GRAPH-O1: MONTE CARLO TREE SEARCH WITH REINFORCEMENT LEARNING FOR TEXT-ATTRIBUTED GRAPH REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Text-attributed graphs, where nodes and edges are enriched with textual information, are widely used across various domains. A central challenge in this setting is question answering, which requires effectively integrating unstructured text with the structured relationships in the graph. Although Large Language Models (LLMs) have achieved remarkable progress in natural language understanding, their direct application to reasoning over text-attributed graphs remains limited. Existing text retrieval–augmented generation methods often treat text passages as independent units, overlooking the rich interconnections within the graph. Similarly, conventional graph RAG approaches that encode large subgraphs as text quickly become impractical due to LLM context-length constraints, leading to fragmented reasoning and reduced accuracy. To address these challenges, we propose GRAPH-O1, an agentic GraphRAG framework that enables LLMs to perform stepwise, interactive reasoning over graphs. Our approach combines Monte Carlo Tree Search (MCTS) with end-to-end reinforcement learning, allowing the model to selectively explore and retrieve only the most relevant subgraph elements. The reasoning process is modeled as a multi-turn agent–environment interaction, and the agent is optimized via an end-to-end reward mechanism. Extensive experiments across multiple LLM backbones show that GRAPH-O1 consistently outperforms state-of-the-art baselines, delivering more accurate, reliable, and interpretable answers.

1 INTRODUCTION

Text-attributed graphs have been widely used across a variety of domains, including scientific knowledge management Jin et al. (2024), biomedical discovery Zeng et al. (2022), and recommender systems Wu et al. (2022). A central challenge in these settings is text-attributed graph question answering, where the goal is to answer complex, multi-hop queries by reasoning over both the structural connections in the graph and the rich textual information associated with nodes and edges. This task is non-trivial: models must not only comprehend and extract knowledge from unstructured text, but also combine it with the structured relationships encoded in the graph, enabling accurate, context-aware reasoning across multiple hops.

Large Language Models (LLMs) such as GPT OpenAI (2024) and LLaMA .et.al. (2024) have demonstrated remarkable capabilities in natural language understanding and generation Radford et al. (2018); Acharya & Adhikari (2021). Despite their impressive performance, these models remain prone to hallucinations Touvron et al. (2023), producing outputs that are fluent and convincing but factually incorrect or misleading. A widely adopted strategy to mitigate this issue is retrieval-augmented generation (RAG) Shi et al. (2023), which augments LLMs with external knowledge sources, allowing models to ground their responses in real-world data during inference. However, most existing RAG approaches treat each retrieved passage as an independent knowledge unit, ignoring the rich structural relationships that often exist between pieces of information. In practice, knowledge in many domains is naturally organized as graphs with intricate interconnections. For example, in academic research, papers are linked through citations, co-authorship, and topic similarity, forming bibliographic graphs where the relevance of a document depends not only on its content but also on its relationships to others. Reasoning effectively over such structured knowledge requires models to integrate both node content and graph structure, enabling more accurate and contextually grounded responses.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Directly applying RAG to graph-structured data presents two fundamental challenges. First, graph structure encodes essential contextual information—such as citations, semantic dependencies, or functional relationships—that cannot be fully captured by treating nodes in isolation. Ignoring these connections leads to fragmented reasoning, as the interdependencies between nodes provide critical cues for understanding complex relationships and drawing accurate conclusions. Second, scalability poses a significant hurdle: as the neighborhood of a node expands, the corresponding subgraph can grow exponentially in size. Encoding these large subgraphs as input text quickly becomes infeasible due to the context-length limitations of LLMs, diluting the model's focus and degrading reasoning performance. Together, these challenges underscore the need for methods that can efficiently leverage both textual content and structural relationships in graph-structured knowledge.

To overcome these limitations, we propose **GRAPH-O1**, an agentic GraphRAG framework that enables LLMs to perform stepwise, interactive reasoning on text-attributed graphs, enhanced by end-to-end reinforcement learning and inspired by GPT-o1. Our approach combines Monte Carlo Tree Search (MCTS) Świechowski et al. (2022) with end-to-end reinforcement learning Shao et al. (2024). Rather than exposing the LLM to an entire subgraph at once, GRAPH-O1 guides exploration by selectively retrieving only the most relevant nodes, neighbors, and attributes at each step. Each iteration follows a three-phase cycle: the LLM first reasons over available evidence and identifies what additional knowledge is needed; it then traverses the graph to uncover relevant information; finally, the system retrieves and incorporates new knowledge into the ongoing reasoning process. This iterative exploration continues until the LLM arrives at a grounded, well-supported answer. To further enhance performance, we introduce a unified reward mechanism that integrates generation quality, retrieval relevance, and structural reliability of graph paths. Through reinforcement learning, the agent learns generalizable graph reasoning strategies that tightly align structured knowledge with natural language generation. Extensive experiments across multiple datasets and LLM backbones show that GRAPH-O1 consistently outperforms conventional RAG methods, producing more accurate, and interpretable answers. To the best of our knowledge, we are the first to apply GPT-o1-style reasoning with GRPO tuning to Text-Attributed Graph reasoning.

In summary, we make the following contributions:

- We introduce **GRAPH-O1**, the first agentic text-attributed graph RAG framework that enables LLMs to perform stepwise, interactive reasoning over text-attributed graphs, guided by Monte Carlo Tree Search (MCTS) for efficient exploration.
- We develop an end-to-end reinforcement learning optimization strategy featuring a unified reward function that jointly evaluates generation quality, answer fidelity, and adherence to the expected answer format.
- We conduct extensive experiments on multiple datasets and LLM backbones, demonstrating that GRAPH-O1 consistently outperforms conventional RAG methods in accuracy, reliability, and interpretability.

2 PROBLEM DEFINITION

In this work, we investigate question answering over text-attributed graphs using Large Language Models, with a particular focus on multi-hop question answering.

Definition 2.1 (Text-Attributed Graph). A graph is defined as $G = (V, E)$, where V is the set of nodes and E is the set of edges. Each node $v_i \in V$ is associated with a textual feature description X_i. For instance, in an e-commerce setting, nodes represent products, edges capture co-purchase relationships, and X_i may include product titles, descriptions, prices, and categories. Because node attributes are represented as text, we refer to G as a text-attributed graph.

Problem Statement. Given a multi-hop natural language query q, a text-attributed graph $G = (V, E)$, and access to an LLM agent, the objective is to generate an accurate and well-grounded answer to q. Unlike conventional document retrieval, this task requires reasoning jointly over graph structure and textual attributes. In particular, the LLM must: (i) identify relevant nodes and edges, (ii) integrate textual features with structural relationships, and (iii) synthesize retrieved evidence into a coherent and faithful response.

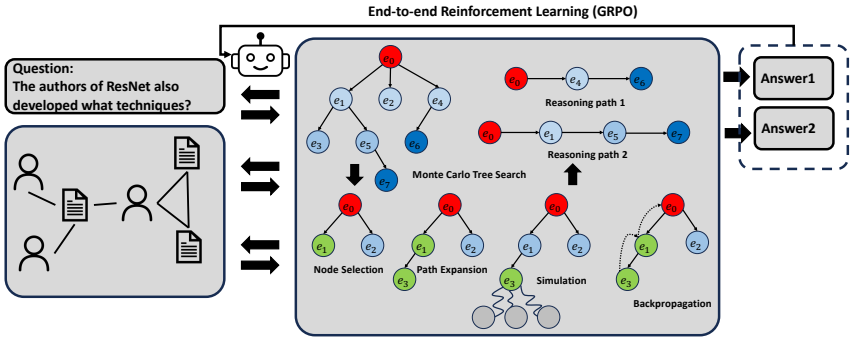


Figure 1: Architecture of our proposed GRAPH-O1 Model.

Formulation as Graph Search. We reformulate the problem as a graph search task. Starting from a topic node mentioned in the question, the agent traverses the graph guided by q until it reaches the target answer v_T. This process can be modeled as a Markov Decision Process (MDP), defined by the tuple (S, A, R, P), where: S is the set of states, A is the set of actions, R is the reward function, and P is the state transition probability.

Let $s_t \in S$ denote the state at step t. Since reasoning depends on both the query and the traversal history, we define the state recursively as: $s_t = s_{t-1} \cup \{a_{t-1}, v_t, \mathcal{N}_{v_t}\}, \quad s_0 = \{q, v_0, \mathcal{N}_{v_0}\}$, where $a_t \in A$ is the action chosen at step t, $v_t \in V$ is the current node, and $\mathcal{N}_{v_t} \subset V$ denotes its neighbors. Thus, s_t captures: (i) the multi-hop query q, (ii) the sequence of visited nodes and actions, and (iii) the neighborhoods encountered so far. At each time step t, the agent selects an action a_t based on s_t, determining how to proceed in the graph. The search concludes when the target entity e_T is reached, and the final answer is produced by synthesizing evidence along the traversed path.

3 PROPOSED METHOD

Combining retrieval-augmented generation Shi et al. (2023) with LLMs enables models to access graph-structured data by retrieving relevant nodes as context. However, text-attributed graphs encode rich relational structures that require multi-step reasoning to fully exploit. Standard approaches such as chain-of-thought prompting Wei et al. (2023) or naive LLM agent frameworks Jin et al. (2025) often struggle with this complexity, producing incomplete or inconsistent answers. Motivated by the GPT-o1 paradigm, we integrate Monte Carlo Tree Search (MCTS) Świechowski et al. (2022) with LLM reasoning to systematically explore graph-structured knowledge. Given a multi-hop query, MCTS expands candidate reasoning paths, evaluates their relevance to the question, and backpropagates this signal to guide further exploration—prioritizing promising paths while avoiding redundant computation. To further improve reasoning quality, we introduce an end-to-end reinforcement learning objective based on Group Relative Policy Optimization Shao et al. (2024), which fine-tunes the reasoning policy to better coordinate exploration, retrieval, and answer generation in a coherent, stepwise manner.

3.1 GRAPH REASONING WITH LLMS VIA MONTE CARLO TREE SEARCH

We propose GRAPH-O1, a framework that integrates LLM-based reasoning with Monte Carlo Tree Search (MCTS) for multi-hop question answering over text-attributed graphs. Given a query q, GRAPH-O1 iteratively decides which graph information to retrieve and uses MCTS to evaluate multiple reasoning trajectories, selecting the most promising path at each step. Each simulation step consists of three stages. First, the **LLM reasoning** phase (a_t^{think}): the LLM examines the current state—previously visited nodes and retrieved information—and decides what to explore next. Second, the **graph interaction** phase (a_t): the decision is translated into concrete graph operations, such as retrieving node attributes or following an edge to a neighboring node. Third, the **MCTS-guided execution** phase (a_t^{out}): the operation is executed, new knowledge is incorporated into the search state, and a rollout value is back-propagated via the UCT algorithm to update $Q(s, a)$ scores. This cycle repeats across multiple simulations, balancing exploration of new paths with exploitation of

promising ones, until the system converges on a well-supported final answer grounded in both textual and structural graph knowledge.

3.1.1 FORMULATING TEXT-ATTRIBUTED GRAPH REASONING AS AN MDP

The question answering process over text-attributed graphs can be naturally formulated as a Markov Decision Process (MDP). In GRAPH-O1, inspired by Jin et al. (2024), we design four core graph functions that capture both semantic and structural aspects of the graph. Each action corresponds to invoking one of these functions to explore the graph and retrieve relevant knowledge, effectively guiding the agent's traversal and reasoning process. These actions are: (1) **RetrieveNode(text)** searches for nodes that are semantically related to a given text query and provides candidate entry points for reasoning. (2) **NodeFeature(id, feature)** retrieves a specific textual attribute of a node, such as its title or description, to supply detailed context. (3) **NeighborCheck(id, type)** returns neighboring nodes of a specified type, allowing the agent to follow meaningful relations in the graph. (4) **NodeDegree(id, type)** counts the number of neighbors of a given type, which serves as a simple measure of node importance and connectivity.

By framing text-attributed graph reasoning as an MDP, we establish a principled foundation for modeling the dynamic interaction between the LLM agent and the graph. Each action incrementally expands the agent's understanding of the graph, while the evolving state preserves both the semantic context of the query and the history of traversed paths. This formulation not only enables systematic exploration of large and complex graphs but also allows reinforcement learning techniques to optimize the agent's decision-making policy for accurate and efficient multi-hop reasoning.

3.1.2 MCTS-GUIDED PLANNING FOR GRAPH REASONING

While LLMs can act as agents to navigate text-attributed graphs, relying on a single greedy reasoning path is inherently fragile: one suboptimal decision early on can propagate through subsequent steps, leading to incorrect or incomplete answers. To address this, we integrate LLM decision-making into a Monte Carlo Tree Search (MCTS) framework, enabling the system to explore multiple reasoning trajectories and balance exploration of new paths with exploitation of rewarding ones—making the agent more robust and accurate. Formally, each tree node represents a state s encapsulating the current query, retrieved graph fragments, and action history. Each edge corresponds to a graph function action $a \in \mathcal{A}$ the agent may execute to gather additional knowledge. Starting from the initial query state, MCTS incrementally builds a search tree, allowing the LLM to consider multiple reasoning paths rather than committing to a single trajectory. Each MCTS iteration proceeds through four stages—selection, expansion, simulation, and backpropagation—which together enable the agent to evaluate diverse actions, assess their downstream effects, and iteratively refine its reasoning strategy. We describe each stage in detail below.

1. Selection. The selection phase initiates from the root node s_0 of the search tree, which represents the initial query state and an empty reasoning context. From this root, the algorithm incrementally traverses the tree by selecting a sequence of actions that balance the trade-off between exploration of new reasoning directions and exploitation of known high-value paths. At each decision point, the action a is chosen according to the Upper Confidence Bound (UCB) criterion:

$$\text{UCB}(s, a) = Q(s, a) + c\sqrt{\frac{\ln N(s)}{N(s, a)}},$$

where $Q(s, a)$ denotes the estimated cumulative reward for performing action a in state s, $N(s, a)$ is the number of times this action has been selected at s, $N(s) = \sum_a N(s, a)$ is the total number of visits to state s, and c is an exploration coefficient that governs the balance between exploitation (favoring actions with high estimated value) and exploration (favoring actions that have been tried less frequently).

Formally, the next action is selected as $a^* = \arg\max_{a \in \mathcal{A}(s)} \text{UCB}(s, a)$, where $\mathcal{A}(s)$ represents the set of all valid graph operations available at state s. The selected action a^* corresponds to a reasoning step that extends the current trajectory, which will later be expanded and simulated in subsequent stages of the MCTS process.

2. Expansion. Once the selection phase identifies a non-terminal leaf node s_L that still contains unexplored actions, the algorithm proceeds to the expansion stage. Here, the search tree is sys-

tematically enlarged by generating one or more child nodes corresponding to the feasible graph function actions $a \in \mathcal{A}(s_L)$ proposed by the LLM. Each of these actions represents a distinct way to extend the reasoning trajectory—such as retrieving related nodes, extracting features, or examining neighborhood relations—and thus provides new contextual knowledge for subsequent reasoning steps. The execution of an action deterministically updates the current search state according to $s_{L+1} = s_L \cup \texttt{graph_call}(a)$, where $\texttt{graph_call}(a)$ denotes a graph operation that returns newly retrieved information, such as node attributes, textual descriptions, neighboring entities, or degree statistics. This updated state s_{L+1} captures both the expanded knowledge context and the logical continuity of the reasoning path up to this point.

For each new child node, we then construct the complete reasoning trajectory by appending the executed action a to the path inherited from its parent. To evaluate the initial promise of these newly expanded nodes, we employ an LLM-based evaluator π_{eval}, which estimates the semantic consistency and relevance of the accumulated reasoning path with respect to the original query q:

$$v(s_{L+1}) = \pi_{\text{eval}}(\text{path}(s_{L+1}, q)),$$

where $v(s_{L+1})$ serves as the initial reward assigned to node s_{L+1}. Additionally, we initialize its visit count as $N(s_{L+1}) = 1$, laying the groundwork for future selection updates based on the UCB criterion.

3. Simulation and Backpropagation. Once a non-terminal leaf node s_L has been expanded, the next step is to estimate the long-term value of the reasoning trajectories emanating from this state. This is achieved through the *simulation* phase, where the agent performs a series of virtual rollouts to explore potential future reasoning paths without executing every action in the real search space. The goal is to approximate how promising the current node and its descendants are in terms of ultimately producing a correct and well-grounded answer.

Starting from s_L, a rollout policy π_{rollout} interacts with the text-attributed graph by sequentially selecting actions until a terminal condition is reached—either the agent generates an answer or the maximum reasoning depth $D_{\max}$ is exceeded. At each step t, given the current state s_t, the accumulated reasoning path p_t, and the query context q, the policy produces a probability distribution over possible graph actions: $\pi_{\text{rollout}}(a_t \mid s_t, p_t, q, G)$, representing the likelihood of choosing each action based on the agent's current knowledge and exploration history. An action a_t is then sampled from this distribution, and the state is deterministically updated according to $s_{t+1} = \texttt{transition}(s_t, a_t)$, which executes the corresponding graph operation (e.g., node retrieval, neighbor exploration) and incorporates the resulting information into the new state representation. Repeating this process produces a simulated reasoning trajectory $p_{\text{sim}} = \{(s_t, a_t, s_{t+1})\}_{t=1}^{T}$, where $T \leq D_{\max}$.

Once the simulated path is generated, we invoke the LLM-based evaluator π_{eval} to assess its semantic relevance and factual alignment with respect to the original question q:

$$v_{\text{sim}} = \pi_{\text{eval}}(p_{\text{sim}}, q),$$

where v_{sim} serves as the estimated reward for the rollout. This value reflects how coherent, contextually appropriate, and informative the reasoning trajectory is in addressing the query.

Next, the *backpropagation* phase propagates this reward value upward along the path from the leaf node back to the root, updating the statistics of all visited state–action pairs. Specifically, for each pair (s, a) encountered along the simulation, the action-value estimates and visit counts are refined as:

$$Q(s,a) \leftarrow \frac{Q(s,a) \cdot N(s,a) + v_{\text{sim}}}{N(s,a) + 1}, \qquad N(s,a) \leftarrow N(s,a) + 1.$$

These updates integrate new information gained from the simulated rollout, gradually improving the accuracy of value estimates for different reasoning paths. As a result, subsequent selection steps—guided by the UCB criterion—are better able to balance exploration of less-visited actions with exploitation of those that have consistently yielded high returns.

Through repeated iterations of simulation and backpropagation, the MCTS framework converges toward identifying reasoning trajectories that are both semantically consistent with the query and structurally grounded in the graph. Over time, this iterative refinement enables the agent to uncover robust, interpretable, and high-quality reasoning paths, thereby improving the reliability and factual correctness of the final answers generated by GRAPH-O1.

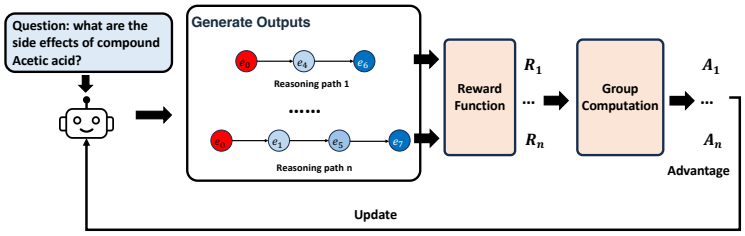


Figure 2: End-to-End Reinforcement Learning to update GRAPH-O1.

3.2 END-TO-END REINFORCEMENT LEARNING

While MCTS provides a principled mechanism for exploring reasoning trajectories, its effectiveness ultimately depends on how well the explored paths are evaluated and refined. To fully exploit this process, we leverage the information collected during MCTS to fine tune the reasoning policy π_θ through end-to-end reinforcement learning. Specifically, we adopt Group Relative Policy Optimization (GRPO) Shao et al. (2024), an advanced policy optimization algorithm that directly optimizes π_θ with respect to the paths discovered by MCTS. For each trajectory, we introduce an outcome-directed reward function $R(\tau)$, which jointly measures the structural validity of the reasoning process and its semantic alignment with the original query. This design ensures that the policy is guided toward producing both faithful and well-structured answers. See Figure 2 for more details.

More specifically, given a dataset of questions $q \in \mathcal{D}_Q$, the agent interacts with the text-attribute graph G via MCTS to generate a collection of reasoning trajectories $\{\tau_i\}_{i=1}^N \subseteq \mathcal{T}_q$, where each trajectory is defined as $\tau_i = \big((s_0^{(i)}, a_0^{(i)}), (s_1^{(i)}, a_1^{(i)}), \ldots, (s_T^{(i)}, a_T^{(i)})\big)$. Based on the collected trajactory, we utilize GRPO to fine tune the model. The GRPO-based training objective is formulated as:

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{\left[s_0 \sim \{P(q) | q \in \mathcal{D}_Q\},\ \{\tau_i\}_{i=1}^N \sim \pi_{\theta_{\text{old}}}(\mathcal{T}_q | s_1; G) \right]}$$
$$\left[\frac{1}{N} \sum_{i=1}^{N} \frac{1}{|\tau_i|} \sum_{t=1}^{|\tau_i|} \min \left(\rho_\theta(a_t^{(i)}) \hat{A}(\tau_i),\ \text{clip}\big(\rho_\theta(a_t^{(i)}), 1 \pm \epsilon\big) \hat{A}(\tau_i) \right) - \beta D_{\text{KL}}(\pi_\theta \parallel \pi_{\text{ref}}) \right],$$

where $\rho_\theta(a_t^{(i)}) = \frac{\pi_\theta(a_t^{(i)} | s_{t-1}^{(i)}; G_H)}{\pi_\theta^{\text{old}}(a_t^{(i)} | s_{t-1}^{(i)}; G_H)}$ and $\hat{A}(\tau_i) = \frac{R(\tau_i) - \text{mean}\left(\{R(\tau_j)\}_{j=1}^N\right)}{\text{F}_{\text{norm}}\left(\{R(\tau_j)\}_{j=1}^N\right)}$. Here, ρ_θ is the importance weight that corrects for distributional shift, $\hat{A}(\tau_i)$ denotes the normalized advantage (e.g., using standard deviation as the normalization function), the clipping operator clip() stabilizes optimization, and the KL penalty encourages π_θ to remain close to a reference policy π_{ref}, and β denotes the regularization strength.

Reward Function. To ensure high-quality reasoning outcomes, we design a trajectory-level reward function $R(\tau)$ that considers both structural coherence and semantic correctness. It combines a *format reward* $R_{\text{format}}(\tau)$ and a *reasoning reward* $R_{\text{reasoning}}(\tau)$, promoting structured multi-step reasoning and accurate answers.

(i) Format Reward. The format reward encourages the model to produce outputs following the required reasoning structure. At each step (s_t, a_t), the output should contain identifiable reasoning components such as `Thought`, `Action`, and `Observation`. Each correctly formatted element contributes 0.5 to the reward, capped at 1.0:

$$R_{\text{format}}(\tau) = \min \left(1.0,\ 0.5 \cdot \sum_{t=1}^{T} \mathbb{I}\{(a_t^{\text{think}}, \alpha_t^{\text{Act}}, a_t^{\text{Obs}}) \text{ is well-formed}\} \right)$$

where $\mathbb{I}\{\cdot\}$ is the indicator function.

(ii) Reasoning Reward. To promote meaningful graph operations, the model is rewarded for using graph-related functions such as `RetrieveNode`, `NeighbourCheck`, `NodeFeature`, and `NodeDegree`. Explicitly correct answers further increase the reward, while obvious errors or failures are penalized:

$$R_{\text{reasoning}}(\tau) = 0.5 \cdot \mathbb{I}\{\text{valid operation}\} + 1.0 \cdot \mathbb{I}\{a_T^{\text{ans}} = y_q^*\} - 0.5 \cdot \mathbb{I}\{a_T^{\text{ans}} \neq y_q^*\}.$$

Table 1: Model performance on GRBENCH Jin et al. (2024) comparing standard LLMs, text retrieval augmented LLMs (Text RAG), graph retrieval augmented LLMs (Graph RAG), Graph-COT. Results are reported using ROUGE-L (R-L) and GPT-4 Score (GPT4score). Mixtral-8x7b-Instruct is used as the backbone for Graph-COT Jin et al. (2024), and GRAPH-O1.

Model	**Academic**		**E-commerce**		**Literature**		**Healthcare**		**Legal**	
	R-L	GPT4score	R-L	GPT4score	R-L	GPT4score	R-L	GPT4score	R-L	GPT4score
Base										
LLaMA-2-13b-chat	8.13	8.03	7.01	12.00	5.32	20.83	5.25	13.70	15.97	16.11
Mixtral-8x7b	9.02	8.14	12.54	18.00	7.50	22.50	3.88	20.00	12.74	16.11
GPT-3.5-turbo	6.05	12.80	9.18	23.50	10.43	26.67	5.83	14.44	10.51	20.00
Text RAG										
LLaMA-2-13b-chat	8.69	8.52	9.23	12.50	7.61	20.00	1.44	5.93	15.37	16.67
Mixtral-8x7b	8.44	8.02	23.14	29.50	13.35	27.92	3.22	16.67	19.69	25.00
GPT-3.5-turbo	5.83	9.91	14.06	20.00	10.04	20.83	4.57	8.52	18.14	23.89
Graph RAG										
LLaMA-2-13b	22.01	22.97	12.48	20.00	9.25	20.00	2.97	4.81	17.98	17.22
Mixtral-8x7b	27.77	31.20	32.87	37.00	20.08	33.33	8.66	15.19	23.48	25.56
GPT-3.5-turbo	18.45	26.98	17.52	28.00	14.94	24.17	8.69	14.07	18.66	22.22
Agent										
GRAPH-MCTS	31.42	32.16	33.71	37.22	35.85	45.83	26.79	32.84	26.39	27.78
GRAPH-COT	27.35	29.43	29.38	32.78	32.44	38.75	23.33	26.02	19.81	22.77
GRAPH-O1	**32.35**	**33.82**	**34.61**	**38.76**	**38.39**	**45.83**	**29.54**	**33.70**	**27.05**	**32.20**
Improvement (%)	*+2.96*	*+5.16*	*+2.67*	*+4.14*	*+7.09*	*+0.00*	*+10.27*	*+2.62*	*+2.50*	*+15.91*

(iii) Overall Reward. The total trajectory reward integrates both format and reasoning components, with semantic correctness contributing only if the reasoning format is valid:

$$R(\tau) = \max\big(-1.0, \min\big(3.0, R_{\text{format}}(\tau) + R_{\text{reasoning}}(\tau)\big)\big), \tag{1}$$

where the reward is clipped to the stable range $[-1.0, 3.0]$ for training stability. This design directly reinforces structurally coherent and semantically accurate multi-step reasoning over the graph G, encouraging faithful and well-grounded answers.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Baselines. To contextualize the performance of GRAPH-O1, we consider three families of approaches. (1) **Base LLMs** tou (2023); jia (2024); Radford et al. (2018), represents models that answer purely from internal parameters with no external knowledge source; a straightforward prompt is issued and the model responds directly. (2) **Text RAG LLMs**, flattens the graph into a searchable text corpus and augments the LLM with passages retrieved at query time. (3) **Graph RAG and Reasoning LLMs** Jin et al. (2024), preserves graph topology by letting the LLM act as a traversal agent, hopping across nodes to gather evidence before producing an answer. All three families are instantiated with the same three backbones, namely LLaMA-2-13b-chat tou (2023), Mixtral-8x7b-Instruct jia (2024), and GPT-3.5-turbo OpenAI (2024), to ensure fair comparison.

Dataset. We conduct experiments on a text-attributed graph QA benchmark that spans five distinct domains: Academic, E-commerce, Literature, Healthcare, and Legal. Questions are grounded in realistic, domain-specific scenarios that cannot be resolved without navigating the underlying graph, making the benchmark a meaningful testbed for graph reasoning. Full dataset statistics appear in Table 6; the data is accessible through GraphCoT.

Evaluation. Answer quality is measured along two axes: ROUGE-L captures surface-level lexical similarity to the reference answer, while GPT-4 Score delegates semantic judgment to GPT-4, recording the proportion of responses it deems correct. Together, these metrics provide both a deterministic and a model-based view of performance.

Setup. Experiments are carried out on NVIDIA H200 GPUs under Python 3.8 and Huggingface 4.36.2. `Mpnet-v2` serves as the shared retriever for all methods, with FAISS Johnson et al. (2019) handling index construction. Unless stated otherwise, GRAPH-O1 is instantiated with Mixtral-8x7b-Instruct as its backbone. Implementation details, prompt templates, and reasoning trajectories are provided in the Appendix.

4.2 OVERALL PERFORMANCE

Table 1 presents the performance of GRAPH-O1 compared to all baselines across the five domains. We can observe several clear trends: First, Base LLMs perform the worst among all methods. This is expected, as they rely solely on their internal knowledge without accessing any external data, limiting their ability to answer domain-specific or complex questions accurately. Text RAG LLMs show improved performance, especially in domains like *E-commerce*, where relevant textual information can be effectively retrieved from the external graph. However, even with retrieval, these models still struggle in domains that require deeper reasoning or structured information, such as *Healthcare* and *Legal*. Graph RAG LLMs generally outperform Text RAG, demonstrating that including the 1-hop subgraph around retrieved nodes provides richer structural context. This structured graph information allows the LLM to reason more effectively over relationships and dependencies, which is particularly useful in domains with complex interconnections. The strongest results come from agent-based approaches. GRAPH-COT combines the reasoning capabilities of LLMs with graph-structured input, achieving significantly higher ROUGE-L and GPT-4 scores than all non-agent baselines. Finally, GRAPH-O1 further improves performance by leveraging a more advanced agent framework alongside graph reasoning. It consistently achieves the highest scores across all domains, showing that integrating structured graph reasoning with an LLM-driven agent is an effective strategy for accurate question answering. These results highlight two key points: (1) incorporating structured graph information is crucial for challenging domains, and (2) an agent-based reasoning framework can fully exploit this information to deliver more reliable and accurate answers.

4.3 RAG VS LLM AS AGENT

We conduct a systematic comparison of retrieval scopes to understand the limitations of RAG. Specifically, we evaluate three retrieval granularities: individual nodes, 1-hop ego-graphs, and 2-hop ego-graphs, where each retrieved subgraph is serialized into a flat text sequence and passed as context to the LLM. Average results across all domains are reported in Table 2. Among the three RAG configurations, 1-hop retrieval strikes the best balance, outperforming both node-level and 2-hop retrieval. However, none of the RAG variants come close to matching GRAPH-O1. The core issue is that wider retrieval scopes produce longer contexts that frequently exceed LLM input limits, hurting performance. GRAPH-O1 avoids this by dynamically selecting only the most relevant graph information at each step.

Table 2: Results for different RAG methods.

Model	**GPT4score**
GPT-3.5-turbo	19.48
+ node retrieval	16.63
+ 1-hop subgraph retrieval	23.09
+ 2-hop subgraph retrieval	22.12
+ GRAPH-O1	36.86

4.4 ABLATION STUDY

In the main results, we adopt Mixtral-8x7b-Instruct as the LLM backbone for GRAPH-O1. In this part, we explore GRAPH-O1 with different LLM backbones, including LLaMA-2-7b-chat, LLaMA-2-13b-chat, Mixtral-8x7b-Instruct. We randomly extract a subset from GRBENCH for experimentation. The results are shown in Table 3. From the results, we find that the choice of LLM backbone matters. LLMs with stronger instruction-following and reasoning abilities lead to better performance in GRAPH-O1.

Table 3: Results of GRAPH-O1 with different LLM backbones.

Model	GPT4 Score
LLaMA-2-7b-chat	7.64
LLaMA-2-13b-chat	14.58
Mixtral-8x7b-Instruct	32.33

Ablation Study on MCTS Parameters. To examine how search settings influence reasoning

performance, we analyze two key MCTS parameters: the maximum depth (`max_depth`) and the search width (`width`). As shown in Table 4, shallow searches (e.g., depth = 6) often stop too early, leading to incomplete reasoning and lower scores. Increasing the depth to 8–10 improves multi-hop reasoning, while deeper searches (> 12) add little benefit but increase cost. Wider searches explore more candidate paths, improving accuracy slightly but requiring much more computation. A moderate width of 2–3 achieves a good balance between performance and efficiency.

Table 4: Effect of search depth and width on reasoning performance in the Healthcare dataset. w means width

Depth	R-L (w=2)	R-L (w=3)	R-L (w=4)
6	17.35	17.51	17.62
8	27.58	27.79	27.94
10	29.54	29.67	30.94
12	29.85	30.39	31.21

GRPO Effecitiveness. We investigate the impact of the GRPO optimization on GRAPH-O1's reasoning performance by comparing results with and without GRPO (Table 5). Removing GRPO leads to drops in ROUGE-L scores. This indicates that the end-to-end reinforcement learning component plays a important role in guiding the LLM agent to select more effective graph traversal and reasoning actions, thereby improving multi-hop question answering performance.

Table 5: Model performance on GRBENCH using ROUGE-L.

Model	Academic	E-commerce	Literature	Healthcare	Legal
Without GRPO	31.42	33.71	35.85	26.79	26.39
With GRPO	**32.35**	**34.61**	**38.39**	**29.54**	**27.05**

5 RELATED WORK

The intersection of LLMs and graph-structured data has attracted growing research interest across multiple directions. Early efforts focus on fine-tuning LLMs directly on graph tasks: KG-GPT Kim et al. (2023) combines LLMs with graph retrieval pipelines for relation identification, CSProm-KG Chen et al. (2023) bridges structural and textual representations through prompt tuning, and ECOLA Han et al. (2022) jointly optimizes over knowledge graphs and textual context to recover missing graph information. A shared limitation of these approaches is the prohibitive cost of retraining as data complexity and graph scale grow. Retrieval-Augmented Generation (RAG) Shi et al. (2023) offers a practical alternative by injecting retrieved knowledge at inference time without modifying model weights, with subsequent work further refining retrieval quality through query rewriting Vakulenko et al., candidate reranking Yin et al. (2024), and local-global graph retrieval strategies Edge et al. (2025). On the reasoning side, Graph-CoT Jin et al. (2024) structures graph traversal as explicit chain-of-thought steps, ReAct Yao et al. (2023) interleaves natural language reasoning with tool-augmented actions, and multi-agent collaboration frameworks Guo et al. (2024) distribute the reasoning workload across specialized roles to tackle more complex queries. Orthogonally, reinforcement learning has emerged as a compelling post-training paradigm for pushing LLM capabilities beyond what standard supervised approaches achieve. Rather than relying solely on PPO or DPO, recent methods introduce richer feedback signals and tree-based exploration: MCTS-DPO Xie et al. (2024) selects high-Q-value Monte Carlo tree nodes as contrastive training pairs, iterative DPO Yuan et al. (2025) closes the loop by having models generate and evaluate their own outputs, and ReST Gulcehre et al. (2023) alternates between growing a candidate pool and fine-tuning exclusively on those exceeding a quality threshold. Process-level supervision provides an additional layer of granularity, with Lightman et al. Lightman et al. (2023) demonstrating that step-by-step feedback substantially outperforms outcome-only reward signals; this insight has since been extended through automated annotation via external LLMs Luo et al. (2025), Monte Carlo rollout estimation Wang et al. (2024), and MCTS-based step evaluation Luo et al. (2024), each substantially reducing dependence on costly human labeling.

6 CONCLUSION

This paper introduces GRAPH-O1, a framework that enables Large Language Models (LLMs) to perform structured, step-by-step reasoning over text-attributed graphs. By integrating Monte Carlo Tree Search (MCTS) with the generative capabilities of LLMs, GRAPH-O1 selectively and iteratively explores the graph, focusing only on the most relevant nodes and textual information. This approach improves both reasoning depth and computational efficiency. To further enhance performance, we employ reinforcement learning with Grouped Reinforcement Policy Optimization (GRPO), which optimizes the model's decision-making during graph traversal. Experimental results demonstrate that GRAPH-O1 achieves superior performance compared to all baseline methods.

AI USE DISCLOSURE

We used generative AI tools to improve the clarity, readability, and grammar of the manuscript. No generative AI tools were used for research ideation, methodology design, experimentation, data analysis, or interpretation of results. All AI-assisted text was reviewed and revised by the authors, who take full responsibility for the final content of this work.

REFERENCES

Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

Mixtral of experts, 2024. URL https://arxiv.org/abs/2401.04088.

A Acharya and S Adhikari. Alexa conversations: An extensible data-driven approach for building task-oriented dialogue systems. 2021.

Chen Chen, Yufei Wang, Aixin Sun, Bing Li, and Kwok-Yan Lam. Dipping plms sauce: Bridging structure and text for effective knowledge graph completion via conditional soft prompting. *arXiv preprint arXiv:2307.01709*, 2023.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization, 2025. URL https://arxiv.org/abs/2404.16130.

Aaron Grattafiori .et.al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, Wolfgang Macherey, Arnaud Doucet, Orhan Firat, and Nando de Freitas. Reinforced self-training (rest) for language modeling, 2023. URL https://arxiv.org/abs/2308.08998.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey of progress and challenges, 2024. URL https://arxiv.org/abs/2402.01680.

Zhen Han, Ruotong Liao, Jindong Gu, Yao Zhang, Zifeng Ding, Yujia Gu, Heinz Koeppl, Hinrich Schütze, and Volker Tresp. Ecola: Enhanced temporal knowledge embeddings with contextualized language representations. *arXiv preprint arXiv:2203.09590*, 2022.

Bowen Jin, Chulin Xie, Jiawei Zhang, Kashob Kumar Roy, Yu Zhang, Zheng Li, Ruirui Li, Xianfeng Tang, Suhang Wang, Yu Meng, and Jiawei Han. Graph chain-of-thought: Augmenting large language models by reasoning on graphs, 2024. URL https://arxiv.org/abs/2404.07103.

Haolin Jin, Linghan Huang, Haipeng Cai, Jun Yan, Bo Li, and Huaming Chen. From llms to llm-based agents for software engineering: A survey of current, challenges and future, 2025. URL https://arxiv.org/abs/2408.02479.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

Jiho Kim, Yeonsu Kwon, Yohan Jo, and Edward Choi. Kg-gpt: A general framework for reasoning on knowledge graphs using large language models. *arXiv preprint arXiv:2310.11220*, 2023.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, Yansong Tang, and Dongmei Zhang. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct, 2025. URL https://arxiv.org/abs/2308.09583.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. Improve mathematical reasoning in language models by automated process supervision, 2024. URL https://arxiv.org/abs/2406.06592.

OpenAI. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

A Radford, J Wu, and R Child. Language models are unsupervised multitask learners. 2018.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. Replug: Retrieval-augmented black-box language models. *arXiv preprint arXiv:2301.12652*, 2023.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

S Vakulenko, S Longpre, Z Tu, and R Anantha. Question rewriting for conversational question answering. WSDM '21. Association for Computing Machinery.

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations, 2024. URL https://arxiv.org/abs/2312.08935.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: A survey. *ACM Comput. Surv.*, 55(5), December 2022. ISSN 0360-0300. doi: 10.1145/3535101. URL https://doi.org/10.1145/3535101.

Yuxi Xie, Anirudh Goyal, Wenyue Zheng, Min-Yen Kan, Timothy P. Lillicrap, Kenji Kawaguchi, and Michael Shieh. Monte carlo tree search boosts reasoning via iterative preference learning, 2024. URL https://arxiv.org/abs/2405.00451.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.

Hang Yin, Zihao Wang, and Yangqiu Song. Rethinking complex queries on knowledge graphs with neural link predictors. In *The Twelfth International Conference on Learning Representations*, 2024.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models, 2025. URL https://arxiv.org/abs/2401.10020.

Zheni Zeng, Yuan Yao, Zhiyuan Liu, and Maosong Sun. A deep-learning system bridging molecule structure and biomedical text with comprehension comparable to human professionals. *Nature communications*, 13(862), 2022.

Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mańdziuk. Monte carlo tree search: a review of recent modifications and applications. *Artificial Intelligence Review*, 56 (3):2497–2562, July 2022. ISSN 1573-7462. doi: 10.1007/s10462-022-10228-y. URL http://dx.doi.org/10.1007/s10462-022-10228-y.

A Appendix

Limitations & Future Works

Despite strong empirical results, GRAPH-O1 has a few limitations worth noting. The multi-step MCTS reasoning incurs higher inference cost compared to standard RAG baselines, making it less suited for latency-sensitive settings. Additionally, GRAPH-O1 currently requires access to model internals, limiting compatibility with closed-source LLMs. Future work could explore more efficient search strategies to reduce runtime, extend the framework to dynamic and heterogeneous graph settings, and investigate black-box adaptation to broaden applicability.

Broader Impact

This paper aims to advance the field of graph-based question answering, text-attributed graph reasoning, and LLM agent planning. Potential applications span a wide range of domains including academic search, e-commerce recommendation, legal analysis, and healthcare decision support. We do not foresee any immediate negative societal consequences arising directly from this work.

A.1 Dataset

Table 6 shows the details of the dataset.

Table 6: Dataset statistics for GRBENCH, as introduced by Jin et al. (2024).

Domain	Topic	# Nodes	# Edges	# Templates	# Questions
Academic	CS	∼8M	∼52M	15	150
	Biology	∼4M	∼39M	14	140
	Chemistry	∼4M	∼30M	14	140
	Material Science	∼3M	∼22M	14	140
	Medicine	∼6M	∼30M	14	140
	Physics	∼2M	∼33M	14	140
E-commerce	Amazon	∼9M	∼313M	20	200
Literature	Goodreads	∼3M	∼22M	24	240
Healthcare	Disease	47K	∼4M	27	270
Legal	Freelaw	∼84M	∼114M	18	180

A.2 Example of GRAPH-O1

Table 7: Template for GRAPH-O1.

Instruction Template:

Solve a question answering task with interleaving **Thought**, **Interaction with Graph**, and **Feedback from Graph** steps.
In the *Thought* step, you should reason about what additional information is required to answer the question. In the *Interaction* step, you can gather feedback from the graph using four available functions: (1) `RetrieveNode[keyword]` retrieves the related node from the graph according to the given query keyword. (2) `NodeFeature[Node, feature]` returns the detailed attribute information of the specified node regarding the given "feature" key. (3) `NodeDegree[Node, neighbor_type]` calculates the number of neighbors of the node that match the specified "neighbor_type." (4) `NeighbourCheck[Node, neighbor_type]` lists all the "neighbor_type" neighbors of the given node and returns them. You may take as many reasoning and interaction steps as necessary until the answer is reached.
Below are several illustrative examples demonstrating the interaction pattern: `graph_definition`
Example: `Question: How many papers are written by author Nicholas Lydon? Thought 1: The question is asking for the number of written papers of a specific author (Nicholas Lydon). We need to find the author node in the graph. Action 1: RetrieveNode[Nicholas Lydon] Observation 1: The ID of this retrieval target node is 53f438c3dabfaedf43596117. Thought 2: The question is asking for the number of papers written by Nicholas Lydon. We need to calculate the "paper" neighbor degree of this node. Action 2: NodeDegree[53f438c3dabfaedf43596117, paper] Observation 2: 2 Thought 3: The number of the paper neighbors is 2. Action 3: Finish[2]` (END OF EXAMPLES)
Definition of the graph: `graph_definition`
Question: `question` Please answer by providing the node's main feature (e.g., name) rather than the node ID. `scratchpad`

Table 8: A real example of MCTS reasoning.

Question	What is the start date of court Oyer and Terminer, Rensselaer County? Please answer by providing node main feature (e.g., names) rather than node IDs.
Answer	1788-01-01
GRAPH-O1 MCTS node for Action 2	Thought 1: The question is about the molecular function of a specific gene, SSX7. I need to look for the corresponding node in the graph. Action 1: RetrieveNode["SSX7"] Observation 1: The ID of this retrieval target node is 280658. **Thought 2: Now I need to check the features of the node "SSX7".** **Action 2: NodeFeature[280658, "molecular function"]**
GRAPH-O1 MCTS node for new Action 2	Thought 1: To answer the question, I need to find the node corresponding to gene SSX7 from the graph. Action 1: RetrieveNode[SSX7] Observation 1: The ID of this retrieval target node is 280658. **Thought 2: To further answer the question, I need to check the node's 'Gene-participates-Molecular Function' neighbors from the graph to get the node names of the neighboring Molecular Function nodes.** **Action 2: NeighbourCheck[280658, Gene-participates-Molecular Function]**

Table 9: Runtime comparison of different models. GraphCoT produces an incorrect answer, while GRAPH-O1 returns the correct one.

Question	what are the side effects of compound Acetic acid?
Wrong reasoning	Thought 1: The question asks for the side effects (a kind of node), we need to check if the node has any side effects. Action 1: NodeDegree[DB01135, 'Compound-causes-Side Effect'] Observation 1: The node or neighbor type does not exist in the graph. This might because your given neighbor type is not correct. Please modify it. Thought 2: The question asks for the side effects (a kind of node), we might want to find the side effects by checking the graph. Action 2: NeighbourCheck[DB01135, 'Compound-causes-Side Effect'] Observation 2: The node or neighbor type does not exist in the graph. This might because your given neighbor type is not correct. Please modify it. Thought 3: The compound 'Acetic acid' does not cause any side effects. Action 3: Finish[There are no side effects of compound 'Acetic acid'.] Observation 3: Answer is INCORRECT Correct answer: Stinging, Pain, Acidosis, Hypernatraemia, Haematuria.
Correct reasoning by GRAPH-O1	Thought 1: The question is related to a compound node (Acetic acid). We need to find the node in the graph. Action 1: RetrieveNode[Acetic acid] Observation 1: The ID of this retrieval target node is DB03166. Thought 2: The question is asking the node's neighbours of 'Compound-causes-Side Effect' relation from the graph. Action 2: NeighbourCheck[DB03166, 'Compound-causes-Side Effect'] Observation 2: The Compound-causes-Side Effect neighbors of DB03166 are: ['C0677500', 'C0030193', 'C0001122', 'C0020488', 'C0018965']. Thought 3: We got the side effect nodeIDs. Now, we need to retrieve the feature 'name' from the node. Action 3: NodeFeature[C0677500, name], NodeFeature[C0030193, name], NodeFeature[C0001122, name], NodeFeature[C0020488, name], NodeFeature[C0018965, name] Observation 3: The name feature of C0677500 are: Stinging. The name feature of C0030193 are: Pain. The name feature of C0001122 are: Acidosis. The name feature of C0020488 are: Hypernatraemia. The name feature of C0018965 are: Haematuria. Thought 4: We got the name of side effects. Action 4: Finish[(Stinging, Pain, Acidosis, Hypernatraemia, Haematuria)] Observation 4: Answer is CORRECT Correct answer: Stinging, Pain, Acidosis, Hypernatraemia, Haematuria.

B IMPLEMENTATION DETAILS

B.1 MCTS CONFIGURATION

Table 10 lists the search settings used for GRAPH-O1. The search stops as soon as any branch produces a final answer, so the simulation count is an upper bound rather than a fixed cost.

B.2 GRPO TRAINING CONFIGURATION

Training data are built from the MCTS search logs on the training questions. Only trajectories whose final answer matches the ground truth are kept, so every training prompt comes from a successful, graph-grounded reasoning trace. Table 11 lists the optimization settings.

Table 10: MCTS hyperparameters.

Setting	Value
Exploration weight c (UCT)	1.414 ($\sqrt{2}$)
Maximum depth	10
Maximum simulations per question	20 (early stop at first final answer)
Discount factor γ	0.9
Final answer	answer of the first completed branch; otherwise most-visited child
Policy decoding temperature	0
Agent step limit	10

Table 11: GRPO hyperparameters (TRL 0.14.0).

Setting	Value
Backbone	Mixtral-8x7B-Instruct-v0.1
Adapter	LoRA, r=8, α=16, dropout 0.05, modules `q_proj`, `v_proj`
Optimizer	AdamW
Learning rate	1×10^{-6}
Per-device batch size / gradient accumulation	1 / 8
Generations per prompt (group size)	8
Sampling temperature (training rollouts)	0.9
Maximum completion length	256 tokens
KL coefficient β	0.04
Epochs	3
Random seed	42

B.3 Reward Function (Implementation Details)

Format reward. Each of the three reasoning components contributes 0.5 when it appears in the completion:

$$R_{\text{format}}(o) = 0.5 \cdot \Big(\mathbb{I}\{\texttt{Thought} \in o\} + \mathbb{I}\{\texttt{Action} \in o\} + \mathbb{I}\{\texttt{Observation} \in o\} \Big) \in [0, 1.5]. \quad (2)$$

Reasoning reward. The completion earns 0.5 for invoking one of the graph operations and loses 0.5 when it reports an error or failure (e.g., a relation or node that does not exist):

$$R_{\text{reasoning}}(o) = 0.5 \cdot \mathbb{I}\{o \text{ invokes a graph operation}\} - 0.5 \cdot \mathbb{I}\{o \text{ reports an error or failure}\}, \quad (3)$$

where the graph operations are `RetrieveNode`, `NeighbourCheck`, `NodeFeature` and `NodeDegree`.

Overall reward. The two components are summed and clipped for stability:

$$R(o) = \max\big(-1.0,\ \min\big(3.0,\ R_{\text{format}}(o) + R_{\text{reasoning}}(o)\big)\big). \quad (4)$$

The attainable range is $[-0.5,\ 2.0]$, so the clip never binds in practice. Advantages are computed within each group as $\hat{A}_i = (R(o_i) - \text{mean}_j R(o_j))/\text{std}_j R(o_j)$ and plugged into the GRPO objective, with KL coefficient β=0.04.

Relation to the design goals in Section 1. *Structural coherence* is rewarded by R_{format}. *Meaningful graph interaction* is rewarded by the graph-operation term, and the error penalty discourages continuations that describe failed or non-existent graph accesses. *Answer fidelity* and *retrieval relevance* are not scored per completion. They come from selecting training states exclusively from trajectories whose retrieved evidence led to the correct answer.

B.4 Retrieval and Evaluation

Node retrieval uses `sentence-transformers/all-mpnet-base-v2` embeddings with FAISS. ROUGE-L is computed with the HuggingFace `evaluate` implementation. GPT4score uses GPT-4 (temperature 0.01, top-p 1.0) as a binary judge with the prompt:

```
Question:{question} Model prediction: {prediction} Ground
truth: {answer}. Please help me judge if the model
prediction is correct or not given the question and ground
truth answer. Please use one word (Yes or No) to answer.
Do not explain.
```

The system message is "You are a generative language model evaluator."

C Statistical Reliability

Table 12 reports the standard error of each score over test questions.

Table 12: Model performance on GRBENCH. R-L: ROUGE-L; G4: GPT-4 score.

Model	**Academic**		**E-commerce**		**Literature**		**Healthcare**		**Legal**	
	R-L	G4	R-L	G4	R-L	G4	R-L	G4	R-L	G4
GRAPH-MCTS	31.42	32.16	33.71	37.22	35.85	45.83	26.79	32.84	26.39	27.78
GRAPH-COT	27.35	29.43	29.38	32.78	32.44	38.75	23.33	26.02	19.81	22.77
GRAPH-O1	**32.35** ±0.13	**33.82** ±0.21	**34.61** ±0.31	**38.76** ±0.25	**38.39** ±0.48	**45.83** ±0.22	**29.54** ±0.37	**33.70** ±0.28	**27.05** ±0.06	**32.20** ±0.18

D Computational Cost

Table 13 reports the wall-clock latency per question under the configuration used for Table **??**. The MCTS methods incur higher latency than the single-trajectory GRAPH-COT because they explore branches that are ultimately discarded. GRAPH-O1 has a latency similar to GRAPH-MCTS, as GRPO updates the policy without changing the search budget.

Table 13: Latency per question (seconds).

Domain	GRAPH-COT	GRAPH-MCTS	GRAPH-O1
Legal	46.6	54.3	51.4
E-commerce	26.9	46.7	51.9
Literature	32.0	32.7	29.3
Healthcare	27.7	44.4	50.4

E Path-Level Diagnostics

Invalid-action rate. Table 14 reports the fraction of graph operations the environment rejected (e.g., a relation type absent from the node's schema). Rejected operations are common because the agent must name relations and features exactly. The agent usually recovers within the same trajectory, and GRAPH-O1 has the lowest rate in every domain.

Table 14: Invalid-action rate per action attempt.

Domain	GRAPH-COT	GRAPH-MCTS	GRAPH-O1
Legal	40.3%	38.3%	35.9%
E-commerce	24.9%	22.2%	20.4%
Literature	29.0%	27.4%	23.5%
Healthcare	48.1%	39.3%	37.6%

Evidence-path agreement (Legal). GRBENCH questions are generated from templates with a known relation chain. For the 130 Legal questions with a well-defined chain, we check whether the relation types a trajectory traverses match the template's chain (Table 15). This is a type-level check:

a trajectory can match the chain while visiting a wrong entity, so the match rate is an upper bound on exact path recall. GRAPH-O1 follows the evidence chain more often, and 84.2% of its correct answers come from chain-matching trajectories (GRAPH-MCTS: 54.5%; GRAPH-COT: 64.0%). Its gains therefore come from following the evidence path rather than from lucky shortcuts.

Table 15: Evidence-chain diagnostics on Legal (130 questions).

	GRAPH-COT	GRAPH-MCTS	GRAPH-O1
Accuracy	19.2%	25.4%	29.2%
Chain-match rate	30.8%	30.8%	42.3%
$P(\text{correct} \mid \text{chain matches})$	40.0%	45.0%	58.2%
$P(\text{correct} \mid \text{no match})$	10.0%	16.7%	8.0%
Correct answers via chain match	64.0%	54.5%	84.2%

F COST-PERFORMANCE TRADE-OFF

Measurement. We measure the inference cost of the runs reported in Table **??**. The counts include every expansion step executed during search, including steps on branches that are not part of the final answer path. Each reasoning step issues two LLM calls, one to generate the Thought and one to generate the Action. We report:

- LLM calls per question;
- generated tokens per question (Mixtral tokenizer);
- graph operations per question (`RetrieveNode`, `NeighbourCheck`, `NodeFeature`, `NodeDegree`), including operations the environment rejected as invalid;
- median wall-clock time per question.

All methods use Mixtral-8x7B-Instruct on NVIDIA H200 NVL GPUs.

Table 16: Inference cost per question and accuracy on Legal (180 questions).

Method	LLM calls	Gen. tokens	Graph ops	Invalid ops	Time (s)	R-L
GRAPH-COT	13.06	508	4.79	2.82	47.5	19.81
GRAPH-MCTS	19.57	755	6.76	4.66	50.5	26.39
GRAPH-O1	17.92	667	6.42	4.22	108.0	27.05

Findings. (i) GRAPH-O1 is cheaper than GRAPH-MCTS by every hardware-independent measure. Under the same search procedure and budget, the trained policy needs 8.4% fewer LLM calls, 11.7% fewer generated tokens, 5.0% fewer graph operations and 9.4% fewer invalid operations, while achieving higher accuracy. Policy training therefore makes search more efficient: fewer expansions are spent on unproductive branches. **(ii) Search stops early.** GRAPH-MCTS executes on average 9.8 and GRAPH-O1 9.0 expansion steps per question. **(iii) The accuracy gain over single-trajectory reasoning costs little.** Compared with GRAPH-COT, GRAPH-O1 uses 37% more LLM calls and 31% more generated tokens and improves R-L by 7.24 points (19.81→27.05) and GPT4score by 9.43 points (22.77→32.20). **(iv) Wall-clock time.** GRAPH-O1 has a higher median time per question despite issuing fewer calls. Wall-clock time depends on how the model is deployed and on other jobs sharing the GPUs, so we treat LLM calls and tokens as the primary cost measures.

Training cost. GRPO training is a one-time cost per domain: 2.6 GPU-hours for E-commerce (9,353 s), and up to 4.4 GPU-hours (15,707 s) for another domain. It is amortized over all inference queries and adds no per-question overhead beyond the search procedure shared with GRAPH-MCTS.

Table 17: Inference cost per question on E-commerce and Literature.

Domain	Method	LLM calls	Gen. tokens	Graph ops	Invalid ops	R-L
E-commerce	GRAPH-COT	12.85	453	4.29	1.94	29.38
	GRAPH-O1	18.40	749	6.20	3.09	34.61
Literature	GRAPH-COT	12.72	622	4.54	1.95	32.44
	GRAPH-O1	17.10	540	6.23	2.72	38.39