Generator-Assistant Stepwise Rollback Framework for Large Language Model Agent

Xingzuo Li[1], **Kehai Chen**[1], **Yunfei Long**[2], **Xuefeng Bai**[1*], **Yong Xu**[1], **Min Zhang**[1]
[1]School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China
[2]School of Electronic Engineering and Computer Science, Queen Mary University of London
24S051028@stu.hit.edu.cn, qp241311@qmul.ac.uk,
{chenkehai, baixuefeng, laterfall, zhangmin2021}@hit.edu.cn

Abstract

Large language model (LLM) agents typically adopt a step-by-step reasoning framework, in which they interleave the processes of thinking and acting to accomplish the given task. However, this paradigm faces a deep-rooted *one-pass* issue whereby each generated intermediate thought is plugged into the trajectory regardless of its correctness, which can cause irreversible error propagation. To address the issue, this paper proposes a novel framework called Generator-Assistant Stepwise Rollback (GA-Rollback) to induce better decision-making for LLM agents. Particularly, GA-Rollback utilizes a generator to interact with the environment and an assistant to examine each action produced by the generator, where the assistant triggers a rollback operation upon detection of incorrect actions. Moreover, we introduce two additional strategies tailored for the rollback scenario to further improve its effectiveness. Extensive experiments show that GA-Rollback achieves significant improvements over several strong baselines on three widely used benchmarks. Our analysis further reveals that GA-Rollback can function as a robust plug-and-play module, integrating seamlessly with other methods. [1]

1 Introduction

Developing Large Language Model (LLM) agents capable of helping humans tackle real-world challenges has become a central focus in current artificial intelligence research (Xi et al., 2023; Zhang et al., 2023). Recently, researchers have leveraged the inherent self-planning abilities of LLMs to integrate thoughts with actions, enabling step-by-step reasoning processes during agent tasks (Song et al., 2022; Yao et al., 2023b; Qiao et al., 2024). This enhancement empowers LLM agents

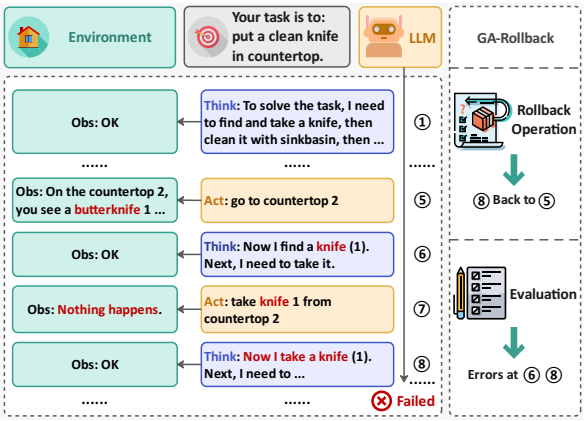


Figure 1: An example of *one-pass* paradigm. The trajectory is generated by LLaMA3.1-8B-Instruct in ReAct-style. Our approach blocks error propagation through rollback operations and ensures the quality of thinking processes through evaluation.

to achieve remarkable improvements on diverse applications, including mathematical reasoning (Hendrycks et al., 2021; Cobbe et al., 2021), web browsing (Yao et al., 2022; Deng et al., 2023; Zhou et al., 2024b), and embodied tasks (Shridhar et al., 2020; Wang et al., 2022).

However, this paradigm is inherently constrained by an *one-pass* limitation, where each generated thought is inserted directly into the trajectory regardless of whether it is correct or not. As a result, incorrect thoughts may persist in the context, influencing subsequent actions and ultimately compromising the outcome. Take an example in Figure 1 generated by LLaMA3.1-8B-Instruct in ReAct-style. In this scenario, the environment responds to all thoughts with a simple "*OK*", which implies that the system assumes the correctness of these thoughts by default. Consequently, a low-quality thought that mistakes "*butterknife*" for "*knife*" leads to the invalid action "*take knife 1 from countertop 2*". To address this issue, many works

*Corresponding author.
[1]Our code is available at https://github.com/wisper12933/GA-Rollback.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 17695–17712
November 4-9, 2025

have been devoted to improving the precision and clarity of reasoning processes. Typically, self-correction methods (Madaan et al., 2023; Shinn et al., 2023; Gou et al., 2024) summarize refined plans using feedback from previous executions and improve the quality of the solution through multiple trials. Despite these achievements, the *one-pass* reasoning pattern within each trial remains prone to error propagation due to its dependence on potentially flawed intermediate steps.

In this paper, we propose a novel framework called Generator-Assistant Stepwise Rollback (GA-Rollback) to induce better decision-making for goal-driven agents. Specifically, the generator interacts with the environment and supplies the assistant with essential contextual information. Meanwhile, the assistant meticulously examines each action produced by the generator along with the corresponding observation. When errors or suboptimal actions are detected, the assistant will provide detailed feedback to guide the generator in performing rollback operations, which revise previous incorrect actions to prevent error propagation. To further enhance our framework, we introduce two key strategies: probability-based feedback evaluation (§3.2) to ensure feedback credibility, and Wait-Info strategy (§3.3) to enrich contextual information in embodied tasks. Experiments on three representative tasks reveal that our method outperforms several strong baselines across various models. Moreover, our analysis demonstrates that GA-Rollback can function as a robust plug-and-play module, integrating seamlessly with other methods. In summary, our main contributions are as follows:

- We propose the GA-Rollback framework for llm-based agents, which separates action and thinking processes to ensure more precise and credible reasoning trajectory.
- We introduce the probability-based feedback evaluation along with Wait-Info strategy designed for embodied environments.
- Experiments on three tasks reveal that our method achieves notable improvement and exhibits stronger robustness, indicating its extensibility as a plug-and-play module.

2 Related Work

Our work is related to LLM agents and self-correction mechanism. In this section, we first review recent advances in LLM agents, followed by an analysis of self-correction mechanism.

2.1 LLM Agents

The reasoning and instruction-following capabilities that have emerged in LLMs (Huang and Chang, 2022; Wei et al., 2022a) make them capable enough to serve as intelligent agents to complete various tasks, such as web navigation (Deng et al., 2023; Zhou et al., 2024b), machine translation (Zhang et al., 2024; Zuo et al., 2025) and role playing (Tang et al., 2025). Standard LLM-agent methods employ structured reasoning frameworks to strengthen their analytical abilities (Yao et al., 2023a; Besta et al., 2024), or apply specialized decoding strategies to improve accuracy (Wang et al., 2024). Recent works have made significant breakthroughs by leveraging the inherent self-planning and reflection abilities of LLMs (Paul et al., 2024; Zhou et al., 2024a). These methods achieve more effective interaction with external environments through task decomposition and planning within or between trials. While these methods have shown great progress, they still face a limitation in the lack of necessary evaluation of the reasoning process. This shortcoming can lead to the generation of low-quality reasoning steps, which may mislead subsequent actions and cause error propagation.

2.2 Self-correction Mechanism

Self-correction represents a feasible approach to improve responses from LLMs by enabling them to refine their outputs during inference (Bai et al., 2022; Madaan et al., 2023). The simplest implementation of self-correction prompts LLMs to provide feedback on their own responses and revise the responses based on the feedback (Huang et al., 2023). Recent studies have been dedicated to improving feedback by using additional resources, including external tools such as code executors (Chen et al., 2024; Gou et al., 2024), knowledge accessed through web browsing (Jiang et al., 2023; Zhao et al., 2023; Peng et al., 2023), and observation gathered from simulation environments (Shinn et al., 2023). Notably, Kamoi et al. (2024) identified feedback generation as the bottleneck in self-correction process, emphasizing that reliable feedback is essential for LLMs to successfully complete assigned tasks. Building upon these insights, our framework leverages LLMs as independent assistants in interactive environments and incorporates appropriate evaluations to enhance the credibility of generated feedback.

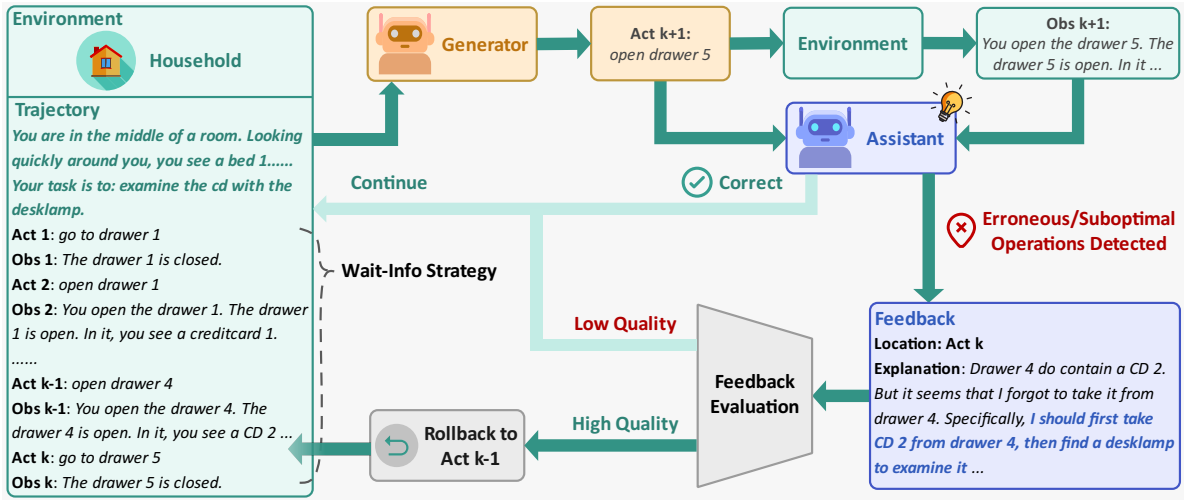


Figure 2: An illustrative overview of GA-Rollback framework. The dark green arrows represent the current workflow. Each action and observation will be reviewed by the assistant, and the feedback provided by the assistant will also be evaluated before being applied to the trajectory.

3 Methodology

The overall architecture of our method is illustrated in Figure 2. In this framework, the generator produces actions along the trajectory, while the assistant serves as a rollback controller that provides detailed feedback upon the detection of errors or suboptimal actions (§3.1). To further enhance the performance of our framework, we propose two key strategies: probability-based feedback evaluation (§3.2) and Wait-Info strategy designed for embodied tasks (§3.3).

We begin by formalizing the agent task and its trajectory. Given a pre-trained LLM denoted as $\mathcal{P}(\cdot)$, an agent needs to find a desired reasoning path toward addressing the given problem. We refer to the whole sequence of reasoning and the corresponding changes in the environment as a **trajectory** $\mathcal{T}_n = [\mathcal{I}, o_0, a_1, o_1, ..., a_n, o_n]$, where $\mathcal{I}$ is the task description, n is the length of the trajectory, o_0 denotes the initial observation of the environment, $a_{1...n}$ are actions made by the agent. Under normal circumstances, the intermediate action a_n is generated as $a_n \sim \mathcal{P}(a_n|\mathcal{T}_{n-1})$, and the current action set is defined as $\mathcal{A}_n = [a_1, a_2, ..., a_n]$. When an erroneous action, denoted as $\widehat{a}_t$, is retained within the trajectory $\mathcal{T}_t$, it is likely to lead to error propagation. Intuitively, the task cannot be accomplished through the erroneous action set $\widehat{\mathcal{A}}_n = [a_1, a_2, ..., \widehat{a}_t, \widehat{a}_{t+1}, ..., \widehat{a}_n]$.

To alleviate the impact of erroneous actions, self-correction mechanism is usually incorporated into the standard framework, whereby an optimization process is implemented after each step or each trial. In the scenario of stepwise self-correction, the agent adjusts its behavior based on current trajectory $\mathcal{T}_n$ and the feedback f_n from optimizer: $a_{n+1} \sim \mathcal{P}(a_{n+1}|f_n, \mathcal{T}_n)$. However, the delayed detection of errors leaves incorrect actions in the trajectory, which may adversely affect subsequent steps. Therefore, we introduce the concept of **rollback** (Chen and Li, 2024) into self-correction. When error occurs, LLM agents are required to create better reasoning path by continuously rolling back from current trajectory $\mathcal{T}_n$ to prior $\mathcal{T}_m$, where $m \in [0, n-1]$.

3.1 Action Rollback based on Feedback

In our framework, we employ an LLM as the assistant to support the generator engaged in the task. Concretely, each time the action a_n generated by the generator and the corresponding observation o_n are updated in the trajectory $\mathcal{T}_n$, the assistant analyzes $\mathcal{T}_n$ in a chain-of-thought (Wei et al., 2022b) manner. For tasks involving typically long trajectories, we enable the assistant review the steps from back to front to efficiently locate the most recent error encountered. If the assistant detects errors or suboptimal actions in $\mathcal{T}_n$ during analysis, it will provide feedback f_n including identified erroneous actions along with specific explanations. We assume that the identified erroneous actions are $[\widehat{a}_t, \widehat{a}_{t+1}, ..., \widehat{a}_n]$ with $\widehat{a}_t$ being the earliest error, and the number of rollback steps is $N = n - t + 1$. Considering that an excessively large rollback span may result in the omission of certain details, we

define an upper bound $\overline{N}$ for N. When N exceeds $\overline{N}$, it will be replaced by $\overline{N}$ to ensure a more precise and meticulous rollback operation.

Upon receiving the assistant's instruction to rollback to $\mathcal{T}_{t-1}$, we first revert the state of the environment. Specially, we retrieve the action set $\mathcal{A}_{t-1} = [a_1, ..., a_{t-1}]$ from $\mathcal{T}_{t-1}$, reset the environment, and sequentially execute the actions in $\mathcal{A}_{t-1}$ to restore the state to that of $\mathcal{T}_{t-1}$. Once this process is completed, the generator produces a new action a_t based on $\mathcal{T}_{t-1}$ and f_n. By rolling back to the previous state, the generator can continue its search iteratively until it finds a feasible path to complete the task. Drawing inspiration from Chen and Li (2024), we also incorporate past mistakes as experiential knowledge into the prompt to avoid the repetition of similar errors.

3.2 Probability-based Feedback Evaluation

Since feedback serves as the trigger for the rollback operation, its quality is of vital importance to the overall performance. If low-quality feedback intervenes in the trajectory, issues such as unnecessary rollbacks may arise. Based on the findings of Wang et al. (2024), we employ a probability-based evaluation to measure the confidence of the feedback generated by the assistant, thereby mitigating this adverse effect.

For a given output $Y = \{y_1, ..., y_m\}$ generated by the assistant, we score its confidence using the mean-pooled probability of its tokens:

$$\text{Score}_\text{Y} = \frac{1}{m}\sum_{i=1}^{m}\mathcal{P}(y_i|y_{<i}, X), \qquad (1)$$

where X denotes the input containing task description and current trajectory. If Score_Y is lower than a predefined threshold θ, the corresponding feedback will be discarded. In such cases, the feedback will neither be passed to the generator nor trigger a rollback operation. To determine the threshold θ, we first analyze the distribution pattern of this confidence metric, then empirically select an appropriate value through systematic testing at certain intervals across the distribution range (§4.3).

3.3 Wait-Info Strategy for Embodied Tasks

In embodied environments, agents need to meticulously check multiple locations and interact with various objects before discovering a viable solution to accomplish the assigned task. These exploratory behaviors represent necessary steps in the task-solving process. Thus, premature intervention during this exploration phase, particularly from the assistant, may be counterproductive and lead to redundant rollback operations, as the assistant might over-analysis these exploratory behaviors.

To resolve the aforementioned problem, our Wait-Info strategy restricts the assistant's involvement in the early stages, allowing the generator to independently generate and execute actions without immediate intervention. The trajectory analysis by the assistant is permitted only after the generator completes the generation and execution of k consecutive actions in the environment. If the rollback operation reduces the trajectory length below k, the assistant will remain locked until the generator extends the trajectory length back to k. This strategy enables the generator to explore its surroundings more thoroughly, providing the assistant with a richer set of information $\mathcal{T}_{\geq k}$ for decision-making.

4 Experiments

4.1 Experimental Settings

Tasks. We evaluate our framework on three representative agent tasks: Game of 24 (Yao et al., 2023a) for mathematical reasoning, ALFWorld (Shridhar et al., 2020) for embodied house hold tasks, Webshop (Yao et al., 2022) for web navigation. Webshop provides a dense reward ranging from 0 to 1 to quantify task completion, while Game of 24 and ALFWorld only provide binary rewards to indicate whether the task is completed. We made slight adjustments to the Game of 24 task to ensure that the agent receives relevant observation for each executed action. During evaluation, we sample 500 instances each from Webshop and Game of 24, and use the out-of-distribution test set of ALFWorld. More details can be found in Appendix A.

Baselines. We compare the GA-Rollback framework with several agent methods that are known for their ability to generalize across diverse tasks. (1) **Few-shot** (Brown et al., 2020) (referred to as **Act-only** in the table) provides the model with a set of examples to capture task-specific patterns. (2) **CoT** (Wei et al., 2022b) guides the model to explicitly generate intermediate rationales before reaching the final answer. (3) **ReAct** (Yao et al., 2023b) enhances task completion by interleaving reasoning with actions during execution, where the

Model	Method	Game of 24	ALFWorld	Webshop		Avg. SR
		SR	SR	Reward	SR	
LLaMA-3.1-8B-Instruct	Act-only	0.4	33.6	62.3	30.6	21.5
	CoT	0.0	32.8	47.8	27.4	20.1
	ReAct	0.4	22.4	47.9	27.0	16.6
	Reflexion	5.2	**44.8**	63.3	32.8	27.6
	ReAct + Reflexion	6.4	26.1	54.2	30.2	20.9
	GA-Rollback	4.2	38.8	61.1	34.0	25.7
	GA-Rollback + ReAct	6.4	18.7	44.8	25.6	16.9
	GA-Rollback + Reflexion	**6.8**	42.5	**68.1**	**40.6**	**30.0**
GLM4-9B-Chat	Act-only	0.0	73.1	65.3	32.6	35.2
	CoT	0.0	70.9	64.8	34.8	35.2
	ReAct	0.2	76.8	56.7	23.0	33.3
	Reflexion	7.2	77.6	67.3	33.6	39.5
	ReAct + Reflexion	6.4	**91.8**	59.9	26.0	41.4
	GA-Rollback	5.4	78.4	66.9	37.8	40.5
	GA-Rollback + ReAct	4.6	80.6	37.0	15.6	33.6
	GA-Rollback + Reflexion	**9.6**	85.8	**70.2**	**41.4**	**45.6**
Qwen2.5-14B-Instruct	Act-only	5.4	78.3	60.7	34.0	39.2
	CoT	7.0	86.6	57.8	36.2	43.3
	ReAct	7.6	89.6	43.6	26.6	41.3
	Reflexion	18.8	91.8	68.5	39.6	50.1
	ReAct + Reflexion	18.8	**95.5**	47.8	29.4	47.9
	GA-Rollback	17.2	89.6	65.7	41.2	48.8
	GA-Rollback + ReAct	19.4	83.6	36.9	23.8	42.3
	GA-Rollback + Reflexion	**23.6**	92.6	**70.4**	**45.2**	**53.8**

Table 1: Performance of different methods on three agent tasks. The best results of each model are marked in **bold** and the second-best results are marked with underline. SR: success rate.

model autonomously generates thinking processes based on the observation to refine subsequent actions. (4) **Reflexion** (Shinn et al., 2023) synthesizes more precise plans between attempts to optimize the trajectory of subsequent trials. (5) **ReAct + Reflexion** combines both methods for more comprehensive thinking and planning. Furthermore, we integrate GA-Rollback with these methods to investigate its extensibility.

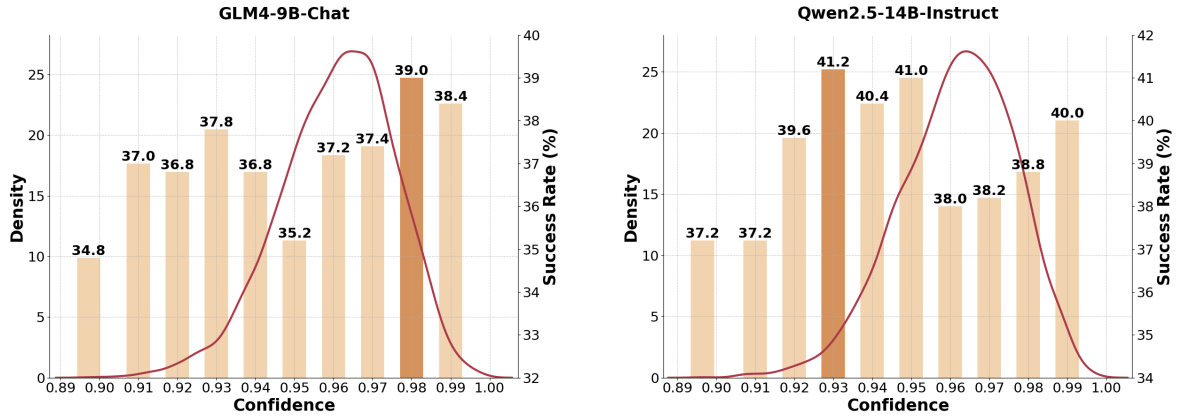
Implementation Details. We conduct our experiments on LLaMA-3.1-8B-Instruct (Dubey et al., 2024), GLM4-9B-Chat (GLM et al., 2024), and Qwen2.5 series (Yang et al., 2024) to provide comprehensive results. Additionally, results for models of varying scales, such as Qwen2.5-3B-Instruct, are provided in Appendix B.3. All experiments are carried out on NVIDIA HGX H20 96G GPUs. In the main experiments, we use the same model as generator and assistant. We set the "max_new_tokens" for the generator to 100 and for the assistant to 500, with both having a "temperature" of 0.1. We also leverage dynamic model compilation to accelerate inference. The number of Wait-Info steps is set to 6, and the quality threshold θ for feedback evaluation is set to 0.93. The number of trials for Reflexion is set to 2. To prevent unlimited rollback operations from causing an infinite loop, we set the maximum number of rollback attempts per task at 6.

4.2 Main Results

Table 1 presents the performance comparison of different methods on three agent tasks. Based on the results, we have the following findings:

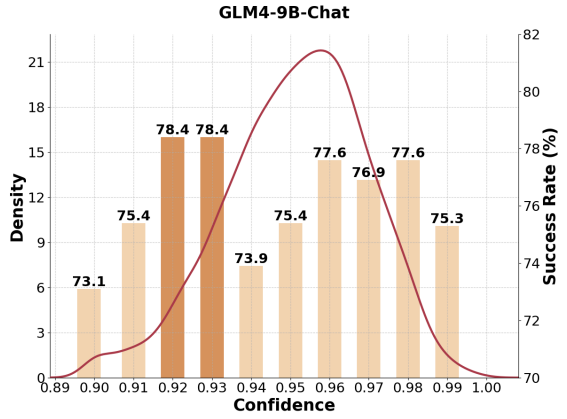
(i) When applied independently, GA-Rollback outperforms several baselines in complex reasoning tasks. For instance, in the Game of 24 benchmark, GA-Rollback alone achieves success rate of 4.2% (LLaMA-3.1-8B-Instruct), 5.4% (GLM4-9B-Chat), and 17.2% (Qwen2.5-14B-Instruct), representing 10.5×, 27.0×, and 2.3× improvements over ReAct. This indicates the effectiveness of our GA-Rollback as a dynamic component in multi-step reasoning.

(ii) The synergistic potential of GA-Rollback is particularly evident when combined with Reflexion. From the results of Webshop, the success rate of GA-Rollback is higher than that of Reflexion, but its reward is lower. This suggests that in the

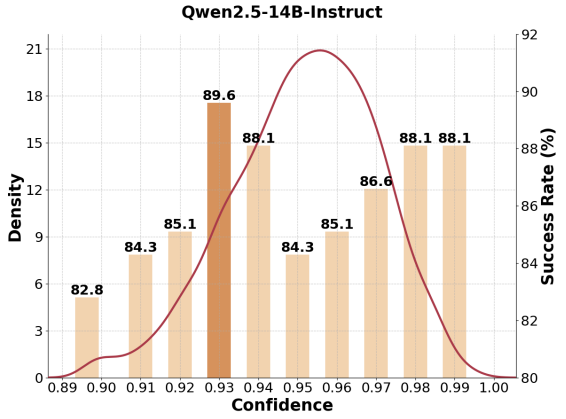


(a) Results of GLM4-9B-Chat on Webshop

(b) Results of Qwen2.5-14B-Instruct on Webshop



(c) Results of GLM4-9B-Chat on ALFWorld



(d) Results of Qwen2.5-14B-Instruct on ALFWorld

Figure 3: The kernel density estimation curve (red) and the success rate (orange) as a function of threshold θ. The leftmost bar in each subgraph corresponds to the performance without feedback evaluation. The darker colored bars highlight the respective performance peaks.

absence of an overall plan, GA-Rollback may cause the agent to miss some necessary intermediate steps. Combining these two methods could leverage their complementary strengths, leading to enhanced performance in agent tasks.

(iii) Even when GA-Rollback is integrated with Reflexion, it attains suboptimal performance on ALFWorld. Such outcomes indicate that embodied tasks involving longer trajectories remain challenging for GA-Rollback, particularly in determining which step to rollback to.

(iv) Surprisingly, we observe a performance decline in all three models when conducting Webshop in ReAct-style. Besides, while ReAct improves the ALFWorld success rate for Qwen2.5-14B-Instruct, it results in a 33% performance drop for LLaMA3.1-8B-Instruct. This demonstrates that the effectiveness of ReAct is contingent on the reasoning capabilities of LLMs. Incorrect reasoning caused by ReAct can adversely affect the outcomes. Moreover, we observe that the thinking processes generated by GA-Rollback and ReAct may conflict with each other, leading to instability when combining the two methods. Appendix C presents an example of such conflict.

4.3 Analysis of Feedback Evaluation

As mentioned in section 3.2, we utilize mean-pooling operations on the assistant's output tokens to represent its overall confidence in the generated feedback. In this section, we investigate the distribution of this confidence metric and its impact on the GA-Rollback framework.

Firstly, we implement the GA-Rollback framework using GLM4-9B-Chat and Qwen2.5-14B-Instruct to conduct Webshop and ALFWorld tasks, during which we collect the feedback confidence scores. As shown in Figure 3, the kernel density estimation curve of confidence approximately follows a normal distribution pattern. This indicates that the model maintains relatively stable confidence levels across different task instances

without exhibiting extreme overconfidence or underconfidence tendencies.

Based on the observed distribution, we leverage a threshold θ to filter out feedback with confidence falling below this value. To determine an optimal filtering criterion, we vary the threshold at 0.01 intervals across the distribution range and evaluate the corresponding success rates. As illustrated in Figure 3, without implementing feedback evaluation, the success rates of the two models on both tasks are relatively low. We also observe two distinct optimal threshold regions for feedback filtering: A threshold of 0.93 effectively eliminates low-confidence noise while preserving the majority of feedback; a stricter threshold of 0.98 retains only the highest-confidence feedback. We ultimately adopt 0.93 for two considerations:

- **Generalizability**: The threshold of 0.93 yields near-optimal results for both models.
- **Efficiency**: The threshold of 0.98 results in a significant portion of generated feedback being discarded, leading to unnecessary computational overhead and reduced efficiency.

4.4 Analysis of Wait-Info Strategy

Figure 4: The performance of GA-Rollback on ALFWorld with different Wait-Info steps.

Figure 4 illustrates how the **Wait-Info** strategy—the minimum number of generator actions delivered to the assistant—affects GLM-4-9B-Chat and Qwen-14B-Instruct on ALFWorld. Performance follows a clear ∩-shaped curve: success rate climbs steadily until $k \approx 6$ (yielding a 4-8 percentage-point gain over the no-wait baseline) and then drops. A moderate horizon enriches the assistant's context with a temporally coherent exploration burst, making causal chains explicit and exposing discrepancies between intended and executed actions; too small a horizon starves the assistant of signal, while too large a horizon floods it with noisy, combinatorial states.

This pattern reveals a principled **information-bandwidth trade-off** rather than a mere implementation tweak. Wait-Info functions as a horizon-gated scheduler that clearly separates the generator's high-entropy exploration phase from the assistant's low-entropy diagnosis phase. Up to the optimal k^*, progressive context enrichment helps the assistant perform targeted rollbacks; beyond k^*, the cognitive burden of analyzing long, error-diffused traces outweigh the benefit of deeper look-ahead, producing diminishing returns. Viewed in this light, Wait-Info becomes a reusable design knob for cooperative LLM paradigms, and we expect the same ∩-shaped relationship to generalize to other agent benchmarks where partial thinking trajectories must be exchanged.

4.5 Comparison with Tree-search Method

We further compare our approach with tree-search method ToT-BFS (Yao et al., 2023a) on Game of 24. As shown in Table 2, GA-Rollback achieves greater performance improvements while resulting in only 2× token cost and 1.5× runtime overhead compared to ToT-BFS.

Model	Method	SR	Avg. Token (K)	Avg. Time (s)
LLaMA3.1-8B-Instruct	Act-only	1.0	10.7	21.1
	ToT-BFS	1.2	22.8	85.2
	GA-Rollback	3.2	38.8	112.3
GLM4-9B-Chat	Act-only	0.0	10.9	27.2
	ToT-BFS	1.6	18.0	83.8
	GA-Rollback	6.0	36.4	123.0
Qwen2.5-14B-Instruct	Act-only	5.8	9.8	29.3
	ToT-BFS	6.4	17.7	82.2
	GA-Rollback	18.4	35.4	148.2

Table 2: Performance comparison with tree-search method ToT-BFS (Yao et al., 2023a) on Game of 24.

Notably, our approach addresses two critical limitations of tree-search methods: (1) Although methods like ToT optimize stepwise decisions through scoring mechanisms, they remain inherently susceptible to error propagation. In contrast, our approach actively mitigates error propagation through rollback-driven self-correction, ensuring more robust performance. (2) In complex environments (e.g., embodied settings), the action space grows exponentially, leading to extremely high search complexity. Instead of exhaustive search, our approach emphasizes dynamic plan adjustment via rollback during exploration.

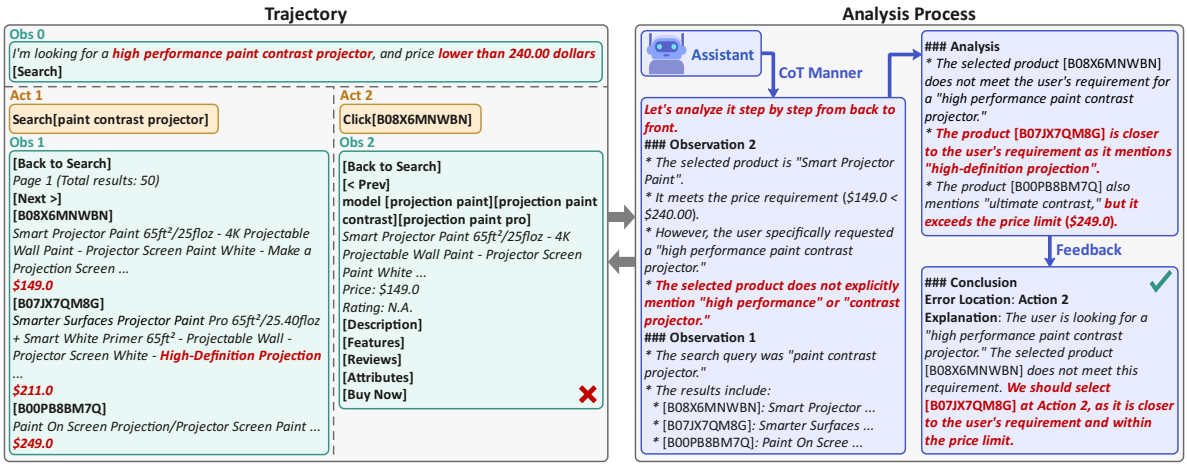


Figure 5: An example of trajectory analysis conducted by the assistant, with key information highlighted in red.

Setting	Game of 24	ALFWorld	Webshop
14B as Generator and Assistant			
GA-Rollback	**17.2**	**88.1**	**41.2**
w/o evaluation	15.4	82.8	37.2
w/o Wait-Info	-	80.6	-
w/o Assistant	5.4	78.3	34.0
Different Assistants			
0.5B	6.8	82.8	38.4
3B	9.4	85.8	39.2
7B	10.2	86.6	39.6
72B	**19.4**	**90.3**	**44.0**

Table 3: The ablation study on each component of GA-Rollback and assistants. The generator is consistently set to Qwen2.5-14B-Instruct, and the assistant is selected from Qwen2.5 Instruct models.

4.6 Ablation Study

We remove feedback evaluation, Wait-Info strategy, and even the entire assistant from our framework to validate their individual contributions. As shown in Table 3, the removal of each component leads to a clear drop in overall success rate, underscoring the effectiveness of our method. Specifically, eliminating the assistant reduces the entire framework to an Act-only form, resulting in a significant performance decline, which highlights the crucial role of the assistant.

To further study the impact of the assistant, we keep the generator fixed and experiment with Qwen2.5 Instruct models of varying scales (ranging from 0.5B to 72B) as the assistant. Notably, as the scale of the assistant increases, its reasoning capability improves correspondingly. This enhancement allows the assistant to analyze the trajectories more thoroughly and provide refined feedback, thereby boosting success rate.

4.7 Case Study

Here, we present an example of the trajectory analysis conducted by the assistant on Webshop. As shown in Figure 5, the user needs a "*high performance paint contrast projector*" below $240. Initially, the generator selects a product that appears to meet these requirements. However, after a thorough examination of the trajectory, the assistant discerns that the chosen item does not satisfy the user's requirement for "*high performance*". Recognizing this issue, the assistant revisits the search results and excludes the product "[B00PB8BM7Q]", which matches the description but exceeds the budget. Then, it selects the appropriate alternative "[B07JX7QM8G]" and provides detailed feedback to inform the generator. This meticulous verification and subsequent rollback operation enable the generator to more effectively achieve its task objectives. More detailed examples are provided in Appendix C.

4.8 Multi-Trial Analysis of GA-Rollback + Reflexion

To further investigate the compatibility of our GA-Rollback and Reflexion, we evaluate its success rate across multiple rounds of trials on three agent tasks. The results, as illustrated in Figure 6, reveal that the success rate of GA-Rollback + Reflexion exhibits a steady increase across three tasks with the rise in trial numbers. Notably, this approach significantly outperforms other multi-trial paradigms on both Game of 24 and Webshop, underscoring the versatility of GA-Rollback as a plug-and-play component that can be seamlessly integrated with other methods. In contrast to the findings reported by Shinn et al. (2023), smaller-

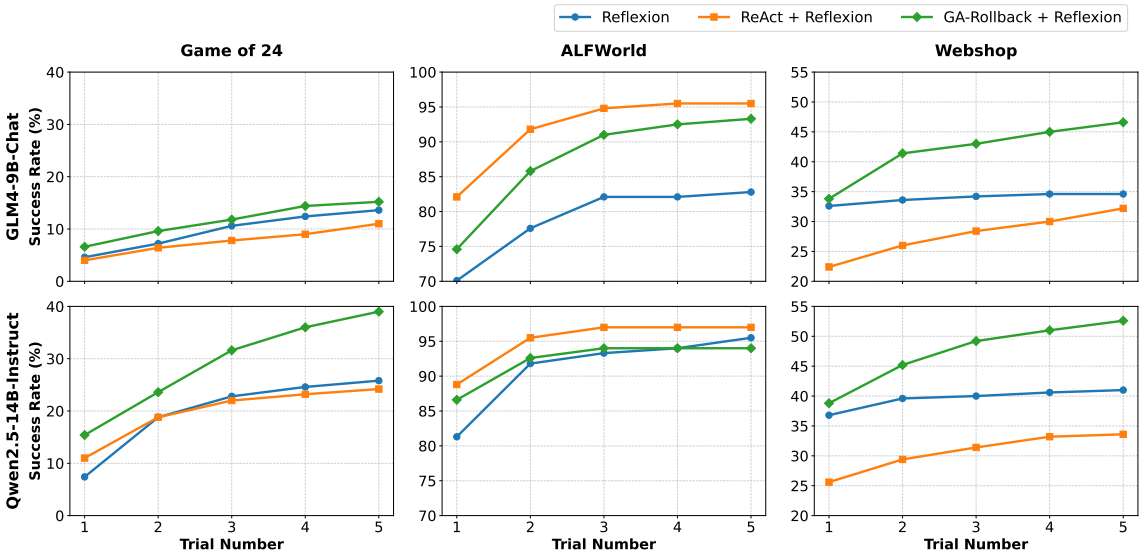


Figure 6: Comparison of success rate across trials for Reflexion, ReAct+Reflexion, and GA-Rollback+Reflexion.

scale LLMs such as GLM4-9B-Chat and Qwen2.5-14B-Instruct demonstrate faster convergence on ALFWorld, typically stabilizing their performance within five trials.

5 Conclusion

This study proposes GA-Rollback, a novel agent framework designed to mitigate the long-standing issue of error propagation, while enhancing the credibility of the reasoning process. GA-Rollback incorporates an independent assistant to support the generator in decision-making and utilizes rollback operations to eliminate potentially erroneous actions. Additionally, we implement probability-based feedback evaluation to improve the assistant's credibility and introduce Wait-Info strategy for embodied tasks. Extensive experiments across three benchmarks demonstrate that GA-Rollback outperforms several strong baselines and exhibits stronger robustness. Subsequent results validate the efficacy of each part within the framework. Moreover, our findings indicate that GA-Rollback can integrate seamlessly with other methods, underscoring its potential as a plug-and-play module for future applications.

Limitations

Despite the improvements achieved by our method, it is important to acknowledge several limitations:

(1) Even with the implementation of feedback evaluation and Wait-Info strategy, GA-Rollback has not yet achieved optimal performance on ALFWorld. This indicates that embodied tasks with longer trajectories pose significant challenges for error detection.

(2) The threshold θ for feedback filtering and the number of Wait-Info steps need to be determined through empirical experiments. Future work could focus on dynamically adjusting these parameters within a single trial.

(3) In real-world scenarios, certain environments may not support full rollback to previous states, which could limit the applicability of our method. To address this limitation in tasks involving irreversible actions, we recommend incorporating additional validation steps before executing such actions.

Acknowledgments

We would like to thank the anonymous reviewers and meta-reviewer for their insightful suggestions. This work is supported in part by the National Natural Science Foundation of China (62276077, 62406091, U23B2055, 62350710797), in part by the Guangdong Basic and Applied Basic Research Foundation under the Grant No. 2024A1515011205, in part by the Shenzhen Science and Technology Program (ZDSYS20230626091203008, KQTD20240291021540 66), and in part by the Shenzhen College Stability Support Plan (GXWD20220817123150002, GXWD20220811170358002). This work is also supported by Guangdong Major Project of Basic and Applied Basic Research (Grant No. 2023B0303000010), and is sponsored by CCF-Baidu Open Fund (CCF-Baidu 202404).

References

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

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Sijia Chen and Baochun Li. 2024. Toward adaptive reasoning in large language models with thought rollback. In *Forty-first International Conference on Machine Learning*.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2024. Teaching large language models to self-debug. In *The Twelfth International Conference on Learning Representations*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems*, volume 36, pages 28091–28114. Curran Associates, Inc.

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

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, et al. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*.

Zhibin Gou, Zhihong Shao, Yeyun Gong, yelong shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2024. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Jie Huang and Kevin Chen-Chuan Chang. 2022. Towards reasoning in large language models: A survey. *arXiv preprint arXiv:2212.10403*.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2023. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*.

Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992, Singapore. Association for Computational Linguistics.

Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. 2024. When can llms actually correct their own mistakes? a critical survey of self-correction of llms. *Transactions of the Association for Computational Linguistics*, 12:1417–1440.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pages 46534–46594. Curran Associates, Inc.

Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. 2024. REFINER: Reasoning feedback on intermediate representations. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1100–1126, St. Julian's, Malta. Association for Computational Linguistics.

Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, et al. 2023. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *arXiv preprint arXiv:2302.12813*.

Shuofei Qiao, Ningyu Zhang, Runnan Fang, Yujie Luo, Wangchunshu Zhou, Yuchen Eleanor Jiang, Chengfei Lv, and Huajun Chen. 2024. Autoact: Automatic agent learning from scratch via self-planning. *arXiv preprint arXiv:2401.05268*.

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

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

Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. 2022. Llm-planner: Few-shot grounded planning for embodied agents with large language models. *arXiv preprint arXiv:2212.04088*.

Yihong Tang, Kehai Chen, Xuefeng Bai, Zhengyu Niu, Bo Wang, Jie Liu, and Min Zhang. 2025. The rise of darkness: Safety-utility trade-offs in role-playing dialogue agents. *arXiv preprint arXiv:2502.20757*.

Han Wang, Archiki Prasad, Elias Stengel-Eskin, and Mohit Bansal. 2024. Soft self-consistency improves language model agents. *arXiv preprint arXiv:2402.13212*.

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

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022b. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. 2023. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv 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, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 11809–11822. Curran Associates, Inc.

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

Hongbin Zhang, Kehai Chen, Xuefeng Bai, Yang Xiang, and Min Zhang. 2024. Paying more attention to source context: Mitigating unfaithful translations from large language model. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 13816–13836.

Zhuosheng Zhang, Yao Yao, Aston Zhang, Xiangru Tang, Xinbei Ma, Zhiwei He, Yiming Wang, Mark Gerstein, Rui Wang, Gongshen Liu, et al. 2023. Igniting language intelligence: The hitchhiker's guide from chain-of-thought reasoning to language agents. *arXiv preprint arXiv:2311.11797*.

Ruochen Zhao, Xingxuan Li, Shafiq Joty, Chengwei Qin, and Lidong Bing. 2023. Verify-and-edit: A knowledge-enhanced chain-of-thought framework. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5823–5840, Toronto, Canada. Association for Computational Linguistics.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. 2024a. Language agent tree search unifies reasoning, acting, and planning in language models. In *Forty-first International Conference on Machine Learning*.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024b. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*.

Fei Zuo, Kehai Chen, Yu Zhang, Zhengshan Xue, and Min Zhang. 2025. InImageTrans: Multimodal LLM-based text image machine translation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20256–20277, Vienna, Austria. Association for Computational Linguistics.

A Task Details

Game of 24 Game of 24 (Yao et al., 2023a) is a mathematical reasoning challenge that tests the ability of agents to solve arithmetic puzzles. The task involves using four given numbers and basic

Method	Avg. Token (K)	Avg. Time (s)
Act-only	5.4	21.9
CoT	9.7	35.6
ReAct	9.6	30.1
Reflexion	7.2	32.2
ReAct + Reflexion	17.1	61.9
GA-Rollback	34.1	115.6
GA-Rollback + ReAct	40.8	162.0
GA-Rollback + Reflexion	49.7	172.7

Table 4: Comparison of token consumption and runtime across different methods on Webshop.

Figure 7: Impact of maximum rollback attempts on success rate and token consumption.

arithmetic operations ($+$, $-$, $*$, $/$) to obtain a target value of 24. The agent must generate intermediate equations as part of the problem-solving process, and the environment provides feedback on whether the final equation correctly equals 24. In our work, we make subtle modifications to the original environment, enabling it to provide corresponding observations for each action taken by the agent.

ALFWorld ALFWorld (Shridhar et al., 2020) presents a series of household tasks that challenge agents to navigate through rooms and apply commonsense reasoning to accomplish tasks, like "put two soapbar in garbagecan". The system evaluates the agent's performance by determining if the task is successfully completed within a specified number of steps. The ALFWorld dataset includes both seen and unseen evaluation sets: the seen set evaluates the agent's ability to generalize within the same distribution, while the unseen set, featuring new task scenarios, tests the agent's capacity for out-of-distribution generalization.

Webshop Webshop (Yao et al., 2022) is an online simulation environment designed to replicate e-commerce interactions. It features a virtual website containing 1.8 million real-world products, each annotated with unique labels and attributes. Within this environment, agents can perform actions such as "search" or "click" to navigate and select products that align with given instructions. When the agent chooses the "buy" option, the system computes a final reward based on how well the product's attributes and price match the specified criteria. Concretely, the reward is designed as:

$$r = r_{\text{type}} \cdot \frac{|U_{\text{att}} \cap Y_{\text{att}}| + |U_{\text{opt}} \cap Y_{\text{opt}}| + \mathbf{1}[y_{\text{price}} \leq u_{\text{price}}]}{|U_{\text{att}}| + |U_{\text{opt}}| + 1}, \tag{2}$$

where the type reward $r_{\text{type}} = \text{TextMatch}(\overline{y}, \overline{y}^*)$, U_{att} and U_{opt} denote the attributes and options of the target product, Y_{att} and Y_{opt} represent the attributes and options of the product selected by the agent.

B Additional Results

B.1 Consumption Analysis

Since rollback operations increase communication overhead and time complexity, we further compare the token consumption and runtime of GA-Rollback with other methods using Qwen2.5-14B-Instruct. As shown in Table 4, our GA-Rollback consumes approximately 5–6× more tokens and runtime than Act-only, but achieves significantly higher success rates. To balance success rate and computational cost, we set the maximum rollback attempts per task at 6. Figure 7 demonstrates that this configuration achieves the optimal trade-off between system performance and resource efficiency, maintaining high success rate while keeping computational cost within reasonable limits.

B.2 Performance on GPT-4o

To further validate the generalizability of GA-Rollback, we conduct experiments on the API-based model GPT-4o. We sample 200 instances from the Webshop benchmark to evaluate our approach and baseline methods. Table 5 shows the effectiveness of our approach.

Method	Webshop	
	Reward	SR
Act-only	51.0	31.0
ReAct	36.2	21.0
Reflexion	55.5	34.5
ReAct + Reflexion	40.6	25.5
GA-Rollback	47.4	32.0
GA-Rollback + ReAct	42.6	28.5
GA-Rollback + Reflexion	54.2	38.0

Table 5: Performance of different methods on GPT-4o.

Model	Method	Game of 24	ALFWorld	Webshop		Avg. SR
		SR	SR	Reward	SR	
Qwen2.5-3B-Instruct	Act-only	1.0	26.1	59.9	26.4	17.8
	CoT	0.0	37.3	62.2	30.6	22.6
	ReAct	1.6	44.0	39.4	17.2	20.9
	Reflexion	5.6	53.7	62.1	28.0	<u>29.1</u>
	ReAct + Reflexion	<u>5.8</u>	**59.0**	41.7	14.6	26.5
	GA-Rollback	5.4	41.8	<u>64.4</u>	<u>34.6</u>	27.3
	GA-Rollback + ReAct	3.4	56.7	57.8	25.6	28.6
	GA-Rollback + Reflexion	**6.4**	<u>57.5</u>	**67.4**	**39.6**	**34.5**
Qwen2.5-32B-Instruct	Act-only	11.0	83.6	60.4	32.4	42.3
	CoT	22.8	85.1	65.5	37.8	48.6
	ReAct	15.8	76.9	41.3	25.4	39.4
	Reflexion	22.4	<u>92.5</u>	63.7	34.8	<u>49.9</u>
	ReAct + Reflexion	<u>25.4</u>	90.3	44.2	24.4	46.7
	GA-Rollback	20.8	88.1	<u>66.8</u>	<u>38.2</u>	49.0
	GA-Rollback + ReAct	23.2	76.1	42.4	24.8	41.4
	GA-Rollback + Reflexion	**33.0**	**94.8**	**67.1**	**43.2**	**57.0**

Table 6: The best results of each model are marked in **bold** and the second-best results are marked with <u>underline</u>.

B.3 Results on Different Model Sizes

To evaluate the generalizability of our proposed method, we conduct experiments across a broader range of model sizes. As presented in Table 6, the results demonstrate that GA-Rollback consistently brings stable performance gains on both 3B and 32B models. This observation aligns with the trends observed in our main experiments on 8B to 14B models. Moreover, as the model size increases, the overall performance of GA-Rollback improves across all three tasks. These findings confirm the robustness and scalability of our approach in enhancing reasoning accuracy across models of varying sizes.

C More Cases

Here, we present more examples to demonstrate the assistant's analysis process on Game of 24 and ALFWorld. The examples are illustrated in Figure 8 and 9, respectively.

In the **Game of 24** example, the assistant conducts a thorough analysis and recognizes that the current approach would not yield the desired result of 24. Therefore, it decides to modify the initial step and develops a comprehensive strategy to guide the generator.

In the **ALFWorld** scenario, although the generator locates both "*alarmclock 1*" and "*desklamp 1*", it forgets to pick up "*alarmclock 1*". Through a step-by-step inspection, the assistant identifies this oversight and triggers a rollback operation, effectively preventing subsequent invalid actions. These two examples demonstrate the assistant's critical role in identifying and rectifying errors, which contributes significantly to optimizing the overall trajectory of task execution.

Furthermore, in Figure 10, we present an example of a conflict between ReAct and GA-Rollback. In this case, the generator concludes that none of the retrieved items match the requirement "*redwood color*", while the assistant maintains that a check of at least one item is necessary before making such a determination. This divergence in judgment leads to redundant rollback operations, which could potentially disrupt subsequent actions.

Case Study for Game of 24

Observation 0:
Here is the task:
Input: 1 2 10 13
Action 1: 13 - 10 = 3
Observation 1: numbers left: 1 2 3
Action 2: 3 * 2 = 6
Observation 2: numbers left: 1 6

Analysis Process by Assistant:
Let's analyze the current trajectory step by step:
- Action 1: 13 - 10 = 3
  - This operation is valid and reduces the numbers to 1, 2, and 3.
- Action 2: 3 * 2 = 6
  - This operation is also valid and further reduces the numbers to 1 and 6.

However, **the trajectory ends here without reaching the target value of 24.** We need to continue the steps to achieve the final result.

Next Steps:
To reach 24, we need to use the remaining numbers (1 and 6) in a way that results in 24. One possible continuation could be:
- Action 3: 6 * 1 = 6 (This doesn't help, so we need another approach)
- Alternatively, we can try:
  - Action 3: 6 + 1 = 7 (Still not helping, so we need another approach)

Since the current trajectory does not lead to 24, we need to re-evaluate the initial steps to find a more effective path.

Correct Approach:
Given the numbers 1, 2, 10, and 13, one effective way to reach 24 is:
- Action 1: 13 - 1 = 12
- Action 2: 12 + 2 = 14
- Action 3: 14 + 10 = 24

Thus, the correct trajectory should be:
- Action 1: 13 - 1 = 12
- Action 2: 12 + 2 = 14
- Action 3: 14 + 10 = 24
- Action 4: answer: (13 - 1) + 2 + 10 = 24

Conclusion
Error Location: Action 1
Explanation: The initial subtraction in Action 1 (13 - 10 = 3) led to a dead end because it made it difficult to reach 24 with the remaining numbers. A better approach would be to subtract 1 from 13 to get 12, then add 2 and 10 to reach 24. **The correct expression should be: "answer: (13 - 1) + 2 + 10 = 24".** This ensures that all numbers are used correctly to achieve the target value of 24.

– Rollback Happened –
Action 1: 13 - 1 = 12

Figure 8: Case study for Game of 24.

Case Study for ALFWorld

Observation 0:
You are in the middle of a room. Looking quickly around you, you see a bed 1, a desk 2, 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 laundryhamper 1, a safe 1, a shelf 6, a shelf 5, a shelf 4, a shelf 3, a shelf 2, and a shelf 1.
Your task is to: examine the alarmclock with the desklamp.
Action 1: go to desk 1
Observation 1: On the desk 1, you see a bowl 1, a desklamp 1, and a laptop 2.
Action 2: use desklamp 1
Observation 2: You turn on the desklamp 1.
Action 3: go to desk 2
Observation 3: On the desk 2, you see a alarmclock 1, a keychain 2, and a mug 1.
Action 4: examine alarmclock 1 with desklamp 1
Observation 4: Nothing happens.
Action 5: go to desk 1
Observation 5: On the desk 1, you see a bowl 1, a desklamp 1, and a laptop 2.
Action 6: take desklamp 1 from desk 1
Observation 6: Nothing happens.
Action 7: use desklamp 1
Observation 7: You turn on the desklamp 1.

Analysis Process by Assistant:
Let's analyze it step by step from back to front.
Step-by-Step Analysis
1. Action 7: use desklamp 1
   - Observation 7: You turn on the desklamp 1.
   - This action is valid, but it does not help in examining the alarmclock.
2. Action 6: take desklamp 1 from desk 1
   - Observation 6: Nothing happens.
   - This indicates that the desklamp 1 cannot be taken from the desk 1. This could be because the desklamp is fixed or not designed to be moved.
3. Action 5: go to desk 1
   - Observation 5: On the desk 1, you see a bowl 1, a desklamp 1, and a laptop 2.
   - This confirms that the desklamp 1 is still on the desk 1.
4. Action 4: examine alarmclock 1 with desklamp 1
   - Observation 4: Nothing happens.
   - **This indicates that the action is invalid. The desklamp 1 must be used in a different way or the alarmclock 1 must be brought to the desklamp 1. .....**

Conclusion
Error Location: Action 4
Explanation: The desklamp 1 cannot be used directly to examine the alarmclock 1 while it is on desk 2. **Instead, I should take the alarmclock 1 from desk 2 and bring it to desk 1 where the desklamp 1 is located.** After that, I can use the desklamp 1 to examine the alarmclock 1.

– Rollback Happened –
Action 4: take alarmclock 1 from desk 2

Figure 9: Case study for ALFWorld.

A Conflict Example of ReAct and GA-Rollback

Observation 0:
i'm looking for a queen size bedspread set in the color redwood, and price lower than 60.00 dollars
[Search]
Action 1: search[queen size bedspread set redwood]
Observation 1:
[Back to Search]
Page 1 (Total results: 50)
[Next >]
[B07B4KXQZV]
Lodge Bedspread Full/Queen Size Quilt with 2 Shams. Cabin 3-Piece Reversible All Season Quilt Set. Rustic Quilt Coverlet Bed Set. Stonehurst Collection.
$44.99
[B07PGQ3JYB]
Lodge Bedspread Full/Queen Size Quilt with 2 Shams. Cabin 3-Piece Reversible All Season Quilt Set. Rustic Quilt Coverlet Bed Set. Wilderness Collection (Moose - Grey)
$44.99
[B07WZBXSK3]
Travan 3-Piece Queen Quilt Set Quilted Bedspread Oversized Microfiber Lightweight Bedspread Coverlet Set with Shams Bedding Set for All Season, Grey, Queen
$64.99
Action 2: think[None of the products returned in the search results match the color redwood. I should consider searching for similar colors or broader terms.]
Observation 2: OK.

Analysis Process by Assistant:
Let's analyze it step by step from back to front.
- Observation 2 shows that the agent is thinking about checking for similar colors since there are no items matching the exact color redwood. However, the agent hasn't taken any action yet.
- Observation 1 shows that there are three options available, but none of them are in the color redwood. The agent should proceed to check for similar colors as mentioned in Observation 2.
Conclusion
Error Location: Action 2
Explanation: **The agent thought about checking for similar colors but did not take any action. The agent should click on one of the products and check if the color is similar to redwood or modify the search query to include similar colors.** For example, the agent could click on **[B07B4KXQZV]** and check if the color is similar to redwood or search for "queen size bedspread set redwood similar".

Figure 10: A conflict example of ReAct and GA-Rollback.

D Prompt for Assistant

The instruction prompts specifically designed for the assistant in Game of 24, ALFWorld, and Webshop are presented in Figures 11, 12, and 13, respectively.

Instruction Prompt for the Assistant in Game of 24

 # Task Description
You will analyze trajectories for potential errors and provide analysis in a standardized format.

 # Analysis Rules
1. Focus on Error Detection: Analyze each step in the trajectory to identify errors, regardless of the current task status.

2. Analysis Method:
- Analyze the given trajectory step by step
- Identify where the trajectory first deviated from correct behavior

3. Required Checks:
- If the number obtained after three operations is not 24, the calculation process should be changed
- If the final result is 24, then it should be displayed as a complete expression using the four numbers in the input, such as "answer: (6 - 4) * (4 + 8) = 24"

4. Output Must Include:
- ** Error Location **: Specific step where error occurred
- ** Explanation **: Error explanation and correction method

 # Examples
{Analysis Examples}

 # Current Task
{Trajectory to be analyzed}

Figure 11: Instruction prompt for the assistant in Game of 24.

Instruction Prompt for the Assistant in ALFWorld

 # Task Description
You will analyze trajectories for potential errors and provide analysis in a standardized format.

 # Analysis Rules
1. Focus on Error Detection: Analyze each step in the trajectory to identify errors, regardless of the current task status.

2. Analysis Method:
- Start from the final outcome
- Work backwards through each action
- Identify where the trajectory first deviated from correct behavior

3. Required Checks:
- Verify that you have reached the correct location before picking up objects

- Confirm destination exists in the environment before navigation
- If the action 'go to [location]' results in 'Nothing happens', it means you are already at that location or the location doesn't exist in the environment

4. Output Must Include:
- ** Error Location **: Specific step where error occurred
- ** Explanation **: Error explanation and correction method

 # Examples
{Analysis Examples}

 # Current Task
{Trajectory to be analyzed}

Figure 12: Instruction prompt for the assistant in ALFWorld.

Instruction Prompt for the Assistant in Webshop

 # Task Description
You will analyze trajectories for potential errors and provide analysis in a standardized format.

 # Analysis Rules
1. Focus on Error Detection: Analyze each step in the trajectory to identify errors, regardless of the current task status.

2. Analysis Method:
- Start from the final outcome
- Work backwards through each action
- Identify where the trajectory first deviated from correct behavior

3. Required Checks:
- Verify all required product attributes are selected
- Confirm specifications match the user's requirements
- Ensure necessary clicks/selections are made before purchase

4. Output Must Include:
- ** Error Location **: Specific step where error occurred
- ** Explanation **: Error explanation and correction method

 # Examples
{Analysis Examples}

 # Current Task
{Trajectory to be analyzed}

Figure 13: Instruction prompt for the assistant in Webshop.