B`) exhibits a more pronounced decrease. This susceptibility confirms that environmental noise amplifies action errors, corroborating the existence of a vicious circle between suboptimal actions and corrupted reasoning contexts. To mitigate this, we employ a critic to monitor suboptimal actions and refine them in real time to avoid adverse effects, thereby improving exploration efficiency and outcomes.

[2]https://you.com/

[3]https://huggingface.co/datasets/PeterJinGo/wiki-18-corpus

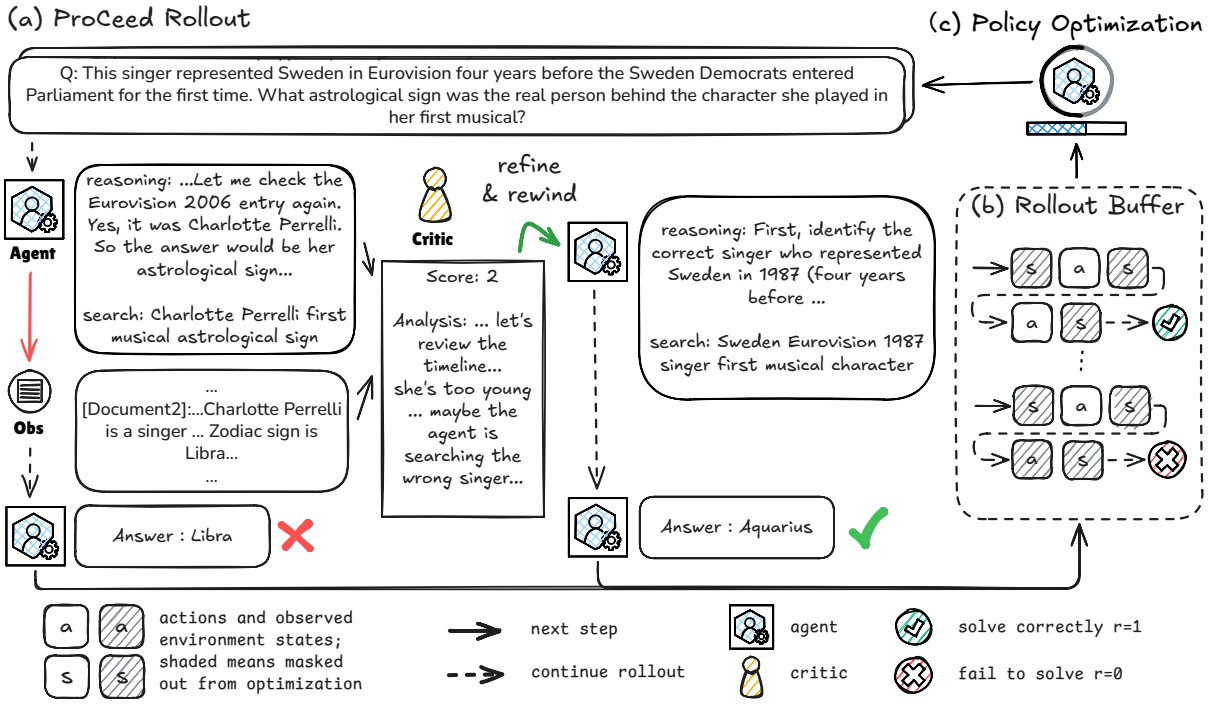

Figure 3: The overall workflow of our method with an example. **(a)** When the agent makes a suboptimal action, the observation reinforces this path and derails subsequent model reasoning, resulting in low exploration efficiency (Sec. 3.2). We employ a process-level critic to identify flawed steps, in which the agent refines its actions and reruns the adverse actions, thereby breaking the vicious circle and improving exploration and reasoning limits (Sec. 3.3). **(b)** After collecting trajectories with ProCeed rollout, we incorporate them with directly generated samples to form meaningful groups for subsequent policy optimization in **(c)**, as detailedly described in Sec. 3.4.

3.3 ProCeed Rollout

In this section, we present the core design of the ProCeedRL rollout phase, illustrated in Fig. 2. We detail how our method identifies and rectifies errors in real time, thereby preventing error propagation.

Process Level Critic To provide step-granular supervision and intercept potential reasoning errors, we employ a critic ϕ to evaluate the quality of each intermediate step. Formally, at each timestep t, we ask the critic to output an integer scalar score l_t and a textual critique c_t, based on both history τ and the latest interaction a_t, s_{t+1}: $l_t, c_t = \phi(\tau_t, a_t, s_{t+1})$.

By incorporating the subsequent observation s_{t+1}, the critic can ground its evaluation in the action's actual effect rather than just its plausibility. Specifically, for irreversible environments, such as the real world, we evaluate a_t directly without s_{t+1}. An action a_t is deemed adverse if its score falls below a specific threshold, $l_t \leq l_{\text{th}}$. In practice, we calibrate l_{th} dynamically across tasks according to the critic and agent's behavior, as discussed in the ablation in Sec. 4.3.2.

Refined Exploratory Demonstration Upon detecting an adverse step, we trigger an intervention to break the "vicious cycle" of misleading observations. Instead of allowing the agent to proceed with a suboptimal action a_t, we perform an on-the-fly rectification. We utilize a refining policy μ to generate a refined action a'_t, conditioned on the critic's specific feedback: $a'_t = \mu(\tau_{t-1}, a_t, l_t, c_t)$.

We then replace the original action a_t with the refined demonstration a'_t. They help the agent avoid suboptimal actions and break the vicious cycle of potentially misleading contexts, thereby exhibiting exploratory benefits. Crucially, our framework is model-agnostic: while ϕ and μ can be instantiated with strong external models to maximize supervision quality, we empirically find that using the policy model π_θ itself is also highly effective. This enables ProCeedRL to operate as a scalable, self-contained pipeline that is independent of external knowledge. Prompts for both the critic and refinement modules are detailed in Appendix E.

3.4 ProCeedRL training

Training ProCeedRL The training process for ProCeedRL, depicted in Fig. 3, extends the standard group-based RLVR framework. We augment each data group in the replay buffer with critic

scores and refined demonstrations, while also including trajectories sampled directly from the current policy. These direct samples serve as negative and on-policy references, which are crucial for group-based RLVR algorithms. The model is subsequently trained on this buffer using an optimization algorithm such as SFT or DAPO. Notably, we mask the demonstration steps in trajectories that fail during optimization since these demonstrations are off-policy relative to the model. Reducing their probability may cause instability and confusion during optimization.

Policy Optimization The demonstration steps collected through refinement do not align with the current policy distribution, resulting in a distributional shift between the training data and the target policy. Directly incorporating misaligned demonstrations d_i with on-policy samples o_i into the RL objective negatively affects performance (Yan et al., 2025; Zhang et al., 2025a). We adopt the method chord-ϕ (Zhang et al., 2025a), which applies a coefficient to weight the demonstration loss, down-weighting both highly probable and unlikely demonstrations. Specifically, for a rollout buffer $\mathcal{B} = \{o_i\} \cup \{d_j\}$, the optimization object in our method when using DAPO is:

$$\mathbb{E}_{x\sim\mathcal{B},\{o_i\}\sim\pi_{\theta_{\text{old}}},\{d_j\}\sim\mu}\left[\frac{1}{|G|}\sum_{i=1}^{|G|}\sum_{t=1}^{|d_i|}\sigma(d_{j,t})\hat{A}_{i,t}\min\right.$$
$$\left.\left(\text{is}_{i,t}(\theta),\ \text{clip}\Big(\text{is}_{i,t}(\theta), 1-\varepsilon_{\text{low}}, 1+\varepsilon_{\text{high}}\Big)\right)\right], \tag{1}$$

where $A_i = \frac{r_i - \text{mean}(r_i)}{\text{std}(r_i)}$ is the relative advantage for response i, $\text{is}_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t}|x,o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|x,o_{i,<t})}$ is the importance sampling ratio. The coefficient $\sigma(d_{j,t}) = \pi_\theta(d_{j,t}|x,d_{j,<t}) \cdot (1-\pi_\theta(d_{j,t}|x,d_{j,<t}))$ stabilizes training with demonstration, and we set it to be 1 for all on-policy steps. We summarize our method as pseudocode in Alg. 1.

4 Experiments

In this section, we design extensive experiments to investigate:

1. Does ProCeedRL improve the overall performance over baselines when variables are controlled (Sec. 4.2)?
2. What are the benefits of introducing ProCeedRL during test time and training (Sec. 4.3.1)?
3. How do the key design factors affect the performance of ProCeedRL(Sec. 4.3.2)?

4.1 Experiment Settings

Datasets and Benchmarks We assess ProCeedRL on two types of challenging agentic reasoning tasks: deep search QA and embodied agents.

In deep search QA, or search augmented QA, LLMs must automatically plan and invoke a search tool to find a final answer to the QA across multiple turns. Building on HotpotQA (Yang et al., 2018), we first categorize the queries in its training set by topic. Then, we obtain a curated training subset of 4000 question-answer pairs with balanced difficulties and topics as our training set. We evaluate our model on several challenging question answering benchmarks: MuSiQue (Trivedi et al., 2022), WebWalkerQA (Wu et al., 2025), GAIA (Mialon et al., 2024), Frames (Krishna et al., 2024), and Bamboogle (Press et al., 2023).

We also evaluate our method on the embodied benchmark ALFWorld (Shridhar et al., 2021). ALFWorld comprises a broad range of household settings in which LLMs must plan with a long horizon to solve different tasks. It includes 3553 training settings, covering various household configurations and instructions, and we evaluate on both in- and out-of-distribution test set splits. More details on the artifacts are provided in Appx. D.

Implementation Our implementation is based on verl (Sheng et al., 2024) and rllm (Tan et al., 2025). We adapt the ReAct (Yao et al., 2023) prompting framework. For deep search QA, we formulate searching and answering as function calls. We use the search engine You, and utilize the top 3 results' snippets as responses to each query. To improve the model's multi-turn function-calling ability, we cold-start with samples generated by `DeepSeek-V3-0528`, which is utilized as the critic in our method. We adopt the AlfWorld formulation as in Feng et al. (2025b) and use the model itself as the critic. We instruct both critics to output an integer rating between 0 and 10 in a zero-shot manner, with a threshold of 3.

We train our model using DAPO (Yu et al., 2025) with a batch size of 32 and a group size of 8. Half of each group is collected via ProCeedRL, with the other half collected directly. For Qwen3-8B on AlfWorld, we additionally train our method with SFT on the correct samples collected by our method during rollout. More details and hyperparameters are provided in Appx. C.

	Bamboogle	MuSiQue	Frames	GAIA	WebwalkerQA
*related works**					
GiGPO (2025b)	68.90%	18.90%	-	-	-
DeepResearcher (2025)	72.80%	29.30%	-	-	-
ARPO(Deep Research) (2025)	58.40%	24.96%	41.18%	12.52%	21.39%
backbone: Qwen3-8B					
ReAct Prompting	50.93%	18.64%	31.92%	9.69%	18.62%
Qwen3-8B-v3-SFT	62.13%	20.07%	37.50%	10.51%	<u>19.85%</u>
+RFT	64.27%	22.49%	40.65%	**14.98%**	19.56%
+DAPO/Search-R1	<u>70.83%</u>	<u>23.60%</u>	<u>43.59%</u>	10.10%	19.51%
+ProCeedRL	**73.87%**	**29.52%**	**46.42%**	<u>13.79%</u>	**23.01%**
ablation: Rewinding More	65.87%	19.11%	40.13%	10.10%	20.16%

Table 1: Performance of different methods on deep search tasks. **Bold** and <u>underlined</u> results represent the best and the second best on each benchmark in our controlled experiments. We report the average accuracy of three runs. Our method outperforms baseline methods across most benchmarks, particularly on challenging ones such as MuSiQue. *For related works, we cite the metrics reported or run open-sourced models (if available) in our settings only for reference, as tool configurations like search engines differ across different works, which affect final performance.

Models and Baselines We conduct experiments on `Qwen3-1.7B/8B` (Team, 2025a), and compare ProCeedRL with the following baselines:

1. **SFT-only**: The model fine-tuned with samples collected from the critic's reasoning, to rule out the effect of knowledge distillation. Additionally, we train the model using RFT (Yuan et al., 2023) on the correct data via rejection sampling from expert-generated trajectories; each group has the same size as in our method.
2. **RL algorithms**: We compare with LLM-based agents trained with standard exploration RL, like GRPO and DAPO. In particular, we adjust the group size for each prompt to align the computational requirements for generation across methods. This is equivalent to several related works (Song et al., 2025; Feng et al., 2025a) that first successfully apply RLVR to agent domains.
3. **Other baselines**: We also include several other related works (Feng et al., 2025a; Zheng et al., 2025; Dong et al., 2025) on LLM agent reasoning for reference, as shown in Tab. 1.

4.2 Results

Metrics We employ the LLM-as-judge approach with `gpt-4o` to determine the correctness of generated answers for the deep search QA task, which we empirically find highly aligns with human judgment and gold reward. For the embodied AlfWorld task, a solution's success is determined by the environment with a PDDL solver. Details and prompts are provided in Appx. E. We report the average accuracy of three runs.

	AlfWorld	
	in distribution	out of distribution
Qwen3-1.7B	11.07%	12.69%
DAPO	18.69%	**24.12%**
ProCeedRL	**20.35%**	23.33%
Qwen3-8B	44.22%	47.07%
RFT	47.38%	50.25%
DAPO	45.23%	53.24%
ProCeedRL	51.43%	55.22%
ProCeedSFT	**57.14%**	**58.95%**

Table 2: Main results on the embodied AlfWorld task.

Main Results The effectiveness of ProCeedRL is validated on deep search (Tab. 1) and embodied planning tasks (Tab. 2), where it substantially outperforms standard RL algorithms that utilize independent repeated sampling. Specifically, ProCeedRL achieves an average improvement of 3.72% on deep search QA, particularly on complex benchmarks such as MuSiQue, and increases performance by over 10% on the embodied AlfWorld task when combined with SFT. Furthermore, its superiority over RFT, which fine-tunes on expert trajectories selected via rejection sampling, underscores the critical role of on-policy data. This result suggests that enhancing the model's reasoning capabilities is more effectively achieved by breaking the vicious cycle and improving exploration efficiency, rather than by merely distilling knowledge from complete, pre-validated expert solutions.

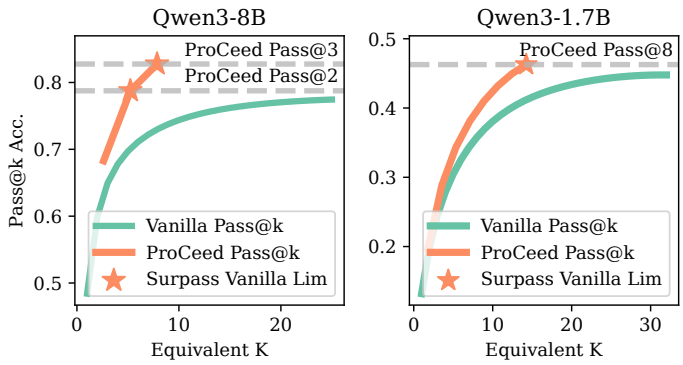

Figure 4: Pass@k of ProCeed and vanilla rollout. The x-axis denotes the number of equivalent vanilla samples in terms of generation. Our method significantly improves exploration efficiency, matching pass@k with less computation. Notably, it exceeds the saturation ceiling of vanilla exploration with a few samples (denoted by stars).

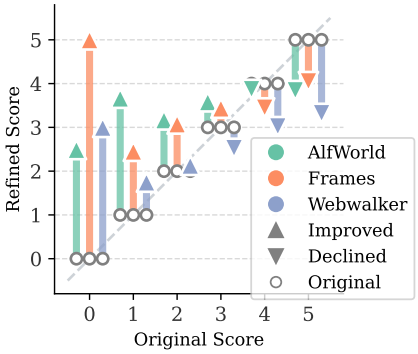

Figure 5: Improvement of refined actions over the original ones. Our method improves suboptimal actions with low ratings, whereas improvements gradually diminish as better original actions.

4.3 Analysis and Ablation Study

To further validate the importance of breaking the vicious circle between suboptimal actions and adverse contexts, and to demonstrate the effectiveness of our method, we conduct extensive analyses and ablation studies to support our design.

4.3.1 Benefits of ProCeedRL

Exploration Effectiveness To verify that our method exceeds the model's reasoning upper bound due to the vicious circle, we test the pass@k accuracy of ProCeed rollout against repeated sampling in AlfWorld using `Qwen3-1.7B/8B`. The other settings are the same as in the main experiment. We align the computational cost of pass@k in Fig. 4 to verify the exploratory gains of investing computation in this pipeline, rather than in generating additional trajectories. Based on our statistics (more details in Appx. B), we find that one ProCeed trajectory costs approximately 2.5 and 1.8 vanilla samples for 8B and 1.7B models, respectively (e.g., ProCeed pass@2 is equivalent to vanilla pass@5 in terms of generations for the 8B model). However, this overhead is only computed as per-trajectory overhead with a thinking critic. In terms of the response number and total tokens needed, our method requires less computation for the same exploration performance due to improved exploration efficiency.

As shown in Fig 4, though each trajectory requires nearly twice the computational effort, we only need one-quarter of the trajectory count. Ultimately, this results in achieving the same exploration performance with only half the computational load. Crucially, ProCeed surpasses the saturation ceiling of vanilla sampling with only a few samples (2 and 8 samples for 8B and 1.7B models), demonstrating its ability to overcome the upper bounds of vanilla exploration. This finding substantiates the benefits of our method and its capacity to break vicious cycles in exploration, thereby surpassing the upper bound of vanilla exploration. In our main experiments, we also controlled the generation for constructing the training set to be roughly the same for our method and baselines. The overall computation reduction and superior policy performance make the additional computation per trajectory especially worthy.

Improvement of Refined Demonstration To directly quantify the efficacy of our refinement module, we evaluate the change in action quality for suboptimal steps identified by the critic. We first isolate states with an initial critic score below five from the WebwalkerQA, Frames, and AlfWorld benchmarks. For each state, we generate a refined action and then re-evaluate it using the same process critic, enabling a direct comparison of pre- and post-refinement scores.

As illustrated in Fig. 5, most steps with low initial scores show substantial improvement after refinement. This enhancement is evident in the mitigation of error propagation from flawed intermediate actions to incorrect contextual information. However, the utility of refinement displays a diminishing return as the quality of the original action increases. Specifically, enforcing reflection on an already high-quality action can sometimes introduce confusion rather than refinement, potentially undermining performance. These observations underscore the importance of determining an appropriate threshold for applying refinement, which we

address in detail in a later discussion.

Critic Type	Frames	WebwalkerQA	AlfWorld OOD
/	31.92%	18.64%	47.07%
Self Critic	38.83%	21.18%	69.21%
Homogeneous	37.54%	21.32%	64.19%
External	41.18%	21.03%	64.55%

Table 3: ProCeed rollout accuracies with different critic choice. All the critic settings improve test-time rollout accuracy, demonstrating the effectiveness and model-agnosticity of our method, as well as the importance of breaking the vicious circle during exploration.

4.3.2 Ablation Study

Critic Choice The critic plays a vital role in the framework, identifying critical steps that significantly influence the exploration process. To rule out the influence of the critic's knowledge, we test the following candidates:

1. **Self-critic**, where the model critique itself on its performance;
2. **Homogeneous Critic**, using a larger model of the same family;
3. **External Critic**, where we leverage another powerful model.

We accordingly employ `Qwen3-8B`, `Qwen3-30B-A3B-Thinking-2507`, and `DeepSeek-V3-0528` as both critics and refinement policy. ProCeed rollout accuracy for `Qwen3-8B` is evaluated across these settings on datasets from Section 4.3.1.

Tab. 3 shows that all critic settings are effective, although with varying performance across benchmarks and models. Crucially, the self-critic configuration alone yields substantial gains over the base model, even surpassing stronger models on specific tasks. This highlights that the method's efficacy relies on the procedural pipeline of identifying and correcting flawed actions to prevent cascading errors in reasoning contexts, rather than on oracle supervision. In practice, we select critics comprehensively based on their exploration gains, as well as their similarity to the policy model, which helps ensure stable learning from demonstrations.

Knowledge Distillation Effect In our main experiments, we design a controlled group to quantify the distillation-like effect from leveraging a stronger model. To rule out the effect of distilled knowledge in critic's rating and analysis, we collect demonstrations from the critic we used, and apply knowledge distillation with SFT on these demonstrations (the RFT variant in Tab 1,2). Based on the findings in the experiments, our method outperforms distillation by significant margins, which attributes improvements primarily to our proposed method.

What's more, in the alfworld main experiments, our method is trained with the self-critic variant, which further shows that our method does not depend on the knowledge of superior critic models. The bonus comes from the process critic and reflection pipeline that refines misleading actions and avoids accumulating errors, rather than from an external model's knowledge. It also demonstrates that our method is a scalable pipeline, which shows continuous improvement when the policy itself is critic and does not rely on stronger models.

When to Rewind We conduct an ablation study of the rewind threshold, l_{th}, which determines the fraction of low-scoring trajectory steps to be refined. To contextualize this analysis, we first examine the distribution of critic scores, which are assigned integer values from 0 to 10 for all steps, as shown in histograms in Fig. 6. We measure the resulting ProCeed rollout accuracy by varying the threshold l_{th} in our method, shown as the line chart in Fig. 6. Other settings are held constant with the main experiments.

Results show that performance improves as more steps are reverted, but with diminishing returns. This suggests that following the demonstration in a few steps is effective in preventing most cascading errors and misleading contexts. More reverted steps help prune more incorrect steps; however, excessive reversion not only increases computational overhead but can also disrupt the model's reasoning by replacing adequate actions with suboptimal ones (as in Fig. 5). We further validate this by training a model on deep search tasks with approximately twice as many reverted actions (as in Tab. 1). Its significantly poorer performance corroborates our analysis. These findings support our use of granular critic ratings over binary labels, as this approach enables fine-grained control over the process, and we set l_{th} to balance all these effects in practice.

Reducing Computation We here propose ways to reduce the additional computation per trajectory. One clear way is to switch the critic to conciser modes. We calculate the performance of proceed rollout loop with Qwen3-8B as well as the additional tokens incurred under such cases:

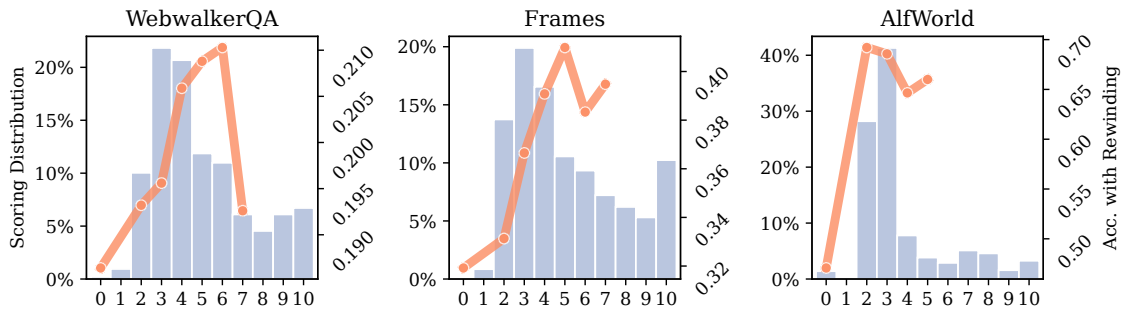

Figure 6: Ablation on rewind thresholds (l_{th}). Blue histograms display the critic score distributions, and the orange line shows rollout accuracy for rewinding and refining all steps $\leq l_{th}$. While pruning low-scoring steps improves exploration, excessive rewinding at higher thresholds yields diminishing returns and performance degradation.

Critic mode	Frames		AlfWorld	
	critic tokens	acc	critic tokens	acc
Thinking	760.84	38.83%	1320.59	65.29%
Non-thinking	246.02	38.01%	211.69	62.19%

Table 4: Critic tokens required and resulting accuracies on Frames and AlfWorld benchmark under different critic mode. It shows that with similar or slightly reduced performance, the computation overhead per trajectory can be greatly reduced.

only changing the critic Qwen3-8B from thinking to non-thinking mode, with everything else the same, as shown in Tab 4. It shows that with similar or slightly reduced performance, the computation overhead per trajectory can be greatly reduced (80% in alfworld), making our method more computational efficient and more scalable.

5 Conclusions

In this paper, we identify a "vicious cycle" in agentic reasoning in which suboptimal actions generate noisy feedback, degrade context, and hinder subsequent exploration. To address this, we propose ProCeedRL, which employs a process critic to rewind and refine adverse steps in real time, thereby preventing error propagation. Extensive experiments and analysis on deep search and embodied tasks demonstrate that ProCeedRL significantly outperforms standard RLVR baselines, surpassing its upper bounds in exploration and reasoning with superior efficiency. Ultimately, ProCeedRL offers a scalable, self-contained method to advance agentic reasoning beyond the inherent upper bounds of independent repeated sampling.

Limitations

Our method incurs per-trajectory computational overhead compared with vanilla exploration RL. The extra computation, introduced by the criticism and possible reflection at each step, incurs an asymptotic cost of $\mathcal{O}(1)$ per step and does not change the overall asymptotic behavior. However, as we demonstrated, this improves exploration efficacy and exceeds the upper bound of vanilla exploration with less overall computation, thereby justifying the additional cost per-trajectory. Moreover, our method does not require a critic at test time, although using one would improve reasoning.

Another limitation of our method is the lack of a guarantee of improvement. Our method leverages the LLM's internal knowledge to identify suboptimal actions and refine them. Although such an approach can improve step quality on average, it lacks theoretical guarantees and may lead to failures, thereby hindering exploration and affecting outcomes. Applying hierarchical critics may help mitigate this issue, and we hope that future work on critics and LLM mechanisms can further address it and improve the scalability of our method.

Our method does not introduce additional risk. While we use established public datasets that have mitigated privacy and toxicity risks, deploying autonomous agents may require ongoing safeguards to prevent unintended behavior in open-ended, real-world environments.

References

Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. 2025. Research: Learning to reason with search for llms via reinforcement learning.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Ruo Yu Tao, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler.

2018. Textworld: A learning environment for text-based games.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning.

Yong Deng, Guoqing Wang, Zhenzhe Ying, Xiaofeng Wu, Jinzhen Lin, Wenwen Xiong, Yuqin Dai, Shuo Yang, Zhanwei Zhang, Qiwen Wang, Yang Qin, Yuan Wang, Quanxing Zha, Sunhao Dai, and Changhua Meng. 2025. Atom-searcher: Enhancing agentic deep research via fine-grained atomic thought reward. *Preprint*, arXiv:2508.12800.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. 2025. Agentic reinforced policy optimization. *ArXiv preprint*, abs/2507.19849.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. 2025a. Retool: Reinforcement learning for strategic tool use in llms.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. 2025b. Group-in-group policy optimization for llm agent training. *ArXiv preprint*, abs/2505.10978.

Daocheng Fu, Jianbiao Mei, Licheng Wen, Xuemeng Yang, Cheng Yang, Rong Wu, Tao Hu, Siqi Li, Yufan Shen, Xinyu Cai, and 1 others. 2025. Re-searcher: Robust agentic search with goal-oriented planning and self-reflection. *ArXiv preprint*, abs/2509.26048.

Jingyue Gao, Runji Lin, Keming Lu, Bowen Yu, Junyang Lin, and Jianyu Chen. 2025. Marge: Improving math reasoning for llms with guided exploration. *ArXiv preprint*, abs/2505.12500.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *ArXiv preprint*, abs/2503.09516.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. 2024. Vineppo: Unlocking rl potential for llm reasoning through refined credit assignment.

Satyapriya Krishna, Kalpesh Krishna, Anhad Mohananey, Steven Schwarcz, Adam Stambler, Shyam Upadhyay, and Manaal Faruqui. 2024. Fact, fetch, and reason: A unified evaluation of retrieval-augmented generation.

Shiyu Li, Yang Tang, Yifan Wang, Peiming Li, and Xi Chen. 2025. Reseek: A self-correcting framework for search agents with instructive rewards.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2024. GAIA: a benchmark for general AI assistants. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, and 244 others. 2024. Openai o1 system card.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Singapore. Association for Computational Linguistics.

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

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 10737–10746. IEEE.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew J. Hausknecht. 2021. Alfworld: Aligning text and embodied environments for interactive learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Joykirat Singh, Raghav Magazine, Yash Pandya, and Akshay Nambi. 2025. Agentic reasoning and tool integration for llms via reinforcement learning. *Preprint*, arXiv:2505.01441.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025. R1-searcher: Incentivizing the search capability in llms via reinforcement learning.

Sijun Tan, Michael Luo, Colin Cai, Tarun Venkat, Kyle Montgomery, Aaron Hao, Tianhao Wu, Arnav Balyan, Manan Roongta, Chenguang Wang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. 2025. rllm: A framework for post-training language agents. Notion Blog.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, Chuning Tang, Congcong Wang, Dehao Zhang, Enming Yuan, Enzhe Lu, Fengxiang Tang, Flood Sung, Guangda Wei, Guokun Lai, and 77 others. 2025. Kimi k1.5: Scaling reinforcement learning with llms.

Qwen Team. 2025a. Qwen3 technical report.

Qwen Team. 2025b. Qwq-32b: Embracing the power of reinforcement learning.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2023. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *ArXiv preprint*, abs/2312.08935.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. 2025a. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning.

Ziliang Wang, Xuhui Zheng, Kang An, Cijun Ouyang, Jialu Cai, Yuhang Wang, and Yichao Wu. 2025b. Stepsearch: Igniting llms search ability via step-wise proximal policy optimization. *ArXiv preprint*, abs/2505.15107.

Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He, Deyu Zhou, Pengjun Xie, and 1 others. 2025. Webwalker: Benchmarking llms in web traversal. *ArXiv preprint*, abs/2501.07572.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. 2025. Learning to reason under off-policy guidance. *Preprint*, arXiv:2504.14945.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

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, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, and 16 others. 2025. Dapo: An open-source llm reinforcement learning system at scale.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2023. Scaling relationship on learning mathematical reasoning with large language models.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. 2025. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model?

Wenhao Zhang, Yuexiang Xie, Yuchang Sun, Yanxi Chen, Guoyin Wang, Yaliang Li, Bolin Ding, and Jingren Zhou. 2025a. On-policy rl meets off-policy experts: Harmonizing supervised fine-tuning and reinforcement learning via dynamic weighting. *Preprint*, arXiv:2508.11408.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025b. The lessons of developing process reward models in mathematical reasoning. *ArXiv preprint*, abs/2501.07301.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. 2025. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments.

A Algorithm

We summarize our overall pipeline and algorithmic details as in Alg. 1.

B Computation for Critics

tasks	policy model	policy tokens	critic type	critic tokens	ratio
AlfWorld	Qwen3-8B	857.36	thinking	1320.59	2.54
			instruct	211.69	1.25
	Qwen3-1.7B	1410.51	thinking	1111.39	1.79
Frames	Qwen3-8B	1010.36	thinking	760.84	1.75
			instruct	246.02	1.24

Table 5: The computation overhead per step for critics under different settings.

Algorithm 1 ProCeedRL Training Loop

Require: Environment `env`, Policy π_θ, Critic ϕ, Refine Policy μ, Threshold l_{th}, Max Steps $m_{\max}$, Batch Size bs, Group Size g.

for Each Batch with bs **do**
  $\mathcal{B} \leftarrow \{\}$
  $s \leftarrow$ `env.reset()`, $\tau \leftarrow \{s_0\}$
  for Each Trajectory in Group g **do**
    while $i \leq m_{\max}$ **do**
      $a_i \leftarrow \pi_\theta(\tau)$
      if Reversible `env` **then**
        s_{i+1}, done $\leftarrow$ `env.step(`a_i`)`
      else
        $s_{i+1} \leftarrow None$
      end if
      $l_t, c_t \leftarrow \phi(\tau, a_i, s_{i+1})$
      if activate reflection and $l_t \leq l_{\text{th}}$ **then**
        Rewind agent and environment
        $a_i \leftarrow \mu(\tau, a_i, l_t, c_t)$
      end if
      s_{i+1}, done $\leftarrow$ `env.step(`a_i`)`
      $\tau \leftarrow \tau + \{a_i, s_{i+1}\}$, $i \leftarrow i + 1$
      if done **then**
        break
      end if
    end while
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{\tau\}$
  end for
  Calculate group-based A_i as DAPO
  Update θ with $\mathcal{B}$ and Eq. 1.
end for
return π_θ

The step-level critic in our method requires an additional generation stage to obtain feedback, resulting in an increased computational cost relative to vanilla exploration. Here, we have tallied the average number of tokens generated by the agent and critic per step during actual operation to quantify computational overhead, as shown in Tab. 5. We calculate the ratio of computation required for collecting a trajectory in our method and in standard ways as $1 + \frac{\sharp(\text{average critic tokens})}{\sharp(\text{average policy tokens})}$. We find that our method incurs computational overhead, with each sample requiring generations equivalent to 1.8 to 2.5 standard trajectories. But such costs in each sample result in overall improvement in exploration efficiency and saturation performance, as discussed in Sec. 4.3.1.

We also collect the previous metrics when switching the critic to a non-thinking one, as shown in the "instruct critic" rows of Tab. 5. The reduced ratio suggests that we can also mitigate computational costs by replacing the critic with a non-thinking instruction model that produces shorter outputs.

C Implementation Details

We implement our training based on verl (Sheng et al., 2024) and rllm (Tan et al., 2025), and we adopt the formulation of the AlfWorld benchmark from verl-agent (Feng et al., 2025b). We use the default thinking mode for the `Qwen3-1.7B/8B` models. We select the critic by comprehensively evaluating candidates based on their exploration gains and their similarity to the strategy model, thereby stabilizing learning with demonstrations. The key parameters for running different algorithms and environment settings are listed in the following tables (Tab. 6, 7, 8).

	value
epochs	3
learning rate	1e-5
batch size	64
max tokens per step	4096
temperature	0.7
top p	0.95

Table 6: Key parameters for SFT

	value
KL penalty	0.01
train batch size	16
ppo micro batch size	256
group size	8
learning rate	1e-6
temperature	0.7
top p	1.00
max length per step	4096

Table 7: Key parameters for RL

	value (deep search/embodied)
max steps	10 / 50
threshold	3

Table 8: Key parameters for environment configuration

Our experiments are conducted on Nvidia H100 and A100-80G GPUs. The overall computation budget takes about 10 gpu-months, most of which are conducted with 2 for 1.7B models and 8 for 8B models.

D Artifact Details

Bamboogle(Press et al., 2023) Bamboogle, distributed under an MIT License, contains 125 multi-hop questions to test an agent's ability to do multi-turn searching and knowledge combination. The dataset is used in alignment with its license and the intended purpose of evaluating whether models can combine known facts to answer 2-hop questions. The dataset is constructed through a human-verification process using popular Wikipedia entities, thereby mitigating potential privacy or harm issues.

MuSiQue(Trivedi et al., 2022) MuSiQue is a large-scale English benchmark and is released under a CC BY 4.0 license. It contains approximately 2,500 test questions, each requiring an answer based on 2 to 4 distinct paragraphs. Its use is consistent to test an agent's ability to perform multi-turn, search-augmented reasoning. The dataset was generated based on the Wikipedia corpus and does not contain private or toxic content.

GAIA(Mialon et al., 2024) GAIA is a benchmark that aims to evaluate next-generation LLMs with augmented tool capabilities. It includes 450 non-trivial questions with unambiguous answers, ranging in difficulty and requiring different levels of tooling and autonomy to solve. We test on the 165 validation set. It is publicly released on Hugging Face, and our use aligns with its intended purpose.

Frames(Krishna et al., 2024) Frames is an English benchmark for retrieval agents, comprising 824 multi-hop questions that require handling complex constraints. Released under the Apache-2.0 license, our use aligns with the license and its purpose of testing the factuality and reasoning ability of search agents. The dataset was built using Wikipedia articles, ensuring high content quality. It mitigates risks associated with private personal information or offensive material by selecting neutral, factual topics.

WebwalkerQA(Wu et al., 2025) WebwalkerQA is a bilingual (English and Chinese) benchmark comprising 680 search agent queries, released under the Apache-2.0 license. It is designed and used to evaluate autonomous web navigation and long-context decision-making in dynamic environments. The dataset was constructed by crawling publicly accessible webpages across safe, distinct domains to ensure that private user data or other sensitive personal information was excluded.

AlfWorld(Shridhar et al., 2021) ALFWorld contains interactive TextWorld (Côté et al., 2018) environments that parallel embodied household environments in the ALFRED (Shridhar et al., 2020) dataset. AlfWorld tests an agent's ability to reason and learn high-level policies in an abstract space before solving embodied tasks through low-level actuation. It contains 3553 training configurations, 120 in-distribution configurations, and 134 out-of-distribution configurations. It is released under the MIT license and does not contain any personal information.

HotPotQA (Yang et al., 2018) HotpotQA is a foundational dataset for diverse, explainable multi-hop question answering, released under a CC BY-SA 4.0 license, with 113k Wikipedia-based English question-answer pairs. Its questions require finding and reasoning over multiple supporting documents, aligned with our purpose. We used a subset of approximately 4000 questions, balanced for topic and difficulty, as our training set, thanks to its diverse question set.

Qwen3 Models (Team, 2025a) We utilize some models in the Qwen3 family in our experiment and training, including `Qwen3-1.7B`, `Qwen3-8B`, and `Qwen3-30B-A3B-Thinking-2507`. These models are distributed under the Apache-2.0 license. Our use in research aligns with their intended use.

E Prompts

Given the environment settings and the agent's tasks across different environments, we design a separate prompt for the process-level critic. It is based on our principles for evaluating steps by both their reasoning and their consequences in the following context, as discussed in Sec. 3.3. In practice, we ask the critic to produce additional analysis as CoT to improve rating accuracy.

Deep Search QA critic prompt

You are an expert in solving difficult problems through reasoning and retrieval.
Currently, an agent is attempting to answer a question through multi-round calls to search tools. The question, the agent's current action and retrieved documents, and history turns are provided.
Your task is to evaluate and score one round of the agent's calls, determining whether the search queries in the current round is correct for answering the question.
The score ranges from 0 to 10, where 0 means completely incorrect and 10 means completely correct.
Your score and analysis will serve as feedback to help the agent decide whether to revise the search terms or proceed to the next round of retrieval and reasoning. If feeling necessary, you can also output your revised search terms and a supporting thinking process for these search queries.
Output format: The final output is list data in JSON format as
``` json
{{"score": score, "critique": "your analysis", "suggestion_search_keywords": "[your revised queries,...]", "suggestion_search_reasoning": "step-by-step reasoning that supports the improved query as if you are the agent for it to learn from, without mentioning the current tool call and you are a critic."}}
```
Question: {problem}
Search Query: {tool_results}
History Turns: {history}

AlfWorld critic prompt

You are an expert evaluator observing an agent trying to complete a task in a household environment.
You will be provided with the agent's task description, the environment's configuration, a history of its previous actions and observations, current admissible actions, and the agent's most recent action.
- Agent's Task: {task_description}
- Environment Configuration: {environment_config}
- History Turns (observations and the corresponding actions the agent took): {action_history}
- Admissible Actions: {admissible_actions}
- Agent's Latest Observation: {latest_observation}
- Agent's Latest Action: {agent_action}
Your goal is to evaluate and score the quality of the agent's latest action, determining if it is a good action for accomplishing the task. The score ranges from 0 to 10, where 0 means completely incorrect and 10 means completely correct. Also give a consice yet complete analysis of the agent's last action. Was it logical? Did it move closer to the goal? Did it make a mistake? If feeling neccessary (like the action is misleading), you can suggest a better candidate action FROM admissible actions together with your step-by-step reasoning. Your suggested action and reasoning may be used by the agent to improve its performance, so ensure the action is from the admissible actions and the reasoning is from an expert agent's first-person view.
Output Format: Output your critic result in JSON format as
```json
"score": score, "critique": "your analysis", "suggestion_action": "The better admissible action you suggest", "suggestion_thought": "Your detailed step-by-step reasoning for the better action. "
```
Ensure to enclose the entire JSON output within a single markdown code block.

As for refinement prompts, we use a similar structure for both tasks, with the only difference

being the instructions for action output, which is based on the task settings in different environments.

Refining Prompt 1

An expert critic has commented on your latest step latest_step, judging the quality of your latest step and whether it helps finishing your task.
- Critic's Overall Ratings (0-10): {score}.
- Critic's Analysis on your previous step: {critic_content}
Your admissible actions of the current situation are: {admissible_actions}.
Carefully read and understand the critic's feedback, and it's your turn to refine the step and retake an feasible action based on the feedback.
You should first reason step-by-step about the previous situation. This reasoning process MUST be enclosed within <think> </think> tags, and do not include any information about the critic, as if it's your first time making the action.
Once you've finished your reasoning, you should choose an admissible action for current step and present it within <action> </action> tags.

Refine Prompt 2

A critic has read and analysed your previous step:
- Critic's Overall Ratings (0-10): {critique_score}
- Critic's Analysis on your previous step: {critique_content}
- Critic's Suggestion search keywords: {critique_suggestion}
Based on the critic's feedback, redo your previous tool call to improve its correctness and quality, in order to better solve the problem.
Note: When you redo and refine the tool call, you should act as if you are making the action the first time, do not add any information about the critic.

LLM-as-judge prompt

You are an impartial judge evaluating the correctness of an AI assistant's answer.
[Question]
{question}
[Correct Answer]
{reference_answer}
[Assistant's Answer]
{assistant_answer}
Task: Determine if the assistant's answer is correct by comparing it to the correct answer.
Instructions:
1. Extract the final answer from the assistant's response
2. Compare it with the correct answer
3. Provide your reasoning
4. Answer with "yes" if correct, "no" if incorrect