

Super Paths, Better Reasoning: Multi-Hop Knowledge Graphs Reasoning with LLM via Super-Path Ranking and Constraint Verification

Kunpeng Du^{1,*}, Yuhang Zheng^{1,*}, Xiaopeng Li¹, Chen Gao², Jishu Wang³,
Rui Zhu¹, Xuan Zhang^{1,†}

¹Yunnan University ²South China University ³Yunnan Normal University

*Equal contribution. †Corresponding author: Xuan Zhang, Email: zhxuan@ynu.edu.cn

Abstract

Knowledge graph reasoning (KGR) composes multi-hop facts over KGs to support complex knowledge-intensive tasks, providing reliable external factual support for large language models (LLMs) and helping mitigate hallucinations. However, building compact, faithful, and high-quality contexts from large-scale KGs remains challenging. We observe a non-monotonic relationship between retrieval scale and reasoning quality: insufficient retrieval misses critical paths, while excessive retrieval introduces redundant and noisy paths that burden LLMs reasoning. Motivated by this observation, we propose super-path ranking and constraint verification (SPaRC), an efficient framework that constructs, ranks, and verifies super-paths before LLMs reasoning. SPaRC compresses redundant paths into super-paths, ranks them with contrastive learning to prioritize high-recall reasoning paths, and provides the LLM with the top-ranked paths after constraint verification for answer reasoning. Experiments on WebQSP and CWQ show that SPaRC achieves state-of-the-art performance, improving Hit@1 by up to 12.0 points and F1 by up to 5.9 points on CWQ. On challenging settings covering deeper hop reasoning, multiple topic entities, and multi-answer questions, SPaRC further improves F1 on two benchmarks by up to 17.6% and 42.1%, 26.7% and 5.9%, and 10.4% and 79.4%, respectively, while reducing reasoning tokens by 29.7% and retrieval time by 25%¹.

1 Introduction

Large language models (LLMs) (OpenAI et al., 2024; Guo et al., 2025) have demonstrated unprecedented advantages in a wide range of natural language processing tasks, owing to their strong semantic understanding and reasoning capabilities (Brown et al., 2020). However, in knowledge-intensive tasks, LLM reasoning still heavily relies

¹Our code is available here.

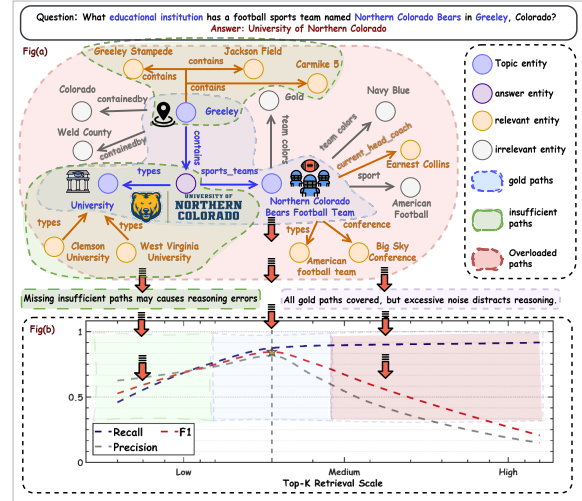


Figure 1: A motivation example of multi-hop KGR: insufficient retrieval misses necessary paths, while over-expanded retrieval introduces noisy paths that distract LLM reasoning.

on static knowledge acquired during pre-training, making LLMs vulnerable to outdated knowledge and factual hallucinations, thereby limiting the reliability and interpretability of LLMs responses (Pan et al., 2024; Jiang et al., 2025b). Knowledge graphs (KGs) organize factual knowledge in a structured and updatable form and can ground LLMs reasoning in verifiable external facts (Jiang et al., 2025b; Peng et al., 2025). Knowledge graph reasoning (KGR) further composes multi-hop facts from KGs to support complex knowledge-intensive tasks with LLMs (Niu et al., 2026). However, effectively exploiting large-scale KGs is non-trivial: the retrieved information must be sufficient to cover the necessary reasoning facts, while remaining compact enough to prevent redundant or noisy content from distracting the LLMs reasoning and reducing answer reliability (Lin et al., 2026; Liu et al., 2026; Zhang et al., 2026b).

As shown in Fig. 1, the key to KGR lies in identifying a set of factual paths that are both sufficient

and compact from a large-scale KG (Wang et al., 2026). In Fig. 1(a), the blue region illustrates the golden oracle subgraph required for the example question, where the answer entity is obtained by composing 1-hop and 2-hop query results over type, location, and team-related paths. However, obtaining such an ideal oracle subgraph is difficult in practice. The green region shows insufficient retrieval, where missing necessary paths leave the reasoning chain incomplete and may cause reasoning errors. In contrast, the red region denotes over-expanded retrieval, where the required paths are covered but many irrelevant paths are also introduced, making the context noisy and distracting LLMs reasoning. Fig. 1(b) quantifies this trade-off: as the retrieval scale increases, the recall curve rises but eventually plateaus, whereas precision drops significantly due to the influx of redundant information, which subsequently drags down the F1 score. For LLMs, this accumulation of noise leads to the lost in the middle phenomenon (Liu et al., 2024), where vital paths are buried under irrelevant content, making them harder to identify and use during reasoning.

This challenge naturally raises a broader question: how LLMs should interact with KGs to obtain and use KG facts for multi-hop reasoning. Existing studies differ mainly in how LLMs are involved in the interaction with KGs. When LLMs are placed inside the retrieval loop, this study treats LLMs as planning agents (Luo et al., 2024) to decompose questions (Chen et al., 2024; Zhang et al., 2026b), perform reflection before verification (Sun et al., 2024; Zhao et al., 2026), and iteratively explore facts (Yan et al., 2026) from KGs. This interaction design is flexible, but repeated LLM calls incur high latency and token costs, and stepwise decisions from partial context may lack a global view, leading to seemingly plausible but incomplete path reasoning (Chen et al., 2026; Wang et al., 2026). When LLMs are kept outside the retrieval loop, lightweight retrievers can pre-extract candidate subgraphs or paths for one-time reasoning (Li et al., 2025; Liu et al., 2025; Mavromatis and Karypis, 2025), improving efficiency but often introducing reasoning-irrelevant noise due to insufficient alignment between question semantics and KG structural constraints (Xiao et al., 2026; Liu et al., 2026). Therefore, the core challenge of KGR lies not in the quantity of retrieved knowledge, but in constructing a compact, faithful, and high-quality context for LLMs reasoning (Yang et al., 2026). This motivates our investigation into how retrieval scope

influences the trade-off between key path recall and noise accumulation.

These observations suggest that multi-hop KGR should not simply pursue a larger retrieval scale, but should instead focus on selectively providing truly important paths (Zhang et al., 2026a). Inspired by this, we propose **super-path ranking and constraint verification (SPaRC)**, an efficient decoupled framework for multi-hop KGR. SPaRC first compresses paths with the same anchor entity and relation sequence into super-paths, reducing redundant intermediate and tail entities while preserving key relational logic. It then trains a contrastive ranker over super-paths, where the current question’s ground-truth super-paths are used as positive samples, and two types of negatives are introduced: first, non-ground-truth super-paths within the same question as hard negatives to distinguish surface-form similar paths that cannot support answer reasoning; second, ground-truth super-paths from other questions in the same batch as easy negatives to learn global decision boundaries across different questions. Finally, SPaRC verifies top-ranked paths with query semantic constraints and provides high-quality context to the LLMs for answer reasoning. By moving super-path construction, ranking, and verification into lightweight modules before reasoning, SPaRC allows LLMs to focus on final complex reasoning rather than costly graph exploration and noise filtering. Our contributions are as follows:

(1) Super paths, better reasoning. We propose SPaRC, an efficient decoupled multi-hop KGR framework with LLMs that introduces super-path to reduce redundancy among key facts while preserving multi-hop reason logic, allowing LLMs to focus on reliable and cost-effective reasoning.

(2) Contrastive ranker, higher recall. We design a super-path-based contrastive ranker that learns to distinguish paths both within the same question and across different questions, enabling more accurate ranking of valid paths against misleading paths with surface-form similarity while helping to improve the recall of top-ranked candidate super-paths.

(3) Significant performance and efficiency gains. Experiments on WebQSP and CWQ show that SPaRC achieves SOTA performance. In challenging settings, SPaRC further improves F1 on the two benchmarks by up to 17.6% and 42.1% for multi-hop reasoning, 26.7% and 5.9% for multi-entity cases, and 10.4% and 79.4% for multiple-

answer cases, while reducing reasoning tokens by 29.7% and retrieval time by 25%.

2 Related Work

Recent multi-hop KGR studies have explored different ways of integrating KGs with LLMs. These methods use KG facts, paths, or executable queries to ground LLMs reasoning. Including agent-based methods, information retrieval (IR)-based methods, and semantic parsing (SP)-based methods.

(1) Agent-based KGR methods place LLMs in the retrieval loop and use them to guide KG exploration (Sun et al., 2024; Luo et al., 2024; Chen et al., 2024; Zhang et al., 2026b; Zhao et al., 2026). They usually decompose questions into intermediate goals, select relations or entities step by step, and refine the exploration trajectory according to previously retrieved facts. This interactive process enables flexible reasoning, but repeated LLM calls make it computationally expensive and highly dependent on the capability of the backbone LLMs.

(2) IR-based KGR methods decouple retrieval from generation by first retrieving triples, paths, or subgraphs from KGs, and then using LLMs to reason over answers (Li et al., 2025; Liu et al., 2025; Mavromatis and Karypis, 2025). Retrieval can be implemented with sparse (Robertson and Zaragoza, 2009) or dense similarity (Gao et al., 2021), graph search (Li et al., 2025), or GNN-based retrievers (Liu et al., 2025; Mavromatis and Karypis, 2025). This design is efficient and scalable, but its effectiveness is limited by retrieval quality, where insufficient retrieval misses key paths while excessive retrieval introduces noisy context.

(3) SP-based KGR method translates given questions into executable logical forms, such as S-expressions or SPARQL queries, and obtains answers by executing them over KGs (Ye et al., 2022; Zhang et al., 2023; Yu et al., 2023). This method offers explicit and interpretable reasoning when the logical form is correct, but generating executable queries that accurately align with the query intent remains challenging (Yan et al., 2026).

Inspired by these lines of work, SPaRC retains the efficiency of retrieval-based reasoning while making path selection more discriminative and constraint-aware before LLMs reasoning. SPaRC reconstructs candidate paths into compact super-paths, ranks them to prioritize intent-supporting paths over those with misleading surface-form similarity, and verifies the top-ranked paths under se-

mantic constraints, thereby providing LLMs with a compact and reliable path context for efficient multi-hop KGR.

3 Preliminaries

Definition 1: Knowledge graph A knowledge graph is denoted as $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, where $\mathcal{E}$ is the entity set, $\mathcal{R}$ is the relation set, and $\mathcal{T} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is the set of factual triples. Each triple $(e_h, r, e_t) \in \mathcal{T}$ represents a relation r between a head entity e_h and a tail entity e_t . A k -hop path is defined as $p = (e_0, r_1, e_1, \dots, r_k, e_k)$, where $(e_{j-1}, r_j, e_j) \in \mathcal{T}$ for $1 \leq j \leq k$.

Definition 2: Super-path. Given an anchor entity $e_i \in \mathcal{E}$ and a relation sequence $[r_1, r_2, \dots, r_k]$, a super-path is defined as $s = (e_i, [r_1, r_2, \dots, r_k])$. It abstracts a group of instantiated paths that share the same anchor entity and relation sequence, where the anchor can correspond to either the source side or the target side of the matched path pattern. Compared with instantiated paths, a super-path compresses repeated intermediate and terminal entities while preserving the relational logic used for multi-hop reasoning.

Multi-hop KGR task statement. Given a natural language question q , a knowledge graph $\mathcal{G}$, and an LLM f_{LLM} , multi-hop knowledge graph reasoning aims to infer the answer entity or entity set $\mathcal{A}_q \subseteq \mathcal{E}$ by composing multiple facts from $\mathcal{G}$. In SPaRC, our goal is to construct a compact and faithful reasoning context $\mathcal{C}_q$ from candidate super-paths before LLMs reasoning, and predict the answer as $\hat{\mathcal{A}}_q = f_{\text{LLM}}(q, \mathcal{C}_q)$, where $\mathcal{C}_q$ should contain sufficient relational constraints for answering q while reducing redundant or noisy paths.

4 Methodology

As shown in Fig. 2, SPaRC first retrieves a question-relevant subgraph from the KG as the initial reasoning basis. Based on this subgraph, SPaRC reconstructs candidate paths into super-paths, ranks them with a contrastive ranker, and verifies the top-ranked paths under query-semantic constraints before LLMs reasoning. The verified path context is then fed into the LLM to reason the final answer.

4.1 Question-relevant Subgraph Retrieval

Given a question q and a knowledge graph $\mathcal{G}$, the first step of SPaRC is to retrieve a question-relevant subgraph as the initial candidate space for multi-hop reasoning. To better utilize KG knowledge

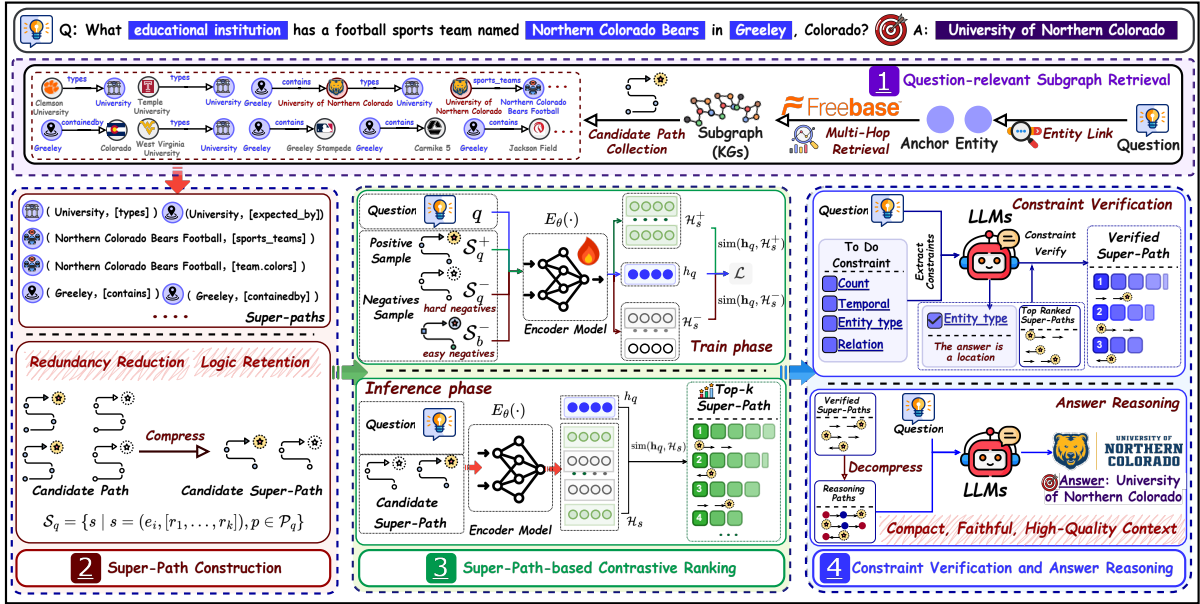


Figure 2: Overall framework of SPaRC. Given a question and an external knowledge base, SPaRC first retrieves a question-relevant subgraph (KGs), then constructs compact super-paths, ranks them with a contrastive ranker, and verifies the top-ranked paths under query-semantic constraints before LLMs reasoning.

related to the question, we first identify topic entities in q using entity linking tools, where ELQ (Li et al., 2020) and FACC1² are commonly used for entity retrieval. The linked topic entities are denoted as $\mathcal{E}_q^0$.

Starting from $\mathcal{E}_q^0$, we perform bounded multi-hop expansion over $\mathcal{G}$ to collect candidate triples and paths. The expansion depth H is set according to the dataset: following common practice, we use a 2-hop expansion for WebQSP (Yih et al., 2016) and a 4-hop expansion for CWQ (Talmor and Berant, 2018). Formally, the retrieved subgraph is denoted as $\mathcal{G}_q = (\mathcal{E}_q, \mathcal{R}_q, \mathcal{T}_q)$, where $\mathcal{T}_q \subseteq \mathcal{T}$ contains triples collected within the H -hop neighborhood of the topic entities. The corresponding candidate path set is denoted as $\mathcal{P}_q = \{p \mid p \subseteq \mathcal{G}_q, |p| \leq H\}$, where each path is associated with at least one topic entity and contains at most H relations. This stage aims to provide a high-recall candidate space, while super-path construction, ranking, and constraint verification further refine the retrieved paths before LLMs reasoning.

4.2 Super-Path Construction

Given the retrieved subgraph $\mathcal{G}_q$ and its candidate path set $\mathcal{P}_q$, this stage aims to reduce path redundancy before ranking and LLMs reasoning. A conventional instantiated path explicitly enumerates

entities and relations as

$$p = (e_0, r_1, e_1, \dots, r_k, e_k). \quad (1)$$

In large retrieved subgraphs, many paths share similar relational sequence and anchor entity but differ only in intermediate or tail entities, leading to unnecessary context expansion. To address this, we construct super-paths by abstracting candidate paths into compact path units defined by an anchor entity and a relation sequence.

For each $p \in \mathcal{P}_q$, we select an anchor entity e_i and represent its relation sequence as

$$s = (e_i, [r_1, r_2, \dots, r_k]), \quad (2)$$

where $[r_1, r_2, \dots, r_k]$ denotes the relation sequence associated with the path. Paths with the same anchor entity and relation sequence are merged into the same super-path,

$$\mathcal{S}_q = \{s \mid s = (e_i, [r_1, \dots, r_k]), p \in \mathcal{P}_q\}. \quad (3)$$

For each super-path s , we maintain its instantiated path set $\mathcal{I}(s)$ for later constraint verification and answer recovery. Compared with instantiated paths, super-paths compress repeated intermediate and tail entities while preserving the core relational logic, providing a more compact input path for subsequent ranking.

4.3 Super-Path-based Contrastive Ranking

After super-path construction, the candidate set $\mathcal{S}_q$ may still contain paths that are related to the ques-

²FACC1: Freebase Annotations of the ClueWeb Corpora.

tion only by surface-form similarity, but cannot support the final answer reasoning. To prioritize more useful super-paths before constraint verification, we train a contrastive ranker over super-paths. For a question q , we denote its ground-truth super-paths as $\mathcal{S}_q^+$, and the non-ground-truth super-paths from the same question as $\mathcal{S}_q^-$. In addition, the ground-truth super-paths of other questions in the same mini-batch are denoted as $\mathcal{S}_b^-$. Here, $\mathcal{S}_q^-$ provides same-question hard negatives, while $\mathcal{S}_b^-$ provides cross-question easy negatives.

We use a shared encoder $E_\theta(\cdot)$ to project both questions and super-paths into the same semantic space. The question embedding is defined as $\mathbf{h}_q = E_\theta(q)$. For any super-path s , its embedding is $\mathbf{h}_s = E_\theta(s)$. Accordingly, the positive and negative super-path embedding sets are defined as

$$\mathcal{H}_s^+ = \{\mathbf{h}_s \mid s \in \mathcal{S}_q^+\}, \quad (4)$$

and

$$\mathcal{H}_s^- = \{\mathbf{h}_s \mid s \in \mathcal{S}_q^- \cup \mathcal{S}_b^-\}. \quad (5)$$

Since the encoder output is normalized, the similarity between q and a super-path embedding $\mathbf{h}_s$ is computed as $\text{sim}(\mathbf{h}_q, \mathbf{h}_s) = \mathbf{h}_q^\top \mathbf{h}_s / \tau$, where τ is the temperature coefficient. The contrastive objective maximizes the probability of positive super-paths against all negative super-paths (Oord et al., 2018). We first define the normalization term as:

$$Z_q = \sum_{\mathbf{h}_s \in \mathcal{H}_s^+} e^{\text{sim}(\mathbf{h}_q, \mathbf{h}_s)} + \sum_{\mathbf{h}_s \in \mathcal{H}_s^-} e^{\text{sim}(\mathbf{h}_q, \mathbf{h}_s)}, \quad (6)$$

then the training loss is:

$$\mathcal{L} = -\log \frac{\sum_{\mathbf{h}_s \in \mathcal{H}_s^+} e^{\text{sim}(\mathbf{h}_q, \mathbf{h}_s)}}{Z_q}, \quad (7)$$

this objective encourages the ranker to assign higher scores to ground-truth super-paths, while reducing the scores of both same-question hard negatives and cross-question easy negatives.

During inference, we use the trained encoder $E_\theta(\cdot)$ to encode q and each candidate super-path $s_i \in \mathcal{S}_q$, and compute its probability under the question as:

$$P_\theta(s_i \mid q) = \frac{e^{\text{sim}(\mathbf{h}_q, \mathbf{h}_{s_i})}}{\sum_{s_j \in \mathcal{S}_q} e^{\text{sim}(\mathbf{h}_q, \mathbf{h}_{s_j})}}. \quad (8)$$

Let $s^1, s^2, \dots, s^{|\mathcal{S}_q|}$ be the super-paths sorted in descending order of $P_\theta(s_i \mid q)$. The top- k super-path set is then obtained as $\mathcal{S}_q^k = \{s^1, s^2, \dots, s^k\}$. The selected $\mathcal{S}_q^k$ is passed to the constraint verification module.

4.4 Constraint Verification and Answer Reasoning

The top-ranked super-paths obtained by the contrastive ranker are semantically relevant to the question, but they may still contain paths that do not satisfy the answer constraints implied by the query. Therefore, SPaRC further introduces a constraint verification stage before final LLMs reasoning. This stage uses extracted constraints to rerank candidate paths and then constructs a compact, faithful, and high-quality context for answer reasoning.

4.4.1 Constraint Verification

Given a question q , we first use the LLM to extract its semantic constraints. The extracted constraint set is denoted as $\mathcal{Z}_q$, where each extracted constraint describes an expected property of the final answer, with supported constraint types including entity type, relation, temporal, and count constraints.

For each top-ranked super-path $s^i \in \mathcal{S}_q^k$, the LLM assigns a constraint score $v(q, s^i) \in [0, 1]$ based on the extracted constraints $\mathcal{Z}_q$. A higher score indicates that the super-path better matches the semantic constraints implied by the question. We combine the ranker probability and the constraint score as follows:

$$\text{Score}(q, s^i) = (1 - \alpha)P_\theta(s^i \mid q) + \alpha v(q, s^i), \quad (9)$$

where α controls the contribution of constraint verification. This step promotes super-paths that are both highly ranked by the contrastive ranker and consistent with the query constraints, while down-weighting surface-form similar but constraint-irrelevant super-paths.

The verified super-path set is obtained as:

$$\mathcal{V}_q = \{s^i \in \mathcal{S}_q^k \mid \text{Score}(q, s^i) \geq \delta\}, \quad (10)$$

where δ is a verification threshold. The resulting $\mathcal{V}_q$ contains the verified super-paths that will be used for final answer reasoning.

4.4.2 Answer Reasoning

After constraint verification, SPaRC decompresses each verified super-path $s^i \in \mathcal{V}_q$ into its corresponding instantiated paths through $\mathcal{I}(s^i)$, and organizes these paths with the question q to form the final reasoning context $\mathcal{C}_q$. Finally, the LLM takes $q, \mathcal{C}_q$ as input and generates the predicted answer $\mathcal{A}_q$. In this way, SPaRC allows the LLM to focus on final answer reasoning over a compact, faithful, and high-quality path context, rather than filtering noisy paths from the retrieved path set.

5 Experiment

We evaluate SPaRC on standard multi-hop KGR benchmarks, analyzing its performance, efficiency, and the individual contributions of its components. Our experimental analysis aims to address the following research questions: **RQ1.** Does SPaRC outperform existing baselines on multi-hop KGR benchmarks? **RQ2.** Does each core component effectively contribute to SPaRC? **RQ3.** Can SPaRC handle complex multi-hop, multi-entity, and multi-answer questions? **RQ4.** How effective is SPaRC compared with representative retrieval strategies? **RQ5.** Does SPaRC reduce token consumption and retrieval time?

5.1 Experimental Setup

Datasets. We evaluate SPaRC on two Freebase-based KGQA benchmarks: WebQuestionsSP (WebQSP) (Yih et al., 2016) and ComplexWebQuestions (CWQ) (Talmor and Berant, 2018). WebQSP contains 4,737 questions with up to 2-hop reasoning, while CWQ contains 34,689 questions with up to 4-hop reasoning and more complex constraints. Detailed statistics are reported in Table 1 and A.1.

Dataset	Composition		Max hop	Hop in test (%)		
	Train	Test		1-hop	2-hop	>2-hop
WebQSP	3,098	1,639	2	63.8	36.2	0.0
CWQ	31,158	3,531	4	21.5	55.3	23.2

Table 1: Statistics of datasets.

Baselines. We compare SPaRC with traditional KGR methods and LLM-based KGR methods. Traditional KGR methods perform multi-hop reasoning over KG structures without relying on LLMs, including KV-Mem (Miller et al., 2016), GraftNet (Sun et al., 2018), EmbedKGQA (Saxena et al., 2020), NSM (He et al., 2021), ReaRev (Mavromatis and Karypis, 2022), SubgraphRetrieval (Zhang et al., 2022), UniKGQA (Jiang et al., 2023), FC-KBQA (Zhang et al., 2023), and DecAF (Yu et al., 2023). According to how LLMs interact with KGs, LLM-based KGR methods are divided into two categories: (1) **Agent-based methods** place LLMs in the reasoning loop to guide iterative KG exploration, including ToG (Sun et al., 2024), RoG (Luo et al., 2024), Interact-KBQA (Xiong et al., 2024), KG-Agent (Jiang et al., 2025a), AgenticKBQA (Chen et al., 2026), and VoG (Zhao

	Method	WebQSP		CWQ	
		Hit@1	F1	Hit@1	F1
Traditional KGR	KV-Mem	46.7	34.5	21.1	15.7
	GraftNet	66.7	62.4	36.8	32.7
	EmbedKGQA	66.6	—	45.9	—
	NSM	68.7	62.8	47.6	42.4
	ReaRev	76.4	70.9	52.9	47.8
	SubgraphRetrieval	69.5	64.1	50.2	47.1
	UniKGQA	77.2	70.2	51.2	48.0
	FC-KBQA	—	76.9	—	56.4
	DecAF	82.1	78.8	70.4	—
KGR with LLMs	Agent-based methods				
	ToG+ChatGPT	76.2	—	57.6	—
	ToG+GPT-4	82.6	—	68.5	—
	RoG	85.7	70.8	62.6	56.2
	Interact-KBQA	72.5	71.2	—	49.1
	KG-Agent	83.3	—	72.2	—
	AgenticKBQA	80.1	77.2	—	68.3
	VoG+GPT-4	88.7	73.2	77.6	67.5
	IR-based methods				
	DualR+GPT-4	87.6	—	73.6	—
	SubgraphRAG*	86.6	70.6	57.0	47.2
	SubgraphRAG†	90.1	77.5	62.0	54.1
	GNN-RAG	85.7	71.3	66.8	59.4
	ReKnoS	81.1	<u>79.5</u>	58.5	56.8
	GCR+ChatGPT	92.6	73.2	72.7	60.9
	GCR†	92.2	74.1	75.8	61.7
	PoG+ChatGPT	91.6	—	74.7	—
	RRP†	91.1	72.5	71.7	56.5
	SCR†	90.8	74.8	68.4	57.3
	STEM*	86.6	71.1	68.8	60.8
	STEM+GPT-4o	90.9	76.2	74.1	65.3
Ours	Our*	92.5	78.1	87.4	68.6
	Our†	93.4	79.7	89.6	74.2

Table 2: Performance comparison of different methods on WebQSP and CWQ. **Bold** and underline denote the best and second-best results, respectively. * and † mark results reproduced using Llama-3.1-8B and GPT-4o-mini for fair comparison.

et al., 2026). (2) **IR-based methods** retrieve relevant KG contexts and then use LLMs for answer reasoning, including DualR (Liu et al., 2025), SubgraphRAG (Li et al., 2025), GNN-RAG (Mavromatis and Karypis, 2025), ReKnoS (Wang et al., 2025), GCR (Luo et al., 2025), PoG (Tan et al., 2025), RRP (Xiao et al., 2026), SCR (Lin et al., 2026), and STEM (Yu et al., 2026). Detailed descriptions of these baselines are provided in Appendix A.2.

Evaluation metrics and Implementation details.

We use Hit@1 and F1 to evaluate the overall answer performance. For retrieval evaluation, we report Hit@ k , precision@ k , recall@ k , and F1@ k , where k denotes the number of top-ranked paths retained for evaluation. We use RoBERTa-large as the shared encoder for the super-path contrastive ranker. The ranker is trained with $\tau = 0.05$, batch size 32, and

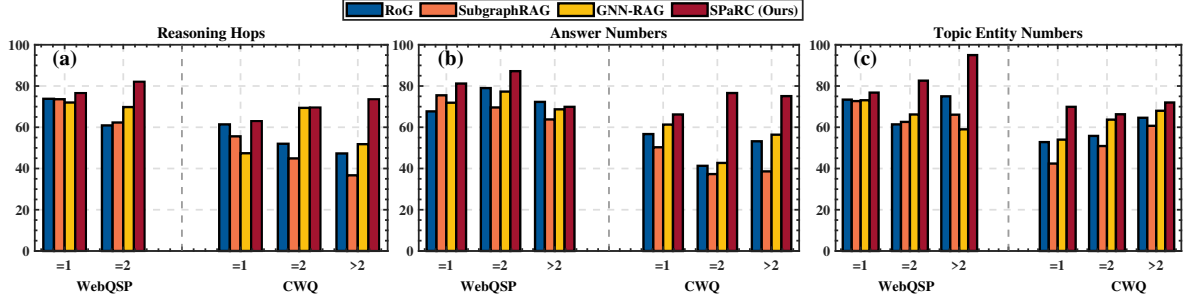


Figure 3: Performance on complex questions: (a) multi-hop, (b) multi-answer, and (c) multi-topic entity settings.

Dataset	Top-1				Top-5			
	Hit	P.	R.	F1	Hit	P.	R.	F1
WebQSP	90.8	90.8	80.5	83.9	99.4	24.5	98.7	38.4
CWQ	96.9	96.9	74.9	81.9	99.5	38.4	98.0	52.1

Table 3: Retrieval performance of SPaRC.

8 negative super-paths per question. GPT-4o-mini and Llama-3.1-8B are used for constraint verification and answer reasoning. All experiments are conducted on NVIDIA 2*A800 GPU using PyTorch and HuggingFace Transformers. More details are provided in Appendix A.3 and A.4.

5.2 Main Results

RQ1: KGR Performance. The main results show that SPaRC achieves SOTA performance on WebQSP and CWQ, as reported in Table 2. Compared with the best baseline results, SPaRC brings larger gains on CWQ, improving Hit@1 from 77.6% to 89.6% and F1 from 68.3% to 74.2%, with absolute improvements of 12.0 and 5.9 points, respectively. On WebQSP, SPaRC further improves Hit@1 to 93.4% and F1 to 79.7%. The gains are therefore more evident on CWQ. This is reasonable because CWQ requires deeper hop expansion and contains more complex questions than WebQSP. As the hop depth increases, the retrieved subgraph usually becomes larger, making useful path identification and irrelevant path filtering more difficult. By constructing a compact path context with super-path ranking and constraint verification, SPaRC is more effective in handling such complex reasoning scenarios.

Under the same LLM backbone, SPaRC consistently outperforms representative baselines. With GPT-4o-mini, SPaRC outperforms GCR and SubgraphRAG on both datasets, with gains on CWQ, +12.55 F1 over GCR and +20.15 F1 over SubgraphRAG. With Llama-3.1-8B, SPaRC also surpasses STEM by +7.05 F1 on WebQSP and +7.85

F1 on CWQ. These results suggest that the gains come not only from the backbone LLM, but also from providing the LLMs with a compact and constraint-aware factual context.

RQ3: Complex Questions Reasoning. Fig. 3 reports F1 results under different question complexity settings, including hop depth, answer number, and topic entity number. SPaRC achieves the best or comparable-best results across most groups, with clearer advantages on the harder CWQ subsets. For deeper-hop reasoning (≥ 2 hops), it improves F1 by up to 17.6% on WebQSP and 42.1% on CWQ. For multi-answer questions, SPaRC further improves F1 by up to 10.4% on WebQSP and 79.4% on CWQ. For multi-topic-entity questions, the relative improvements reach 26.7% and 5.9% on the two datasets, respectively. These results indicate that SPaRC remains effective when questions require longer reasoning paths, multiple topic entities, or multiple answers. This benefits from its design: super-path ranking selects more useful path structures before LLM reasoning, while constraint verification further refines the selected paths according to the question requirements.

RQ4: Retrieval Results. Table 3 reports the retrieval performance of the super-path-based contrastive ranking module. Here, Hit@ k measures whether at least one useful super-path is included in the top- k ranked results. The ranker achieves high Hit@1 on both WebQSP and CWQ, and Hit@5 further reaches 99.4 and 99.5, indicating that useful paths are almost always retained within a small top- k set. Precision@1 is identical to Hit@1 because only one super-path is evaluated at $k = 1$, so the top-ranked result is either useful or not. When k increases from 1 to 5, recall increases substantially, but precision naturally drops because more super-paths are retained and not all of them are ground-truth paths. This also lowers F1@5 despite the higher coverage. This trend is consistent

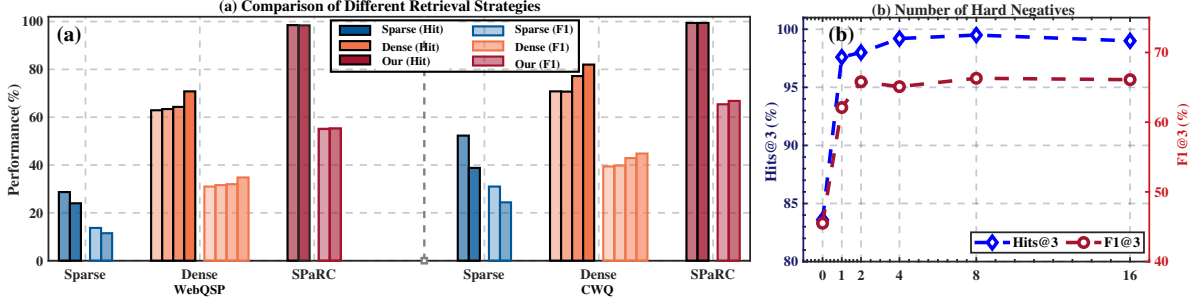


Figure 4: Super-path-based retriever performance analysis: (a) comparison with representative sparse (BM25, TF-IDF), dense (MiniLM, MPNet, SimCSE, BGE) retrieval models and our (using BERT and RoBERTa backbones). (b) Effect of hard negatives on CWQ Top-3 retrieval.

Method	#KG Tokens	Time (s)	LLM Calls
RoG	622	2327	2
SubgraphRAG	1229	12	1
GNN-RAG+RA	655	160	2
SPaRC	437	9	3

Table 4: Efficiency comparison of different methods.

with our motivation: the contrastive ranker provides high-coverage candidate super-paths, while the subsequent constraint verification module further filters them to reduce irrelevant paths before LLMs reasoning. Further detailed analysis can be found in Section 5.5.

RQ5: Efficiency Analysis. Table 4 shows that SPaRC achieves a better efficiency balance than existing methods. Compared with RoG and GNN-RAG+RA, SPaRC avoids expensive graph exploration and reduces runtime from 2327s and 160s to 9s, respectively, while using fewer KG tokens. Compared with SubgraphRAG, which uses only one LLM call, SPaRC still reduces KG tokens and runtime. This indicates that the extra verification call does not introduce a large token overhead, because super-path construction and ranking substantially reduce the path context before LLMs reasoning. Overall, SPaRC improves efficiency by compressing the retrieved path space rather than relying on long contexts or iterative KG exploration. More analysis see Appendix D.2.

5.3 RQ2: Ablation Study

To examine the contribution of each core component, we remove constraint verification (CV), super-path-based contrastive ranking (CR), super-path construction (SP), and both CR and CV from SPaRC. As shown in Table 5, all variants lead to performance degradation. Removing SP replaces super-paths with original paths, causing F1 to drop

Variant	WebQSP		CWQ	
	Hit	F1	Hit	F1
SPaRC*	92.5	78.1	87.4	68.6
w/o CV	89.4	73.6	85.8	67.3
w/o CR	80.3	63.4	83.7	67.4
w/o SP	86.0	66.3	85.8	58.6
w/o CR + CV	69.2	42.5	64.0	49.3

Table 5: Ablation study of SPaRC. CR denotes super-path-based contrastive ranking, SP denotes super-paths, and CV denotes constraint verification.

from 78.1 to 66.3 on WebQSP and from 68.6 to 58.6 on CWQ, which shows that super-path construction helps reduce path redundancy before ranking and verification. Removing CR or CV also decreases performance, indicating that ranking and verification are both necessary for selecting useful paths and filtering constraint-irrelevant ones. When both CR and CV are removed, all constructed super-paths are directly fed into the LLM, introducing more noisy and irrelevant paths; this results in the largest drop, with F1 decreasing to 42.5 on WebQSP and 49.3 on CWQ. These results show that SP, CR, and CV jointly produce a compact and reliable reasoning context for LLMs reasoning. More detailed analyses are provided in Appendix D.

5.4 RQ4: Retrieval Strategies Comparison.

Fig. 4(a) shows that the proposed super-path-based retriever clearly outperforms representative sparse and dense retrieval models. Sparse methods such as TF-IDF and BM25 perform poorly, especially on WebQSP, indicating that lexical matching is insufficient for identifying useful reasoning paths. Dense retrievers improve the results, with BGE achieving the strongest baseline performance, but they still lag behind our trained retriever by a large margin. Our retriever shows that task-specific contrastive

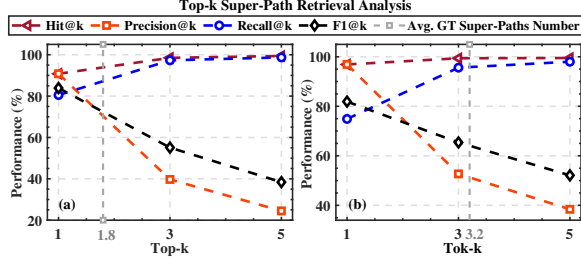


Figure 5: Additional analysis of top- k super-path retrieval results: (a) WebQSP and (b) CWQ.

training better aligns questions with useful super-paths than general retrieval models.

Fig. 4(b) shows that introducing hard negatives improves the retrieval result. Compared with using only easy negatives, adding hard negatives substantially increases both metrics, and the performance becomes stable when the number of hard negatives reaches 8. This indicates that the same question hard negatives help the retriever distinguish useful super-paths from surface-form similar paths.

5.5 RQ4: Analysis of Contrastive Ranker

To examine how retrieval scope affects key path recall and noise accumulation, we analyze the super-path ranker under different top- k settings. As shown in Fig. 5, Hit@ k is high at $k = 1$ and increases at $k = 3$ and $k = 5$, indicating that top-ranked results usually contain at least one ground-truth super-path. However, questions contain on average 1.8 and 3.2 ground-truth super-paths on WebQSP and CWQ, respectively. A single top-ranked path may therefore not cover all required facts, especially on CWQ. Thus, top- k retrieval seeks not one correct path but a small candidate set covering the required reasoning facts. When k increases from 1 to 3, Recall@ k improves substantially, while Precision@ k decreases as non-ground-truth paths are added. Increasing k further to 5 brings only marginal recall gains, with continued declines in precision and F1. This shows that top- k retrieval provides a high-coverage but still coarse candidate set. SPaRC therefore applies constraint verification to rerank these paths before constructing the final context for LLM reasoning.

5.6 RQ2: Effect of Constraint Verification

Constraint verification further improves the top-ranked super-paths produced by the contrastive ranker. As shown in Fig. 6, increasing α from zero generally improves precision and F1, indicating

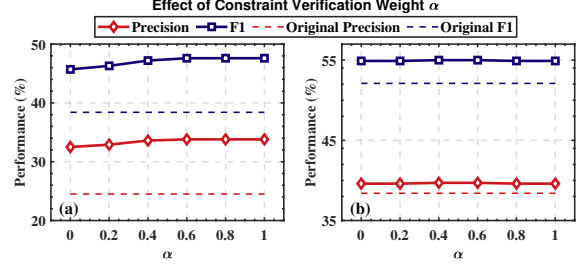


Figure 6: Effect of constraint verification weight α on Top-5 retrieval results: (a) WebQSP and (b) CWQ.

that the constraint score complements the ranking score by promoting paths more compatible with the extracted constraints. The improvements are more visible on WebQSP, whereas the curves on CWQ remain relatively stable, suggesting that the contrastive ranker already provides strong top-ranked super-paths under this setting, and constraint verification mainly serves as a supplementary refinement rather than causing large ranking changes.

6 Conclusion

In this paper, we propose SPaRC, an efficient decoupled framework for multi-hop KGR with LLMs. SPaRC constructs compact and constraint-aware path contexts by compressing instantiated paths into super-paths, ranking them with contrastive learning, and verifying top-ranked paths before LLM reasoning. Experiments on WebQSP and CWQ show that SPaRC improves both effectiveness and efficiency, especially for complex questions. These results highlight the value of refining paths before LLM reasoning for reliable multi-hop knowledge graph reasoning.

Limitations

SPaRC is designed to refine retrieved KG paths before LLMs reasoning, but its performance still depends on whether the initial subgraph retrieval stage preserves the necessary reasoning facts. As shown in the failure analysis in Appendix E, errors caused by missing relevant paths remain an important source of failure, since later super-path ranking and constraint verification cannot recover facts that are absent from the candidate path set. In addition, the current super-path representation abstracts candidate paths by anchor entities and relation sequences, which improves compactness but may not capture all fine-grained distinctions among structurally similar paths. Further discussion of these directions is provided in Appendix F.

Ethics Statement

This work studies multi-hop knowledge graph reasoning with LLMs using existing public benchmarks and structured knowledge resources. It does not involve the collection of personal or sensitive user data. Since SPaRC relies on an external knowledge base and LLMs, its outputs may still inherit factual incompleteness, outdated knowledge, or biases from these resources. Therefore, its outputs should still be used with appropriate verification in applications that require high factual reliability.

Acknowledgments

This work is supported by the Yunnan Science and Technology Major Project under Grant No. 202602AD080003; Xingdian Talent Support Program Industrial Innovation Talent Project of Yunnan under Grant No. yfgc202422; Science Foundation of Young and Middle-aged Academic and Technical Leaders of Yunnan under Grant No. 202205AC160040; China Central Fund for Guiding Development of Local Science and Technology under Grant No. 202407AB110010; Knowledge-driven Smart Energy Science and Technology Innovation Team of Yunnan Provincial Department of Education; Open Foundation of Yunnan Key Laboratory of Software Engineering under Grant No. 2023SE101;

References

- Manuel Borroto, Francesco Ricca, Bernardo Cuteri, and Vito Barbara. 2022. [Sparql-qa enters the qald challenge](#). In *Proceedings of the 7th Natural Language Interfaces for the Web of Data (NLIWoD) co-located with the 19th European Semantic Web Conference, Hersionissos, Greece*, volume 3196, pages 25–31.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. [Language models are few-shot learners](#). *Advances in neural information processing systems*, 33:1877–1901.
- Liyi Chen, Panrong Tong, Zhongming Jin, Ying Sun, Jieping Ye, and Hui Xiong. 2024. [Plan-on-graph: Self-correcting adaptive planning of large language model on knowledge graphs](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 37665–37691.
- Zhuo Chen, Fei Wang, Zixuan Li, Zhao Zhang, Weiwei Ding, Chuanguang Yang, Yongjun Xu, and Xiaolong Jin. 2026. [Incentivizing agentic reasoning capability with outcome supervision for knowledge base question answering](#). In *Proceedings of the ACM Web Conference 2026*, pages 4302–4313.
- Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. [Simcse: Simple contrastive learning of sentence embeddings](#). In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 6894–6910.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 163 others. 2025. [Deepseek-r1 incentivizes reasoning in llms through reinforcement learning](#). *Nature*, 645(8081):633–638.
- Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. [Improving multi-hop knowledge base question answering by learning intermediate supervision signals](#). In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 553–561.
- Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, Yang Song, Chen Zhu, Hengshu Zhu, and Ji-Rong Wen. 2025a. [Kg-agent: An efficient autonomous agent framework for complex reasoning over knowledge graph](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9505–9523.
- Jinhao Jiang, Kun Zhou, Xin Zhao, and Ji-Rong Wen. 2023. [UniKGQA: Unified retrieval and reasoning for solving multi-hop question answering over knowledge graph](#). In *The Eleventh International Conference on Learning Representations*.
- Pengcheng Jiang, Siru Ouyang, Yizhu Jiao, Ming Zhong, Runchu Tian, and Jiawei Han. 2025b. [Retrieval and structuring augmented generation with large language models](#). In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pages 6032–6042.
- Belinda Z Li, Sewon Min, Srinivasan Iyer, Yashar Mehdad, and Wen-tau Yih. 2020. [Efficient one-pass end-to-end entity linking for questions](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6433–6441.
- Mufei Li, Siqi Miao, and Pan Li. 2025. [Simple is effective: The roles of graphs and large language models in knowledge-graph-based retrieval-augmented generation](#). In *The Thirteenth International Conference on Learning Representations*.
- Chenxiao Lin, Ye Luo, KunHong Liu, and Qingqiang Wu. 2026. [Stepwise contrastive reasoning for retrieval-augmented generation over knowledge graphs](#). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 15243–15251.

- Guangyi Liu, Yongqi Zhang, Yong Li, and Quanming Yao. 2025. [Dual reasoning: A gnn-llm collaborative framework for knowledge graph question answering](#). In *Conference on Parsimony and Learning*, volume 280, pages 351–372.
- Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. [Lost in the middle: How language models use long contexts](#). *Transactions of the association for computational linguistics*, 12:157–173.
- Yu Liu, Xixun Lin, Yanmin Shang, Yangxi Li, Shi Wang, and Yanan Cao. 2026. [Pathmind: A retrieve-prioritize-reason framework for knowledge graph reasoning with large language models](#). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 15386–15393.
- Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2024. [Reasoning on graphs: Faithful and interpretable large language model reasoning](#). In *The Twelfth International Conference on Learning Representations*.
- Linhao Luo, Zicheng Zhao, Gholamreza Haffari, Yuan-Fang Li, Chen Gong, and Shirui Pan. 2025. [Graph-constrained reasoning: Faithful reasoning on knowledge graphs with large language models](#). In *Forty-second International Conference on Machine Learning*.
- Costas Mavromatis and George Karypis. 2022. [Rearev: Adaptive reasoning for question answering over knowledge graphs](#). In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2447–2458.
- Costas Mavromatis and George Karypis. 2025. [Gnn-rag: Graph neural retrieval for efficient large language model reasoning on knowledge graphs](#). In *Findings of the Association for Computational Linguistics*, pages 16682–16699.
- Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. [Key-value memory networks for directly reading documents](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409.
- Guanglin Niu, Bo Li, and Yangguang Lin. 2026. [A comprehensive survey of knowledge graph reasoning: Approaches and applications](#). *IEEE Transactions on Big Data*, pages 1–20.
- Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. [Representation learning with contrastive predictive coding](#). *arXiv preprint arXiv:1807.03748*.
- OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. [Gpt-4 technical report](#). *Preprint*, arXiv:2303.08774.
- Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jipapu Wang, and Xindong Wu. 2024. [Unifying large language models and knowledge graphs: A roadmap](#). *IEEE Transactions on Knowledge and Data Engineering*, 36(7):3580–3599.
- Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. 2025. [Graph retrieval-augmented generation: A survey](#). *ACM Transactions on Information Systems*, 44(2):1–52.
- Stephen Robertson and Hugo Zaragoza. 2009. [The probabilistic relevance framework: Bm25 and beyond](#). *Foundations and Trends in Information Retrieval*, 4(1-2):1–174.
- Gerard Salton and Christopher Buckley. 1988. [Term-weighting approaches in automatic text retrieval](#). *Information processing & management*, 24(5):513–523.
- Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. [Improving multi-hop question answering over knowledge graphs using knowledge base embeddings](#). In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 4498–4507.
- Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. [Mpnet: Masked and permuted pre-training for language understanding](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 16857–16867. Curran Associates, Inc.
- Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. 2018. [Open domain question answering using early fusion of knowledge bases and text](#). In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 4231–4242.
- Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. 2024. [Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph](#). In *The Twelfth International Conference on Learning Representations*.
- Alon Talmor and Jonathan Berant. 2018. [The web as a knowledge-base for answering complex questions](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 641–651.
- Xingyu Tan, Xiaoyang Wang, Qing Liu, Xiwei Xu, Xin Yuan, and Wenjie Zhang. 2025. [Paths-over-graph: Knowledge graph empowered large language model reasoning](#). In *Proceedings of the ACM on Web Conference 2025*, pages 3505–3522.

- Song Wang, Junhong Lin, Xiaojie Guo, Julian Shun, Jundong Li, and Yada Zhu. 2025. [Reasoning of large language models over knowledge graphs with super-relations](#). In *The Thirteenth International Conference on Learning Representations*.
- Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. 2021. [Minilmv2: Multi-head self-attention relation distillation for compressing pre-trained transformers](#). In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2140–2151.
- Yingxu Wang, Shiqi Fan, Mengzhu Wang, Siyang Gao, Chao Wang, and Nan Yin. 2026. [Damr: Efficient and adaptive context-aware knowledge graph question answering with llm-guided mcts](#). In *The Fourteenth International Conference on Learning Representations*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. [Chain-of-thought prompting elicits reasoning in large language models](#). In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837.
- Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muenighoff, Defu Lian, and Jian-Yun Nie. 2024. [C-pack: Packed resources for general chinese embeddings](#). In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pages 641–649.
- Yilin Xiao, Chuang Zhou, Qinggang Zhang, Bo Li, Qing Li, and Xiao Huang. 2026. [Reliable reasoning path: Distilling effective guidance for llm reasoning with knowledge graphs](#). *IEEE Transactions on Knowledge and Data Engineering*, 38(4):2380–2393.
- Guanming Xiong, Junwei Bao, and Wen Zhao. 2024. [Interactive-kbqa: Multi-turn interactions for knowledge base question answering with large language models](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10561–10582.
- Shiqi Yan, Yubo Chen, Ruiqi Zhou, Zhengxi Yao, Shuai Chen, Tianyi Zhang, Shijie Zhang, Wei-Qiang Zhang, Yongfeng Huang, Haixin Duan, and Yunqi Zhang. 2026. [Explore-on-graph: Incentivizing autonomous exploration of large language models on knowledge graphs with path-refined reward modeling](#). In *The Fourteenth International Conference on Learning Representations*.
- Hao Yang, Zhiyu Yang, Xupeng Zhang, Wei Wei, Yunjie Zhang, and Lin Yang. 2026. [Compactrag: Reducing llm calls and token overhead in multi-hop question answering](#). In *Proceedings of the ACM Web Conference 2026*, pages 2240–2251.
- Xi Ye, Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, and Caiming Xiong. 2022. [Rng-kbqa: Generation augmented iterative ranking for knowledge base question answering](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6032–6043.
- Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. [The value of semantic parse labeling for knowledge base question answering](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206.
- Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Yang Wang, Zhiguo Wang, and Bing Xiang. 2023. [Decaf: Joint decoding of answers and logical forms for question answering over knowledge bases](#). In *The Eleventh International Conference on Learning Representations*.
- Peng Yu, En Xu, Bin Chen, Haibiao Chen, and Yinfei Xu. 2026. [Stem: Structure-tracing evidence mining for knowledge graphs-driven retrieval-augmented generation](#). *arXiv preprint arXiv:2604.22282*.
- Jing Zhang, Xiaokang Zhang, Jifan Yu, Jian Tang, Jie Tang, Cuiping Li, and Hong Chen. 2022. [Subgraph retrieval enhanced model for multi-hop knowledge base question answering](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5773–5784.
- Lingxi Zhang, Jing Zhang, Yanling Wang, Shulin Cao, Xinmei Huang, Cuiping Li, Hong Chen, and Juanzi Li. 2023. [Fc-kbqa: A fine-to-coarse composition framework for knowledge base question answering](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1002–1017.
- Qianchi Zhang, Hainan Zhang, Liang Pang, Yongxin Tong, Hongwei Zheng, and Zhiming Zheng. 2026a. [Less is more: Compact clue selection for efficient retrieval-augmented generation reasoning](#). In *Proceedings of the ACM Web Conference 2026*, pages 1971–1982.
- Yonggang Zhang, Jianqi Gao, and Jie Lu. 2026b. [Enhancing llms for knowledge base question answering by chain-of-decomposition](#). In *The Fourteenth International Conference on Learning Representations*.
- Wenxin Zhao, Jiachuan Wang, Yongqi Zhang, Shuangyin Li, Cheng Deng, Jun Wang, and Lei Chen. 2026. [Vog: Enhancing llm reasoning through step-wise verification on knowledge graphs](#). In *The Fourteenth International Conference on Learning Representations*.

Appendix

A Experimental Setup

A.1 Datasets

In addition to the basic dataset statistics, we further report the distribution of topic entities and answer

Dataset	Topic entity (%)			Answer number (%)		
	1	2	> 2	1	2	> 2
WebQSP	78.4	21.0	0.6	51.2	13.9	34.9
CWQ	52.0	41.2	6.9	74.3	8.0	17.7

Table 6: Statistics of the Number of Topic Entities and Answers.

numbers in the test sets. As shown in Table 6, WebQSP is dominated by single-topic entity questions, while CWQ contains a much larger proportion of questions with two or more topic entities. This indicates that CWQ requires more complex entity interaction and path composition. For answer distribution, both datasets contain a considerable number of multi-answer questions, which further increases the difficulty of producing complete answers in multi-hop KGR.

A.2 Baselines

We evaluate SPaRC against representative traditional KGR and LLM-based KGR baselines.

Traditional KGR methods.

- **KV-Mem** (Miller et al., 2016): Uses key-value memory slots to store KB triples, where subject-relation pairs serve as keys and objects serve as values, and performs multi-hop memory retrieval for answer prediction.
- **GraftNet** (Sun et al., 2018): Builds a question-specific subgraph with KB facts and entity-linked text, and applies graph neural networks to propagate information across entities, relations, and text nodes.
- **EmbedKGQA** (Saxena et al., 2020): Leverages KG embeddings to score candidate answers in the embedding space, improving multi-hop KGR over sparse KGs without requiring external text or a fixed local answer neighborhood.
- **NSM** (He et al., 2021): Introduces a teacher-student framework for neural-symbolic multi-hop reasoning, where intermediate supervision guides the student model to learn more reliable reasoning trajectories.
- **ReaRev** (Mavromatis and Karypis, 2022): Refines question instructions with KG-aware information and dynamically executes them over the graph, improving reasoning over complex questions and incomplete KGs.
- **SubgraphRetrieval** (Zhang et al., 2022): Retrieves and prunes a k -hop subgraph around the topic entity, then combines KG embeddings and relevance scoring to select candidate answers.
- **UniKGQA** (Jiang et al., 2023): Unifies subgraph retrieval and reasoning through question-relation matching and matching information propagation over KGs.
- **FC-KBQA** (Zhang et al., 2023): Extracts relevant fine-grained knowledge components from the KB and reformulates them into middle-grained knowledge pairs to generate logical expressions with improved generalization and executability.
- **DecAF** (Yu et al., 2023): Uses free-text retrieval over linearized KBs to jointly generate logical forms and direct answers, and combines logical-form-executed answers with directly generated answers to produce the final prediction.

KGR with LLMs. We compare with LLM-based KGR methods that differ in how LLMs interact with KGs. Specifically, we consider agent-based methods and IR-based methods.

Agent-based methods. These methods place LLMs in the reasoning loop to guide iterative KG exploration.

- **ToG** (Sun et al., 2024): Treats the LLM as an active reasoning agent that iteratively explores entities and relations in KGs, using beam search to discover promising reasoning paths before generating answers.
- **RoG** (Luo et al., 2024): Generates relation paths as reasoning plans, retrieves valid paths from the KG according to these plans, and uses the retrieved paths to guide LLMs reasoning.
- **Interact-KBQA** (Xiong et al., 2024): Uses LLMs to generate logical forms through direct interactions with KBs, where generic KB operation APIs and few-shot exemplars guide the reasoning process for complex questions.
- **KG-Agent** (Jiang et al., 2025a): Uses LLMs as autonomous agents equipped with KG operation tools and memory, enabling multi-step reasoning through iterative tool selection, execution, and memory updates over KGs.

- **AgenticKBQA** (Chen et al., 2026): Trains LLMs for autonomous agentic reasoning over KBs, using outcome-supervised curriculum reinforcement learning to iteratively decompose questions, generate logical queries, and interact with the KB to obtain answers.
- **VoG** (Zhao et al., 2026): Verifies and revises LLM-generated reasoning plans over KGs, using stepwise graph-based verification and adaptive revision to improve multi-hop reasoning reliability.
- **PoG** (Tan et al., 2025): Dynamically explores and prunes multi-hop reasoning paths from KGs, providing relevant path-based evidence for LLM reasoning over multi-hop and multi-entity questions.
- **RRP** (Xiao et al., 2026): Combines LLM semantics with KG structural information through relation embeddings and bidirectional distribution learning to extract reasoning paths, and employs a rethinking module to evaluate and rank them according to their contribution to answering the question.

IR-based methods. These methods first retrieve relevant KG contexts and then use LLMs for answer reasoning.

- **DualR** (Liu et al., 2025): Integrates GNN-based explicit reasoning with LLM-based implicit reasoning, where the GNN extracts reasoning chains from KGs and the LLM uses them to produce the final answer.
- **SubgraphRAG** (Li et al., 2025): Retrieves question-relevant subgraphs with a lightweight triple-scoring retriever and feeds the retrieved KG context into LLMs for answer reasoning.
- **GNN-RAG** (Mavromatis and Karypis, 2025): Uses GNNs to retrieve candidate answers from KG subgraphs, extracts shortest reasoning paths connecting question entities to candidates, and verbalizes these paths as context for LLMs reasoning.
- **ReKnoS** (Wang et al., 2025): Introduces super-relations to summarize and connect multiple relation paths, enabling both forward and backward reasoning over KGs to improve retrieval coverage and reasoning efficiency.
- **GCR** (Luo et al., 2025): Constrains LLMs decoding with a KG-Trie to generate KG-grounded reasoning paths and hypothesis answers, which are then used by a general LLMs for final graph-constrained reasoning.
- **SCR** (Lin et al., 2026): Performs stepwise contrastive reasoning by aligning the question representation with neighboring KG entities at each step, progressively selecting the next topic entity until reaching the answer.
- **STEM** (Yu et al., 2026): Reframes multi-hop KGR as schema-guided graph search, using semantic-to-structural projection, globally aware subgraph retrieval, and Triple-GNN guidance to construct evidence reasoning graphs.

Unsupervised retrieval methods. To evaluate the effectiveness of super-path ranking, we include representative retrieval methods as reference baselines, including sparse retrieval methods and dense retrieval methods.

Sparse retrieval methods represent queries and candidate texts with term-based sparse features and compute relevance mainly through lexical matching.

- **TF-IDF** (Salton and Buckley, 1988): TF-IDF estimates the importance of a term by considering its frequency in a document and discounting terms that frequently appear across the corpus, thereby providing a simple lexical relevance score between queries and documents.
- **BM25** (Robertson and Zaragoza, 2009): Best Matching 25 is a probabilistic ranking function widely used in information retrieval. It extends TF-IDF by considering term-frequency saturation and document-length normalization, providing a stronger lexical matching score between a query and candidate documents.

Dense retrieval methods encode queries and documents into low-dimensional dense vectors using neural encoders and retrieve by vector similarity, capturing semantic relevance beyond surface-form matching.

- **MiniLM** (Wang et al., 2021): A compact sentence transformer model obtained through self-attention relation distillation. It transfers fine-grained query-key-value interaction knowledge from large pretrained teacher models to smaller student models, producing efficient sentence-level representations with reduced model size.
- **MPNet-base-v2**: MPNet (Song et al., 2020) combines masked language modeling and permuted language modeling to improve pretrained language representations. The MPNet-base-v2 model is further used for sentence embedding by fine-tuning on large-scale sentence-pair data with a contrastive learning objective, enabling effective semantic representation learning.
- **SimCSE** (Gao et al., 2021): SimCSE is a contrastive learning framework for sentence embeddings. Its unsupervised version uses dropout noise to construct positive pairs from the same sentence, while its supervised version leverages natural language inference pairs to improve alignment and uniformity in the embedding space.
- **BGE** (Xiao et al., 2024): BGE provides a family of general text embedding models based on BERT-like architectures and trained through a multi-stage training recipe. It offers multiple model scales to balance representation quality and efficiency, and can be used directly for embedding-based retrieval or further fine-tuned for downstream applications. The bge-large-en-v1.5 is english version 1.5 model with a more reasonable similarity distribution.

A.3 Evaluation Metrics

For retrieval evaluation, we report $\text{Hit}@k$, $\text{precision}@k$, $\text{recall}@k$, and $\text{f1}@k$, where k denotes the number of top-ranked super-paths retained for evaluation. Consistent with the notation used in the methodology, let $\mathcal{S}_q^k$ be the top- k super-path set selected by the ranker and $\mathcal{S}_q^+$ be the ground-truth super-path set of question q . We define the number of correctly retrieved super-paths as $c_q^k = |\mathcal{S}_q^k \cap \mathcal{S}_q^+|$. The per-question retrieval metrics are defined as follows:

$$\text{Hit}@k(q) = \mathbf{1}(c_q^k > 0), \quad (11)$$

Setting	Value
<i>Super-path contrastive ranker</i>	
Encoder $E_\theta(\cdot)$	{RoBERTa, BERT}
Projection Dimension	256
Max length	128
Temperature τ	0.05
Batch size	{16, 32, 64}
Learning rate	2×10^{-5}
Epochs	{5, 10, 15}
Hard negatives	{0, 1, 2, 4, 8, 16}
Easy negatives	{0, 1, 2, 4, 8, 16}
<i>Retrieval and verification</i>	
Top- k	{1, 3, 5, 10}
Hop depth H	{2, 4}
Weight α	{0.2, 0.4, 0.6, 0.8, 1}
Threshold δ	0.5
LLM f_{LLM}	{GPT-4o-mini, Llama-3.1-8B}

Table 7: Key implementation settings of SPaRC.

$$\text{P}@k(q) = \frac{c_q^k}{|\mathcal{S}_q^k|}, \quad \text{R}@k(q) = \frac{c_q^k}{|\mathcal{S}_q^+|}, \quad (12)$$

$$\text{F1}@k(q) = \frac{2 \cdot \text{P}@k(q) \cdot \text{R}@k(q)}{\text{P}@k(q) + \text{R}@k(q)}. \quad (13)$$

The final reported values are averaged over all evaluation questions with ground-truth super-path sets.

A.4 Implementation Details

We summarize the key implementation settings of SPaRC in Table 7. The settings mainly cover the super-path contrastive ranker, question-relevant subgraph retrieval, and constraint verification.

B Ground-Truth Super-Path

Ground-truth super-paths are constructed from the dataset-provided SPARQL queries. Unlike GNN-RAG (Mavromatis and Karypis, 2025), which constructs ground-truth paths using shortest paths from topic entities to answer entities, we construct them based on the sub-query clauses extracted from the SPARQL queries, since multiple shortest paths may exist and not all of them necessarily satisfy the query intent.

Specifically, we use regular expressions to extract (subject, predicate, object) expressions from each SPARQL query, such as (ns:m.0160w, ns:location.country.currency_used, ?x). Given the retrieved subgraph described in Section 4.1, we identify the triples that match these expressions. For missing matches, the required

Algorithm 1 SPaRC Framework

Input: question q , KG $\mathcal{G}$, hop depth H , top- k , encoder $E_\theta(\cdot)$, LLM f_{LLM} , α , δ .
Output: predicted answer $\hat{\mathcal{A}}_q$.

// Question-relevant subgraph retrieval

- 1: $\mathcal{E}_q^0 \leftarrow \text{EntityLinking}(q)$.
- 2: $\mathcal{G}_q \leftarrow \text{SubgraphRetrieve}(\mathcal{G}, \mathcal{E}_q^0, H)$, $\mathcal{P}_q \leftarrow \text{PathCollection}(\mathcal{G}_q)$.

// Super-path construction

- 3: $\mathcal{S}_q \leftarrow \emptyset$, $\mathcal{I} \leftarrow \emptyset$.
- 4: **for each** $p = (e_0, r_1, e_1, \dots, r_k, e_k) \in \mathcal{P}_q$ **do**
- 5: $s \leftarrow (e_i, [r_1, \dots, r_k])$.
- 6: **if** $s \notin \mathcal{S}_q$ **then** $\mathcal{S}_q \leftarrow \mathcal{S}_q \cup \{s\}$, $\mathcal{I}(s) \leftarrow \emptyset$.
- 7: $\mathcal{I}(s) \leftarrow \mathcal{I}(s) \cup \{p\}$.
- 8: **end for**

// Super-path-based contrastive ranking

- 9: $\mathbf{h}_q \leftarrow E_\theta(q)$.
- 10: **for each** $s_i \in \mathcal{S}_q$ **do**
- 11: $\mathbf{h}_{s_i} \leftarrow E_\theta(s_i)$.
- 12: Compute $P_\theta(s_i | q)$ by Eq. (8).
- 13: **end for**
- 14: $\mathcal{S}_q^k \leftarrow \text{TopK}(\mathcal{S}_q, P_\theta(\cdot | q), k)$.

// Constraint verification

- 15: $\mathcal{Z}_q \leftarrow \text{ConstraintExtract}(q, f_{\text{LLM}})$, $\mathcal{V}_q \leftarrow \emptyset$.
- 16: **for each** $s^i \in \mathcal{S}_q^k$ **do**
- 17: $v(q, s^i) \leftarrow \text{Verify}(q, s^i, \mathcal{Z}_q, f_{\text{LLM}})$.
- 18: Compute $\text{Score}(q, s^i)$ by Eq. (9).
- 19: **if** $\text{Score}(q, s^i) \geq \delta$ **then** $\mathcal{V}_q \leftarrow \mathcal{V}_q \cup \{s^i\}$.
- 20: **end for**

// Answer reasoning

- 21: $\mathcal{C}_q \leftarrow \text{Decompress}(\mathcal{V}_q, \mathcal{I})$.
- 22: $\hat{\mathcal{A}}_q \leftarrow f_{\text{LLM}}(q, \mathcal{C}_q)$.
- 23: **return** $\hat{\mathcal{A}}_q$.

SPARQL-matched triples are retrieved from Freebase before constructing the positive path. The matched triples are then organized into ground-truth paths and converted into ground-truth super-paths following the construction procedure described in Section 4.2.

C Algorithm for SPaRC

Algorithm 1 summarizes the Framework of SPaRC, including subgraph retrieval, super-path construction, ranking, constraint verification, and final LLMs reasoning.

Algorithm 2 presents the training process of the super-path based contrastive ranker, which learns to distinguish ground-truth super-paths from same-

Algorithm 2 Super-Path-based Contrastive Ranker Training

Input: training set $\mathcal{D}$, encoder $E_\theta(\cdot)$, temperature τ , batch size B .
Output: trained encoder $E_\theta(\cdot)$.

- 1: **for each** mini-batch $\mathcal{B} \subset \mathcal{D}$ **do**
- 2: $\mathcal{L}_\mathcal{B} \leftarrow 0$.
- 3: **for each** question $q \in \mathcal{B}$ **do**
- 4: Construct $\mathcal{S}_q^+$, $\mathcal{S}_q^-$, and $\mathcal{S}_b^-$.
- 5: $\mathbf{h}_q \leftarrow E_\theta(q)$.
- 6: $\mathcal{H}_s^+ \leftarrow \{\mathbf{h}_s \mid \mathbf{h}_s = E_\theta(s), s \in \mathcal{S}_q^+\}$.
- 7: $\mathcal{H}_s^- \leftarrow \{\mathbf{h}_s \mid \mathbf{h}_s = E_\theta(s), s \in \mathcal{S}_q^- \cup \mathcal{S}_b^-\}$.
- 8: Compute $\mathcal{Z}_q$ by Eq. (6).
- 9: Compute $\mathcal{L}$ by Eq. (7).
- 10: $\mathcal{L}_\mathcal{B} \leftarrow \mathcal{L}_\mathcal{B} + \mathcal{L}$.
- 11: **end for**
- 12: Update θ by minimizing $\mathcal{L}_\mathcal{B}/|\mathcal{B}|$.
- 13: **end for**
- 14: **return** $E_\theta(\cdot)$.

Method	WebQSP		CWQ	
	Hit@1	F1	Hit@1	F1
Our [*]	92.2 $\pm$ 0.3	78.6 $\pm$ 0.5	87.9 $\pm$ 0.6	68.2 $\pm$ 0.5
Our [†]	93.3 $\pm$ 0.1	79.5 $\pm$ 0.3	89.6 $\pm$ 0.1	74.1 $\pm$ 0.2

Table 8: Mean and standard deviation over multi-runs.

question and cross-question negative super-paths.

D Additional Experiment**D.1 RQ1: Additional Main Results**

As shown in Table 8, SPaRC results are reported as the mean and sample standard deviation over three runs. Table 9 reports SPaRC results with different LLM backbones to support controlled same-backbone comparisons with existing methods.

D.2 RQ2: Effect of Super-Path Construction

RQ5: Path compression. The results in table 10 show that super-path construction substantially reduces the size of the candidate path set. On WebQSP, the average number of paths decreases from 2391.1 to 45.5, achieving about $52.6\times$ compression. On CWQ, it decreases from 1908.9 to 33.9, achieving about $56.3\times$ compression. This indicates that many instantiated paths share the same anchor entity and relation sequence, and super-paths can effectively remove such redundant path enumerations while preserving the main relational structure.

LLM	WebQSP		CWQ	
	Hit@1	F1	Hit@1	F1
GPT-4	93.1	82.7	89.3	79.2
GPT-4o	92.9	83.3	88.3	79.8
ChatGPT	92.4	84.5	87.1	79.7
GPT-4o-mini	93.4	79.7	89.6	74.2
Llama-3.1-8B	92.5	78.1	87.4	68.6

Table 9: Performance of SPaRC with different LLM backbones.

Dataset	Type	Avg. #	Avg. Tok.	F1	Hit
WebQSP	Path	2391.1	53479.8	66.3	86.0
	Super-path	45.5	4864.7	78.1	92.5
	GT super-path	1.8	113.6	—	—
CWQ	Path	1908.9	42910.8	58.6	85.8
	Super-path	33.9	3011.5	68.6	87.4
	GT super-path	3.2	58.3	—	—

Table 10: Effect of super-path construction under Llama-3.1-8B backbones. Avg. # and Avg. Tok. denote the average number of paths and KG tokens, respectively. GT super-path only reports Oracle statistics.

RQ5: Token reduction. Super-path construction also greatly reduces KG token consumption. On WebQSP, the average token number decreases from 53479.8 to 4864.7, reducing the context size by about $11.0\times$. On CWQ, it decreases from 42910.8 to 3011.5, reducing tokens by about $14.2\times$. The oracle statistics further show that ground-truth super-paths are highly compact, with only 113.6 and 58.3 tokens on WebQSP and CWQ, respectively. This suggests that the useful reasoning context is much smaller than the raw retrieved path set.

Reasoning performance. Despite using far fewer paths and tokens, super-paths improve final reasoning performance. On WebQSP, F1 increases from 66.3 to 78.1, and Hit increases from 86.0 to 92.5. On CWQ, F1 increases from 58.6 to 68.6, and Hit increases from 85.8 to 87.4. These results indicate that super-path construction does not merely reduce input length but also improves context quality by removing redundant paths and providing a more compact path representation for subsequent ranking and verification.

D.3 RQ2: Super-Path Granularity Analysis

Since triples, relation sequences, paths, and super-paths differ substantially in retrieval scale, we evaluate them by retaining candidates whose predicted probability exceeds 0.6 rather than using a fixed top-k cutoff. Table 11 reports recall under these four retrieval granularities. Here, triples are col-

Retrieval granularity	WebQSP	CWQ
Triples	73.9	51.4
Relation sequence	80.0	75.1
Paths	85.6	86.8
Super-paths	97.2	95.6

Table 11: Recall comparison across different retrieval granularity under the same probability threshold.

Method	Accuracy (%)
CoT (Wei et al., 2022)	42.9
ToG (Sun et al., 2024)	50.2
VoG (Zhao et al., 2026)	57.4
PPR (Xiao et al., 2026)	69.1
Our	73.6

Table 12: Accuracy comparison of different methods on the QALD10-en dataset.

lected through bounded multi-hop expansion over the KG; relation sequences are obtained by deduplicating the relations appearing in these triples; paths refer to instantiated multi-hop sequences; and super-paths are formed by compressing paths with the same anchor entity and relation sequence. The results show that recall increases consistently from triples to super-paths on both WebQSP and CWQ. This suggests that, compared with isolated triples or relation-level representations, path-based retrieval better preserves multi-hop reasoning information, while super-paths further reduce redundancy among instantiated paths and retain the information most relevant to subsequent ranking and reasoning.

D.4 RQ1: Generalization on Wikidata

To evaluate the generalization ability of SPaRC beyond Freebase, we further conduct experiments on the QALD10-en (Borrito et al., 2022) benchmark over Wikidata. As shown in Table 12, SPaRC achieves the best accuracy of 73.6%, outperforming the strongest baseline PPR by 4.5 percentage points. These results demonstrate the generalization ability of SPaRC beyond Freebase-based benchmarks.

E Failure Analysis and Case Study

We provide qualitative case studies and a detailed failure analysis. This section addresses the following research question:

RQ6. What insights can we obtain from case studies and failure analysis?

Failure Analysis. The remaining errors come from two sources. As shown in Table 13, first, com-

Dataset	AR		PM		CM	
	#	Err. (%)	#	Err. (%)	#	Err. (%)
WebQSP	35	28.5	80	65.0	8	6.5
CWQ	165	37.6	266	61.0	6	1.4

Table 13: Failure analysis on WebQSP and CWQ. AR denotes answer reasoning failure, PM denotes partial path missing, and CM denotes complete path missing.

Error Type	#	Err. (%)
Answer Reasoning Failure	92	20.9
Constraint Error	73	16.7
Path Missing	272	62.4

Table 14: Error Attribution on CWQ.

plete path missing is the dominant failure type on both WebQSP and CWQ, accounting for 65.0% and 61.0% of all errors, respectively. When the retrieved path set contains no correct answer, subsequent ranking, verification, and LLM reasoning have little opportunity to recover the result. In contrast, partial path missing contributes only a small fraction of failures on both datasets, indicating that incomplete coverage in multi-answer cases is not the primary source of residual errors. Finally, answer reasoning failure occupies a considerable proportion, especially on CWQ, where it rises to 37.6%. This shows that even when all correct answers are already covered by the retrieved paths, the LLM may still fail to derive the correct answer, suggesting that reasoning over complex retained contexts remains challenging.

To further localize these errors, we conduct a finer-grained attribution of 436 unsuccessful CWQ cases. Table 14 decomposes the previous answer reasoning failure category into constraint errors and final answer reasoning failures, which account for 16.7% and 20.9%, respectively. This distinction shows that some failures after correct paths are retrieved arise during constraint-based reranking rather than final LLM reasoning.

Case study: interpretable reasoning. We conduct a series of case studies to qualitatively examine the interpretability of the SPaRC framework. Tables 15 and 16 present representative successful cases, while Tables 17 and 18 illustrate cases that reveal the limitations of SPaRC.

The successful cases show that SPaRC can provide an interpretable reasoning process from ranked super-paths to constraint-verified paths and final answer generation. In both cases, the contrastive

ranker places the required super-paths within the top-ranked candidates, ensuring that the essential relational structure is preserved before LLM reasoning. Constraint verification then further refines the ranked super-paths by retaining those that better satisfy the question constraints and filtering out less relevant paths. As a result, the LLM receives a more focused reasoning context in which the key supporting paths are clearly presented rather than obscured by a large set of loosely related paths. These cases illustrate how super-path ranking and constraint verification jointly improve the usability of retrieved KG information and make the supporting basis for the final answer clearer.

The failure cases reveal two representative sources of error. In retrieval failure cases, the required super-path is not retained among the ranked candidates, so later verification and LLMs reasoning cannot recover the missing reasoning fact. In reasoning failure cases, the relevant super-path has already been retrieved and retained after verification, but the LLM still produces an incorrect or incomplete answer from the provided context. This indicates that SPaRC effectively improves path selection, but final answer quality can still be affected by the completeness of retrieved super-paths and the LLM’s ability to correctly use the verified context. Overall, these case studies complement the quantitative results by illustrating how SPaRC supports interpretable multi-hop reasoning and by revealing its remaining limitations.

F Future Work

In future work, we plan to extend SPaRC in two directions. First, we will study incomplete KGR settings, where necessary entities or relations may be missing from the KG, and explore how super-path construction can be combined with link prediction or external textual evidence to improve path coverage. Second, we will investigate more adaptive path refinement strategies, such as dynamically adjusting retrieval depth, top- k , and verification strength according to question complexity, so that SPaRC can construct suitable reasoning contexts for different types of multi-hop questions.

G Prompt

We provide the complete prompt templates for all three LLM calls used in SPaRC, covering constraint extraction in Fig. 7, constraint verification in Fig. 8, and answer reasoning in Fig. 9, respectively.

Field	Content
Question	What type of cancer did Gilda Radner die of
Answer	Ovarian cancer
Topic Entity	Gilda Radner, Cancer
Ground-Truth Paths	Gilda Radner → people.deceased_person.cause_of_death → Ovarian cancer Cancer → people.cause_of_death.parent_cause_of_death → Ovarian cancer
Ranked Top-5 Super-Path	Cancer → people.cause_of_death.parent_cause_of_death Gilda Radner → people.deceased_person.cause_of_death Gilda Radner → medicine.notable_person_with_medical_condition.condition Gilda Radner → people.person.sibling_s Gilda Radner → people.person.spouse_s
Constraint Description	The answer is a disease The head entity is a person
Constraint Verification Super-Path	Cancer → people.cause_of_death.parent_cause_of_death → Ovarian cancer Gilda Radner → people.deceased_person.cause_of_death → Ovarian cancer Gilda Radner → medicine.notable_person_with_medical_condition.condition → Bulimia nervosa
Answer Reasoning	Ovarian cancer

Table 15: A representative successful case study from the WebQSP dataset.

Field	Content
Question	What European Union country sharing borders with Germany contains the Lejre Municipality?
Answer	Denmark
Topic Entity	Lejre Municipality, Country, Germany
Ground-Truth Paths	Lejre Municipality → location.country.administrative_divisions → Denmark Germany → location.location.adjoin_s, location.adjoining_relationship.adjoins → Denmark Country → common.topic.notable_types → Denmark
Ranked Top-5 Super-Path	Country → common.topic.notable_types Lejre Municipality → location.country.administrative_divisions Germany → location.location.adjoin_s, location.adjoining_relationship.adjoins Germany → base.aareas.schema.administrative_area.administrative_children Germany → location.country.administrative_divisions
Constraint Description	The answer is a country The location is a municipality
Constraint Verification Super-Path	Country → common.topic.notable_types → Denmark Germany → location.location.adjoin_s, location.adjoining_relationship.adjoins → Denmark Germany → base.aareas.schema.administrative_area.administrative_children → Bremen, Berlin, and 14 other candidate answers Germany → location.country.administrative_divisions → Dortmund, and 28 other candidate answers
Answer Reasoning	Denmark

Table 16: A representative successful case study from the CWQ dataset.

Field	Content
Question	What year was the first Miss America pageant held
Answer	1921
Topic Entity	Miss America
Ground-Truth Paths	Miss America → time.recurring_event.date_of_first_occurrence → 1921
Ranked Top-5 Super-Path	Miss America → time.recurring_event.instances Miss America → time.recurring_event.current_frequency Miss America → award.recurring_competition.individual_competitions Miss America → common.topic.notable_types Miss America → organization.organization.headquarters
Constraint Description	The answer is a year
Constraint Verification Super-Path	Miss America → time.recurring_event.current_frequency → Yearly Miss America → time.recurring_event.instances → Miss America 2007, Miss America 2006, and 93 other candidate answers Miss America → award.recurring_competition.individual_competitions → Miss America 2001, 1964 Miss America, and 7 other candidate answers Miss America → common.topic.notable_types → Recurring competition
Answer Reasoning	1932, 1930

Table 17: A representative retrieval failure case study from the WebQSP dataset.

Field	Content
Question	What is the name of the mascot by the university which includes Syracuse University Orange Football
Answer	Syracuse University Otto the Orange
Topic Entity	Syracuse Orange football
Ground-Truth Paths	Syracuse Orange football → education.educational_institution.sports_teams, education.educational_institution.mascot → Syracuse University Otto the Orange
Ranked Top-5 Super-Path	Syracuse Orange football → education.educational_institution.sports_teams, education.educational_institution.mascot Syracuse Orange football → sports.sports_team.arena_stadium Syracuse Orange football → sports.sports_team.colors Syracuse Orange football → sports.school_sports_team.school Syracuse Orange football → sports.school_sports_team.athletics_brand
Constraint Description	The answer is a university The answer is a sports team
Constraint Verification Super-Path	Syracuse Orange football → education.educational_institution.sports_teams, education.educational_institution.mascot → Syracuse University Otto the Orange Syracuse Orange football → sports.sports_team.arena_stadium → Carrier Dome Syracuse Orange football → sports.sports_team.colors → Blue, White, Orange Syracuse Orange football → sports.school_sports_team.athletics_brand → Syracuse Orange Syracuse Orange football → sports.school_sports_team.school → Syracuse University
Answer Reasoning	Otto the Orange, Syracuse Orange

Table 18: A representative reasoning failure case study from the CWQ dataset.

Prompt Template for Constraint Extraction
System Instruction: You are a constraint analyzer for a knowledge graph QA system. Given a natural language question, extract constraints mentioned in the question. Constraints are of the following types: entity_type: Specifies the category of any entity mentioned in the question or the target answer. relation: Captures a relational constraint in the question. temporal: Involves a time condition that applies to any part of the question. count: Involves a numerical quantity, comparison, or ordering relevant to any part of the question. Each constraint must be a JSON object containing exactly the following three fields: constraint_type: string, one of entity_type, relation, temporal, count. description: a short natural language description of the constraint. value: the concrete value of the constraint. Output Format: Output a JSON array of constraints extracted from the question. Do not introduce constraints beyond the defined constraint types. User: Given Question: <i>q</i>

Figure 7: Prompt template used for constraint extraction.

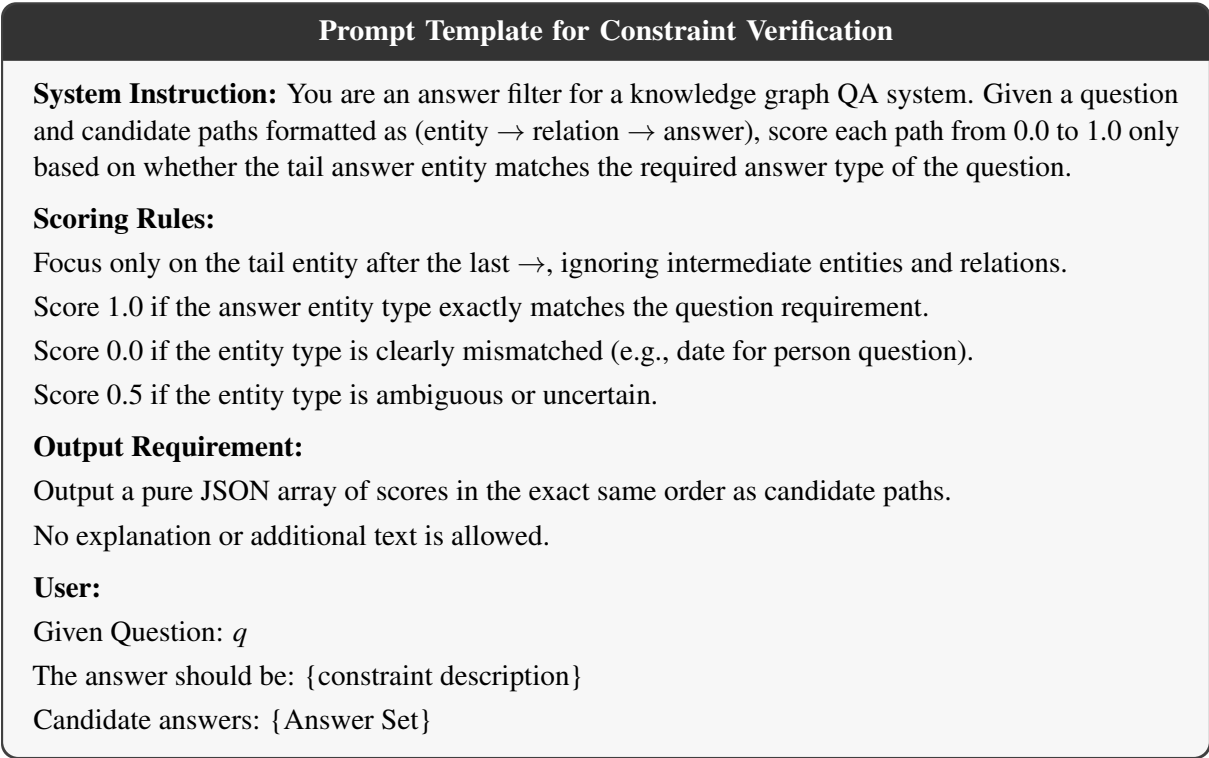


Figure 8: Prompt template used for constraint verification.

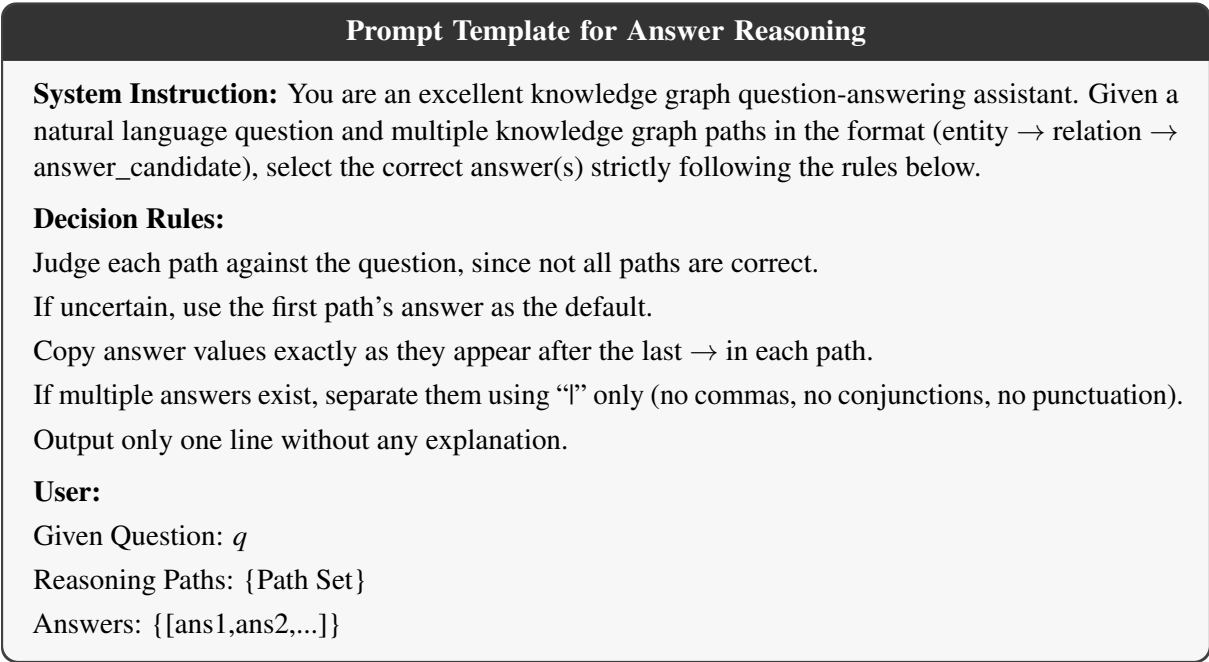


Figure 9: Prompt template used for answer reasoning.