From Chains to Cycles: Improving LLM Graph Reasoning with a Reason-Retrospect-Replan Approach

Rong Zhou[1*], Duanyang Yuan[1*], Sihang Zhou[1†], Wenxuan TU[2†], Yanning Hou[1], Hao Chen[1], Quan Liu[1], Jian Huang[1]

[1] College of Intelligence Science and Technology, National University of Defense Technology, Changsha, China
[2] School of Computer Science and Technology, Hainan University, Hainan, China

*These authors contributed equally.
†E-mail: shzhou@nudt.edu.cn, twx@hainanu.edu.cn

Abstract. Knowledge Graph Question Answering (KGQA) aims to answer complex questions by integrating Large Language Models (LLMs) with structured facts from Knowledge Graphs (KGs). To enhance reasoning robustness, existing methods often employ step-by-step reflection, where the model adjusts its plan after each action by appending thoughts and observations to its context. However, this strategy can lead to excessive context length when handling long-chain reasoning tasks, thereby hindering model performance. To bridge this gap, we propose R3-Cycle, a framework that employs a Reason-Retrospect-Replan cycle. R3-Cycle first guides an agent through an initial round of reasoning, during which a memory module captures the complete reasoning trajectory. If this round fails, a dedicated reflection module is activated. It performs a global retrospective analysis of the trajectory to diagnose the root cause of the failure and generate a clear rectification plan. This plan then guides the model to execute a second round of reasoning, managed by an adaptive step allocation mechanism, in a reset context to complete the task. Experiments on the GRBENCH benchmark show R3-Cycle significantly outperforms the Graph-CoT baseline (e.g., 37.5 vs. 28.7 ROUGE-L on Goodreads). These gains are consistent across both closed-source and open-source models, and are achieved at a manageable computational cost, validating the effectiveness and practical viability of our global retrospective approach.

1. Introduction

Large Language Models (LLMs) have demonstrated remarkable performance in various natural language processing and data science tasks, particularly in Question Answering (QA), owing to their exceptional language understanding and generation capabilities. However, inherent limitations of LLMs, such as outdated knowledge, factual hallucinations, and opaque decision-making processes, severely restrict their application in complex tasks demanding precise and verifiable knowledge. Knowledge Graphs (KGs), as large-scale structured knowledge bases, offer explicit and editable facts, providing a powerful means to mitigate these shortcomings. This has led to the emergence of Knowledge Graph Question Answering (KGQA) as a significant research direction.

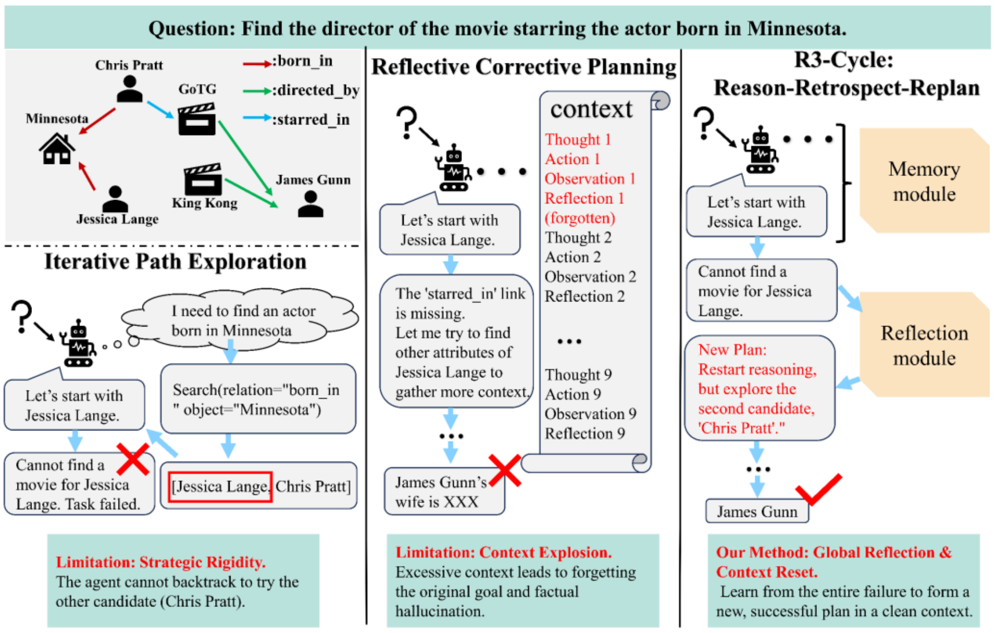

Question: Find the director of the movie starring the actor born in Minnesota.
Chris Pratt
GoTG
Minnesota
King Kong
James Gunn
Jessica Lange
:born_in
:directed_by
:starred_in
Iterative Path Exploration
I need to find an actor born in Minnesota
Let's start with Jessica Lange.
Search(relation="born_in" object="Minnesota")
Cannot find a movie for Jessica Lange. Task failed.
[Jessica Lange, Chris Pratt]
Limitation: Strategic Rigidity.
The agent cannot backtrack to try the other candidate (Chris Pratt).
Reflective Corrective Planning
context
Thought 1
Action 1
Observation 1
Reflection 1
(forgotten)
Thought 2
Action 2
Observation 2
Reflection 2
Thought 9
Action 9
Observation 9
Reflection 9
Let's start with Jessica Lange.
The 'starred_in' link is missing. Let me try to find other attributes of Jessica Lange to gather more context.
James Gunn's wife is XXX
Limitation: Context Explosion.
Excessive context leads to forgetting the original goal and factual hallucination.
R3-Cycle: Reason-Retrospect-Replan
Let's start with Jessica Lange.
Memory module
Cannot find a movie for Jessica Lange.
Reflection module
New Plan: Restart reasoning, but explore the second candidate, 'Chris Pratt'.
James Gunn
Our Method: Global Reflection & Context Reset.
Learn from the entire failure to form a new, successful plan in a clean context.

Figure 1. An illustration comparing R3-Cycle with two existing paradigms on a multi-hop KGQA task.

To leverage KGs for enhancing the reasoning capabilities of LLMs, a core technical direction is to use them as "Reasoning Guidelines". This direction has evolved through several paradigms, distinguished by the LLM's autonomy and error-handling capabilities. Initial approaches employed Static Knowledge Injection, where the KG acts as a one-time knowledge provider. In these methods, an independent module retrieves a static subgraph or a set of paths before the LLM begins its final reasoning. For instance, SR (Zhang et al., 2022) trains an independent subgraph retriever, while Keqing (Wang et al., 2023) uses predefined templates to decompose questions and retrieve candidate knowledge. The limitation of this approach is that the LLM passively receives knowledge and cannot directly participate in the dynamic exploration of the graph, making the reasoning process inflexible.

To overcome this inflexibility, subsequent work has focused on Iterative Path Exploration. This paradigm positions the LLM as an active agent that progressively explores reasoning paths on the KG through a multi-step "thought-action" loop. A representative work in this category is GRAPH-COT, which employs a "Thought-Action-Observation" cycle for iterative reasoning. However, a common problem with this paradigm is its strategic rigidity. Rooted in its nature as Forward-Chaining Search, when a reasoning path is obstructed by fundamental issues like KG incompleteness, the agent lacks the ability to review and adjust its overall strategy. It can only abandon the failing path rather than learning from the failure to try entirely new alternatives, often leading to premature termination or deadlock.

To address the strategic rigidity of pure exploration, the Reflective Corrective Planning paradigm emerged, enabling the agent to self-correct during reasoning. A representative work in this field is Plan-on-Graph (PoG) (Chen et al., 2024a), which introduces memory and reflection mechanisms to allow the LLM to backtrack and re-explore when it deems information insufficient. Despite this progress, such step-wise local adjustment methods face a critical challenge in complex tasks: as the number of reasoning steps increases, each "thought-action-observation-reflection" cycle is appended to the model's context, causing the context length to expand dramatically. When the context becomes too long, the LLM is prone to the "lost in the middle" problem, where it may ignore or forget crucial early information, thereby significantly increasing the risk of factual hallucinations.

To resolve these issues, we propose the Reason-Retrospect-Replan Cycle (R3-Cycle). The framework first executes an initial reasoning round, with a memory module capturing the entire trajectory. If this attempt fails, a reflection module performs a global retrospective analysis on the trajectory to diagnose the failure's root cause and generate a rectification plan. This plan then guides a second reasoning attempt in a clean, reset context, managed by an adaptive step allocation mechanism that adjusts the reasoning budget based on the initial failure type. This cycle-based approach prevents excessive context growth and enhances reasoning robustness in complex, long-chain tasks.

Our main contributions are as follows:

(1) We propose R3-Cycle, a novel graph reasoning framework that overcomes the limitations of both rigid forward-chaining and context-heavy step-wise reflection. It introduces a global, cycle-based correction mechanism that operates in a clean context to ensure robust and efficient problem-solving.

(2) We design a complete "Reason-Retrospect-Replan" workflow, featuring: (a) a memory module for capturing the full trajectory, (b) a reflection module for global, root-cause analysis, and (c) an adaptive step allocation mechanism to intelligently guide the re-reasoning process.

(3) Extensive experiments on the GRBENCH benchmark demonstrate that R3-Cycle significantly outperforms the strong Graph-CoT baseline (e.g., 37.5 vs. 28.7 ROUGE-L on Goodreads). We further validate that these performance gains are model-agnostic, generalizing to open-source models, and are achieved with a manageable computational overhead, confirming the framework's practical viability.

2. Methodology

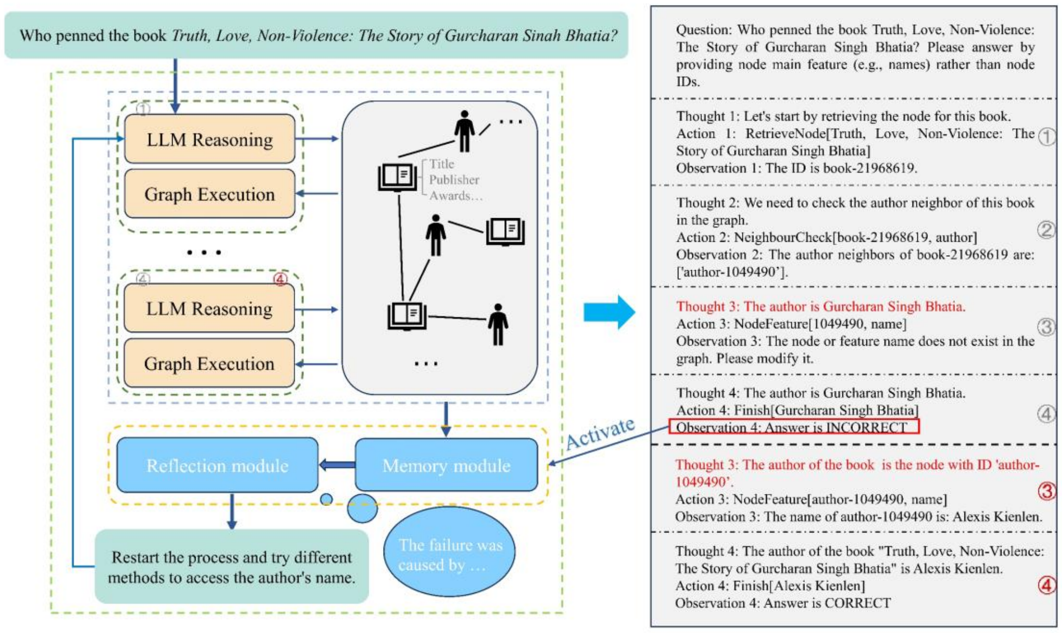
Our proposed framework, R3-Cycle, is a Graph Chain-of-Thought framework enhanced with reflection and re-planning capabilities. R3-Cycle is designed to improve the reasoning robustness and accuracy of Large Language Models (LLMs) on complex Knowledge Graph Question Answering (KGQA) tasks. R3-Cycle builds upon the iterative agent-based framework of Graph-CoT and integrates meta-cognitive principles. Specifically, we design and implement three core enhancement modules: a Memory Module to record the reasoning trajectory, a Reflection and Re-planning Module that activates upon reasoning failures to self-correct, and an Adaptive Step Allocation mechanism to intelligently manage retries. This combination enables the LLM agent not only to interactively explore the graph but also to self-critique, correct its errors, and initiate a new, targeted round of reasoning, thereby significantly improving its ability to solve complex, multi-hop problems.

As illustrated in Figure 2, the overall workflow of R3-Cycle is a dynamic and rectifiable iterative loop. The process continues until the LLM agent deems the collected information sufficient to generate a final answer or a dynamically determined retry limit is reached. Each iteration consists of the following core components:

(1)**Core Reasoning Loop**: Comprising Reasoning, Interaction, and Execution, this loop constitutes the primary exploration action on the graph.

(2)**Memory Updating**: After each successful execution, this component records the current reasoning state and newly acquired knowledge.

(3) **Reflection and Adaptive Re-planning**: Triggered when the reasoning process fails, this component analyzes the cause of failure and orchestrates an intelligent, adaptive re-reasoning attempt.


Who penned the book *Truth, Love, Non-Violence: The Story of Gurcharan Sinah Bhatia?*
LLM Reasoning
Graph Execution
Title
Publisher
Awards…
LLM Reasoning
Graph Execution
Reflection module
Memory module
Activate
Restart the process and try different methods to access the author's name.
The failure was caused by …

Question: Who penned the book Truth, Love, Non-Violence: The Story of Gurcharan Singh Bhatia? Please answer by providing node main feature (e.g., names) rather than node IDs.

Thought 1: Let's start by retrieving the node for this book.
Action 1: RetrieveNode[Truth, Love, Non-Violence: The Story of Gurcharan Singh Bhatia]
Observation 1: The ID is book-21968619.

Thought 2: We need to check the author neighbor of this book in the graph.
Action 2: NeighbourCheck[book-21968619, author]
Observation 2: The author neighbors of book-21968619 are: ['author-1049490'].

Thought 3: The author is Gurcharan Singh Bhatia.
Action 3: NodeFeature[1049490, name]
Observation 3: The node or feature name does not exist in the graph. Please modify it.

Thought 4: The author is Gurcharan Singh Bhatia.
Action 4: Finish[Gurcharan Singh Bhatia]
Observation 4: Answer is INCORRECT

Thought 3: The author of the book is the node with ID 'author-1049490'.
Action 3: NodeFeature[author-1049490, name]
Observation 3: The name of author-1049490 is: Alexis Kienlen.

Thought 4: The author of the book "Truth, Love, Non-Violence: The Story of Gurcharan Singh Bhatia" is Alexis Kienlen.
Action 4: Finish[Alexis Kienlen]
Observation 4: Answer is CORRECT

Figure 2. The workflow of R3-Cycle, detailing its components and self-correction mechanism.

2.1 Core Reasoning Loop

We adhere to the Graph-CoT paradigm by modeling the LLM as an intelligent agent that interacts with the knowledge graph environment. Its fundamental exploration process is a concise and effective three-step loop.

(1)**Reasoning**: At the beginning of each iteration, the LLM acts as a "thinker." Based on the initial question and the observation from the previous step, it generates a natural language "Thought" to analyze the current state, decompose the problem, and plan the next action.

(2)**Interaction**: The LLM translates this "Thought" into a concrete, machine-executable "Action." This is achieved by prompting the LLM to generate a call to a predefined graph function. We provide the agent with four standardized tools that constitute its action space for graph interaction:

[RetrieveNode]: Retrieves the node for a given entity name.

[NodeFeature]: Obtains the value of a specific attribute for a given node.

[NeighbourCheck:] Lists the neighbors of a node connected by a specific relation.

[NodeDegree]: Counts the number of neighbors for a node under a specific relation.

(3)**Execution**: The system executes the function call generated by the LLM on the knowledge graph. The result of this operation is formatted as an "Observation" and fed back to the LLM. This observation serves as the critical input for the next "Reasoning" step, thereby closing the loop.

This cycle repeats until the agent determines it has gathered sufficient information to conclude the task. At this point, it generates a final "Action" to provide the answer. In its final "Thought," the agent synthesizes all information gathered throughout its exploration to confidently formulate the answer. This process continues until the agent deems the collected information sufficient to generate a final answer. R3-Cycle's design ensures that the final answer is grounded in evidence from the knowledge graph, rather than being a product of the model's hallucination.

2.2 Memory Module

To provide a foundation for higher-level cognitive activities like reflection, we introduce a persistent Memory Module. This module is invoked after each successful execution step to record the complete state of the reasoning process. It stores two primary types of information:

First, **Complete Reasoning Log**: An archive of the entire reasoning trajectory, including all "Thoughts," "Actions," and "Observations." Second, **Structured Knowledge Path**: An extraction of the key knowledge acquisition path from the interaction steps, representing the agent's exploration trajectory in a standardized format.

2.3 Reflection and Adaptive Re-planning

This is the core innovation of R3-Cycle, endowing the agent with the ability to self-correct and re-reason from failures. When an initial reasoning attempt fails, the reflection module is automatically triggered. The reflection process is initiated via a specially designed prompt that provides the LLM with the original question, the incorrect final answer, and the complete reasoning log and structured knowledge path from the Memory Module. The LLM is required to generate a structured output containing a root cause diagnosis, specific corrective advice, and a judgment on whether to initiate a retry.

If a retry is warranted, the system initiates a re-reasoning process. This process resets the agent's state but injects the reflective guidance into its initial context. This guidance, containing the cause analysis and corrective advice, aims to steer the LLM away from previous mistakes. To move beyond fixed retry limits and enhance the intelligence of this process, we introduce an adaptive step allocation mechanism. This mechanism dynamically adjusts the step limit for the retry attempt based on the nature of the initial failure, allowing R3-Cycle to tailor its computational budget to the task's complexity. The mechanism classifies initial failures into two types:

Incorrect Answer: The process finished but yielded a wrong answer. This indicates a flawed reasoning path, not an insufficient budget. Thus, the retry is allocated the same step limit.**Halted Reasoning**: The process exceeded the step limit without an answer. This suggests the task requires deeper exploration and the initial step limit was insufficient. Therefore, the retry is granted an extended budget (+50%).

This dynamic strategy ensures R3-Cycle intelligently allocates resources, preventing premature termination on complex queries while avoiding wasteful exploration on simpler ones. It enables R3-Cycle to learn from its mistakes and adjust its reasoning strategy more effectively. As validated by our ablation studies, this complete framework demonstrates superior performance and robustness.

3. Experiments

To comprehensively evaluate the effectiveness and practicality of R3-Cycle, we conducted a series of extensive experiments. Our experiments are designed to answer the following key research questions:

(1) How do the Memory, Reflection, and Adaptive Step Allocation modules contribute to the performance of the baseline Graph-CoT framework?

(2) Is R3-Cycle's effectiveness model-agnostic? Can its benefits generalize from proprietary models like GPT to open-source alternatives?

(3) What is the computational overhead introduced by the reflection cycle, and is it a worthwhile trade-off for the performance gains?

Table 1. Performance Comparison (ROUGE-L Score) of R3-Cycle Against Baseline Models

Backbone LLM	Dataset	LLM	textRAG	1-hop RAG	2-hop RAG	Graph-Cot	**R3-Cycle**
GPT-3.5-turbo	Amazon	10.7	16.9	22.8	19.3	21.7	**36.9**
	Biomedical	8.7	7.3	12.7	7.0	5	**29.8**
	Goodreads	9.7	11.3	13.9	15.0	28.7	**37.5**
GPT-4o-mini	Amazon	11.2	19.9	27.9	22.5	38	**42.0**
	Biomedical	8.1	7.4	13.9	7.8	37.6	**39.8**
	Goodreads	7.4	13.7	18.0	17.7	40.4	**44.1**
Qwen2.5-7b-Instruct	Amazon	9.1	19.1	24.6	22.8	34.2	**36.7**
	Biomedical	9.2	7.4	10.5	8.7	26.4	**33.7**
	Goodreads	6.6	11.5	17.6	16.9	30.9	**36.6**

3.1 Experimental Setup

The detailed experimental setup is as follows:

Datasets: We conduct our experiments on the GRBENCH benchmark, following the setup of the original Graph-CoT paper. For a representative evaluation, we selected three graphs from distinct domains: Amazon (e-commerce), Biomedical (healthcare), and Goodreads (literature).

Evaluation Metrics: We adopt ROUGE-L (R-L) as our primary evaluation metric to measure the lexical overlap between the model-generated answer and the ground-truth answer. It is a widely used metric in text generation and question answering tasks, effectively reflecting the content accuracy of the generated answers.

Implementation Details: We conduct experiments using GPT-3.5-turbo, GPT-4o-mini, and Qwen2.5-7b-Instruct. All experimental hyperparameters are kept consistent with those in the original Graph-CoT paper to ensure a fair comparison.

Ablation Study Methods: To systematically validate the contribution of each component in our framework, we designed the following model variants for our ablation study:

[Original]: The baseline Graph-CoT framework.

[Original + Memory]: The baseline framework augmented solely with the Memory module.

[Original + Reflection]: The baseline framework augmented with the Reflection module but lacking persistent memory.

[Original + Mem+Ref]: This variant integrates both Memory and Reflection modules but utilizes a fixed retry limit. It serves as a direct ablation for our adaptive mechanism.

[R3-Cycle]: Our proposed full framework, which integrates all modules, including the Adaptive Step Allocation mechanism.

Table 2. Ablation study results for the Memory and Reflection modules (ROUGE-L Score).

Backbone LLM	Dataset	Original	Original + Memory	Original + Reflection	Original + Mem+Ref	**R3-Cycle**
GPT-3.5-turbo	Amazon	21.7	24.9	32.4	35.4	**36.9**
	Biomedical	5.0	6.2	10.3	13.4	**29.8**
	Goodreads	28.7	29.7	35.6	37.2	**37.5**
GPT-4o-mini	Amazon	38.0	38.2	38.8	40.4	**42.0**
	Biomedical	37.6	37.7	37.9	37.9	**39.8**
	Goodreads	40.4	41.1	42.3	43.2	**44.1**

3.2 Comparison with Baselines

To situate R3-Cycle within a broader academic context, we conduct a comparative analysis against a series of representative baseline models:

LLM: Utilizes only the large language model's internal knowledge to answer questions, without any external information retrieval.

textRAG: A standard retrieval-augmented generation paradigm that retrieves relevant snippets from a text corpus via vector similarity to aid in generation.

1-hop RAG: A simplified graph reasoning method that only performs a single-hop exploration of neighboring nodes and retrieves information from them.

2-hop RAG: Builds upon 1-hop RAG by extending the exploration to two hops, simulating a deeper level of graph reasoning.

Graph-CoT: The baseline Graph-CoT framework.

As shown in Table 1, R3-Cycle consistently and significantly outperforms all baseline models across all datasets and LLM configurations. The performance gains are particularly pronounced in complex reasoning scenarios. For instance, on the Goodreads dataset with GPT-3.5-turbo, R3-Cycle achieves a ROUGE-L score of 37.5, substantially outperforming Graph-CoT's 28.7 and more than doubling the 15.0 score of the best-performing RAG baseline. This result strongly validates the superiority of our global retrospective correction approach.

Furthermore, experiments with the open-source Qwen2.5-7b-Instruct model demonstrate the model-agnostic benefits of our framework. On the Goodreads dataset, R3-Cycle boosts the performance of Qwen from 30.9 (Graph-CoT) to 36.6, showcasing a similar performance uplift as observed with GPT models. This confirms that the advantages of R3-Cycle stem from its robust framework design rather than being tied to specific proprietary model architectures.

3.3 Ablation Study

To analyse the contribution of each component, we conducted a detailed ablation study with results presented in Table 2. The analysis reveals a clear, hierarchical contribution from our proposed modules.

Reflection is the primary driver of performance. Across all scenarios, adding the Reflection module (Original + Reflection) yields a larger improvement than adding Memory alone, demonstrating that empowering the agent with self-assessment is crucial.

Table 3. Efficiency and Time Comparison on Qwen2.5-7b-Instruct. Resource utilization reflects the average percentage during execution.

Backbone LLM	Dataset	Method	CPU	GPU	Time/**Retry Time**
Qwen2.5-7b-Instruct	Amazon	Graph-CoT	24.6	46.2	14.3
		R3-Cycle	**22**	**38.3**	**18.2**
	Biomedical	Graph-CoT	1.6	48.2	11.9
		R3-Cycle	**0.5**	**48.6**	**18.9**
	Goodreads	Graph-CoT	18.7	40.1	9.1
		R3-Cycle	**12.1**	**40.3**	**17.3**

Memory and reflection are complementary. The [Original + Mem+Ref] variant, which combines both modules, consistently outperforms variants with only one. The Memory module provides the Reflection process with the critical historical context of the entire reasoning trajectory, enabling the agent to 'learn from its mistakes' more effectively and making the reflection process more targeted.

Adaptive allocation provides the final performance edge. Our full R3-Cycle model, equipped with the adaptive step allocation mechanism, consistently achieves the best results. For example, on the challenging Biomedical dataset, the adaptive strategy elevates the score from 13.4 (Original + Mem + Ref) to 29.8 (R3-Cycle). This significant jump validates that intelligently adjusting the retry budget based on failure type is superior to a fixed-limit approach.

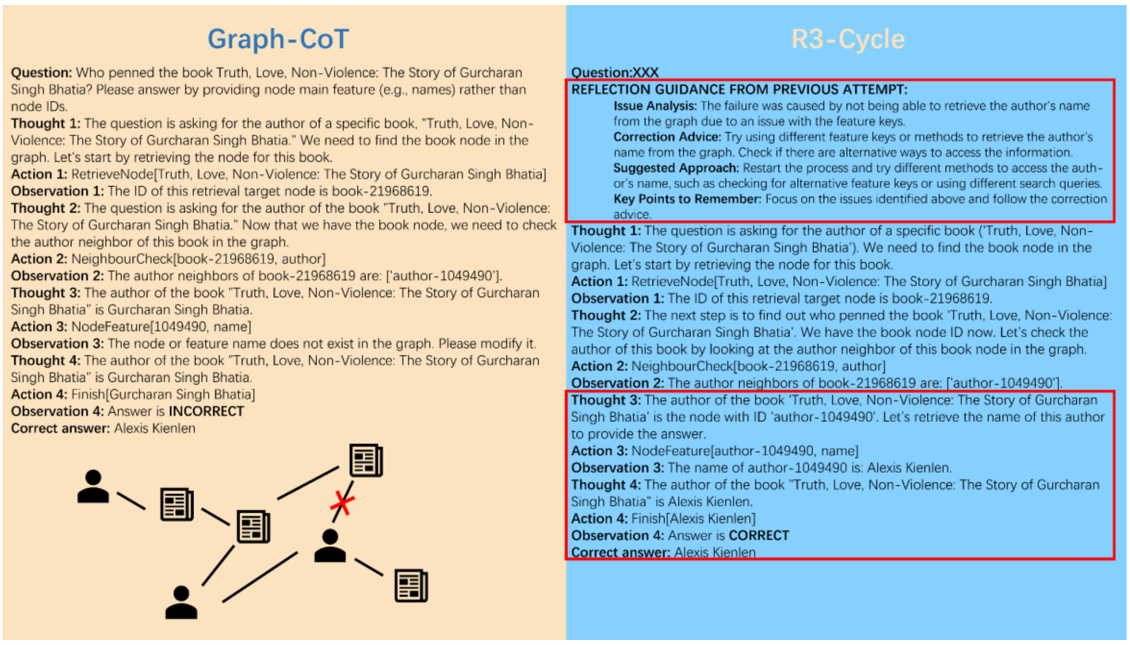
3.4 Efficiency and Cost Analysis

We analysed the computational overhead of R3-Cycle by measuring Retry-Time and CPU/GPU utilization (Table 3). The analysis reveals that the retry phase introduces a manageable latency (e.g., 18.2s on Amazon), which we frame as a deliberate trade-off: investing a modest, predictable time cost for a significant gain in accuracy. Crucially, the reflection process itself is computationally lightweight. Average CPU and GPU usage is comparable to the baseline, indicating the overhead stems from LLM inference latency rather than heavy system load. This efficient design confirms R3-Cycle's practical viability for complex tasks where correctness is important.

3.5 Case Study

To visually demonstrate R3-Cycle's enhanced reasoning robustness, we present a case study where the question contains a "trap" entity designed to mislead the model: "Who penned the book Truth, Love, Non-Violence: The Story of Gurcharan Singh Bhatia?" The challenge lies in resisting the hallucination that "Gurcharan Singh Bhatia" is the author and instead identifying the true author through KG exploration.

As illustrated in Figure 3, the reasoning paths of the baseline Graph-CoT and R3-Cycle diverge critically at the third step. Initially, both frameworks successfully identify the book's node ID and the associated author's node ID. However, the baseline agent is then distracted by the name in the question, incorrectly concluding the author is "Gurcharan Singh Bhatia." More critically, even when faced with explicit negative feedback from the environment (Observation 3: The node or feature name does not exist...), its rigid, unidirectional reasoning process prevents it from recovering, leading to an incorrect final answer.



Question: Who penned the book Truth, Love, Non-Violence: The Story of Gurcharan Singh Bhatia? Please answer by providing node main feature (e.g., names) rather than node IDs.
Thought 1: The question is asking for the author of a specific book, "Truth, Love, Non-Violence: The Story of Gurcharan Singh Bhatia." We need to find the book node in the graph. Let's start by retrieving the node for this book.
Action 1: RetrieveNode[Truth, Love, Non-Violence: The Story of Gurcharan Singh Bhatia]
Observation 1: The ID of this retrieval target node is book-21968619.
Thought 2: The question is asking for the author of the book "Truth, Love, Non-Violence: The Story of Gurcharan Singh Bhatia." Now that we have the book node, we need to check the author neighbor of this book in the graph.
Action 2: NeighbourCheck[book-21968619, author]
Observation 2: The author neighbors of book-21968619 are: ['author-1049490'].
Thought 3: The author of the book "Truth, Love, Non-Violence: The Story of Gurcharan Singh Bhatia" is Gurcharan Singh Bhatia.
Action 3: NodeFeature[1049490, name]
Observation 3: The node or feature name does not exist in the graph. Please modify it.
Thought 4: The author of the book "Truth, Love, Non-Violence: The Story of Gurcharan Singh Bhatia" is Gurcharan Singh Bhatia.
Action 4: Finish[Gurcharan Singh Bhatia]
Observation 4: Answer is **INCORRECT**
Correct answer: Alexis Kienlen

Question:XXX
REFLECTION GUIDANCE FROM PREVIOUS ATTEMPT:
Issue Analysis: The failure was caused by not being able to retrieve the author's name from the graph due to an issue with the feature keys.
Correction Advice: Try using different feature keys or methods to retrieve the author's name from the graph. Check if there are alternative ways to access the information.
Suggested Approach: Restart the process and try different methods to access the author's name, such as checking for alternative feature keys or using different search queries.
Key Points to Remember: Focus on the issues identified above and follow the correction advice.
Thought 1: The question is asking for the author of a specific book ('Truth, Love, Non-Violence: The Story of Gurcharan Singh Bhatia'). We need to find the book node in the graph. Let's start by retrieving the node for this book.
Action 1: RetrieveNode[Truth, Love, Non-Violence: The Story of Gurcharan Singh Bhatia]
Observation 1: The ID of this retrieval target node is book-21968619.
Thought 2: The next step is to find out who penned the book 'Truth, Love, Non-Violence: The Story of Gurcharan Singh Bhatia'. We have the book node ID now. Let's check the author of this book by looking at the author neighbor of this book node in the graph.
Action 2: NeighbourCheck[book-21968619, author]
Observation 2: The author neighbors of book-21968619 are: ['author-1049490'].
Thought 3: The author of the book 'Truth, Love, Non-Violence: The Story of Gurcharan Singh Bhatia' is the node with ID 'author-1049490'. Let's retrieve the name of this author to provide the answer.
Action 3: NodeFeature[author-1049490, name]
Observation 3: The name of author-1049490 is: Alexis Kienlen.
Thought 4: The author of the book "Truth, Love, Non-Violence: The Story of Gurcharan Singh Bhatia" is Alexis Kienlen.
Action 4: Finish[Alexis Kienlen]
Observation 4: Answer is **CORRECT**
Correct answer: Alexis Kienlen

Figure 3. A side-by-side comparison of the reasoning paths of the original Graph-CoT (left) and R3-Cycle (right).

In contrast, R3-Cycle's Reflection module activates after the initial failure. It performs a global analysis of the trajectory and generates a clear corrective plan: "failed to retrieve the author's name from the graph, try different methods to access...". Guided by this reflection, the agent ignores the textual trap in its second attempt. It correctly executes the NodeFeature tool on the retrieved author ID and successfully identifies the true author, "Alexis Kienlen." This case vividly illustrates how R3-Cycle transforms a fragile, one-way reasoning process into a robust, dynamic, and self-correcting system capable of overcoming hallucinations by learning from its failures.

4. Related Work

Combining Large Language Models (LLMs) with Knowledge Graphs (KGs) to solve complex question answering problems has become an active research area. Existing methods for KG-Augmented LLMs for Question Answering can be divided into three main paradigms based on the role the KG plays in the system (Ma et al., 2025).

4.1 KGs as Background Knowledge

This paradigm, rooted in Retrieval-Augmented Generation (RAG) (Lewis et al., 2020), first retrieves relevant information from the KG. Early methods integrated knowledge during model training, for instance by using adaptation layers (Tian et al., 2024) or fine-tuning on SPARQL-question pairs (Zhang et al., 2024d). More recent approaches treat the KG as an external source, where methods like GraphRAG (Hu et al., 2024b; Cao et al., 2024) and KG-RAG (Sanmartin, 2024) retrieve relevant subgraphs. This structured information is then linearized into text for the LLM's context (Christmann et al., 2022). However, this paradigm's performance is bottlenecked by the single retrieval step, leaving the LLM a passive information receiver.

4.2 KGs as Reasoning Guidelines

This paradigm encapsulates the LLM as an agent, enabling it to perform dynamic, multi-step exploration on the KG. Based on its development, it can be subdivided into:

(1)**Static Knowledge Injection**: In this approach, an independent module pre-extracts a fixed subgraph or paths before LLM inference. Methods range from training dedicated retrievers like SR (Zhang et al., 2022) and using templates like Keqing (Wang et al., 2023), to employing

GNNs (Liu et al., 2024a), trie structures (Luo et al., 2024a), or relation-aware anchors (Yuan et al., 2025) to extract knowledge. This one-shot retrieval process, however, fundamentally limits reasoning flexibility.

(2)**Iterative Path Exploration**: To enhance dynamism, this paradigm positions the LLM as an active explorer. Works like Graph-CoT (Jin et al., 2024) and ToG (Sun et al., 2024a) use a step-by-step "thought-action-observation" loop or beam search. However, this forward-chaining approach is rigid, lacking the ability to backtrack or adjust its overall strategy upon failure. Even attempts to mitigate this with learnable "soft reasoning paths" (Hou et al., 2025) do not fully resolve this strategic inflexibility.

(3)**Reflective Corrective Planning**: This paradigm endows the agent with self-correction capabilities through step-wise local reflection, as seen in ReAct (Yao et al., 2022) and Plan-on-Graph (PoG) (Chen et al., 2024a). However, this mechanism causes the context to expand dramatically in long-chain tasks. This leads to the "lost in the middle" problem (Liu et al., 2023), where the model forgets early information, increasing the risk of factual hallucinations.

Our work, R3-Cycle, directly addresses this challenge. By introducing a global retrospective analysis and replanning within a reset context, we achieve effective correction while avoiding the problem of context overload, representing a significant advancement over existing reflective planning methods.

4.3 KGs as Refiners and Validators

This approach uses a "generate-then-verify" strategy: the LLM first generates answers, which are then checked (Guan et al., 2024) or improved (Ko et al., 2024) using KG facts. While this improves the final answer's reliability, it cannot correct flawed reasoning paths during the generation stage. Specific methods include filtering or re-ranking candidates (Salnikov et al., 2023; Zhang et al., 2024c) and refining outputs through summarization or user feedback (Ko et al., 2024; Xiong et al., 2024). Unlike these methods that correct the final output, R3-Cycle's reflection mechanism fixes the reasoning process itself to prevent errors from happening in the first place.

5. Conclusion

In this paper, we introduced R3-Cycle, a framework that transforms the fragile, linear reasoning of LLM-based graph agents into a robust, cyclical process of self-correction. By employing a global retrospective analysis on failed trajectories and re-planning in a clean context, R3-Cycle overcomes the critical limitations of both rigid forward-chaining methods and context-heavy local reflection methods. Our experiments on the GRBENCH benchmark demonstrate that R3-Cycle significantly outperforms existing baselines. Ablation studies confirmed that this success is driven by a powerful synergy: the Reflection module enables root-cause diagnosis, the Memory module provides the necessary historical context, and our adaptive step allocation mechanism intelligently manages the retry budget. Crucially, these gains are achieved at a manageable computational cost, confirming the framework's practical viability. Looking ahead, we will focus on enhancing efficiency through optimized reflection triggers and exploring sophisticated memory structures like vector databases. We also aim to improve reflection quality via advanced prompting or specialized models. Ultimately, we plan to extend this generalizable global "memory-reflection" paradigm to a broader range of complex agent tasks, such as API manipulation and code generation, to validate its universal value in building more intelligent and reliable AI systems.

References

[1] Jing Zhang, Xiaokang Zhang, Jifan Yu, Jian Tang, Jie Tang, Cuiping Li, and Hong Chen. 2022. Subgraph retrieval enh anced model for multi-hop knowledge base question answering. In ACL.

[2] Chaojie Wang, Yishi Xu, Zhong Peng, Chenxi Zhang, Bo Chen, Xinrun Wang, Lei Feng, and Bo An. 2023. keqing: kn owledge-based question answering is a nature chain-of-thought mentor of LLM. arXiv:2401.00426.

[3] Bowen Jin, Chulin Xie, Jiawei Zhang, Kashob Kumar Roy, Yu Zhang, Zheng Li, Ruirui Li, Xianfeng Tang, Suhang Wa ng, Yu Meng, and Jiawei Han. 2024. Graph Chain-of-Thought: Augmenting Large Language Models by Reasoning o n Graphs. arXiv preprint arXiv:2404.07103.

[4] Liyi Chen, Panrong Tong, Zhongming Jin, Ying Sun, Jieping Ye, and Hui Xiong. 2024a. Plan-on-Graph: Self-correcti ng adaptive planning of large language model on knowledge graphs. In NeurIPS, pages 37665–37691.

[5] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 202 3. Lost in the middle: How language models use long contexts. arXiv preprint arXiv:2307.03172.

[6] Chuangtao Ma, Yongrui Chen, Tianxing Wu, Arijit Khan, and Haofen Wang. 2025. Large Language Models Meet Kn owledge Graphs for Question Answering: Synthesis and Opportunities. arXiv preprint arXiv:2505.20099.

[7] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttle r, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented gen eration for knowledge-intensive nlp tasks. In Advances in Neural Information Processing Systems, 33, pages 945 9–9474.

[8] Shiyu Tian, Yangyang Luo, Tianze Xu, Caixia Yuan, Huixing Jiang, Chen Wei, and Xiaojie Wang. 2024. KG-Adapter: Enabling knowledge graph integration in large language models through parameter-efficient fine-tuning. In ACL Findings, pages 3813–3828.

[9] Zhiqiang Zhang, Liqiang Wen, and Wen Zhao. 2024d. A GAIL fine-tuned LLM enhanced framework for low-resour ce knowledge graph question answering. In CIKM, pages 3300–3309.

[10] Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. 2024b. GRAG: Graph retrieval-augmen ted generation. arXiv:2405.16506.

[11] Yukun Cao, Zengyi Gao, Zhiyang Li, Xike Xie, and S Kevin Zhou. 2024. LEGO-GraphRAG: Modularizing graph-base d retrieval-augmented generation for design space exploration. arXiv:2411.05844.

[12] Diego Sanmartin. 2024. KG-RAG: Bridging the gap between knowledge and creativity. arXiv:2405.12035.

[13] Philipp Christmann, Rishiraj Saha Roy, and Gerhard Weikum. 2023. Explainable conversational question answeri ng over heterogeneous sources via iterative graph neural networks. In SIGIR, pages 643–653.

[14] Guangyi Liu, Yongqi Zhang, Yong Li, and Quanming Yao. 2024a. Explore then determine: A GNN-LLM synergy fra mework for reasoning over knowledge graph. arXiv:2406.01145.

[15] Linhao Luo, Zicheng Zhao, Chen Gong, Gholamreza Haffari, and Shirui Pan. 2024a. Graph-constrained reasoning: Faithful reasoning on knowledge graphs with large language models. arXiv:2410.13080.

[16] Duanyang Yuan, Sihang Zhou, Xiaoshu Chen, Dong Wang, Ke Liang, Xinwang Liu, and Jian Huang. 2025. Knowledg e graph completion with relation-aware anchor enhancement. In Proceedings of the AAAI Conference on Artifici al Intelligence, 39(14).

[17] Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. 2024a. Think-on-Graph: Deep and responsible reasoning of large language model with knowledge graph. In ICLR.

[18] Yanning Hou, Sihang Zhou, Ke Liang, Lingyuan Meng, Xiaoshu Chen, Ke Xu, Siwei Wang, Xinwang Liu, and Jian Hu ang. 2025. Soft reasoning paths for knowledge graph completion. In International Joint Conference on Artificial I ntelligence.

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

[20] Xinyan Guan, Yanjiang Liu, Hongyu Lin, Yaojie Lu, Ben He, Xianpei Han, and Le Sun. 2024. Mitigating large langua ge model hallucinations via autonomous knowledge graph-based retrofitting. In AAAI, pages 18126–18134.

[21] Sungho Ko, Hyunjin Cho, Hyungjoo Chae, Jinyoung Yeo, and Dongha Lee. 2024. Evidence-focused fact summariza tion for knowledge-augmented zero-shot question answering. In EMNLP, pages 10636–10651.

[22] Mikhail Salnikov, Maria Lysyuk, Pavel Braslavski, Anton Razzhigaev, Valentin A Malykh, and Alexander Panchenk o. 2023. Answer candidate type selection: Text-to-text language model for closed book question answering meet s knowledge graphs. In KONVENS, pages 155–164.

[23] Yu Zhang, Kehai Chen, Xuefeng Bai, Zhao Kang, Quanjiang Guo, and Min Zhang. 2024c. Question-guided knowled ge graph re-scoring and injection for knowledge graph question answering. In EMNLP, pages 8972–8985.

[24] Guanming Xiong, Junwei Bao, and Wen Zhao. 2024. Interactive-KBQA: Multi-turn interactions for knowledge bas e question answering with large language models. In ACL, pages 10561–10582.