PathMind: A Retrieve-Prioritize-Reason Framework for Knowledge Graph Reasoning with Large Language Models

Yu Liu[1, 2], **Xixun Lin**[1, 2*], **Yanmin Shang**[1, 2], **Yangxi Li**[3], **Shi Wang**[4], **Yanan Cao**[1, 2]

[1]Institute of Information Engineering, Chinese Academy of Sciences
[2]School of Cyber Security, University of Chinese Academy of Sciences
[3]Peking University
[4]Institute of Computing Technology, Chinese Academy of Sciences
{liuyu2022, linxixun, shangyanmin, caoyanan}@iie.ac.cn, liyangxi@outlook.com, wangshi@ict.ac.cn

Abstract

Knowledge graph reasoning (KGR) is the task of inferring new knowledge by performing logical deductions on knowledge graphs. Recently, large language models (LLMs) have demonstrated remarkable performance in complex reasoning tasks. Despite promising success, current LLM-based KGR methods still face two critical limitations. First, existing methods often extract reasoning paths indiscriminately, without assessing their different importance, which may introduce irrelevant noise that misleads LLMs. Second, while some methods leverage LLMs to dynamically explore potential reasoning paths, they require high retrieval demands and frequent LLM calls. To address these limitations, we propose PathMind, a novel framework designed to enhance faithful and interpretable reasoning by selectively guiding LLMs with important reasoning paths. Specifically, PathMind follows a "Retrieve-Prioritize-Reason" paradigm. First, it retrieves a query subgraph from KG through the retrieval module. Next, it introduces a path prioritization mechanism that identifies important reasoning paths using a semantic-aware path priority function, which simultaneously considers the accumulative cost and the estimated future cost for reaching the target. Finally, PathMind generates accurate and logically consistent responses via a dual-phase training strategy, including task-specific instruction tuning and path-wise preference alignment. Extensive experiments on benchmark datasets demonstrate that PathMind consistently outperforms competitive baselines, particularly on complex reasoning tasks with fewer input tokens, by identifying essential reasoning paths.

Introduction

Reasoning, the ability to derive logical conclusions from available knowledge, has been a long-standing goal in the pursuit of artificial intelligence (Halpern 1986). Knowledge graphs (KGs) represent structured relationships between entities and serve as a fundamental foundation for reasoning (Kim et al. 2023; Pan et al. 2024). Knowledge graph reasoning (KGR) aims to infer new knowledge or answer complex queries based on KGs, facilitating various practical applications such as recommendation systems (Zhou et al. 2020), question answering (Liu et al. 2024), and biomedical inference (Jiang et al. 2025). Despite great achievements, KGR remains a challenging task, as real-world KGs are often large-scale and inherently incomplete, making reasoning unreliable (Pan et al. 2024).

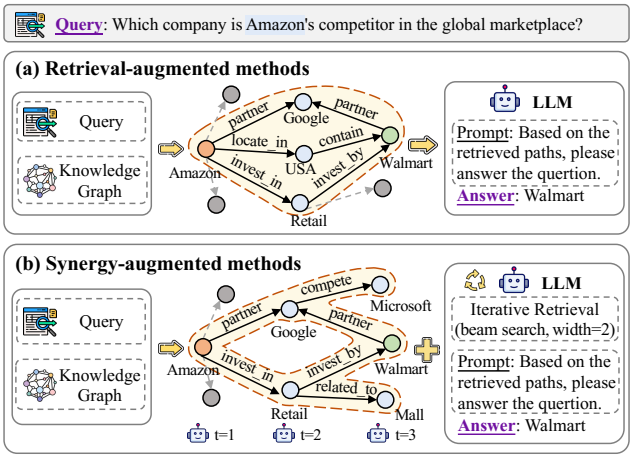

Figure 1: Illustration of LLM-based KGR methods. (a) Retrieval-augmented methods retrieve relevant information from KGs for LLMs. (b) Synergy-augmented methods integrate KGs and LLMs through iterative interaction.

Recently, large language models (LLMs) have demonstrated remarkable performance on complex reasoning tasks (Meyer et al. 2023), representing a significant advancement toward artificial general intelligence. These models leverage advanced pre-training techniques and vast amounts of unlabeled text data to understand and generate human language with impressive fluency and coherence. Consequently, LLM-based KGR has attracted increasing interest, aiming to leverage the powerful generalizability of LLMs to perform knowledge reasoning over KGs (Jiang et al. 2023; Sun et al. 2024; Ao et al. 2025).

Existing LLM-based KGR methods can be broadly divided into two main lines: *retrieval-augmented* and *synergy-augmented* paradigms, as illustrated in Figure 1(a) and (b). Retrieval-augmented methods (Kim et al. 2023; Luo et al. 2024; Long et al. 2025) retrieve query-relevant triples or

*Corresponding author.

multi-hop paths from KGs, which are then verbalized into LLMs to improve reasoning abilities. In contrast, synergy-augmented methods (Sun et al. 2024; Chen et al. 2024) treat LLMs as agents that iteratively interact with KGs to explore possible reasoning paths and generate answers. In general, these two methods effectively leverage the complementary strengths of LLMs and KGs, achieving superior performance compared to previous strong approaches.

Despite their success, existing methods still suffer from several critical limitations. First, retrieval-augmented methods often extract potential reasoning paths indiscriminately, failing to evaluate their importance for answer generation, which may introduce irrelevant or noisy information that misleads LLMs. For instance, when querying *"Which company is Amazon's competitor in the global marketplace?"* in Figure 1, the path $Amazon \xrightarrow{invest\_in} Retail \xrightarrow{invest\_by} Walmart$ clearly indicates a competitive relationship between Amazon and Walmart, whereas less relevant paths like $Amazon \xrightarrow{partner} Google \xleftarrow{partner} Walmart$ might misleadingly imply collaboration. Second, although synergy-augmented methods leverage LLMs to dynamically discover reasoning paths, they often encounter significant computational challenges, such as substantial retrieval demands in large search spaces and high overhead from multiple LLM calls, which severely limit their scalability and practicality in real-world scenarios.

To overcome these limitations, we propose **PathMind**, a novel framework that enhances faithful and interpretable reasoning by selectively guiding LLMs with important reasoning paths. Specifically, PathMind follows a "Retrieve-Prioritize-Reason" paradigm. Given a question, we first retrieve the query subgraph from the KG and encode it into graph representations. Next, we introduce a path prioritization mechanism that identifies important reasoning paths using a semantic-aware path priority function, which simultaneously considers the accumulative cost up to the current node and the estimated future cost to reach the target entity. Finally, we leverage these retrieved paths to guide LLMs via a dual-phase training strategy: task-specific instruction tuning and path-wise preference alignment, enabling LLMs to generate accurate and logically consistent responses without incurring the overhead of multiple LLM calls. Extensive experiments and comprehensive analyses on benchmark datasets show the superiority of PathMind.

In summary, our contributions are as follows:

- We introduce a new framework named PathMind, which leverages important reasoning paths to effectively guide LLMs toward accurate logical reasoning, enhancing both faithfulness and interpretability.
- We propose an effective path prioritization mechanism that simultaneously models the accumulative cost and estimates future cost to identify important reasoning paths, and further improve LLMs via task-specific instruction tuning and path-wise preference alignment.
- Experimental results on widely-used KGR benchmarks demonstrate that PathMind achieves competitive performance compared to strong baselines, particularly on complex reasoning tasks with fewer input tokens.

Related Work

Retrieval-Augmented Methods. These methods typically retrieve factual knowledge from KGs and integrate it into LLMs to improve reasoning abilities. Some methods (Kim et al. 2023; Li, Miao, and Li 2025) retrieve relevant facts by evaluating semantic similarities between the question and associated triples. Other approaches (Luo et al. 2024; Long et al. 2025; Liu et al. 2025b) extract explicit paths to exploit structures within KGs. In particular, RoG (Luo et al. 2024) proposes a planning-retrieval-reasoning framework to generate relational paths. GNN-RAG (Mavromatis and Karypis 2025) retrieves the shortest paths between topic entities and answer candidates. GCR (Luo et al. 2025) generates reliable reasoning paths grounded in KGs. EPERM (Long et al. 2025) and ORT (Liu et al. 2025b) discover more reasoning chains to support inference. Meanwhile, certain approaches (Liu et al. 2025c; Ao et al. 2025) employ graph neural networks (GNNs) to learn graph embeddings for retrieved subgraphs. Overall, these methods depend heavily on the information retrieved from KGs. However, current approaches often treat them equally, failing to distinguish the varying contributions of reasoning paths, which can result in extraneous or noisy information that may mislead LLMs.

Synergy-Augmented Methods. These methods combine LLMs and KGs via an interaction mechanism that iteratively explores KGs to discover potential reasoning paths. Early approaches, such as ChatKBQA (Luo et al. 2023) and FlexKBQA (Li et al. 2024), utilize LLMs to convert natural language questions into structural queries (e.g., SPARQL), which are then executed on KGs. More recently, most works (Sun et al. 2024; Ma et al. 2025) treat LLMs as agents (Lin et al. 2025) that interact with KGs to derive reasoning paths and generate target responses. For example, ToG (Sun et al. 2024) employs LLMs to iteratively perform beam search on KGs; its successor, ToG-2 (Ma et al. 2025), extends this by incorporating external unstructured knowledge. PoG (Chen et al. 2024) proposes a self-correcting adaptive planning paradigm. Additionally, techniques such as KD-CoT (Wang et al. 2023) and SymAgent (Liu et al. 2025a) explore chain-of-thought prompting to elicit deep reasoning of LLMs. Despite their success, these methods still face significant computational challenges, stemming from vast search spaces and multiple LLM API calls, which seriously limit their practical applications.

Preliminary

Knowledge Graphs. KGs represent factual knowledge with structured relationships. Formally, a KG can be defined as $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, where $\mathcal{E}$ and $\mathcal{R}$ are the sets of entities and relations, respectively, and $\mathcal{T} = \{(e, r, e') \mid e, e' \in \mathcal{E}, r \in \mathcal{R}\}$ represents the set of relational triples. Given a query q and a KG $\mathcal{G}$, the KGR task is to design a function f to predict answers $a \in \mathcal{A}_q$ based on knowledge from $\mathcal{G}$, i.e., $a = f(q, \mathcal{G})$. Following previous work (Luo et al. 2024), we assume the topic entity $e_q \in \mathcal{O}_q$ observed in q and the answer $a \in \mathcal{A}_q$ are labeled and linked to the corresponding entities in $\mathcal{G}$, i.e., $\mathcal{O}_q, \mathcal{A}_q \subseteq \mathcal{E}$.

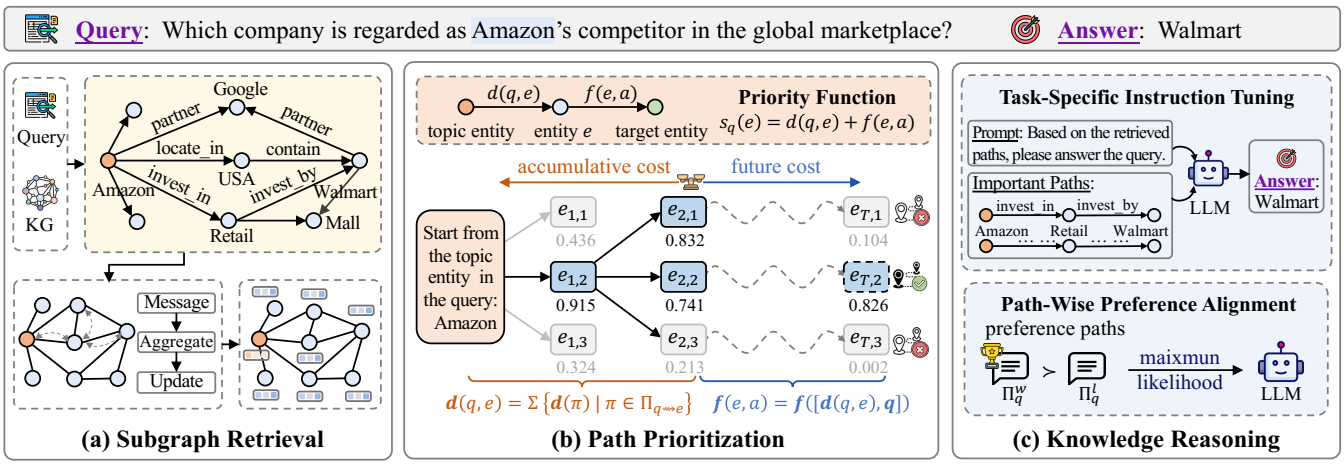

Figure 2: The overall framework of our **PathMind**. (a) Subgraph Retrieval extracts the query subgraph from the KG and encodes it into graph representations. (b) Path Prioritization identifies important reasoning paths according to a path priority function. (c) Knowledge Reasoning enhances LLM reasoning via task-specific instruction tuning and path-wise preference alignment.

Reasoning Paths. Reasoning paths are sequences of consecutive triples in KGs. More concretely, a reasoning path can be defined as $\pi = e_0 \xrightarrow{r_1} e_1 \xrightarrow{r_2} \dots \xrightarrow{r_l} e_l$, where $\forall (e_{i-1}, r_i, e_i) \in \mathcal{T}$. These paths reveal the connections among entities that can support knowledge reasoning. However, in real-world KGs, the number of possible paths is extremely large, and only a small set of them is important for reasoning. Based on this observation, we introduce the concept of *important reasoning paths*—those that are particularly necessary for reasoning. For example, in Figure 1, the path $Amazon \xrightarrow{invest\_in} Retail \xrightarrow{invest\_by} Walmart$ suggests that Amazon and Walmart are in a competitive relationship. In contrast, the path $Amazon \xrightarrow{partner} Google \xleftarrow{partner} Walmart$ might even misleadingly imply collaboration.

Task Formulation. Typically, the KGR task can be formulated as an optimization problem that maximizes the probability of inferring the answer from the KG $\mathcal{G}$ given the query q. By retrieving reasoning paths Π as supporting evidence, we empower LLMs to perform structured reasoning. Formally, the task can be defined as follows:

$$P(a|q,\mathcal{G}) = \sum_{\pi \in \Pi} P_\phi(a|q,\pi) P_\varphi(\pi|q,\mathcal{G}), \tag{1}$$

where $P_\varphi(\pi|q,\mathcal{G})$ denotes the probability of discovering a reasoning path π from the KG $\mathcal{G}$ given the query q using the retriever model parameterized by φ. $P_\phi(a|q,\pi)$ is the probability of generating an answer a using the LLM parameterized by ϕ, given the query q and the reasoning path π.

To acquire reasoning paths for reasoning, previous studies have followed *retrieval-augmented* or *synergy-augmented* paradigm. However, the former typically extracts reasoning paths equally, without evaluating their different importance to answer generation, while the latter is computationally intensive and leads to high costs. To address these issues, we propose a novel method that prioritizes important reasoning paths, enabling more reliable reasoning.

Methodology

In this paper, we introduce PathMind, a new framework that integrates LLMs and KGs to enhance faithful and interpretable reasoning. The overall architecture of PathMind is illustrated in Figure 2. Specifically, our model consists of three key components: (1) Subgraph Retrieval, (2) Path Prioritization, and (3) Knowledge Reasoning. In the following, we explain these technical details.

Subgraph Retrieval Module

Given a query q, the first step is to extract the relevant subgraph $\mathcal{G}_q = (\mathcal{E}_q, \mathcal{R}_q, \mathcal{T}_q)$ from the KG, aiming to reduce the search space while preserving essential information.

Query Subgraph Extraction. For each topic entity $e_q \in \mathcal{O}_q$ in the query q, we first retrieve its k-hop neighborhood $\mathcal{N}_k(e_q)$. We then take the union of these neighborhoods as the set of subgraph nodes, denoted as $\mathcal{E}_q = \bigcup_{e_q \in \mathcal{O}_q} \mathcal{N}_k(e_q)$. Next, we extract the set of edges $\mathcal{R}_q$ from $\mathcal{G}$ that connect nodes within $\mathcal{E}_q$. Using these nodes and edges, we obtain the triple set $\mathcal{T}_q = \{(e, r, e') \mid e, e' \in \mathcal{E}_q, r \in \mathcal{R}_q\}$. Finally, the query subgraph $\mathcal{G}_q = (\mathcal{E}_q, \mathcal{R}_q, \mathcal{T}_q)$ is constructed.

Graph Representation Learning. After constructing the query subgraph $\mathcal{G}_q$, we utilize a graph neural network (GNN) to learn the representations of nodes and relations. The GNN leverages message passing and aggregation mechanisms to capture complex relationships and structural dependencies inherent in $\mathcal{G}_q$. Specifically, at each layer l, the representation of node $e \in \mathcal{E}_q$ is updated as follows:

$$\begin{aligned} \boldsymbol{m}_e^{(l)} &= \mathrm{AGG}^{(l)}\left(\left\{\boldsymbol{W}_r^{(l)} \boldsymbol{h}_{e'}^{(l-1)} \mid (e', r, e) \in \mathcal{T}_q\right\}\right), \\ \boldsymbol{h}_e^{(l)} &= \mathrm{UPDATE}^{(l)}\left(\boldsymbol{h}_e^{(l-1)}, \boldsymbol{m}_e^{(l)}\right), \end{aligned} \tag{2}$$

where $\boldsymbol{h}_e^{(l)}$ is the representation of node e at layer l, and $\boldsymbol{W}_r^{(l)}$ is the learnable relation matrix. The functions $\mathrm{AGG}^{(l)}$

and UPDATE$^{(l)}$ represent aggregation and update operations. Initial representations $\boldsymbol{h}_e^{(0)}$ and $\boldsymbol{W}_r^{(0)}$ are randomly initialized. After L layers of message passing, $\boldsymbol{h}_e^{(L)}$ and $\boldsymbol{W}_r^{(L)}$ encode rich structural information, which will be used for subsequent reasoning tasks. For brevity, we omit the superscripts and denote them as $\boldsymbol{h}_e$ and $\boldsymbol{W}_r$, respectively.

Path Prioritization Module

Based on the query subgraph $\mathcal{G}_q$, we introduce a path prioritization module to identify multi-hop reasoning paths, which serve as logical chains of thought and support explainable reasoning. Unlike existing methods that indiscriminately retrieve all possible reasoning paths, often introducing substantial irrelevant or noisy data, our approach prioritizes more important reasoning paths, enabling more faithful and interpretable reasoning for LLMs.

Inspired by the A* algorithm for path planning (Zhu et al. 2023; Meng et al. 2024; Zhuang et al. 2024), we argue that *an effective path prioritization mechanism should simultaneously consider the accumulative cost up to the current step and the estimated cost to reach the target*. Building on this insight, we design a novel path priority function to guide the search towards important reasoning paths. Specifically, it comprises two aspects: **the accumulative cost** $d(q, e)$, which measures the cost from the query q to the current entity e, and **the estimated future cost** $f(e, a)$, which estimates the remaining cost to reach the target answer a. The overall priority score is defined as $s_q(e) = d(q, e) + f(e, a)$. Figure 3 illustrates this process on the KG.

However, designing an effective path priority function presents two key challenges. First, unlike the traditional A* algorithm typically applied to grid graphs, KGs are heterogeneous graphs where edges represent semantic relationships rather than geometric distances, making it non-trivial to define an appropriate cost function that measures the "semantic distances" between entities. Second, KGs are often large-scale with vast numbers of entities and relations. Consequently, directly applying the A* algorithm can easily lead to an intractably large search space, rendering the search inefficient or even infeasible.

To address these challenges, we propose a semantic-aware path priority function that efficiently discovers important reasoning paths. To be concrete, we define the accumulative cost $d(q, e)$ as the aggregation of the paths that connect the topic entities in the query q to the current entity e. We compute its corresponding representation $\boldsymbol{d}(q, e)$, as follows:

$$\begin{aligned} \boldsymbol{d}(q, e) &= \sum_{\pi \in \Pi_{q \rightsquigarrow e}} \boldsymbol{d}(\pi) \\ &= \sum_{\pi \in \Pi_{q \rightsquigarrow e}} \sum_{(e_{i-1}, r_i, e_i) \in \pi} \boldsymbol{w}_q(e_{i-1}, r_i, e_i), \end{aligned} \tag{3}$$

where $\Pi_{q \rightsquigarrow e}$ denotes the set of paths from the query q to the entity e, and $\boldsymbol{w}_q(e_{i-1}, r_i, e_i) = (\boldsymbol{h}_{e_{i-1}} \boldsymbol{W}_{r_i} \boldsymbol{h}_{e_i})^\top \boldsymbol{q}$ denotes the semantic representation of the triple (e_{i-1}, r_i, e_i) in the path π, conditioned on the query q.

For the estimated future cost $f(e, a)$, since the target answer a is unknown in advance, we reparameterize the answer a using the topic entities and the query relation. Thus, we define the representation $\boldsymbol{f}(e, a)$ as:

$$\boldsymbol{f}(e, a) = \boldsymbol{f}([\boldsymbol{d}(q, e), \boldsymbol{q}]), \tag{4}$$

where $\boldsymbol{f}(\cdot)$ is a feed-forward neural network, and $[\cdot, \cdot]$ concatenates two representations. Intuitively, the function $\boldsymbol{f}([\boldsymbol{d}(q, e), \boldsymbol{q}])$ compares the current representation $\boldsymbol{d}(q, e)$ with the query $\boldsymbol{q}$ to estimate the remaining cost. If $\boldsymbol{d}(q, e)$ is close to $\boldsymbol{q}$, the remaining cost will be close to 0, indicating that the entity e is likely close to the correct answer. Finally, the overall priority score is computed as:

$$s_q(e) = \sigma\big(\text{MLP}\big(\boldsymbol{d}\,(q, e) + \boldsymbol{f}\,(e, a)\big)\big), \tag{5}$$

where MLP denotes a multi-layer perceptron, and $\sigma(\cdot)$ is the sigmoid function that normalizes the output to [0, 1]. In this way, we effectively measure semantic distances while significantly reducing the search space by restricting the retrieval scope to the query subgraph $\mathcal{G}_q$.

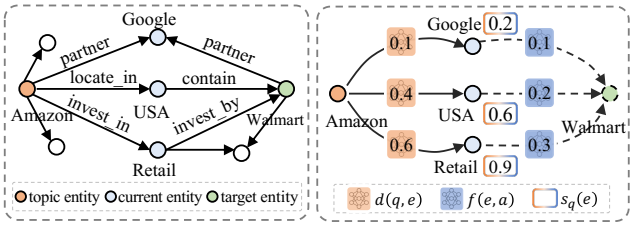

Figure 3: Illustration of the path priority function on the KG, where the current entity e is evaluated based on the accumulative cost $d(q, e)$ and the estimated future cost $f(e, a)$.

Learning. To learn the path priority function, we use the knowledge reasoning task as supervision, encouraging the model to assign higher priorities $s_q(e)$ to entities that are more important for reasoning. The loss is as follows:

$$\mathcal{L} = -\sum_{e \in \mathcal{A}_q} \log\left(s_q(e)\right) - \sum_{e \in \mathcal{G}_q \setminus \mathcal{A}_q} \log\left(1 - s_q(e)\right), \tag{6}$$

where $\mathcal{A}_q$ is the set of answers to the query q. Once the priority function $s_q(e)$ is learned, we repeatedly select the top-K entities based on the learned priority scores during each iteration $t \in T$. As a result, we identify important reasoning paths $\Pi_q = \{\pi_i \mid \pi_i = e_0 \xrightarrow{r_1} e_1 \xrightarrow{r_2} \ldots \xrightarrow{r_T} e_T\}_{i=1}^{N}$, where N is the number of reasoning paths.

Knowledge Reasoning Module

Upon obtaining the important reasoning paths, we fine-tune LLMs via a two-phase training strategy: task-specific instruction tuning and path-wise preference alignment. This ensures that the model not only generates accurate responses but also develops logically consistent reasoning. In addition, it avoids the high overhead from multiple LLM calls.

Task-Specific Instruction Tuning. Initially, the instruction tuning phase is designed to improve the model's ability to interpret and solve reasoning problems by adhering to task-specific instructions. In this phase, we employ Supervised Fine-Tuning (SFT) to train the model on various KGR

tasks. Specifically, the model takes the query q and the important reasoning paths Π_q as input, and generates the corresponding answers $\mathcal{A}_q$ as output. The SFT loss is formally defined as follows:

$$\mathcal{L}_{\text{SFT}} = -\mathbb{E}_{(q,\mathcal{A}_q)\sim\mathcal{D}_{\text{SFT}}}\big[\log P_\phi(\mathcal{A}_q \mid q, \Pi_q)\big], \quad (7)$$

where ϕ denotes the model parameters, and the task dataset is defined as $\mathcal{D}_{\text{SFT}} = \{q^{(i)}, \mathcal{A}_q^{(i)}\}_{i=1}^{N}$. The paths Π_q are verbalized into textual prompts using an instruction template. The model trained in this phase is denoted as $\mathcal{M}_{\text{sft}}$.

Path-Wise Preference Alignment. The second phase applies Direct Preference Optimization (DPO) to further align the model with more accurate and logically consistent reasoning based on preference paths. For each query q, we consider the input pairs labeled (Π_q^w, Π_q^l), where Π_q^w and Π_q^l represent the preferred and less preferred paths, respectively. DPO refines the learned SFT model, $\mathcal{M}_{\text{sft}}$, by aligning it more closely with the preferred paths. The training objective for DPO is defined as follows:

$$\mathcal{L}_{\text{DPO}}\left(\mathcal{M}; \mathcal{M}_{\text{sft}}\right) = -\mathbb{E}_{\left(q,\Pi_q^w,\Pi_q^l\right)\sim\mathcal{D}_{\text{DPO}}} \left[\log\sigma\left(\beta\log\frac{\mathcal{M}\left(\Pi_q^w \mid q\right)}{\mathcal{M}\left(\Pi_q^l \mid q\right)} - \beta\log\frac{\mathcal{M}_{\text{sft}}\left(\Pi_q^w \mid q\right)}{\mathcal{M}_{\text{sft}}\left(\Pi_q^l \mid q\right)}\right)\right], \quad (8)$$

where β is a parameter controlling the deviation from the base reference model $\mathcal{M}_{\text{sft}}$, and the comparison dataset is defined as $\mathcal{D}_{\text{DPO}} = \{q^{(i)}, \Pi_q^{w(i)}, \Pi_q^{l(i)}\}_{i=1}^{M}$. To construct these preference pairs (Π_q^w, Π_q^l), we treat the retrieved important reasoning paths Π_q as the preferred paths Π_q^w, while the less preferred paths Π_q^l are sampled from the remaining candidate paths in the subgraph $\mathcal{G}_q$.

Experiments

We conduct experiments to validate the effectiveness of our model and address key research questions. **RQ1:** How does PathMind perform compared to existing KGR methods? **RQ2:** What are the contributions of different modules to the overall performance? **RQ3:** How generalizable, scalable, and efficient is our PathMind framework? **RQ4:** How does PathMind perform interpretable reasoning?

Experimental Settings

Datasets. We evaluate PathMind on two popular benchmark datasets: WebQuestionSP (WebQSP) (Yih et al. 2016) and Complex WebQuestions (CWQ) (Talmor and Berant 2018). See Luo et al. (2024) for more details.

Baselines. We compare our method against representative baselines, categorized into two groups:

- **Traditional KGR methods.** *(i) Embedding-based methods:* KVMem (Miller et al. 2016) and NSM (He et al. 2021) *(ii) Retrieval-based methods:* GraftNet (Sun et al. 2018), SR+NSM (Zhang et al. 2022), ReaRev (Mavromatis and Karypis 2022)
- **LLM-based KGR methods.** *(i) Retrieval-Augmented methods:* BGE (Zhang et al. 2023), MindMap (Wen, Wang, and Sun 2024), RoG (Luo et al. 2024), GNN-RAG (Mavromatis and Karypis 2025), SubgraphRAG (Li, Miao, and Li 2025), LightPROF (Ao et al. 2025), EPERM (Long et al. 2025), GCR (Luo et al. 2025) *(ii) Synergy-Augmented methods:* KD-CoT (Wang et al. 2023), EffiQA (Dong et al. 2025), StructGPT (Jiang et al. 2023), ToG (Sun et al. 2024), PoG (Chen et al. 2024), KnowPath (Zhao et al. 2025),

Evaluation Protocols. Following prior work, we evaluate model performance using two standard metrics: Hits@1 and F1. Hits@1 measures the proportion of instances where the top-1 predicted answer is correct, while F1 provides a balanced measure of answer coverage.

Implementation Details. In our implementation, we use Llama3.1-8B as the LLM backbone. For subgraph retrieval, we sample 3-hop neighborhoods to construct query subgraphs. In path prioritization, node and relation representations are learned using GNN, while query representations are encoded with the pre-trained BERT. We select the top-3 nodes at each iteration and set the maximum number of iterations to 2 for WebSQP and 4 for CWQ. For knowledge reasoning, we train the model for 3 epochs with a batch size of 2. The learning rate is set to 2e-5 with a warm-up ratio of 3e-2. In DPO training, the learning rate is 5e-6, and the hyperparameter β is set to 0.1. The maximum input length for the LLM is set to 2048 tokens. Our model is implemented in PyTorch and trained on two NVIDIA A800 GPUs.

Overall Comparison (RQ1)

Table 1 presents the main results on KGR using WebQSP and CWQ. (1) *Overall Performance:* PathMind significantly outperforms all compared baselines, demonstrating superior reasoning capabilities over KGs. Specifically, our method improves Hits@1 by 0.8% over the second-best method EPERM, on WebQSP. On the more challenging CWQ dataset, which involves multi-hop questions, PathMind achieves improvements of 5.1% in Hits@1 and 3.9% in F1 compared to the strong baseline GNN-RAG. These results highlight the effectiveness of our method, especially in handling complex reasoning tasks. (2) *Analysis of Other Methods*: Interestingly, while GCR excels in Hits@1, its F1 score is relatively low, likely because the model tends to focus on the most probable answer. Among traditional methods, retrieval-based approaches outperform embedding-based methods. For example, SR+NSM, which uses path retrieval, achieves better performance, emphasizing the significance of reasoning paths. (3) *LLM Performance and Challenges:* Despite the strong general capabilities of LLMs such as Llama-3.1-8B and GPT-4o, they still exhibit a significant performance gap compared to the best fine-tuned models. This underscores the inherent difficulty of answering complex queries using LLMs alone.

Ablation Study (RQ2)

Performance on Model Ablation. To evaluate the effect of different components of PathMind, we compare three variants: (1) *w/o Prioritization*, removing the path prioritization module; (2) *w/o Alignment*, excluding path-wise prefer-

Methods	WebQSP		CWQ	
	Hits@1	F1	Hits@1	F1
KV-Mem	0.467	0.345	0.184	0.157
GraftNet	0.664	0.604	0.368	0.327
NSM	0.687	0.628	0.476	0.424
SR+NSM	0.689	0.641	0.502	0.471
ReaRev	0.764	0.709	0.529	0.478
Qwen2-7B	0.508	0.355	0.253	0.216
Llama-2-7B	0.564	0.365	0.284	0.214
Llama-3.1-8B	0.555	0.348	0.281	0.224
GPT-4o	0.618	0.436	0.382	0.329
KD-CoT	0.686	0.525	0.557	–
StructGPT	0.726	0.637	0.543	0.496
MindMap	0.649	0.471	0.488	0.433
ToG	0.826	–	0.685	–
RoG	0.857	0.708	0.626	0.562
LightPROF	0.838	–	0.593	–
KnowPath	0.841	–	0.679	–
GNN-RAG*	0.864	0.690	0.673	0.591
SubgraphRAG	0.866	0.706	0.472	0.570
EPERM	0.888	0.724	0.662	0.589
GCR*	0.883	0.654	0.686	0.532
PathMind (ours)	**0.895**	**0.728**	**0.707**	**0.614**

Table 1: Performance comparison of PathMind with baselines on two datasets. (**bold** denotes the best results, underline denotes the second best results, and "*" marks results reproduced using Llama3.1-8B for fair comparison.).

ence alignment; and (3) *w/o Training*, without the two-phase training strategy. As shown in Table 2, removing path prioritization makes the model rely solely on the query subgraph, resulting in a significant decline in performance on WebQSP and CWQ. This finding validates the critical role of identifying and selecting explicit reasoning paths. The absence of alignment (via DPO) yields inferior results, indicating the necessity of aligning the model with preferred reasoning paths. Furthermore, without training, the model struggles to interpret the structural knowledge of KGs, leading to substantial performance degradation.

Variants	WebQSP		CWQ	
	Hits@1	F1	Hits@1	F1
PathMind	0.895	0.728	0.707	0.614
w/o Priorization	0.840	0.662	0.643	0.561
w/o Alignment	0.871	0.695	0.672	0.586
w/o Training	0.668	0.480	0.413	0.274

Table 2: Model ablation study of our PathMind framework.

Performance on Path Prioritization. To further investigate the necessity of path prioritization, we compare three path selection strategies: *random paths*, *shortest paths*, and *important paths (ours)*. Random paths are constructed via

Strategies	WebQSP		CWQ	
	Hits@1	F1	Hits@1	F1
Random Paths	0.356	0.104	0.268	0.079
Shortest Paths	0.854	0.681	0.662	0.578
Important Paths	0.895	0.728	0.707	0.614

Table 3: Impact on different path prioritization strategies.

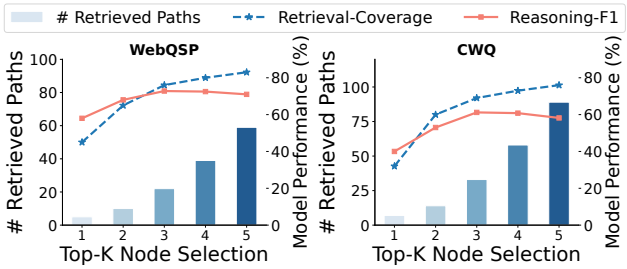

Figure 4: Effect on varying numbers of node selection.

random walks, while shortest paths are obtained using the GNN-RAG method. As shown in Table 3, our method significantly outperforms the other two strategies, demonstrating the effectiveness of prioritizing semantically relevant paths. Although other strategies enable efficient traversal, they often introduce irrelevant noise and hinder accurate reasoning. In addition, we find that path selection has a more pronounced impact on CWQ compared to WebQSP. We hypothesize that this is due to the inherently multi-hop nature of complex questions in CWQ, where effective path selection is crucial for supporting logical reasoning.

Performance on Hyperparameter Setting. We consider two key hyperparameters: T, the maximum number of iterations, and K, the maximum number of nodes selected per iteration. Based on the characteristics of each dataset, we set $T = 2$ for WebQSP and $T = 4$ for CWQ. To assess the impact of varying K on model performance, we conduct a series of experiments. As depicted in Figure 4, performance improves with increasing K. However, when K exceeds 3, the F1 scores on both datasets tend to decline, likely due to the inclusion of irrelevant entities that obscure critical information. Consequently, we select $K = 3$ as the optimal setting to identify important reasoning paths.

Further Analysis (RQ3)

Generalizability Across LLMs. To assess the generalizability of PathMind across various LLMs, we conduct experiments using representative models, including Qwen2-7B, Llama2-7B, and Llama3.1-8B. From the results in Table 4, we can see that PathMind achieves outstanding performance across these LLMs, underscoring the effectiveness and adaptability of our proposed framework. In particular, when fine-tuned on Llama3.1-8B, our method attains higher accuracy than other LLMs, suggesting that more advanced models can significantly boost overall performance.

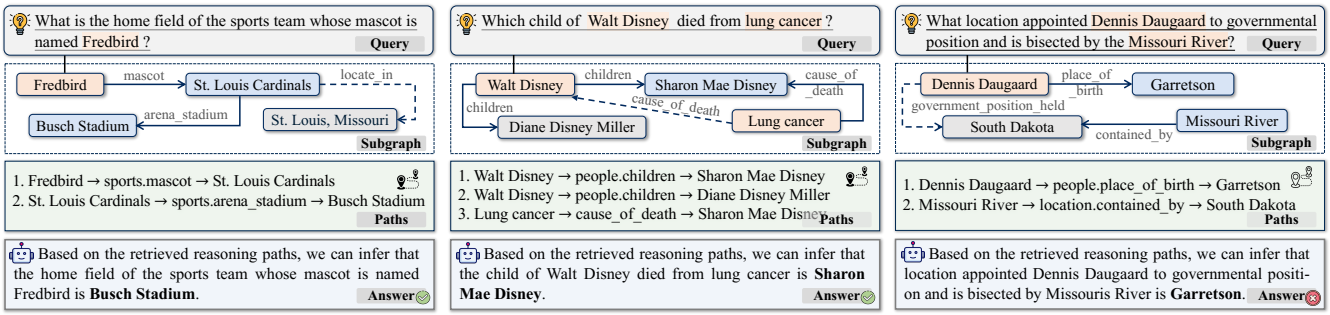

Figure 5: Illustrations of interpretable reasoning of PathMind on CWQ (solid lines indicate retrieved important paths).

Backbones	WebQSP		CWQ	
	Hits@1	F1	Hits@1	F1
Llama2-7B	0.864	0.687	0.652	0.573
Qwen2-7B	0.872	0.693	0.665	0.580
Llama3.1-8B	0.895	0.728	0.707	0.614

Table 4: Transferability of PathMind across various LLMs.

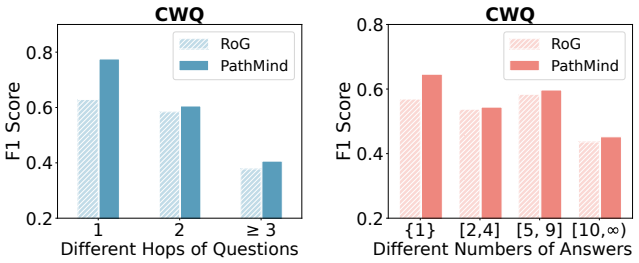

Figure 6: Scalability of PathMind with different question hops and different answer numbers on CWQ.

Reasoning Scalability Analysis. To better understand the scalability of our model, we analyze its performance across reasoning hops and answer numbers. As shown in Figure 6, PathMind consistently surpasses RoG, demonstrating strong robustness and scalability. In simple scenarios, our method effectively identifies important paths, yielding substantial improvements. As task complexity increases, PathMind maintains its leading performance, thanks to its ability to disregard numerous irrelevant paths, thereby reducing interference from extraneous information.

Model Efficiency Evaluation. To evaluate model efficiency, we analyze the average runtime, number of LLM calls, and number of input tokens in Table 5. Specifically, synergy-augmented methods, such as PoG, achieve superior results but require more LLM calls and input tokens due to their iterative reasoning process. In contrast, retrieval-augmented methods benefit from efficient subgraph retrieval, which significantly reduces runtime and computational costs. Our method strikes a favorable balance between performance and efficiency, particularly by reducing input tokens through the identification of important paths.

Methods	Hit (%)	Time (s)	# Calls	# Tokens
Synergy-Augmented Methods				
ToG	0.751	16.14	11.6	7,069
EffiQA	0.829	–	7.3	–
PoG	0.873	16.80	9.0	5,518
Retrieval-Augmented Methods				
RoG	0.857	2.60	2	521
GNN-RAG	0.864	**1.52**	**1**	414
GCR	0.883	3.60	2	231
PathMind	**0.895**	2.23	**1**	**216**

Table 5: Efficiency comparison of PathMind on WebQSP.

Case Study (RQ4)

We further present three complex cases in CWQ to demonstrate the interpretable reasoning process of PathMind. As illustrated in Figure 5, in Case 1, the method correctly extracts the two-hop reasoning path $Fredbird \xrightarrow{mascot} St.\ Louis\ Cardinals \xrightarrow{arena\ stadium} Busch\ Stadium$, which is consistent with human cognition. In Case 2, although the retrieved reasoning paths contain noises, such as "the child of Walt Disney" corresponding to two facts: $Walt\ Disney \xrightarrow{children} Sharon\ Mae\ Disney$ and $Walt\ Disney \xrightarrow{children} Diane\ Disney\ Miller$, the reasoning module can successfully discern valuable evidence and perform faithful reasoning. In Case 3, our model makes an incorrect prediction due to the missing important path $Dennis\ Daugaard \xrightarrow{gov.pos.held} South\ Dakota$, which highlights the significance of path identification from KGs to augment the reasoning capabilities of LLMs.

Conclusion

In this paper, we introduce PathMind, a novel framework designed to enhance LLMs for KGR. The core idea is to guide LLMs using important reasoning paths, enabling more faithful and interpretable reasoning. Our method comprises three key components: subgraph retrieval, path prioritization, and knowledge reasoning. Extensive results and model analysis demonstrate that PathMind consistently outperforms competitive baselines across various KGR benchmarks.

References

Ao, T.; Yu, Y.; Wang, Y.; et al. 2025. LightPROF: A Lightweight Reasoning Framework for Large Language Model on Knowledge Graph. In *AAAI*.

Chen, L.; Tong, P.; Jin, Z.; et al. 2024. Plan-on-Graph: Self-Correcting Adaptive Planning of Large Language Model on Knowledge Graphs. In *NIPS*.

Dong, Z.; Peng, B.; Wang, Y.; et al. 2025. EffiQA: Efficient Question-Answering with Strategic Multi-Model Collaboration on Knowledge Graphs. In *COLING*.

Halpern, J. Y. 1986. Reasoning about knowledge: An overview. In *Theoretical aspects of reasoning about knowledge*, 1–17. Elsevier.

He, G.; Lan, Y.; Jiang, J.; et al. 2021. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In *WSDM*.

Jiang, J.; Zhou, K.; Dong, Z.; et al. 2023. StructGPT: A General Framework for Large Language Model to Reason over Structured Data. In *EMNLP*.

Jiang, P.; Xiao, C.; Jiang, M.; et al. 2025. Reasoning-Enhanced Healthcare Predictions with Knowledge Graph Community Retrieval. In *ICLR*.

Kim, J.; Kwon, Y.; Jo, Y.; and Choi, E. 2023. KG-GPT: A General Framework for Reasoning on Knowledge Graphs Using Large Language Models. In *EMNLP*.

Li, M.; Miao, S.; and Li, P. 2025. Simple is effective: The roles of graphs and large language models in knowledge-graph-based retrieval-augmented generation. In *ICLR*.

Li, Z.; Fan, S.; Gu, Y.; et al. 2024. Flexkbqa: A flexible llm-powered framework for few-shot knowledge base question answering. In *AAAI*.

Lin, X.; Ning, Y.; Zhang, J.; et al. 2025. LLM-based Agents Suffer from Hallucinations: A Survey of Taxonomy, Methods, and Directions. *arXiv preprint*.

Liu, B.; Zhang, J.; Lin, F.; et al. 2025a. SymAgent: A Neural-Symbolic Self-Learning Agent Framework for Complex Reasoning over Knowledge Graphs. In *WWW*.

Liu, R.; Luo, B.; Li, J.; et al. 2025b. Ontology-Guided Reverse Thinking Makes Large Language Models Stronger on Knowledge Graph Question Answering. *ACL*.

Liu, Y.; Cao, Y.; Lin, X.; et al. 2025c. Enhancing Large Language Model for Knowledge Graph Completion via Structure-Aware Alignment-Tuning. In *EMNLP*.

Liu, Y.; Cao, Y.; Wang, S.; et al. 2024. Generative models for complex logical reasoning over knowledge graphs. In *WSDM*.

Long, X.; Zhuang, L.; Li, A.; et al. 2025. EPERM: An Evidence Path Enhanced Reasoning Model for Knowledge Graph Question and Answering. In *AAAI*.

Luo, H.; Tang, Z.; Peng, S.; Guo, Y.; Zhang, W.; et al. 2023. Chatkbqa: A generate-then-retrieve framework for knowledge base question answering with fine-tuned large language models. *arXiv preprint*.

Luo, L.; Li, Y.-F.; Haf, R.; and Pan, S. 2024. Reasoning on Graphs: Faithful and Interpretable Large Language Model Reasoning. In *ICLR*.

Luo, L.; Zhao, Z.; Gong, C.; et al. 2025. Graph-constrained reasoning: Faithful reasoning on knowledge graphs with large language models. *ICLR*.

Ma, S.; Xu, C.; Jiang, X.; et al. 2025. Think-on-Graph 2.0: Deep and Faithful Large Language Model Reasoning with Knowledge-guided Retrieval Augmented Generation. *ICLR*.

Mavromatis, C.; and Karypis, G. 2022. ReaRev: Adaptive Reasoning for Question Answering over Knowledge Graphs. In *EMNLP*.

Mavromatis, C.; and Karypis, G. 2025. Gnn-rag: Graph neural retrieval for efficient large language model reasoning on knowledge graphs. In *ACL*.

Meng, S.; Wang, Y.; Yang, C.-F.; et al. 2024. LLM-A*: Large Language Model Enhanced Incremental Heuristic Search on Path Planning. In *EMNLP*.

Meyer, J. G.; Urbanowicz, R. J.; Martin, P. C.; et al. 2023. ChatGPT and large language models in academia: opportunities and challenges. *BioData Mining*, 16(1): 20.

Miller, A.; Fisch, A.; Dodge, J.; et al. 2016. Key-Value Memory Networks for Directly Reading Documents. In *EMNLP*.

Pan, S.; Luo, L.; Wang, Y.; et al. 2024. Unifying large language models and knowledge graphs: A roadmap. *TKDE*.

Sun, H.; Dhingra, B.; Zaheer, M.; et al. 2018. Open Domain Question Answering Using Early Fusion of Knowledge Bases and Text. In *EMNLP*.

Sun, J.; Xu, C.; Tang, L.; et al. 2024. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *ICLR*.

Talmor, A.; and Berant, J. 2018. The Web as a Knowledge-Base for Answering Complex Questions. In *NAACL*.

Wang, K.; Duan, F.; Wang, S.; et al. 2023. Knowledge-driven cot: Exploring faithful reasoning in llms for knowledge-intensive question answering. *arXiv preprint*.

Wen, Y.; Wang, Z.; and Sun, J. 2024. MindMap: Knowledge Graph Prompting Sparks Graph of Thoughts in Large Language Models. In *ACL*.

Yih, W.-t.; Richardson, M.; Meek, C.; et al. 2016. The value of semantic parse labeling for knowledge base question answering. In *ACL*.

Zhang, J.; Zhang, X.; Yu, J.; et al. 2022. Subgraph Retrieval Enhanced Model for Multi-hop Knowledge Base Question Answering. In *ACL*.

Zhang, P.; Xiao, S.; Liu, Z.; et al. 2023. Retrieve anything to augment large language models. *arXiv preprint*.

Zhao, Q.; Yang, H.; Song, Q.; et al. 2025. KnowPath: Knowledge-enhanced Reasoning via LLM-generated Inference Paths over Knowledge Graphs. *arXiv preprint*.

Zhou, S.; Dai, X.; Chen, H.; et al. 2020. Interactive recommender system via knowledge graph-enhanced reinforcement learning. In *SIGIR*.

Zhu, Z.; Yuan, X.; Galkin, M.; et al. 2023. A* net: A scalable path-based reasoning approach for knowledge graphs. *NIPS*.

Zhuang, Y.; Chen, X.; Yu, T.; et al. 2024. ToolChain*: Efficient Action Space Navigation in Large Language Models with A* Search. In *ICLR*.