Grounding LLM Reasoning with Knowledge Graphs

Alfonso Amayuelas[*]
UC Santa Barbara
amayuelas@ucsb.edu

Joy Sain, Simerjot Kaur, Charese Smiley
JP Morgan AI Research
joy.sain, simerjot.kaur, charese.h.smiley@jpmchase.com

Abstract

Large Language Models (LLMs) excel at generating natural language answers, yet their outputs often remain unverifiable and difficult to trace. Knowledge Graphs (KGs) offer a complementary strength by representing entities and their relationships in structured form, providing a foundation for more reliable reasoning. We propose a novel framework that integrates LLM reasoning with KGs by linking each step of the reasoning process to graph-structured data. This grounding turns intermediate "thoughts" into interpretable traces that remain consistent with external knowledge. Our approach incorporates multiple reasoning strategies, Chain-of-Thought (CoT), Tree-of-Thought (ToT), and Graph-of-Thought (GoT), and is evaluated on GRBench, a benchmark for domain-specific graph reasoning. Our experiments show state-of-the-art (SOTA) performance, with at least 26.5% improvement over CoT baselines. Beyond accuracy, we analyze how step depth, branching structure, and model size influence reasoning quality, offering insights into the conditions that support effective reasoning. Together, these contributions highlight how grounding LLMs in structured knowledge enables both higher accuracy and greater interpretability in complex reasoning tasks.

1 Introduction

LLMs have shown remarkable versatility in answering questions posed in natural language. This is mainly due to their ability to generate text, their broad internal knowledge, and their capacity to access external information [Zhuang et al., 2023, Lewis et al., 2020]. However, a significant area for improvement is their tendency to produce information that, while plausible-sounding, is often unverifiable and lacks traceable origins and sources [Tonmoy et al., 2024]. This limitation highlights a deeper issue in how LLMs organize and apply reasoning, especially when reliable and accountable outputs are required.

The LLM generation process heavily relies on their internal parameters, making it difficult to link their outputs to external sources [Ko et al., 2024, Zheng et al., 2024]. This limitation challenges their reliability in industrial applications [Luo and Specia, 2024]. In applied settings, where LLMs handle critical operations, integrating them with domain-specific knowledge is essential. Fine-tuning LLMs for new domains is labor-intensive, especially for companies with proprietary data facing privacy and legal issues. As a result, there is a need for interventions that guide reasoning processes so that outputs remain accurate and transparent without requiring exhaustive re-training.

Methods such as Retrieval-Augmented Generation (RAG) [Lewis et al., 2020] and SQL-based querying [Rajkumar et al., 2022] address this gap partially. However, they often fail to capture the dynamic relationships between concepts that are necessary for comprehensive understanding. These approaches typically assume that knowledge is well-represented in discrete units, such as documents or tables, which can lead to incomplete insights when dealing with interconnected knowledge that spans multiple sources. This limits the ability to support reasoning over complex queries.

[*]Work completed during an internship at JP Morgan AI Research

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Foundations of Reasoning in Language Models.

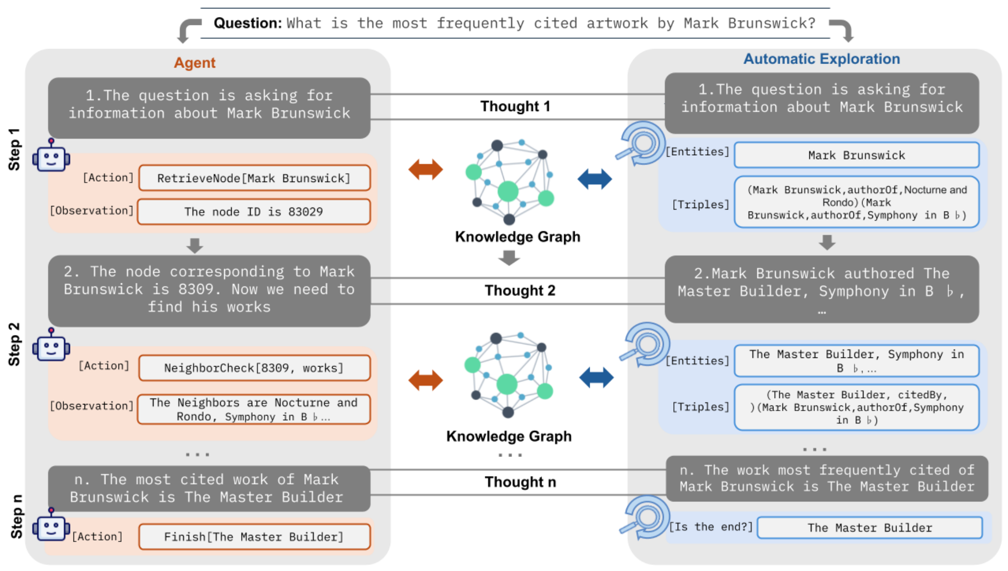

Question: What is the most frequently cited artwork by Mark Brunswick?
Agent
Automatic Exploration
Step 1
1.The question is asking for information about Mark Brunswick
Thought 1
[Action] RetrieveNode[Mark Brunswick]
[Observation] The node ID is 83029
Knowledge Graph
[Entities] Mark Brunswick
[Triples] (Mark Brunswick,authorOf,Nocturne and Rondo)(Mark Brunswick,authorOf,Symphony in B ♭)
Step 2
2. The node corresponding to Mark Brunswick is 8309. Now we need to find his works
Thought 2
2.Mark Brunswick authored The Master Builder, Symphony in B ♭, …
[Action] NeighborCheck[8309, works]
[Observation] The Neighbors are Nocturne and Rondo, Symphony in B ♭…
[Entities] The Master Builder, Symphony in B ♭, …
[Triples] (The Master Builder, citedBy,)(Mark Brunswick,authorOf,Symphony in B ♭)
Knowledge Graph
Step n
n. The most cited work of Mark Brunswick is The Master Builder
Thought n
n. The work most frequently cited of Mark Brunswick is The Master Builder
[Action] Finish[The Master Builder]
[Is the end?] The Master Builder

Figure 1: Methods for Question-Answering in KGs (Section 4). **Left**: *Agent*. LLM decides to take one of the predefined actions to connect with the graph. **Right:** *Automatic Graph Exploration*. Entities are extracted in each reasoning step, triggering a search for each identified entity.

KGs capture such relationships by organizing entities and their connections in a structured representation. Recent work has begun to explore how KGs can guide reasoning in LLMs [Luo et al., 2024]. Building on this direction, we propose a framework that integrates reasoning strategies with domain-specific KGs from the GRBench dataset. Each reasoning step is explicitly connected to the graph, producing answers that are both accurate and traceable. We evaluate three strategies, CoT, ToT, and GoT, which provide different ways of organizing reasoning traces (Figure 2). Our framework achieves SOTA performance on GRBench, improving by at least 26.5% over CoT baselines. Beyond accuracy, we analyze how reasoning behaviors vary with step depth, branching, and model size (Section 7), offering insights into how reasoning evolves during inference and how it can be shaped by structured knowledge. Our contributions can be summarized as follows:

- We present a versatile framework that links reasoning steps with graph search, providing an intervention that can be applied across domains and knowledge settings.
- We show substantial gains over existing reasoning strategies, with state-of-the-art performance on GRBench and an improvement of at least 26.5% compared to CoT baselines.
- We conduct a detailed study of how reasoning quality is influenced by step depth, branching structure, and model size, offering insights into the conditions under which different interventions succeed (Section 7).
- By systematically grounding each step in a knowledge graph, we improve both the accuracy and the traceability of outputs, creating reasoning traces that are interpretable and verifiable.

2 Related Work

LLMs require substantial data and resources for training Villalobos et al. [2022]. Retrieval-Augmented Generation (RAG) enable the models to incorporate external evidence at inference time [Lewis et al., 2020, Khattab et al., 2022]. Recent work further combines RAG with structured knowledge, such as ontologies and KGs, to improve factuality and reasoning Li et al. [2024], underscoring the growing importance of structured data for robust and domain-adaptable LLMs.

Structured Knowledge Structured knowledge, such as databases or KGs, provides organizations with reliable sources of information that can be systematically maintained and automatically updated. KGs in particular offer an adaptable knowledge model that captures complex relationships between interconnected concepts. Research has explored models that can interact with multiple types of structured knowledge, such as StructLM [Zhuang et al., 2024], and approaches that incorporate structured knowledge during pretraining to improve model performance [Moiseev et al., 2022].

Integrating KGs with LLMs The integration of KGs with LLMs has emerged as a promising direction to strengthen reasoning capabilities and reliability [Peng et al., 2024]. In general, four primary methods can be distinguished: (1) learning graph representations [Fatemi et al., 2024, Chai et al., 2023], though latent representations often underperform compared to text-based methods on Knowledge Graph Question Answering (KGQA) tasks; (2) using Graph Neural Network (GNN) retrievers to extract relevant entities and feeding them as text-based input to the model [He et al., 2024, Mavromatis and Karypis, 2024]; (3) generating code, such as SPARQL queries, to directly retrieve information from graphs [Li et al., 2023]; and (4) step-by-step interaction methods that allow iterative reasoning over graphs [Luo et al., 2024, Yu et al., 2022, Luo et al., 2023], which currently achieve the strongest results on KGQA benchmarks.

LLM Reasoning with Graphs Beyond retrieval, KGs have also been studied as a means to structure and analyze the reasoning processes of LLMs themselves Wang et al. [2024]. This integration enables more coherent and contextually relevant outputs while also supporting the tracing and verification of reasoning steps. The most effective methods typically rely on interactive, step-by-step engagement between LLMs and graphs, as discussed above. Examples of this approach include systems such as Sun et al. [2024], Wen et al. [2024], Luo et al. [2024], Kim et al. [2023], Li et al. [2025], which demonstrate improved reasoning performance through graph-based scaffolding. More recent work has further investigated the integration of traditional reasoning strategies, such as CoT and tree-structured reasoning, into KG-based interaction [Jin et al., 2024, Markowitz et al., 2024].

Building on these advances, our framework integrates established reasoning strategies directly with domain-specific KGs. Unlike previous methods that treat KGs as retrieval tools or rely on latent representations, our approach systematically links each reasoning step to graph entities and relations.

3 Background

In this section, we formalize the prerequisite knowledge relevant to this paper. We use p_θ to denote a pre-trained language model with parameters θ, and letters x, y, z to refer to a language sequence. $x = (x_1, x_2, ..., x_n)$, where each generated token is x_i is a such that $p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i|x_{1...i-1})$.

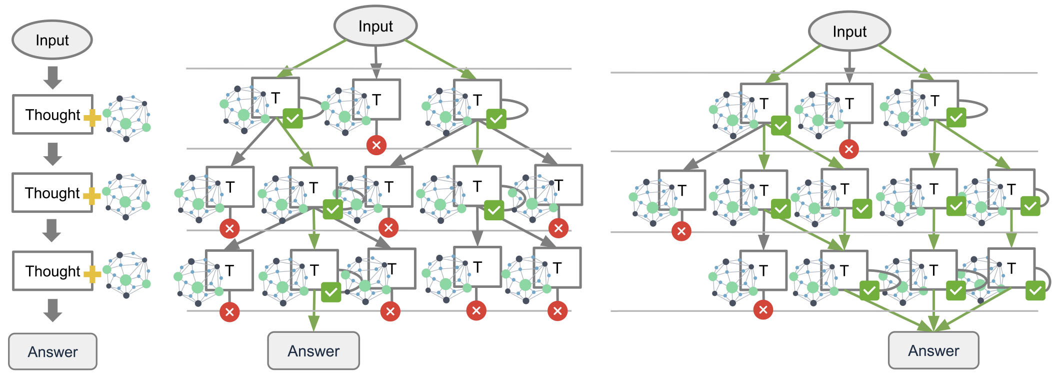
Knowledge Graphs (KGs) A KG is a heterogeneous directed graph that contains factual knowledge to model structured information. Nodes represent entities, events, or concepts, while edges represent the connection and types of relations between them. Formally, a KG is represented as $\mathcal{G}$, defined by a set of triples $\mathcal{G} = \{(h, r, t) \mid h, t \in \mathcal{E}, r \in \mathcal{R}\}$, where $\mathcal{E}$, $\mathcal{R}$ denote the set of entities and relations, respectively. KGs provide a structured framework that can guide reasoning processes by explicitly representing the relationships between concepts.

Knowledge Graph Question-Answering (KGQA) It is a reasoning task that leverages KGs. Given a natural language question, q, and an associated KG, $\mathcal{G}$, the goal is to develop a method that retrieves the correct answer, a, based on the knowledge extracted from the KG: $a = f(q, \mathcal{G})$. Beyond retrieving facts, KGQA often requires integrating multiple reasoning steps that traverse the graph to connect related concepts.

Step-by-step Reasoning with LLMs To improve the reasoning capabilities of LLMs at inference time, a common approach is to generate intermediate reasoning steps. The key idea is the introduction of intermediate steps, $Z_{p_\theta} = z_1, ..., z_n$, to add inference sources to bridge the q and a. This decomposition allows models to tackle complex, multi-step problems incrementally, focusing computational effort on parts of reasoning chain that require deeper analysis. Stepwise reasoning over KGs offers a natural mechanism to track, guide, and interpret the reasoning process.

4 Method

This work demonstrates how progressively conditioning LLM reasoning at each step can enhance performance on domain-specific question answering over knowledge graphs. By structuring reasoning into incremental steps that interact with graph data, the model can manage complex dependencies and dynamically refine its conclusions. Our method combines reasoning strategies for LLMs: *CoT, ToT, GoT* with 2 graph interaction methods: (1) Agent, an agent to navigate the graph; and (2) Automatic Graph Exploration, an automatic graph traversal mechanism based on the generated text.



(a) Chain of Thought (CoT) (b) Tree of Thought (ToT) (c) Graph of Thought (GoT)

Figure 2: Reasoning Strategies: This figure illustrates different LLM reasoning strategies to navigate the potential answer space: CoT, ToT, GoT. Each strategy consists of "thoughts" connected to the KG through search methods (Section 4.2) illustrating stepwise reasoning over structured knowledge.

4.1 Reasoning Strategies

Chain-of-Thought (CoT) CoT is a well-known reasoning method that involves generating a sequence of logical steps, where each step builds upon previous ones, ultimately leading to a conclusion. Formally, it generates a sequence of reasoning steps $Z_{p_\theta}(q) = \{z_1, z_2, \dots, z_n\}$, where each step z_i is sampled sequentially given the input query q, all previous steps and graph information from all steps , $\mathcal{G}'$, as $z_i \sim p_\theta^{\text{CoT}}(z_i|q, \mathcal{G}', z_{1\dots i-1})$. The final answer a is derived from this reasoning process given all the generated thoughts $a \sim p_\theta^{\text{CoT}}(a|q, \mathcal{G}', z_{1\dots n})$. In practice, it is sampled as a continuous language sequence. Figure 2a represents this method, where each step is linked to the KG.

Tree-of-Thought (ToT) ToT generalizes CoT by modeling the reasoning process as a tree, enabling simultaneous exploration of multiple reasoning paths. Starting from an initial state $s_0 = [q]$, where q is the input, ToT incrementally expands each state by generating multiple candidate thoughts:

$$z_{i+1}^{(j)} \sim p_\theta(z_{i+1} \mid s_i), \quad j = 1, \dots, k \tag{1}$$

Each candidate thought represents a node in the tree, forming new states. These states are evaluated by a heuristic scoring function $V(p_\theta, s)$, guiding the selection and pruning of branches. Search strategies, such as breadth-first search (BFS), systematically explore this tree:

$$S_t = \text{argmax}_{S' \subseteq \hat{S}_t, |S'|=b} \sum_{s \in S'} V(p_\theta, s) \tag{2}$$

where $\hat{S}_t$ denotes candidate states at step t, and b limits the breadth. We implement two versions of heuristic functions V to select the top t states:

1. Selection: The LLM directly chooses the top t states to proceed, discarding the others.
2. Score: The states are ranked by a heuristic voting mechanism: $V(p_\theta, S)(s) = \mathbb{P}[s = s^*]$ where the LLM is prompted to estimate probability of the current state solving the given input question.

This structured search and pruning strategy allows the model to evaluate multiple candidate reasoning paths, enabling more deliberate and interpretable reasoning.

Graph-of-Thought (GoT) GoT extends ToT by organizing reasoning into a directed graph structure $G = (V, E)$, where each node represents a thought and edges reflect dependencies. Starting from initial thought, new thoughts are generated similarly to ToT and added to the graph. Each new thought is connected to its parent, and additional reasoning chains can be formed through merging operations:

$$z_{i+1} = A(z_i^{(a)}, z_i^{(b)}) \tag{3}$$

where A denotes a merge operation that integrates two thought chains into a single coherent reasoning step. The merged thought is added as a new node with edges from both parents. In our implementation, thoughts are evaluated using either Selection- or Score-based strategy as in ToT. Merged thoughts inherit information from both parents and can enhance robustness. At each depth, a fixed number of thoughts are retained using breadth-first traversal and evaluated for progression. The graph-based organization captures dependencies and merges information from multiple reasoning chains, supporting dynamic refinement and structured exploration of the reasoning space.

4.2 LLM + KG Interaction Methods

We implement methods to connect reasoning strategies with KGs. The LLM interacts with the KG at every step. This retrieves new information and conditions the model for subsequent steps. We present 2 methods to achieve this interaction, both illustrated in Appendix B.

4.2.1 Agent

This approach creates an agent that interacts with the graph, following the methodology initially described in ReACT [Yao et al., 2022]. After generating a thought, the LLM selects from a set of actions based on the given thought. Each step in the reasoning chain consists of an interleaved sequence: *thought* $\rightarrow$ *action* $\rightarrow$ *retrieved data*. This method implements four actions as described in GraphCoT [Jin et al., 2024]: (a) `RetrieveNode`(Text): Identifies the related node in the graph using semantic search, (b) `NodeFeature`(NodeID, FeatureName): Retrieves textual information for a specific node from the graph, (c) `NeighborCheck`(NodeID, EdgeType): Retrieves neighbors' information for a specific node, (d) `NodeDegree`(NodeID, EdgeType): Returns the degree (#neighbors) for a given node and edge type. These actions collectively enable the agent to navigate and extract meaningful information from the graph, enhancing the reasoning capabilities of the LLM by grounding its thoughts in structured, retrievable data.

4.2.2 Automatic Graph Exploration

This method incrementally searches the graph by interleaving language generation with structured retrieval. At each step, the LLM generates a new "thought" based on previous thoughts and retrieved triples. Entities mentioned in the generated text are automatically extracted using LLM prompts and serve as anchors for further graph exploration.

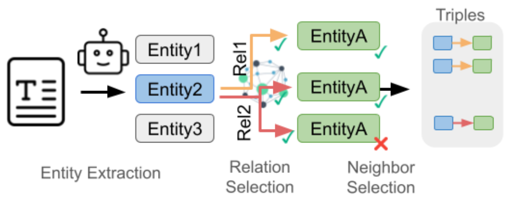


Figure 3: Automatic Graph Exploration. It extracts entities from text (query/thought), then select relevant relations and neighbors with the LLM. The resulting entity-relation-entity combinations form triples to expand the reasoning chain.

Graph exploration proceeds through a multi-step Search + Prune pipeline, inspired by the process described in Sun et al. [2024]. For each unvisited entity, the system first retrieves and prunes relation types using LLM guidance. Then, for each selected relation, neighboring entities are discovered and filtered using a second round of pruning. The model selects only the most relevant neighbors based on their contextual fit with the question and previous reasoning steps. This hierarchical pruning – first on relations, then on entities – ensures the method remains computationally tractable while preserving interpretability. The overall traversal follows a breadth-first search (BFS) pattern, with pruning decisions at each level directed by LLM. This process is shown in Figure 3. This iterative reasoning and retrieval process allows the model to condition future steps on progressively relevant subgraphs, shaping the reasoning trajectory. Unlike agentic methods that rely on predefined actions, the automatic approach operates in the graph space guided by the natural language, providing more freedom in the generation. The mechanism is designed to maximize information gain at each step while avoiding graph overgrowth. More details are provided in Algorithm 1.

5 Experiments

Benchmark We use the GRBench dataset to evaluate our methods. This dataset is specifically designed to evaluate how effectively LLMs can perform stepwise reasoning over domain-specific graphs. It includes several graphs spanning various general domains. For our evaluation, we selected 7 graphs across multiple domains, excluding those with excessively high RAM requirements that exceed our available resources. Comprehensive graph statistics are provided in Appendix A.

Baselines The proposed methods, *Agent* and *Automatic Graph Exploration*, applied to CoT, ToT, and GoT, are compared against the following baseline methods: (1) **Zero-Shot**: Directly querying the model to answer the question without additional context. (2) **Text RAG** Gao et al. [2023]: Text-retrieval method that uses text representation of nodes as input for query, with retrieved data serving as context for the model. (3) **Graph RAG**: Includes node neighbors (1-hop) for additional context beyond Text RAG. (4) **Graph CoT (Agent)**: Implements Graph CoT as an agent for CoT reasoning, utilizing the actions described in Section 4.2. These baselines allow us to measure impact of stepwise, knowledge-grounded reasoning versus simple retrieval-augmented or zero-shot approaches.

Experimental methods We implement the methods described in Section 4, extending (1) **Agent** and (2) **Automatic Graph Exploration** with various reasoning strategies during inference: (1) **CoT**, (2) **ToT**, and (3) **GoT**. For ToT and GoT, we evaluate the impact of stepwise decision-making using State Evaluation methods: (1) *Selection* and (2) *Score*. In the results presented in Table 1, we set $n = 10$ steps for all methods. ToT and GoT use a branching factor and Selection of $k = t = 3$. Our experiments focus on the effect of structured reasoning interventions on performance and stepwise refinement of answers. We use only open-access Llama 3.1 (Instruct) Dubey et al. [2024] as the backend models, which enhances reproducibility and allows for unlimited free calls. Specifically, we employ the 8B, 70B, and 405B versions, using the FP8 variant for the 405B model.

Evaluation We use rule-based and model-based metrics to evaluate the models, following GRBench paper Jin et al. [2024]. For the rule-based metric, we use Rouge-L (R-L) [Lin, 2004], which measures the longest sequence of words appearing in same order in both generated text and ground truth answer. For model-based metric, we prompt GPT-4o to assess if the model's output matches ground truth answer. GPT4Score is percentage of answers that GPT-4o identifies as correct. These evaluation methods capture not only final answer accuracy but also the fidelity of reasoning steps, reflecting the effectiveness of our interventions in guiding LLM reasoning over structured knowledge.

Implementation Details The experiments are run on NVIDIA TITAN RTX or NVIDIA A100 using Python 3.8. The models are deployed with vLLM Kwon et al. [2023], a memory-efficient library for LLM inference and serving. For the baseline, Mpnet-v2 is used as the retriever, and FAISS Johnson et al. [2019] is employed for indexing.

6 Results

The main results from both the baselines and experimental methods, evaluated using R-L, are presented in Table 1. For brevity, additional results using GPT4Score can be found in Appendix D. Together, these findings allow us to compare different forms of reasoning interventions, agentic action selection, automatic graph exploration, and structured multi-path search, on their ability to guide LLMs toward accurate answers. We highlight three key insights from the findings: (1) The agentic method generally outperformed automatic graph exploration, indicating that targeted interventions on knowledge graph traversal enhance answer accuracy. (2) The ToT strategy demonstrated superior performance by effectively exploring multiple reasoning paths, showcasing the benefits of inference-time interventions that diversify reasoning trajectories. (3) Although GoT strategy showed potential, it did not significantly outperform ToT, suggesting that merging divergent reasoning paths remains a challenging intervention design problem. These results show the importance of reasoning strategies in enabling LLMs to navigate multiple paths in the graph, while also illustrating the limits of current intervention techniques.

	Method		Model	Healthcare	Goodreads	Biology	Chemistry	Materials Science	Medicine	Physics
Baselines	Base		Llama 3.1 8B-Ins	7.32	6.18	10.68	11.69	8.95	8.77	11.52
			Llama 3.1 70B-Ins	9.74	9.79	11.49	12.58	10.40	12.21	12.61
			Llama 3.1 405B-Ins	8.66	12.49	10.52	13.51	11.73	11.82	11.63
	Text-RAG		Llama 3.1 8B-Ins	8.24	14.69	12.43	11.42	9.46	10.75	11.29
			Llama 3.1 70B-Ins	10.32	18.81	11.87	16.35	12.25	12.77	12.54
			Llama 3.1 405B-Ins	11.61	16.23	16.11	13.82	14.23	15.16	16.32
	Graph-RAG		Llama 3.1 8B-Ins	12.94	22.30	30.72	34.46	30.20	25.81	33.49
			Llama 3.1 70B-Ins	17.95	25.36	38.88	40.90	41.09	31.43	39.75
			Llama 3.1 405B-Ins	16.12	23.13	37.57	42.58	37.74	33.34	40.98
Graph CoT	Agent		Llama 3.1 8B-Ins	16.83	30.91	20.15	18.43	26.29	14.95	21.41
			Llama 3.1 70B-Ins	33.48	40.98	50.00	51.53	49.6	48.27	44.35
			Llama 3.1 405B-Ins	28.41	36.56	41.35	48.36	47.81	42.54	35.24
	Graph Explore		Llama 3.1 8B-Ins	25.58	32.34	36.65	35.33	31.06	31.05	35.96
			Llama 3.1 70B-Ins	29.41	29.60	44.63	49.49	39.23	38.87	45.52
			Llama 3.1 405B-Ins	28.45	43.06	36.93	38.71	47.49	55.66	32.73
Graph ToT	Agent	Score	Llama 3.1 8B-Ins	28.91	52.25	43.81	44.18	43.49	36.07	39.56
			Llama 3.1 70B-Ins	38.51	51.58	64.44	61.13	55.19	63.00	55.33
			Llama 3.1 405B-Ins	**47.51**	50.73	**70.34**	64.9	49.02	**65.40**	44.63
		Select	Llama 3.1 8B-Ins	28.67	50.59	42.33	37.07	40.81	33.17	36.50
			Llama 3.1 70B-Ins	40.26	**52.59**	64.53	66.84	61.42	61.21	55.89
			Llama 3.1 405B-Ins	46.90	51.68	70.27	**67.95**	**63.74**	64.23	**59.56**
	Graph Explore	Score	Llama 3.1 8B-Ins	24.49	36.80	35.81	36.41	34.28	34.49	37.69
			Llama 3.1 70B-Ins	32.79	38.19	53.83	58.25	48.55	52.18	48.07
			Llama 3.1 405B-Ins	33.90	42.68	46.87	57.43	50.46	55.56	48.73
		Select	Llama 3.1 8B-Ins	25.04	37.8	36.34	38.5	32.44	33.31	34.85
			Llama 3.1 70B-Ins	33.40	39.13	54.78	58.53	47.19	51.13	47.51
			Llama 3.1 405B-Ins	33.82	43.63	44.47	59.06	48.52	55.62	46.07

Table 1: Rouge-L (R-L) performance results on GRBench, comparing standard LLMs, Text-RAG, Graph-RAG, Graph-CoT, and Graph-ToT. Experiments are described in Section 5, using LLama 3.1 - Instruct backbone models with sizes 8B, 70B, and 405B-FP8.

Agent vs Graph Search In our experimental results, the agentic method outperformed graph exploration approach across most datasets and reasoning strategies. The agent-based method, which involves LLM selecting specific actions to interact with KG, consistently improves performance as the number of reasoning steps increases, as shown in Section 7. This highlights that explicit, model-driven interventions are more effective than passive expansion strategies, as they promote iterative refinement and selective focus. While graph exploration can quickly provide broad coverage, the agentic method's targeted, stepwise interactions yield more accurate and comprehensive answers over longer sequences of reasoning.

Tree of Thought (ToT) The ToT reasoning strategy showed superior performance across its various interaction methods and state evaluators, as summarized in Table 1. ToT achieved performance improvements of 54.74% in agent performance and 11.74% in exploration mode compared to the CoT baseline. However, this improvement comes with the trade-off of increased inference time, highlighting the cost of inference-time reasoning interventions. The success of ToT illustrates how branching interventions that explore multiple candidate paths can substantially enhance reasoning accuracy, especially when coupled with evaluators that prune unpromising trajectories. We also compared the two State Evaluation methods (Selection and Score), finding complementary benefits depending on dataset and scale.

Graph of Thought (GoT) The results for GoT strategy are summarized in Table 2. Due to the additional computational cost, we report results for two datasets only. GoT did not outperform ToT. Our initial hypothesis was that LLMs could integrate divergent results from multiple branches, but in practice the models struggled to merge these effectively. Specifically, in the graph exploration setting, models often failed to combine different triples found in separate branches. This reveals a current limitation of reasoning interventions based on aggregation: while branching helps discover diverse facts, robust mechanisms for synthesis and reconciliation are still underdeveloped. This finding opens a direction for future research into more advanced intervention strategies for merging partial reasoning outcomes.

Method		Model	Healthcare	Biology
Agent	Score	Llama 3.1 8B-Ins	29.11	33.25
		Llama 3.1 70B-Ins	30.88	56.64
		Llama 3.1 405B-Ins	43.53	48.1
	Select	Llama 3.1 8B-Ins	29.05	40.37
		Llama 3.1 70B-Ins	40.74	65.59
		Llama 3.1 405B-Ins	**47.63**	**71.49**
Graph Explore	Score	Llama 3.1 8B-Ins	24.96	21.72
		Llama 3.1 70B-Ins	31.24	50.70
		Llama 3.1 405B-Ins	35.00	39.10
	Select	Llama 3.1 8B-Ins	25.06	21.84
		Llama 3.1 70B-Ins	36.95	52.32
		Llama 3.1 405B-Ins	33.74	54.64

Table 2: Graph-GoT results on GRBench using Rouge-L

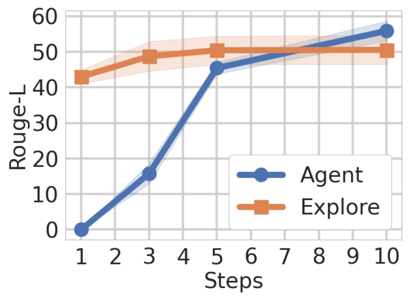


Figure 4: Effect of the number of steps in the LLM-KG Interaction Methods. The Agent requires more steps to obtain the performance of the Graph Exploration, while the Graph Exploration only needs the anchor entities to perform the search within the graph.

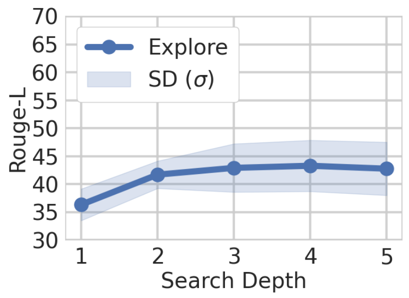


Figure 5: Effect of the Search depth in Graph Exploration interaction method for a fixed steps number. The method can achieve relatively good performance with the anchor entities extracted from the question.

7 Analysis & Ablation studies

In this section, we want to better understand the nuances of our methods for LLM and KG grounding. We conduct an analysis on the Academic datasets from the benchmark, as they all contain the same number of samples and feature questions generated from similar templates to ensure a controlled comparison.

How does the number of steps affect the results? We observe in Figure 4 the effect of varying the number of steps in the KG interaction methods (Agent, Explore) across all academic datasets. The plots indicate that graph exploration performs better with fewer steps, as it automatically traverses the graph for the identified anchor entities. Conversely, the agentic methods improve as the number of steps increases, eventually achieving better performance. This validates our framework's design choice to support both exploration and agentic strategies, each excels in complementary regimes.

What is the effect of Search Depth in Automatic Graph Exploration? We observe the effect of search depth in Figure 5, which presents performance results across various depths, with fixed step size of one. The results demonstrate that the performance of depth-first search plateaus at depth of 3, highlighting the relevance of search exploration with respect to the given query. Beyond this point, deeper traversal yields no significant gains, likely due to diminishing relevance of distant nodes. This shows why shallow, targeted exploration is sufficient in our framework, keeping search efficient without sacrificing accuracy.

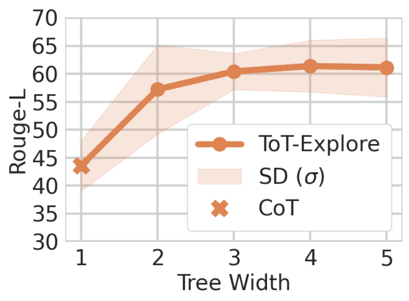


Figure 6: Impact of tree width on Agentic ToT performance. It shows a general trend of performance improvement with increasing tree width.

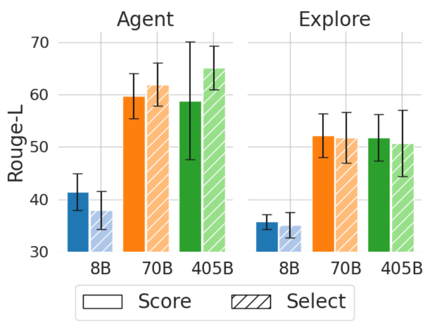


Figure 7: Influence of the State Evaluators in ToT. The Select method obtains better results over Score method.

What is the effect of tree width in the reasoning strategy (ToT)? Based on experimental results across all academic datasets, we observe performance variations among different methods. To gain further insight, we observe in Figure 6 the effect of tree width on results. We notice a slight upward trend in performance as the tree width increases, although the difference is more pronounced between CoT and ToT itself, going from one branch to two. The added computational time and resources likely contribute to this performance enhancement.

What is the influence of the state evaluator? We observe in Figure 7 the impact of state evaluators, specifically Score and Select, within the ToT framework. The analysis indicates that, while there is no significant difference between the two methods, the Select evaluator generally yields slightly better results. This trend is especially evident in the context of the Agent's performance, though the advantage is less pronounced in automatic graph exploration.

How are errors different for each strategy? To understand failure patterns, we define three error types: (1) Reached limit — the reasoning hit the step limit; (2) Answer found but not returned — the correct answer appeared but was not output; (3) Wrong reasoning step — the model followed an illogical step. Using GPT-4o, we labeled a larger set of answers and traces. We observe in Figure 8 that ToT and GoT show more "answer found but not returned" cases than CoT, suggesting better retrieval but occasional failures in synthesis. This comes with a slight rise in logical errors, likely due to the complexity of multiple reasoning paths.

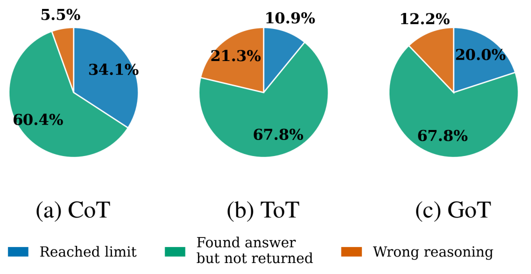


Figure 8: Error distribution across strategies. ToT and GoT reduce unanswered cases but increase logical errors due to more complex reasoning.

8 Limitations

In this work, we demonstrate how LLMs can be used to explore a graph while conditioning the next steps based on the graph's results. We show that the two approaches presented achieve superior results in graph exploration. Integrating KGs with LLMs can provide complex relational knowledge for LLMs to leverage. However, the overall effectiveness depends heavily on both the coverage and quality of the underlying graph, as well as the capabilities of the language model.

Extending inference-time reasoning methods for LLMs is significantly constrained by computational resources and the time available to the user. We analyze the computational complexity of the methods in Appendix E, where we show the exponential growth of ToT and GoT due to its branching structure. GoT further compounds this by allowing merges between reasoning paths, which increases the total number of evaluations. Additionally, loading large graphs into memory introduces substantial RAM overhead, limiting applicability to resource-rich environments.

While LLMs conditioned on external knowledge can generate outputs based on accessed content, their generated output is not strictly limited to that information. Thus, they may still hallucinate. Our framework helps mitigate this risk by grounding reasoning in explicit graph structure, but does not eliminate it.

9 Conclusion

We present a framework for grounding LLM reasoning in KGs by integrating each step with structured graph retrieval. By combining strategies like CoT, ToT, and GoT with adaptive graph search, our method achieves state-of-the-art performance on GRBench. Beyond performance, we find that explicitly linking reasoning steps to graph structure offers a more interpretable view of how LLMs navigate knowledge. The approach enables inference-time reasoning, offering flexibility across domains, and suggests a path toward reasoning interventions that are both systematic and transparent. Future work includes extending this framework to larger and more heterogeneous graphs, and exploring how structured retrieval can guide reasoning in domains where accuracy and verifiability are critical.

References

Ziwei Chai, Tianjie Zhang, Liang Wu, Kaiqiao Han, Xiaohai Hu, Xuanwen Huang, and Yang Yang. Graphllm: Boosting graph reasoning ability of large language model. *ArXiv preprint*, abs/2310.05845, 2023. URL `https://arxiv.org/abs/2310.05845`.

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

Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. Talk like a graph: Encoding graphs for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=IuXR1CCrSi`.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *ArXiv preprint*, abs/2312.10997, 2023. URL `https://arxiv.org/abs/2312.10997`.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 132876–132907. Curran Associates, Inc., 2024. URL `https://proceedings.neurips.cc/paper_files/paper/2024/file/efaf1c9726648c8ba363a5c927440529-Paper-Conference.pdf`.

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

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

Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive nlp. *ArXiv preprint*, abs/2212.14024, 2022. URL `https://arxiv.org/abs/2212.14024`.

Jiho Kim, Yeonsu Kwon, Yohan Jo, and Edward Choi. KG-GPT: A general framework for reasoning on knowledge graphs using large language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9410–9421, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.631. URL `https://aclanthology.org/2023.findings-emnlp.631/`.

Miyoung Ko, Sue Hyun Park, Joonsuk Park, and Minjoon Seo. Investigating how large language models leverage internal knowledge to perform complex reasoning. *ArXiv preprint*, abs/2406.19502, 2024. URL `https://arxiv.org/abs/2406.19502`.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html`.

Feiyang Li, Peng Fang, Zhan Shi, Arijit Khan, Fang Wang, Dan Feng, Weihao Wang, Xin Zhang, and Yongjian Cui. Cot-rag: Integrating chain of thought and retrieval-augmented generation to enhance reasoning in large language models. *arXiv preprint arXiv:2504.13534*, 2025.

Tianle Li, Xueguang Ma, Alex Zhuang, Yu Gu, Yu Su, and Wenhu Chen. Few-shot in-context learning on knowledge base question answering. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6966–6980, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.385. URL `https://aclanthology.org/2023.acl-long.385/`.

Zhuoqun Li, Xuanang Chen, Haiyang Yu, Hongyu Lin, Yaojie Lu, Qiaoyu Tang, Fei Huang, Xianpei Han, Le Sun, and Yongbin Li. Structrag: Boosting knowledge intensive reasoning of llms via inference-time hybrid information structurization. *ArXiv preprint*, abs/2410.08815, 2024. URL `https://arxiv.org/abs/2410.08815`.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL `https://aclanthology.org/W04-1013/`.

Haoran Luo, Zichen Tang, Shiyao Peng, Yikai Guo, Wentai Zhang, Chenghao Ma, Guanting Dong, Meina Song, Wei Lin, et al. Chatkbqa: A generate-then-retrieve framework for knowledge base question answering with fine-tuned large language models. *ArXiv preprint*, abs/2310.08975, 2023. URL `https://arxiv.org/abs/2310.08975`.

Haoyan Luo and Lucia Specia. From understanding to utilization: A survey on explainability for large language models. *ArXiv preprint*, abs/2401.12874, 2024. URL `https://arxiv.org/abs/2401.12874`.

Linhao Luo, Yuan-Fang Li, Reza Haf, and Shirui Pan. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=ZGNWW7xZ6Q`.

Elan Markowitz, Anil Ramakrishna, Jwala Dhamala, Ninareh Mehrabi, Charith Peris, Rahul Gupta, Kai-Wei Chang, and Aram Galstyan. Tree-of-traversals: A zero-shot reasoning algorithm for augmenting black-box language models with knowledge graphs. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12302–12319, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.665. URL `https://aclanthology.org/2024.acl-long.665/`.

Costas Mavromatis and George Karypis. Gnn-rag: Graph neural retrieval for large language model reasoning. *ArXiv preprint*, abs/2405.20139, 2024. URL `https://arxiv.org/abs/2405.20139`.

Fedor Moiseev, Zhe Dong, Enrique Alfonseca, and Martin Jaggi. SKILL: Structured knowledge infusion for large language models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1581–1588, Seattle, United States, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.113. URL `https://aclanthology.org/2022.naacl-main.113`.

Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. Graph retrieval-augmented generation: A survey. *ArXiv preprint*, abs/2408.08921, 2024. URL `https://arxiv.org/abs/2408.08921`.

Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. Evaluating the text-to-sql capabilities of large language models. *ArXiv preprint*, abs/2204.00498, 2022. URL `https://arxiv.org/abs/2204.00498`.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=nnVO1PvbTv`.

SM Tonmoy, SM Zaman, Vinija Jain, Anku Rani, Vipula Rawte, Aman Chadha, and Amitava Das. A comprehensive survey of hallucination mitigation techniques in large language models. *ArXiv preprint*, abs/2401.01313, 2024. URL `https://arxiv.org/abs/2401.01313`.

Pablo Villalobos, Anson Ho, Jaime Sevilla, Tamay Besiroglu, Lennart Heim, and Marius Hobbhahn. Will we run out of data? limits of llm scaling based on human-generated data. *ArXiv preprint*, abs/2211.04325, 2022. URL `https://arxiv.org/abs/2211.04325`.

Xinyi Wang, Alfonso Amayuelas, Kexun Zhang, Liangming Pan, Wenhu Chen, and William Yang Wang. Understanding reasoning ability of language models from the perspective of reasoning paths aggregation. In *Forty-first International Conference on Machine Learning*, 2024. URL `https://openreview.net/forum?id=dZsEOFUDew`.

Yilin Wen, Zifeng Wang, and Jimeng Sun. MindMap: Knowledge graph prompting sparks graph of thoughts in large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10370–10388, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.558. URL `https://aclanthology.org/2024.acl-long.558/`.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *CoRR*, abs/2210.03629, 2022. URL `https://doi.org/10.48550/arXiv.2210.03629`.

Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Wang, Zhiguo Wang, and Bing Xiang. Decaf: Joint decoding of answers and logical forms for question answering over knowledge bases. *ArXiv preprint*, abs/2210.00063, 2022. URL `https://arxiv.org/abs/2210.00063`.

Danna Zheng, Mirella Lapata, and Jeff Z Pan. Large language models as reliable knowledge bases? *ArXiv preprint*, abs/2407.13578, 2024. URL `https://arxiv.org/abs/2407.13578`.

Alex Zhuang, Ge Zhang, Tianyu Zheng, Xinrun Du, Junjie Wang, Weiming Ren, Stephen W Huang, Jie Fu, Xiang Yue, and Wenhu Chen. Structlm: Towards building generalist models for structured knowledge grounding. *ArXiv preprint*, abs/2402.16671, 2024. URL `https://arxiv.org/abs/2402.16671`.

Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. Toolqa: a dataset for llm question answering with external tools. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, 2023.

A GRBench Statistics

Detailed statistics of the graphs in GRBench Jin et al. [2024] are shown in Table 3. **Academic Graphs** contain 3 types of nodes: paper, author, venue. **Literature Graphs** contain 4 types of nodes: book, author, publisher and series. **Healthcare Graph** contains 11 types of nodes: anatomy, biological process, cellular component, compound, disease, gene, molecular function, pathway, pharmacologic class, side effect, and symptom. Questions are created according to multiple templates labeled as easy, medium, and hard, depending on the number of nodes required to give the answer.

Domain	Topic	Graph Statistics		Data	
		# Nodes	# Edges	# Templates	# Questions
Academic	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
Literature	Goodreads	~3M	~22M	24	240
Healthcare	Disease	~47K	~4M	27	270
SUM	-	-	-	**121**	**1210**

Table 3: Detailed statistics of the GRBench Jin et al. [2024].

B LLM <–> KG Interaction Pipelines

Description of the two LLM + KG Interaction Pipelines in their CoT form:

1. Agent – Figure 9A pipeline where the LLM alternates between generating a reasoning step, selecting an explicit action (e.g., retrieving a node, checking neighbors), and observing results from the KG until termination.
2. Automatic Graph Exploration – Figure 10. A pipeline where entities are automatically extracted from the LLM's generated text and used to guide iterative graph traversal with pruning, progressively expanding the reasoning chain.

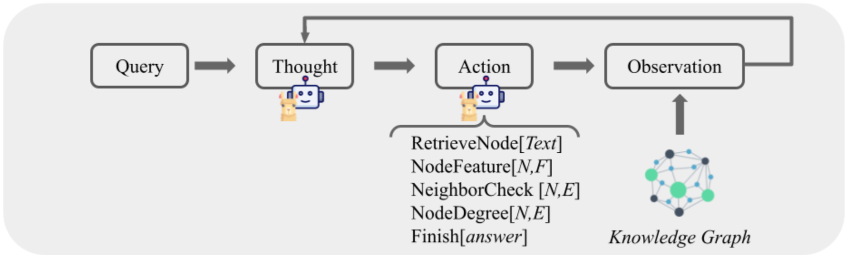


Figure 9: Agent Pipeline: (1) Input Query, (2) Thought Generation (3) Action Selection, (4) Environment Observation from the Knowledge Graph. The process is repeated until termination action is generated or limit reached.

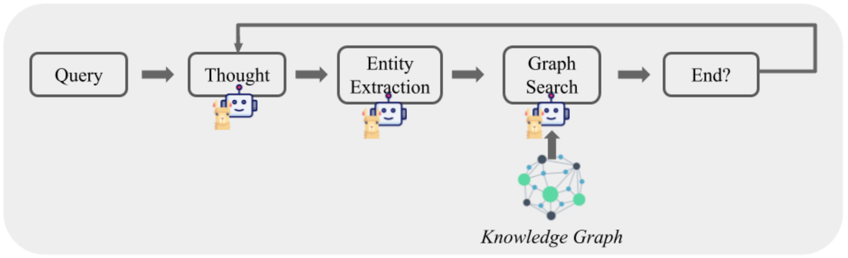


Figure 10: Automatic Graph Exploration Pipeline: (1) Input Query, (2) Thought Generation, (3) Entity Extraction (from query or thought with LLM), (4) Automatic Graph Search as described in Algorithm 1 (5) Query LLM for answer or continue

Algorithm 1 presents the pseudocode for the Automatic Graph Exploration described in Section 4.2.2.

Algorithm 1 Graph Exploration Algorithm

```
1: procedure GraphExplore($LLM, seen\_entities, search\_depth$)
2:     $relevant\_attributes, found\_triples \leftarrow 0$
3:     for $depth$ in $search\_depth$ do
4:         for $entity$ in $seen\_entities$ do
5:             if $seen\_entities[entity\_id].visited$ == True then
6:                 Continue
7:             else
8:                 $seen\_entities[entity] \leftarrow$ Visited
9:             end if
10:            $head\_entity\_name, entity\_attributes, neighbors \leftarrow$ Graph[$entity$]
11:            $pruned\_neighbors \leftarrow$ prune_relations($LLM, neighbors$)
12:            $pruned\_neighbors \leftarrow$ prune_entities($LLM, pruned\_neighbors$)
13:            $found\_triples \leftarrow$ generate_triples($entity, pruned\_neighbros$)
14:        end for
15:        $seen\_entities \leftarrow$ Update($seen\_entities$, neighbors)
16:        if End?($LLM, found\_triples, relevant\_attributes$) == True then
17:            break
18:        end if
19:    end for
20:    return $found\_triples, relevant\_attributes, seen\_entities$
21: end procedure
```

C Performance results in plots

Figures 11 and 12 illustrate the performance results using the Rouge-L and GPT4score metrics, respectively, for the healthcare graph for all methods. The results were run on the LLama 3.1 Instruct models (8B, 70B, and 405B-FP8) and demonstrate the improved performance achieved through more complex reasoning and search strategies during inference.

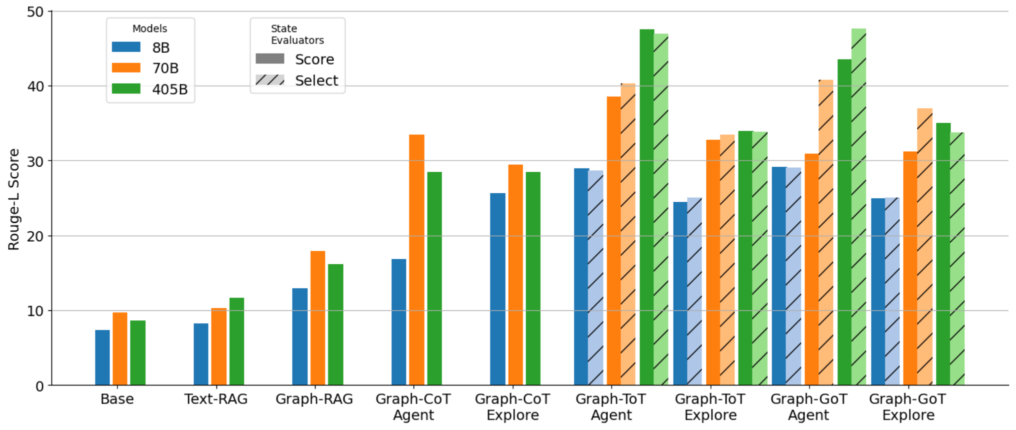


Figure 11: Performance results using the Rouge-L metric on the healthcare graph of GRBench Jin et al. [2024], comparing all methods with LLama 3.1 Instruct models of various sizes (8B, 70B, 405B-FP8). Experimental details are included in Section 5.

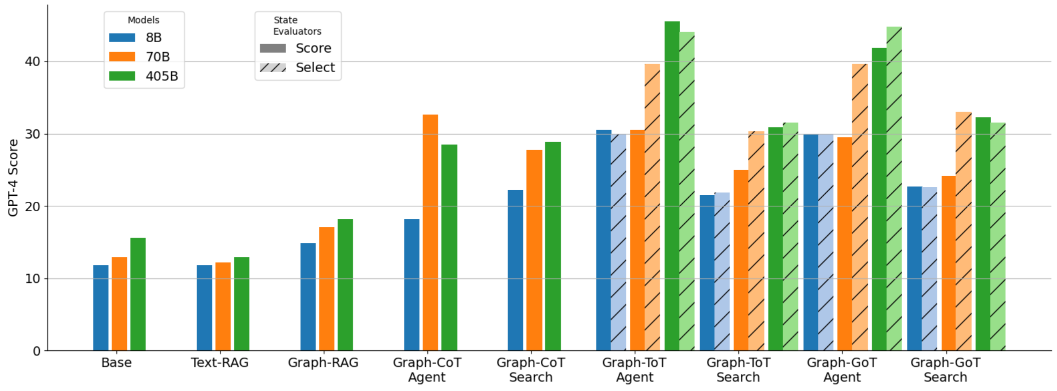


Figure 12: Performance results using the GPT4Score metric on the healthcare graph of GRBench Jin et al. [2024], comparing all methods with LLama 3.1 Instruct models of various sizes (8B, 70B, 405B-FP8). Experimental details are included in Section 5.

D Results on GPT4Score

In this section, we present the results of the experiments described in Section 5 for all methods, using the GPT4Score metric. This metric calculates the percentage of "correct" answers as judged by GPT-4 when presented with both the correct and the generated answer. The tables in this section present the same data as in Tables 1 and 2, but evaluated using GPT4Score.

	Method		Model	Healthcare	Goodreads	Biology	Chemistry	Materials Science	Medicine	Physics
Baselines	Base		Llama 3.1 8B-Ins	11.85	13.33	10.71	11.43	7.86	7.87	9.29
			Llama 3.1 70B-Ins	12.96	19.17	10.00	12.14	11.43	11.43	12.86
			Llama 3.1 405B-Ins	15.55	26.67	12.86	12.14	12.14	13.57	12.14
	Text-RAG		Llama 3.1 8B-Ins	11.85	21.67	12.86	10.00	10.00	8.57	7.86
			Llama 3.1 70B-Ins	12.22	27.5	12.14	13.57	13.57	13.57	12.86
			Llama 3.1 405B-Ins	12.96	26.67	15.00	13.57	12.86	14.29	13.57
	Graph-RAG		Llama 3.1 8B-Ins	14.81	32.50	29.29	29.28	27.86	25.71	29.29
			Llama 3.1 70B-Ins	17.04	32.92	39.29	40.71	43.57	34.29	40.00
			Llama 3.1 405B-Ins	18.15	31.67	37.14	42.86	40.00	36.43	41.43
Graph CoT	Agent		Llama 3.1 8B-Ins	18.15	32.5	20.71	19.28	25.00	14.29	21.43
			Llama 3.1 70B-Ins	32.59	43.75	50.00	51.43	50.00	48.57	46.43
			Llama 3.1 405B-Ins	28.89	48.33	38.57	38.57	47.86	56.43	34.29
	Graph Explore		Llama 3.1 8B-Ins	22.22	36.67	35.00	30.71	29.29	29.29	32.86
			Llama 3.1 70B-Ins	27.78	32.92	45.71	49.29	40.00	40.00	44.29
			Llama 3.1 405B-Ins	28.89	48.33	38.57	38.57	47.86	56.43	34.29
Graph ToT	Agent	Score	Llama 3.1 8B-Ins	30.49	55.14	43.33	41.67	44.05	36.43	39.52
			Llama 3.1 70B-Ins	30.49	54.48	65.48	62.14	55.95	63.57	56.19
			Llama 3.1 405B-Ins	**45.55**	56.53	**71.67**	65.71	52.62	68.81	44.76
		Select	Llama 3.1 8B-Ins	30.00	54.17	40.71	37.14	40.00	32.86	36.43
			Llama 3.1 70B-Ins	39.63	56.67	65.00	67.14	62.86	60.71	55.55
			Llama 3.1 405B-Ins	44.07	**58.75**	71.43	**69.29**	**65.00**	**68.81**	**60.00**
	Graph Explore	Score	Llama 3.1 8B-Ins	21.48	41.10	32.86	31.67	31.43	32.14	35.24
			Llama 3.1 70B-Ins	24.94	40.97	52.38	57.86	49.29	54.29	47.86
			Llama 3.1 405B-Ins	30.86	48.33	47.86	57.14	50.71	56.67	47.14
		Select	Llama 3.1 8B-Ins	21.85	41.67	32.86	31.67	31.43	32.14	35.24
			Llama 3.1 70B-Ins	30.37	42.08	54.29	57.14	47.86	52.14	46.43
			Llama 3.1 405B-Ins	31.48	48.75	45.00	57.86	48.86	57.14	45.71

Table 4: GPT4Score performance results on GRBench Jin et al. [2024], comparing standard LLMs, Text-RAG, Graph-RAG, Graph-CoT, and Graph-ToT. Experiments are described in Section 5, using LLama 3.1 - Instruct backbone models with sizes 8B, 70B, and 405B.

Method		Model	Healthcare	Biology
Agent	Score	Llama 3.1 8B-Ins	29.88	32.86
		Llama 3.1 70B-Ins	29.51	61.69
		Llama 3.1 405B-Ins	41.81	48.33
	Select	Llama 3.1 8B-Ins	30.00	40.71
		Llama 3.1 70B-Ins	39.63	69.83
		Llama 3.1 405B-Ins	**44.81**	**72.86**
Graph Explore	Score	Llama 3.1 8B-Ins	22.72	21.19
		Llama 3.1 70B-Ins	24.20	48.57
		Llama 3.1 405B-Ins	32.22	41.67
	Select	Llama 3.1 8B-Ins	22.59	19.28
		Llama 3.1 70B-Ins	32.96	52.86
		Llama 3.1 405B-Ins	31.48	57.86

Figure 13: Graph-GoT results (GPT4Score) on GRBench with Llama 3.1 Instruct sizes 8B, 70B, and 405B.

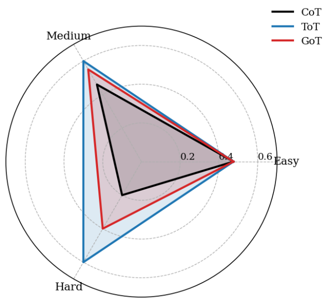


Figure 14: Results decomposed into easy/medium/hard questions according to GPT4Score.

E Computational Analysis

	Method	Key Parameters	Approx. # LLM Calls	Approx. # KG Operations	Primary Growth Driver(s)
CoT	Agent	n	$\mathcal{O}(n)$	$\mathcal{O}(n)$	n (linear)
	Graph Explore	n, d	$\mathcal{O}(n)$	$\mathcal{O}(n \cdot \text{Cost}_{\text{Explore}}(d))$	n, d
ToT	Agent	$D_{\max}, k, t$	$\mathcal{O}\left(k \cdot \frac{t^{D_{\max}}-1}{t-1}\right)$	Same as LLM Calls	$D_{\max}, k, t$ (exponential in $D_{\max}$)
	Graph Explore	$D_{\max}, k, t, d$	$\mathcal{O}\left(k \cdot \frac{t^{D_{\max}}-1}{t-1}\right)$	$\mathcal{O}\left(k \cdot \frac{t^{D_{\max}}-1}{t-1} \cdot \text{Cost}_{\text{Explore}}(d)\right)$	$D_{\max}, k, t, d$
GoT	Agent	$D_{\max}, k, t$	. $\mathcal{O}\left(k \cdot \frac{t^{D_{\max}}-1}{t-1} + \sum_{i=1}^{D_{\max}} \left\lfloor \frac{k \cdot t^i}{2} \right\rfloor\right)$	Same as LLM Calls	$D_{\max}, k, t$ (aggregation adds extra cost)
	Graph Explore	$D_{\max}, k, t, d$	Same as Agent	$\mathcal{O}$ (LLM Calls $\cdot \text{Cost}_{\text{Explore}}(d)$)	$D_{\max}, k, t, d$

Table 5: Theoretical computational complexity comparison of reasoning methods. Parameters: n (reasoning steps), $D_{\max}$ (tree depth), k (branching factor), t (paths retained), d (KG search depth), and $\text{Cost}_{\text{Explore}}(d)$ (cost per KG search). GoT includes pairwise aggregation of thoughts at each depth.

The computational analysis summarized in Table 5 highlights the trade-offs between reasoning power and computational cost when grounding LLMs with Knowledge Graphs (KGs). The *Agent*-based methods scale linearly with the number of reasoning steps or tree nodes, with CoT representing the lowest cost baseline. In contrast, *Automatic Graph Exploration* methods introduce additional overhead via entity extraction, multi-hop traversal up to a fixed `max_depth`, and LLM-based pruning at each step.

Among reasoning strategies, ToT introduces exponential growth in cost with respect to depth due to its exploration of k branches and selection of t continuations per level. GoT further amplifies this by incorporating aggregation transformations that attempt to merge every pair of thoughts at each depth, leading to an additional cost proportional to $\sum_{i=1}^{D_{\max}} \left\lfloor \frac{k \cdot t^i}{2} \right\rfloor$.

Importantly, our experiments reveal that the higher complexity of GoT does not consistently translate to improved accuracy compared to ToT, suggesting diminishing returns. While the model size (e.g., 8B, 70B, 405B) influences the latency and memory footprint of each LLM call, it does not affect the algorithmic complexity classes shown. Hence, selecting a strategy requires balancing reasoning depth with feasible compute budgets.

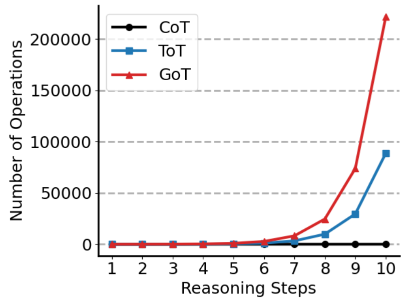


Figure 15: Comparison of computational costs across different reasoning strategies — CoT, ToT, GoT — as a function of reasoning steps. GoT exhibits the highest cost due to merge operations and graph traversal.

F Prompts Archive

In this section, we gather the necessary prompts essential for implementing the proposed methodologies.

Agent

```
 Agent Step:  Solve a question answering task with interleaving
Thought, Interaction with Graph, Feedback from Graph steps.  In
Thought step, you can think about what further information is needed,
and In Interaction step, you can get feedback from graphs with four
functions:
(1) RetrieveNode[keyword], which retrieves the related node from the
graph according to the corresponding query.
(2) NodeFeature[Node, feature], which returns the detailed attribute
information of Node regarding the given "feature" key.  (3)
NodeDegree[Node, neighbor_type], which calculates the number of
"neighbor_type" neighbors of the node Node in the graph.
(4) NeighbourCheck[Node, neighbor_type], which lists the
"neighbor_type" neighbours of the node Node in the graph and returns
them.
You may take as many steps as necessary.
Here are some examples:
{examples}
Please answer by providing node main feature (e.g., names) rather
than node IDs.
Generate the next step.
Definition of the graph:  {graph_definition}
Question:  {question}
{scratchpad}
```

Automatic Graph Exploration

```
Search Thought:  Given the previous thoughts, generate the next
thought to answer the provided question.
Your end goal is to answer the question step by step.  For context,
you are also provided with some knowledge triples from a knowledge
base.
Follow the format of the examples to generate the next thought.

{examples}

Graph Definition:  {graph_definition}
Question:  {question}
Knowledge Triples:
{triples}
Previous thoughts:
{thoughts}
Related Entity Attributes:
{attributes}
Next Thought:

Search End?:  Your are provided with the an original question, the
associated subquestion thoughts and their corresponding knowledge
graph triples (head_entity -> relation -> tail_entity).  Your task
is to answer whether it's sufficient for you to answer the original
question (Yes or No).  You are provided with examples.  You should
follow the same format as in the examples, writing 'Yes' or 'No'
```

```
within brackets at the beginning of the answer.
(Examples)
Task: Question: {question}
Thoughts: {thoughts}
Knowledge Triples: {triples}
Entity Attributes: {attributes}
Answer:

Entity Extraction: Given the provided text, extract the relevant
entities that may appear in a knowledge base. Return the answer at
the end with brackets relevant entities as shown in the following
examples. If there are several entities, separate them with commas.
(Examples)
Task: Text: {text}
Relevant Entities:

Prune Relations: From the given entity and relations, select only
the relevant relations to answer the question. Provide the answer at
the end with bracketsanswer , as shown in the following example.
(Examples)
Question: {question}
Head Entity: {entity}
Relations: {relations} Answer:

Prune entities: You are provided with a question, a head entity,
a relation and tail entity or entities from a knowledge base. Select
the tail entity or entities to answer the question. Return the
tail entity or entities at the end with brackets relevant entity or
entities, as shown in the following examples.
(Examples)
Question: {question}
Head Entity: {head_entity}
Relation: {relation}
Tail Entities: {tail_entities}
Relevant Entities:

Search Attributes: Is any of the attributes relevant to answer the
question? Return the answer at the end with brackets answer , as
shown in the following examples.
(Examples)
Question: {question}
Entity: {entity}
Attributes: {attributes}
Relevant Attributes:
```

State Evaluators

Selection Vote: Given a question, you need to select the possible chain of thought that may lead to the correct answer with higher probablity. You are provided with several choices with thouhgts and related triples from a knowledge base. Decide which choice is most promising to complete the task. Analyze each choice in detail, then conclude in the last line: "The best choice is s" , where s the integer id of the choice.
(*Examples*)
Question: {*question*}
Choices: {*choices*} (+ attributes/triples)
Answer:

Score Vote: Generate a score for the given reasoning chain. The score represents the probability that the chain will lead to the correct answer. The chains contain interleaved thoughts and related triples from a knowledge base. Some chains may not be complete, but you need to judge the steps that are provided. The score can be any floating number between 0 and 1.
(*Examples*)
Question: {*question*}
Thought Chain: {*thoughts*} (+ attributes/triples)
Score:

Graph of Thoughts

Graph of Thought Merge: Generate the next thought for the merged chain of thoughts. You are provided with the question, two chains of thoughts, and the corresponding merged chain of thought. Identify inconsistencies or errors from the previous chains and provide the next thought for the merged chain. You should follow the same format as in the examples.
(*Examples*)
Question: {*question*}
Chain 1: {*chain_1*} (+triples/attributes)
Chain 2: {*chain_2*} (+triples/attributes)
Merged Chain: {*merged_chain*}
Next Thought:

G Examples

Example 1: Multi-step Reasoning over Graph
Question: *What anatomy can be expressed by gene KRT39?*
Ground Truth Answer: *head, skin of body*
Model Answer: *head, skin of body*

Thought 1: The question is related to a gene node (KRT39). We need to find this node in the graph.
Action 1: `RetrieveNode[KRT39]`
Observation 1: The ID of the node is `390792`.

Thought 2: We need to check the 'Anatomy-expresses-Gene' neighbors of this gene node.
Action 2: `NeighbourCheck[390792, Anatomy-expresses-Gene]`
Observation 2: The neighbors are `['UBERON:0000033', 'UBERON:0002097']`.

Thought 3: Retrieve names of the anatomy nodes.
Action 3: `NodeFeature[UBERON:0000033, name]`, `NodeFeature[UBERON:0002097, name]`
Observation 3: `UBERON:0000033` $\rightarrow$ *head*, `UBERON:0002097` $\rightarrow$ *skin of body*.

Thought 4: These are the anatomy terms expressed by the gene.
Action 4: `Finish[head, skin of body]`

Example 2: Direct Triple Lookup
Question: *What anatomy can be expressed by gene KRT39?*
Ground Truth Answer: *head, skin of body*
Model Answer: *head, skin of body*

Thought 1: `KRT39` is a gene that is known to be expressed in two anatomical regions.
Found Triples:

- `"KRT39" -> Anatomy-expresses-Gene -> head`
- `"KRT39" -> Anatomy-expresses-Gene -> skin of body`

Action: `Finish[head, skin of body]`

Table 6: Comparison of two LLM-generated reasoning traces for the same question using different strategies: multi-step graph traversal (Example 1) vs. direct triple lookup (Example 2).