Faico: Faithful and Complete Knowledge Graph Augmented Reasoning

Guo Cheng
guo_cheng@bit.edu.cn
Beijing Institute of Technology
Beijing, China
QiYuan Lab
Beijing, China

Kangfei Zhao*
zkf1105@gmail.com
Beijing Institute of Technology
Beijing, China

Ke Ye
keye@bit.edu.cn
Beijing Institute of Technology
Beijing, China

Pengpeng Qiao
peng2qiao@gmail.com
Institute of Science Tokyo
Tokyo, Japan

Zhiwei Zhang
zwzhang@bit.edu.cn
Beijing Institute of Technology
Beijing, China

Saiguang Che
stonnyaiqi@gmail.com
QiYuan Lab
Beijing, China

Shaonan Ma
msn18@tsinghua.org.cn
QiYuan Lab
Beijing, China

Mingxing Zhang
zhang_mingxing@mail.tsinghua.cn
Tsinghua University
Beijing, China

Abstract

Large language models (LLMs) augmented with knowledge graphs (KGs) have exhibited great potential for complex reasoning tasks. However, existing approaches often struggle with incomplete subgraph retrieval and inaccurate semantic alignment, which hinder reasoning performance and answer quality. In this paper, we present Faico, a KG-enhanced reasoning framework designed to achieve both semantic faithfulness and structural completeness. Faico decouples model inference from graph traversal by integrating a fine-tuned LLM-based relation type generator for accurate semantic mapping and a KG retriever for reasoning subgraph search. Based on the predicted relation types, we model the reasoning subgraph (RS) as a k-bounded edge type (k-BET) subgraph, where k constrains the recurrence of relation types within paths, and devise a budget-dominance-based algorithm to efficiently identify the maximal k-BET subgraph. Our framework ensures comprehensive coverage of relevant multi-hop relations while reducing computational overhead. Through extensive experiments on multiple KGQA benchmarks, Faico demonstrates improvements in both effectiveness and efficiency over LLM-native and state-of-the-art KG-augmented reasoning baselines, delivering more accurate, complete answers and lower inference latency.

CCS Concepts

• **Computing methodologies** → **Knowledge representation and reasoning**.

Keywords

Knowledge Graph Question Answering, Schema-Constrained Decoding, Graph Search

ACM Reference Format:
Guo Cheng, Kangfei Zhao, Ke Ye, Pengpeng Qiao, Zhiwei Zhang, Saiguang Che, Shaonan Ma, and Mingxing Zhang. 2026. Faico: Faithful and Complete Knowledge Graph Augmented Reasoning. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1 (KDD '26), August 09–13, 2026, Jeju Island, Republic of Korea.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3770854.3780336

KDD Availability Link:
The source code of this paper has been made publicly available at https://doi.org/10.5281/zenodo.17935127.

*Corresponding author.

KDD '26, Jeju Island, Republic of Korea

ACM ISBN 979-8-4007-2258-5/2026/08
https://doi.org/10.1145/3770854.3780336

1 Introduction

Large language models (LLMs) have demonstrated remarkable capabilities in a wide range of natural language tasks [14, 22, 23, 31, 45]. Despite impressive performance, LLMs are fundamentally limited by outdated internal knowledge and a propensity to hallucinate, producing plausible but incorrect responses [12, 20, 44]. In contrast to unstructured text, knowledge graphs (KGs) [6, 8, 19, 30, 37] provide a structured and semantically rich representation of facts, explicitly encoding entities and their interrelations. This explicit structure not only supplements LLMs with external, verifiable knowledge, but also naturally supports complex multi-hop and multi-entity reasoning [26, 29, 32]. As a result, the use of KGs has emerged as an effective paradigm for enhancing the factuality, interpretability, and depth of reasoning in LLM-based systems.

Existing KG-augmented reasoning approaches [1, 27, 33, 40] typically follow a two-step process: extracting a reasoning subgraph (RS) from the underlying KG, and subsequently leveraging an LLM to infer the answer from this subgraph. These approaches can be broadly categorized into two paradigms, *Model Pre-select* [7, 17, 26, 32] and *Model Post-filter* [18, 35]. Model Pre-select

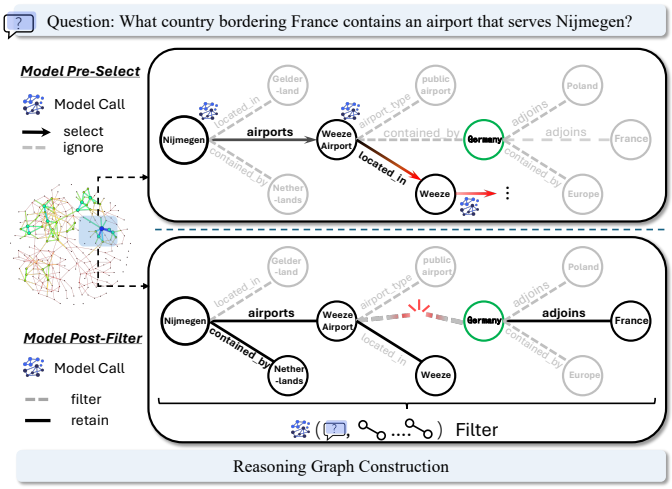


Figure 1: Analysis over current graph reasoning methods. The correct answer (Germany) is marked with a green circle, and error sources are marked in red.

paradigm initiates graph traversal from topic entities identified in the query, employing an LLM or vector similarity search to expand the search path progressively. This process forms a reasoning subgraph, which is then fed into an LLM along with the original question to generate the answer [3, 15, 21]. This paradigm suffers from accumulated errors inherent in cascading LLM calls, as well as substantial computational overhead. Its reliance on inflexible pruning heuristics further exacerbates the issue, leading to incomplete coverage of potential answer candidates within the Knowledge Graph. In contrast, the Model Post-filter paradigm first employs embedding-based traversal to retrieve a large candidate subgraph from the KG, and then uses an LLM or lightweight model to filter out semantically irrelevant information [25, 35]. The post-filtering can disrupt coherent logical chains in the resulting subgraph, often leading to incomplete reasoning chains and leaving the LLM without full context for inference.

A common problem in both paradigms is the excessive coupling between LLM inference and subgraph retrieval [10, 24]. The over-entanglement leads to reasoning subgraphs that are semantically unfaithful, failing to fully capture the intent of the question, or structurally incomplete, missing important logical connections. For Model Pre-select approaches such as ToG [32], sequential invocations of an LLM to select edge types during breadth-first search in the KG can cause cascading errors. Once an incorrect edge type is chosen, the subsequent search follows an incorrect path, undermining semantic fidelity. For Model Post-filter [18, 35] approaches, a candidate subgraph is first constructed around the topic entities, subsequently, a lightweight model is employed to score and prune the subgraph, retaining only the top-k most relevant triples. However, only preserving a fixed-size subgraph may mistakenly prune critical edges in multi-hop reasoning, resulting in incomplete reasoning chains and answer misses.

To address these problems, we aim to design a KG-enhanced reasoning framework that fulfills semantic faithfulness and structural completeness simultaneously. The design faces three challenges: (1) *Decoupling model inference from KG retrieval.* Appropriately separating model invocation and graph algorithms to improve overall performance, balancing the dependency and flexibility of the two components. (2) *Accurate semantic alignment.* Precisely identifying the semantic relationships in the NL questions and mapping them to the complex hierarchical relation schema of the KG. (3) *Efficient and complete subgraph retrieval.* Large-scale KGs entail an efficient subgraph search algorithm, which also ensures that the retrieved subgraphs are structurally complete, i.e., capturing all relevant multi-hop relations without introducing redundancy. The three challenges are intertwined throughout the overall system design.

To this end, we propose a semantically Faithful and structurally complete KG-enhanced reasoning framework, named Faico, that effectively mines high-quality reasoning subgraphs in the KG and generates NL answers by LLMs. Faico employs an edge type generator and a KG retriever, individually specialized in semantic alignment and structure retrieval. For NL question, the generator is a fine-tuned LLM, predicting valid edge types subject to the underlying hierarchical schema of the KG. The produced edge types are utilized by the KG retriever to navigate search paths. Specifically, we model the target RS as a maximal k-bounded edge type (k-BET) subgraph, where k limits the recurrence of edge types in the subgraph. We devise a budget-dominance-based algorithm to efficiently identify the maximal k-BET subgraph. The contributions of this paper are summarized as follows:

- We analyze common failures of prior approaches and propose a new semantically faithful and structurally complete KG-augmented reasoning framework. Our framework decouples model invocation and graph algorithm for edge types mapping and reasoning subgraph retrieval individually.
- We design an edge type generator that leverages a fine-tuned LLM to decode valid edge types, under the routing of a token-trie of the KG schema. In addition, to obtain complete reasoning subgraphs, we formalize (maximal) k-BET subgraph and propose a budget-dominance-based algorithm with a node skipping strategy.
- We conduct extensive experiments to demonstrate the effectiveness and efficiency of the overall framework, as well as the accuracy and completeness of the generated edge types and the subgraph structures. Compared with two LLM-native and six KG-augmented reasoning baselines, Faico achieves 2% to 14.3% improvements in Hit@1 score on CWQ and WebQSP datasets.

2 Preliminaries & Analysis

A Knowledge Graph $\mathcal{G}$ is a directed labeled graph with a node (entity) set $\mathcal{V}$ and an edge (relation) set $\mathcal{E}$. Each relation e is associated with a relation type $\ell(e)$ as its label, which is from a label domain Σ. A KG can also be represented as a set of triples. Each triple (u, e, v) consists of head and tail entities $u, v \in \mathcal{V}$ and their relation $e \in \mathcal{E}$, denoting an item of factual knowledge in $\mathcal{G}$. Detailed notation definitions are provided in Appendix A.

KG-augmented LLM Reasoning. Given an NL query q, the initial step of KG-augmented LLM reasoning is to identify a set of topic entities directly mentioned by q, denoted by T_q, which can be implemented by entity linking models [16, 41]. Retrieval in $\mathcal{G}$ starts from these topic entities, where triples of semantic relevance

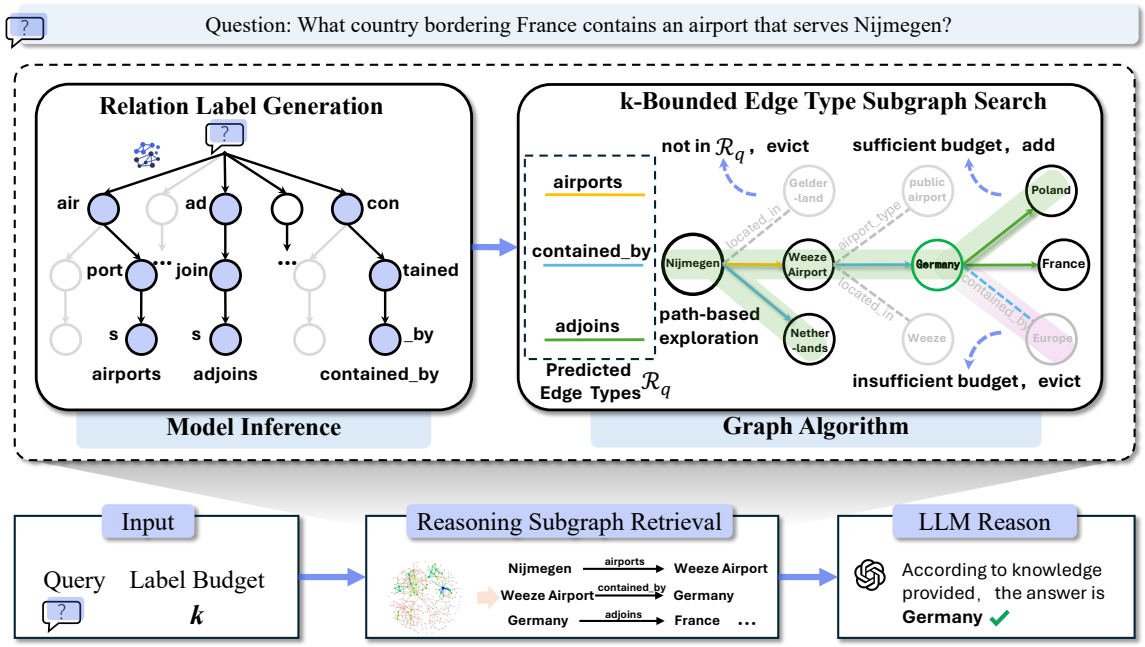


Figure 2: Overview of the knowledge graph reasoning framework of Faico.

and structural connectivity are extracted as a RS. In this paper, we focus on finding high-quality RS in $\mathcal{G}$ efficiently. Semantically, an ideal RS should cover all relation types implied by the question. Structurally, it should include valid paths in the KG from topic entities to potential answers, forming complete logical chains for LLM reasoning.

Existing approaches adopt different strategies to identify an RS; however, they still struggle to find an ideal RS. Figure 1 presents the typical failures of existing approaches on an exemplar question: '*What country bordering France contains an airport that serves Nijmegen?*'. For Model Pre-select approaches [7, 17, 26, 32] (at the top of Figure 1), an LLM is invoked to select relation types (e.g., `airports`, `contained_by`, `adjoins`), but cumulative errors hamper the model to select the crucial `adjoins` relation from Germany to France, leading to the absence of the answer, Germany, in the final RS. For Model Post-filter approaches [18, 35] (at the bottom of Figure 1), a larger candidate subgraph is retrieved and then a lightweight classifier is employed to prune plausible edges. Although all relevant relation types and the answer entity Germany are preserved in the RS, Figure 1, the relation `contained_by` from Weeze Airport to Germany is mistakenly pruned, leading to the break of the reasoning chain. Consequently, the LLM will lack a complete provenance of the answer in the reasoning step.

Thereby, we summarize that an ideal RS should simultaneously satisfy two fundamental characteristics as follows:

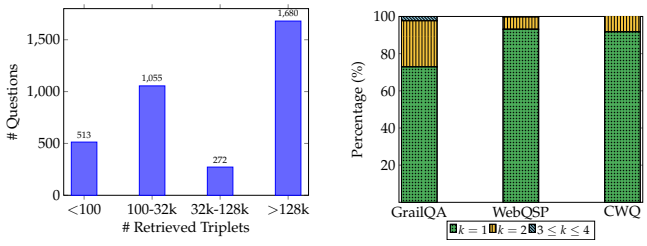
- **Semantic Faithfulness.** The RS should contain all the relation types in $\mathcal{G}$ that are semantically relevant to the question.
- **Structural Completeness.** The RS should contain complete reasoning paths from topic entities in T_q to answer entities that exist in $\mathcal{G}$.

It is worth noting that these two characteristics are closely intertwined rather than independent. During the incremental search process, the absence of key relation types can result in incomplete reasoning paths, while missing critical structural connections may, in turn, lead to insufficient semantic coverage. Such mutual dependency makes it difficult to guarantee either property in isolation. Despite the identified topic entities, searching for RS of semantic coverage and structural completeness in a large KG is non-trivial. In this paper, we present a relatively decoupled framework that divides semantics and structures and conquers individually, aiming to prevent error entanglement and to ensure more faithful and complete reasoning subgraphs for downstream LLM-based reasoning

3 Overview & Reasoning Subgraph Modeling

We propose a two-stage reasoning subgraph search framework that first predicts a set of relevant relation types $\mathcal{R}_q$ for question q and then searches a complete RS in the KG based on $\mathcal{R}_q$. Our design principle is to leverage advances in LLMs for semantic understanding and in graph algorithms for symbolic retrieval, enabling precise and comprehensive navigation of semantic relations and structures in two sequential stages.

Figure 2 delineates an overview of Faico. Specifically, in the first stage, Faico employs a fine-tuned LLM to generate a set of relation types $\mathcal{R}_q$ for a given question q. To generate valid relation types, we represent the hierarchical relations schema of the KG as a token-level prefix tree, a token-trie, which explicitly guides the LLM to decode valid relation types in the KG. We propose a trie-based decoding algorithm that explores multiple semantic branches within the trie while progressively pruning less relevant paths. The LLM serves as a relation-type generator and is deployed in an offline setting by token-trie guided parameter fine-tuning.



(a) Graph Size Analysis of Naive Graph Traversal

(b) Distribution of Relation Label Counts

Figure 3: Empirical statistics on (a) reasoning subgraph size and (b) relation label count in different datasets.

In the second stage, Faico retrieves an RS that encompasses the generated relation types $\mathcal{R}_q$, where each path in the RS connects to at least one of topic entities in T_q. We model the structure of RS as a k-bounded edge type (k-BET) subgraph and propose an efficient algorithm with a node pruning strategy to build the RS given $\mathcal{R}_q$. Finally, Faico issues an LLM call, prompting triples of the RS, alongside the question q, to generate a faithful and sound response. In the following, we formulate the structure of the k-BET RS in §3.1, and elaborate on the algorithms of RS retrieval and relation type generation in §4 and §5, respectively.

3.1 Reasoning Subgraph Formulation

Given topic entities T_q, identifying a structurally complete RS is a non-trivial task. Although the semantics of question q are restricted to a set of relation types $\mathcal{R}_q$, its search space grows exponentially with respect to search depth d. Figure 3a illustrates an empirical study on the CWQ benchmark [34]. Here, we conduct a naive BFS search starting from T_q in the Freebase KG [4]. For 47.72% of questions (1,680 out of 3,531), the number of retrieved triples surpasses 128k, with a search depth $d = 5$, which not only incurs a high latency for KG retrieval but also exceeds the context window of LLMs. Existing approaches [7, 32] set the search depth as a fixed value. However, a fixed search depth cannot generalize well to different questions.

To reduce the search space, we propose a new subgraph model for reasoning subgraphs in the KG, which is based on the observation that few NL questions contain repetitive relation types in one question. In other words, for any relation type $\sigma \in \mathcal{R}_q$, the question q does not involve σ multiple times. Figure 3b illustrates the distribution of the maximum occurrence of σ in the questions, denoted as k, from three widely used datasets. This indicates that most relation types occur at most twice in a question. This observation motivates us to use the occurrence of σ to limit the search space of RS, without compromising the expressiveness of the RS.

Based on the observation, we formalize k-bounded edge type (k-BET) path as a potential reasoning path in the RS, where the edge types fall in $\mathcal{R}_q$ and the cardinality of each edge type is limited by a constant k.

Definition 1 (k-bounded Edge Type Path). *Given a relation type set $\mathcal{R}_q$, a path $\left[v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \cdots \xrightarrow{e_l} v_l\right]$ in a KG is a k-bounded edge type path if $\forall e \in \{e_1, \cdots, e_l\}, \ell(e) \in \mathcal{R}_q$ and $|\ell(e) = \sigma| \leq k$.*

We formulate the RS of a question q as a k-bounded edge type (k-BET) subgraph where the subgraph encompasses the topic entities T_q and each node in the subgraph connects to at least one topic entity via k-BET paths.

Definition 2 (k-bounded Edge Type Subgraph). *Given a topic entity set T_q and a relation label set $\mathcal{R}_q$, a k-bounded edge type subgraph $G' = (V', E')$ satisfies $T_q \subset V'$, and $\forall v \in V' \setminus T_q$, v is reachable to a node $v' \in T_q$ or is reachable from $v' \in T_q$ via a k-BET path.*

In this paper, our goal is to identify the *maximal* k-BET subgraph that cannot be expanded by adding nodes and edges. This ensures that all the entities subject to the semantics of the questions are covered in the RS.

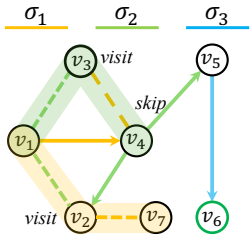
Example 1. *The RS in Figure 2 is a maximal k-BET subgraph for $k = 1$. Specifically, given the topic entity $T_q = \{Nijmegen\}$ and the relation types $\mathcal{R}_q = \{\texttt{airport}, \texttt{contained\_by}, \texttt{adjoins}\}$, other nodes $\{Weeze\ Airport, Netherlands, Germany, France, Poland\}$ connect to Nijmegen by k-BET paths where each relation type appears only once. The subgraph is maximal since adding other nodes such as Europe will violate the cardinality constraint of relation types.*

4 Reasoning Subgraph Retrieval

In this section, we concentrate on the retrieval algorithm for maximal k-BET subgraphs, suppose T_q and $\mathcal{R}_q$ are given. Recall that the formulation of k-BET subgraph highlights the reachability between topic entities to other nodes, under the constraint of the cardinality of edge types in $\mathcal{R}_q$. And the maximal k-BET subgraph underscores an exhaustive search of all k-BET paths. The gist of this subgraph search algorithm, distinguished from general reachability query algorithm, is to leverage the cardinality to narrow down the search space. Specifically, retrieving a k-BET subgraph cannot be implemented by simply marking vertices as visited (as in standard DFS or BFS). A vertex may need to be revisited under different paths that correspond to different remaining budgets. For example, in Figure 4, starting from a topic entity v_1, previous traversal has visited nodes $\{v_1, v_2, v_4, v_5, v_6\}$. When we trace back to v_1, the exploration to v_2 via edge $v_1 \xrightarrow{\sigma_2} v_2$ cannot be simply pruned although v_2 has already been visited. That is because path $[v_1 \xrightarrow{\sigma_1} v_2 \xrightarrow{\sigma_2} v_7]$ (highlighted in yellow) is also a k-BET path that needs exploring.

To achieve fine-grained control of exploration, we introduce the budget of edge types, B, regarding the currently visited node, to represent the usage of bounded edge types in traversal on a k-BET path. The budget of a node v records the remaining budget of each edge type $\sigma \in \mathcal{R}_q$ that can be used to further expand the k-BET path. Example 2 presents an example on this concept.

Example 2. *In Figure 4, given $\mathcal{R}_q = \{\sigma_1, \sigma_2, \sigma_3\}$, $T_q = \{v_1\}$ and $k = 1$, nodes $\{v_2, v_4, v_5, v_6\}$ have already been explored from v_1 via graph traversal. When we trace back to v_1, visiting v_3 and v_4 via path $P = [v_1 \xrightarrow{\sigma_2} v_3 \xrightarrow{\sigma_1} v_4]$ (highlighted in green), the current budget of P is $B = \{\sigma_1 : 0, \sigma_2 : 0, \sigma_3 : 1\}$. Considering the previous path to v_4 $P' = [v_1 \xrightarrow{\sigma_1} v_4]$ with a budget $B' = \{\sigma_1 : 0, \sigma_2 : 1, \sigma_3 : 1\}$, we have $B(\sigma) \leq B'(\sigma)$ for all $\sigma \in \mathcal{R}_q$. After adding the k-BET path P into the RS, it is unnecessary to search the neighbors of v_4 to further extend P, since the previous path P' provides a larger search space.*



Explored Unvisited

v	Budget B		
	σ_1	σ_2	σ_3
v_2	0	0	1
v_4	0	1	1
v_5	0	0	1
v_6	0	0	0

Figure 4: Node skipping by budget dominance (k=1).

Algorithm 1: budget-dominance-based k-BET RS Retrieval

Input: Topic entities T_q, predicted relation types $\mathcal{R}_q$, KG $\mathcal{G}$, relation type bound k.
Output: reasoning subgraph $\mathcal{G}_r$.

1 Initialize $\mathcal{S} \leftarrow$ Stack(), $\mathcal{G}_r \leftarrow \emptyset$, $\mathcal{B} \leftarrow \emptyset$;
2 **for** $v_t \in T_q$ **do**
3 $B \leftarrow \{\sigma : k \mid \sigma \in \mathcal{R}_q\}$;
4 $\mathcal{S}$.push((v_t, B)); ▷ *traverse from a topic entity* v_t
5 **while** $\mathcal{S}$ *is not empty* **do**
6 $(v, B) \leftarrow \mathcal{S}$.pop() ;
7 **for** $(v, e, v') \in$ GetTriples($\mathcal{G}, v$) *and* $\ell(e) \in \mathcal{R}_q$ *and* $B(\ell(e)) > 0$ **do**
8 **if** $(v, e, v') \notin \mathcal{G}_r$ **then**
9 $\mathcal{G}_r \leftarrow \mathcal{G}_r \cup \{(v, e, v')\}$; ▷ *save the triple*
10 $\{B'_1, B'_2, \cdots\} \leftarrow \mathcal{B}$.find($v'$); ▷ *get previous budgets*
11 **if** $\exists B'_i \in \{B'_1, B'_2, \cdots\}, B \preceq B'_i$ **then**
12 **continue**; ▷ *bypass* v' *by budget dominance*
13 $B(\ell(e))$ −= 1; ▷ *update current budget*
14 $\mathcal{S}$.push((v', B));
15 $\mathcal{B}$.insert(v', B); ▷ *update previous budget*
16 **for** $B'_i \in \{B'_1, B'_2, \cdots\}$ *and* $B'_i \preceq B$ **do**
17 $\mathcal{B}$.delete(v', B'_i);
18 **return** $\mathcal{G}_r$

As Example 2 demonstrates, recording the budget along the search of k-BET path provides the opportunity to early stop the exploration on nodes. We formulate the dominance between the budgets of two k-BET paths as below. In brief, if the budget of a k-BET path to node v is dominated by the budget of another path, searching along this path at node v can be stopped.

Definition 3 (Budget dominance). *For two k-BET paths P, P' connecting a topic entity to a node v, their respective budgets regarding $\mathcal{R}_q$ are denoted as B, B'. If $\forall \sigma \in \mathcal{R}_q, B(\sigma) \leq B'(\sigma)$ satisfies, we say B is dominated by B', denoted as $B \preceq B'$.*

Algorithm 1 presents the budget-dominance-based k-BET subgraph retrieval algorithm with node skipping by budget dominance. Initially, we use a stack $\mathcal{S}$ to maintain the search frontier and a data structure $\mathcal{B}$ to persist the budgets of edge types for each visited node, indexed by node id, where each budget corresponds to a path to the node (line 1). In line 4, the traversal starts from a topic entity v_t with the initial budget k for all $\sigma \in \mathcal{R}_q$. Given the search frontier, node v and its current budget B in line 6, the algorithm traverses its adjacency edges with types in $\mathcal{R}_q$ and subject to the current budget (line 5-17). The edge (triple) that forms a k-BET path is added to the RS in line 9. We will skip the traversal along this edge if for the neighbor of v, v', there is a historical k-BET path with budget B'_i dominating the current budget B (line 10-12). Otherwise, the algorithm goes deeper by pushing v' and the updated budget B into the stack. In line 15-17, we maintain the data structure $\mathcal{B}$ by updating possible historical budgets of v' which are dominated by B, to keep the pruning capability of $\mathcal{B}$. The algorithm iterates over all the topic entities until there are no nodes to be visited recursively.

Assume the final RS $\mathcal{G}_r$ contains $|V_r|$ nodes, with each node $v \in V_r$ having at most Δ out-neighbors, the time complexity of Algorithm 1 is $O\left(|T_q| \cdot |V_r| \cdot \Delta^{|\mathcal{R}_q| \times k}\right)$. The RS retrieved by Algorithm 1 is a maximal k-BET subgraph, and the corresponding proof is provided in Appendix B.

5 Trie-based Relation Type Learning

To generate semantically faithful relation types, two criteria must be satisfied. First, predicted edges must be *schema-valid* to avoid hallucinated relation types not present in the KG schema, which would lead to invalid traversal paths. Second, the generated set must achieve *semantic coverage*, capturing all relation types relevant to the query.

However, these requirements face two key challenges. First, hallucination occurs when the model generates relation types that do not exist in the underlying KG schema, breaking downstream reasoning. Second, incomplete relation types arise due to the vast and long-tailed vocabulary of relation types, mapping context to a precise label within such a massive and skewed output space poses a significant generation challenge[2, 39]. This mismatch prevents the LLM from identifying all relevant edges, especially those in domain-specific or low-frequency edge types, ultimately leading to incomplete subgraphs and unreachable answers.

5.1 Relation Type Generation

To mitigate this mismatch, we introduce a token-trie based relation type generation mechanism that constrains the model's output space to the predefined vocabulary of KG relation types. The token-level hierarchical structure of the edges in a KG is represented in a trie $\mathcal{T}$, encapsulating the constraints of the generated types that Faico aims to leverage. We deploy a fine-tuned LLM M to predict the involved relation types for a query q, which composes reasoning paths following the token-trie. Specifically, given the token space of the relation types $\mathcal{R}$, the model predicts the next token of a candidate relation type conditioned on the original query q and the currently predicted tokens as a decoding path $p = [\tau_1, \tau_2, \cdots, \tau_n]$ following the auto-regression decoding process:

$$z = M(q, \text{decoding path} = p) \tag{1}$$

where $z \in \mathbb{R}^{|\mathcal{R}|}$ is the logits across the full token space. In each decoding step, we restrict the model to generate a valid token as a child in the trie of the last token of p. Here, the validity of tokens indicates that in the relation type hierarchy of the KG, the candidate relation type to be generated should be a child of the relation type

corresponding to the last token in p. To achieve validity, we impose a mask on the Softmax output for the logits z as shown in Eq. (2), thereby obtaining the next-token probability distribution regarding the current prefix p, denoted as $P(\tau \mid p, q)$.

$$P(\tau \mid p, q) = \begin{cases} \frac{\exp(z_\tau)}{\sum_{\tau' \in \mathcal{T}.\text{Child}(p[-1])} \exp(z_{\tau'})}, & \text{if } \tau \in \mathcal{T}.\text{Child}(p[-1]) \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Here, $p_{[-1]}$ denotes the last token in the current decoding path p, which corresponds to a specific node n in the token-trie $\mathcal{T}$. The term $\mathcal{T}.\text{Child}(p[-1])$ refers to the immediate child nodes of n, representing the valid next tokens. The generation process terminates when n is a leaf node in the trie.

Due to the intrinsic semantic ambiguity of the question, Faico will generate multiple tokens in parallel in one decoding step. For all valid tokens regarding the current decoding path p, we only preserve those tokens whose conditional probability surpasses a threshold θ. By the chain rule for language model decoding, Eq. (3) formulates the joint probability of a complete reasoning path p which contains the generated relation types. This pruning strategy ensures that the final set of relation types is both schema-valid and high-confidence.

$$P(p \mid q) = \begin{cases} \prod_{i=1}^{n} P(\tau_i \mid p_{[1:i-1]}, q), & \text{if } j < i, P(\tau_j \mid p_{[1:j-1]}, q) > \theta \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

Since one decoding step may possess multiple branches, all the decoding paths p form a hierarchical tree structure, from a special root node as the starting point.

Figure 5 illustrates an example of the relation type generation process for the question: *'What year did the team with mascot named Lou Seal win the World Series?'*. At the beginning, the fine-tuned model starts from a root token '?' and predicts two edge types: `sports` and `time`. Subsequently, the two types, serve as two prefixes that are decoded by the model and generate `sports_team` and `time_event`. Finally, in the last decoding step, three decoding paths are generated, i.e., `sports_team_mascot`, `sports_team_champion`, and `time_event_end_date`, which are organized into a tree structure as shown in Figure 5.

Algorithm 2 presents the generation algorithm that leverages a BFS with a beam width b and the θ-pruning strategy. Given a question q and the token-trie of the KG $\mathcal{T}$, the model generates a set of sequences $\mathcal{R}_q$, where each sequence is a concatenation of the generated relation types in a decoding path. We use a queue Q to maintain the search frontier, where an element contains the current visited token c, and the current decoding path p. At the beginning, we initialize the result $\mathcal{R}_q$ and the queue. In line 3-13, a BFS with a beam width b is performed. Specifically, if the current token c is a leaf node in the token-trie, we persist the generated decoding path p in $\mathcal{R}_q$ (line 5). Otherwise, the LLM generates the next node given p (line 7-13). First, the logits of the token are computed by the LLM and masked for validity check, by Eq. (1) and Eq. (2), respectively. Then, with the probability from the Softmax function (line 9), we select top-b tokens whose generative probability is greater than the threshold θ in line 10. Finally, the b tokens are inserted into the search frontier if they are children of the current token c in the trie in line 11-13. It is worth mentioning that for one question, Algorithm 2 only incurs one LLM call. Given q as the prefix input, the practical decoding sequence of LLM is the BFS order of the tree composed of all the generated decoding paths.

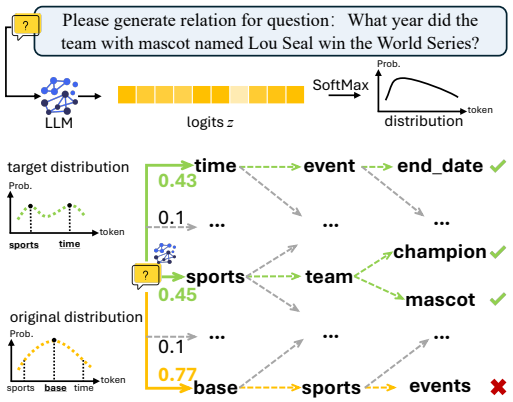


Figure 5: Token-trie based edge type generation.

Algorithm 2: TokenTrieDecode(q, b, θ, $\mathcal{T}$, M)

Input: Question q, beam width b, probability threshold θ, token-trie $\mathcal{T}$, language model M.
Output: Set of predicted relation types $\mathcal{R}_q$.

1 $\mathcal{R}_q \leftarrow \emptyset$; ▷ *Final set of decoded relation types*
2 $Q \leftarrow$ Queue((root, [])); ▷ *Queue of (node, path, depth)*
3 **while** Q *is not empty* **do**
4 $(c, p) \leftarrow Q$.dequeue(); ▷ *c: current token node, p: decoding path*
5 **if** c *is terminal* **then**
6 $\mathcal{R}_q$.add(JoinTokens(p)); ▷ *Save completed edge*
7 $z \leftarrow M(q, p)$; ▷ *Get logits for next token*
8 $\hat{z} \leftarrow$ MaskByTrie(z, $\mathcal{T}$, c); ▷ *Only keep valid children*
9 $P \leftarrow$ Softmax($\hat{z}$);
10 $\{\tau_1, \ldots, \tau_b\} \leftarrow \{i \mid P[i] > \theta \wedge P[i] \in \text{Top-b}(P)\}$;
11 **foreach** $\tau \in \{\tau_1, \ldots, \tau_b\}$ **do**
12 **if** $\tau \in \mathcal{T}.\text{Child}(c)$ **then**
13 Q.enqueue((τ, p.append(τ)));
14 **return** $\mathcal{R}_q$

To further demonstrate the scalability of the token-trie mechanism, we analyze its theoretical complexity. Modeled as an n-ary tree with K relation types and average token length $m \approx O(\log K)$, the trie requires $O(K \cdot m)$ for construction and $O(m)$ for updates.

5.2 Trie-based Path Learning

We elaborate on how to fine-tune a general-purpose LLM to effectively predict a complete relation type set. Specifically, we aim for the model to actively internalize schema constraints, reducing reliance on decoding masks, while accurately distinguishing semantically correct relations from merely plausible ones.

To achieve this goal, we propose a supervised fine-tuning strategy that navigates the model to align NL questions with their precise relation type sets, subject to the constraints of the token-trie. Given a training question q with its ground-truth relation type $\hat{\mathcal{R}}_q$, recall that $\hat{\mathcal{R}}_q$ can be decomposed into multiple sequences of tokens, aligning to the format of the decoding paths. We use $\hat{\tau}^{(i)}$ to represent the set of tokens generated in step i, from which the supervision signal for step i is a binary vector $\hat{y}^{(i)} \in \{0,1\}^{|\mathcal{R}|}$. If a token $\tau_j \in |\mathcal{R}|$ is in $\hat{\tau}^{(i)}$, we set $\hat{y}^{(i)}[j] = 1$, otherwise 0. Training the LLM is to optimize the masked KL divergence in Eq. (4):

$$\mathcal{L}_{\text{trie-KL}}(q,p) = \sum_{i=1}^{n} \sum_{\tau_j \in \hat{\tau}^{(i)}} y^{(i)}[j] \cdot \log \frac{y^{(i)}[j]}{P(\tau_j \mid p, q)} \tag{4}$$

where $P(\tau_j \mid p, q)$ is the model's output on only valid tokens t_j, by the masking Softmax in Eq. (2), where p is the prefix of τ_j in a decoding path. Unlike the conventional KL divergence that computes over the full vocabulary $\mathcal{R}$, this objective ensures that learning is only focused on structurally valid tokens, thereby eliminating gradient noise on invalid tokens and tightly coupling model training with the KG's schema.

To support efficient training, we adopt a parameter-efficient fine-tuning algorithm LoRA [13] to fine-tune a pretrained general-purpose LLM. Our data pipeline parses SPARQL queries from benchmark datasets (e.g., WebQSP, CWQ), extracts edge types, builds the corresponding token-trie, and generates prefix-level supervision examples. We also optionally support masked supervision, where the model predicts target tokens at designated masked positions rather than via autoregressive generation.

This path-level learning framework equips the model with fine-grained knowledge of edge structure and semantics, significantly improving decoding accuracy while reducing edge hallucination during inference.

6 Experimental Studies

In this section, we give the experimental setup in §6.1, and conduct substantial experiments to study the following research questions:

- **RQ1**: How effective is Faico compared to other state-of-the-art (SOTA) graph reasoning approaches? (§6.2)
- **RQ2**: To what extent do semantic faithfulness and structural completeness impact the overall performance? (§6.3)
- **RQ3**: How efficient is Faico regarding resource consumption and end-to-end query latency compared to the baselines? (§6.4)
- **RQ4**: How does each sub-component contribute to the overall performance and efficiency of Faico and are they robust? (§6.5)

6.1 Experimental Setup

6.1.1 Datasets. Our evaluation leverages three widely-used KGQA datasets: CWQ [34], WebQSP [43], and GrailQA [11]. Each dataset provides natural language questions paired with their ground-truth SPARQL queries. The profile of the datasets is given in Appendix C. For all experiments, we use a cleaned subgraph of Freebase as the underlying KG. This subgraph is derived from the original Freebase (1.9B triples) by filtering out non-English and non-numeric triples, resulting in a graph comprising 100.1 million nodes, 736.2 million edges, and over 20,400 relation types.

6.1.2 Baselines. We compare our approach against both LLM-native and recent KG-enhanced reasoning methods. The LLM-native baselines include LLM-direct [5] and Chain-of-Thought (CoT) [9, 38], which assess the inherent reasoning capabilities of LLMs. For KG-enhanced reasoning approaches, we compare with six SOTA methods: ToG [32], RoG [26], Plan-on-Graph [7], GoG [42], Paths-over-Graph [35] and SubgraphRAG [18]. To ensure a fair and robust comparison, each baseline is configured using the runnable best-performing hyperparameters reported in their papers. SubgraphRAG and RoG fail to run on GrailQA as they do not provide the necessary data preprocessing or training pipelines for this dataset.

6.1.3 Implementation details. Our graph search and token-trie learning/decoding algorithms are implemented in Python. For the main experiments, Faico uses $k = 1$ and $\theta = 0.01$ unless otherwise specified. We employ Qwen-Plus [36] as the reasoning LLM to generate answers. All experiments are conducted on a server running Ubuntu 20.04.6 LTS, equipped with 10 NVIDIA A100 GPUs, 1 TB RAM, and 128 CPU cores (Intel Xeon Platinum 8358, 2.60GHz). Method-specific parameter settings are provided in Appendix D.

6.1.4 Evaluation Metrics. To better evaluate the method's ability to mine complete answers, we use complete answer sets for metrics across three datasets. We mainly report **macro-averaged** Hit*, Hit@1, and the precision, recall, and F1 scores across queries, for both the predicted answers and the generated relation types. To assess efficiency, we use the end-to-end **average query time** across different KG-RAG methods. Additionally, we track the average number of **LLM calls** and **token usage** (including both prefill and decode tokens) as indicators of overall resource consumption. Details of the calculation of the metrics can be found in Appendix E.

6.2 Query Performance (RQ1)

We first evaluate Faico in an end-to-end setting against current SOTA approaches. While previous work evaluates the results with only partial (one) answer, we compute **Precision, Recall, F1** and **Hit** with the complete answer set to test knowledge graph reasoning in a strict and comprehensive manner. Due to space constraints, the descriptive statistics, including means and standard deviations, are provided in Appendix F.

As shown in Table 1, we find that Faico is capable of finding the correct and formal answer. The Hit metric measures the model's ability to find at least one relevant answer from the entire answer set, a higher Hit score indicates the model's ability to locate feasible answers. Faico achieves the highest Hit@1 score in both WebQSP and CWQ datasets. On WebQSP, Faico reaches 14.1% improvement over current KG reasoning SOTA (paths-over-graph). On CWQ, Faico achieves 2% (paths-over-graph) improvement in Hit@1 score. Furthermore, results in Precision, Recall and F1 show that Faico achieves higher accuracy and completeness in its answers compared to other methods. For example, Faico surpassed methods like SubgraphRAG by 32.8% and Plan-on-Graph by 45.0% on WebQSP in precision score. This demonstrates Faico's stronger ability to retrieve answers that are both relevant and correct, making it more reliable for tasks requiring precise information retrieval. While Faico ranks second in the **Hit*** metric on GrailQA, this result must be interpreted in the context of the dataset's distribution. As detailed

Table 1: Performance comparison on WebQSP, GrailQA, and CWQ datasets.

Method	WebQSP					GrailQA					CWQ				
Metric	Hit*	Hit@1	PRE	REC	F1	Hit*	Hit@1	PRE	REC	F1	Hit*	Hit@1	PRE	REC	F1
LLM-direct	66.1	37.3	32.2	26.9	26.9	31.7	11.2	10.6	8.2	8.4	40.2	28.7	26.8	25.6	25.5
LLM-COT	65.7	36.9	30.9	26.8	26.2	29.7	8.4	7.7	6.4	6.5	41.6	29.1	26.9	25.8	25.6
ToG	67.2	51.3	47.4	40.3	40.9	64.3	51.4	51.1	46.0	46.9	47.1	36.2	35.3	32.4	33.0
RoG	78.9	73.4	**78.2**	63.7	61.0	-	-	-	-	-	47.6	43.8	37.4	40.5	37.1
SubgraphRAG	87.7	62.7	58.2	56.0	54.4	-	-	-	-	-	62.9	48.2	44.8	44.4	43.0
Plan-on-Graph	75.3	58.8	53.3	46.8	47.1	76.8	65.7	64.0	59.1	59.5	50.9	37.0	35.2	32.9	33.0
GoG	80.3	68.7	63.1	55.2	55.8	77.7	71.3	68.0	64.5	63.8	65.5	57.3	53.8	50.7	50.7
Paths-over-Graph	81.7	73.6	66.9	60.9	60.1	**84.8**	**79.0**	**75.1**	**71.3**	**70.2**	72.1	69.1	66.8	**65.4**	**65.2**
Faico(Ours)	**88.6**	**84.0**	77.3	**75.3**	**73.4**	78.4	68.4	66.1	61.8	62.1	**75.2**	**70.5**	**67.6**	65.3	64.9

Table 2: OOD Generalization of Faico across datasets.

Dataset	Unseen Label Ratio (%)	Label F1	Label Hit	Graph Answer Hit
WebQSP	4.8	13.6	39.2	67.1
CWQ	2.7	38.5	71.6	43.2
GrailQA	**56.2**	34.1	69.0	86.5

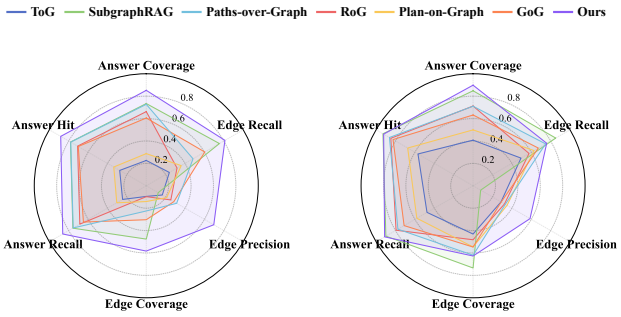
in Table 2, we analyzed the proportion of test queries containing relation types unseen during training. GrailQA presents a significant generalization challenge, with 56.2% of questions containing unseen labels—a stark contrast to WebQSP (4.8%) and CWQ (2.7%). This sparsity of training signals inevitably hampers the prediction of precise semantic edge types. However, despite this difficulty, Faico achieves a Label Hit of 69.0% and a Graph Answer Hit of 86.5%. These results demonstrate that Faico possesses generalization abilities, maintaining retrieval performance even when confronting a majority of OOD relations. To further investigate the failure cases of Faico on GrailQA see Appendix G for details.

Overall, Faico excels in both answer accuracy and completeness, achieving leading scores in current metrics. This makes it a robust method for knowledge graph reasoning tasks, providing both accurate and comprehensive answers.

6.3 Completeness (RQ2)

We extract reasoning subgraphs (RS) from all baseline methods and report additional metrics to evaluate their quality. Specifically, we measure **Edge Recall** (the ratio of correct edges types retrieved to the total number of ground-truth edge types), **Edge precision** (the ratio of correct edges in the retrieved subgraphs), and **Edge Coverage** (the ratio of the RS containing all ground-truth edge types). To assess answer retrieval, we report **Answer Recall** (the ratio of retrieved correct answers to the total number of ground-truth answers), **Answer Hit** (the ratio of the RS from which at least one answer is found) and **Answer Coverage** (the ratio of the RS containing all answers). These metrics collectively evaluate whether the RS supports complete and accurate end-to-end QA.

As shown in Figure 6, the results can be analyzed in two perspectives: (1) how well the retrieved subgraph captures the relevant semantics (Edge Recall, Edge Precision, and Edge Coverage), and (2) how effectively it leads to correct answers (Answer Recall, Answer Hit, and Answer Coverage). First, sufficient coverage of relation types is crucial for effective LLM reasoning. For example, on WebQSP, Faico achieves 75.9% Edge Recall, 58.7% Edge Precision and 62.7% Edge Coverage, outperforming Paths-over-Graph and RoG. Second, complete structural connectivity is essential for supporting valid answers. On CWQ, SubgraphRAG achieves high Edge Recall (75.4%) and Edge Coverage (47.6%), but does not outperform methods like RoG (31.7% Edge Recall, 9.7% Edge Coverage), which provide stronger connectivity guarantees. Overall, these results indicate that Faico excels in both semantic faithfulness and structural completeness, enabling more accurate and comprehensive reasoning for QA tasks.



(a) CWQ (b) WebQSP

Figure 6: RS completeness analysis on WebQSP and CWQ.

6.4 Efficiency (RQ3)

We evaluate the efficiency from two perspectives: LLM resource consumption and average query time $\overline{T}$, both measured alongside answer accuracy. As depicted in Table 3, Faico exhibits low cost and rapid response, consistently achieving top-1 or top-2 in Hit@1 across WebQSP, CWQ, and GrailQA. On WebQSP, Faico reduces input tokens by over 73.9% compared to model pre-selection SOTA methods such as plan-on-graph, and delivers over a 17.5 x speedup in average query time over paths-over-graph on CWQ. With a moderate size of RS (measured by $\overline{|\mathcal{E}_r|}$, the average number of edges in the subgraphs), and minimal token usage, Faico strikes an optimal balance between efficiency, effectiveness, and resource

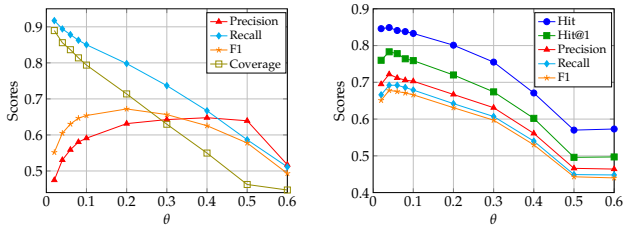
Table 3: Efficiency and resource consumption comparison.

Method	LLM*	Input	Output	$\overline{\lvert\mathcal{E}_r\rvert}$	$\overline{T}$ (sec.)	Hit@1
WebQSP						
ToG	8.5	5755.3	1084.6	2.5	66.05	51.3
RoG	1	641.4	93.9	32.4	3.55	73.4
SubgraphRAG	1	4746.9	139.4	199.2	6.04	62.7
Plan-on-Graph	13.9	8784.2	459.8	8.7	118.51	58.8
GoG	5.72	6879.9	216.6	13.9	12.50	68.7
Paths-over-Graph	10.9	32818.7	886.6	25	144.83	73.6
Faico (Ours)	1	2290.3	159.9	57	6.91	**84.0**
CWQ						
ToG	11.4	7650.5	1689.3	2.4	104.40	36.2
RoG	1	948.3	112.1	56.3	5.18	43.8
SubgraphRAG	1	4445.4	198.4	193.1	7.81	48.2
Plan-on-Graph	20.6	13342.1	660.8	7	232.77	37.0
GoG	11.5	12750	440.9	8.1	28.5	57.3
Paths-over-Graph	9.3	25185.3	1141.9	17	180.06	69.1
Faico (Ours)	1	3295	168.4	101.7	10.27	**70.5**
GrailQA						
ToG	5.9	3366	866.2	2.8	35.66	51.4
Plan-on-Graph	9.1	5142.7	354.6	8.7	36.16	65.7
GoG	4.37	5348.5	180.5	8.1	8.97	71.3
Paths-over-Graph	7.9	18696.8	779.8	26.4	78.15	**79.0**
Faico (Ours)	1	1601.2	98.1	18.4	6.26	68.4

*Report generative LLM calls only.

Table 4: Comparison of model performance under different component substitutions.

Method	Hit@1	Precision	Recall	F1
Faico(TL+k-BET)	**84.0**	**77.3**	75.3	**73.4**
TL → LLM rerank	60.5	54.5	52.0	50.7
TL → Direct SFT	77.3	72.7	70.8	69.6
k-BET → GLO	74.7	70.0	59.3	60.0
k-BET → Bounded BFS	78.6	72.8	62.3	62.9
LLM → GPT-4o-mini	83.1	76.3	74.1	72.3
LLM → Deepseek-v3.1	80.2	73.9	**76.2**	70.6



(a) Relation Type Generator Performance **(b) Overall Performance**

Figure 7: Performance sensitivity to threshold θ.

utilization. In addition, we provide the query time distribution for all baselines on WebQSP in Appendix H.

6.5 Ablation Study & Robustness (RQ4)

To validate the contributions of both token-trie learning (TL) and k-BET retrieval, we conduct ablation studies by replacing each component with alternative baselines. For TL, we consider two alternatives: (1) **Direct supervised fine-tuning (SFT)**, which fine-tunes the LLM to map query prompts directly to target edge labels (using the query as input and edge label as supervision); and (2) **LLM rerank**, which first retrieves the top-20 candidates with NV-Embed-V2, then leverages the LLM to rerank and select the most relevant edge labels. For k-BET retrieval, we evaluate (1) **Bounded BFS**, a breadth-and-depth limited search (with both width and depth bounded to 3 in our setting), and (2) **Graph-level Occurrence (GLO)**, which selects labels by globally constraining the allowed occurrence of each edge label across the explored subgraph. The ablation results on WebQSP are reported in Table 4. The performance drop ranges from 6.4% to 27.9% (Hit@1), confirming the effectiveness of our k-BET retrieval model and token-trie-based generation mechanisms. Notably, the replacement of the retrieval model leads to a disproportionate decline in Recall and F1 compared to Precision. This suggests that while baseline methods may identify some correct answers, they fail to achieve completeness, highlighting the critical role of the RS construction in mining the full set of ground-truth entities.

We further examine the effect of different LLM backbones by replacing Qwen-Plus with GPT-4o-mini and Deepseek-v3.1. The performance remains largely stable: Hit@1 shifts from 84.0% to 83.1% with GPT-4o-mini and to 80.2% with Deepseek-v3.1. This demonstrates that Faico is resilient to LLM backbone changes.

In addition, we study the interplay between semantic threshold θ and edge type budget k, which control relation label quality and subgraph size, respectively. As θ increases, Precision and Recall for relation label generation generally improve, enhancing the accuracy and completeness of generated relations. For instance, at $\theta = 0.06$, Precision and Recall reach 55.9% and 87.8%, yielding an F1 of 63.0% and Coverage of 83.7%. However, when Precision falls below an acceptable level (e.g., $\theta = 0.5$), both F1 and Coverage drop sharply, indicating that low Precision introduces incorrect or irrelevant relations. The E2E performance peaks at $\theta = 0.02$ and then consistently declines, which highlights the importance of maintaining high Precision to balance semantic coverage and correctness and ensure robust E2E performance. Further analysis of the effect of the edge bound k is provided in Appendix H.

7 Conclusion

This paper presents Faico, a KG-enhanced reasoning framework that achieves both semantic faithfulness and structural completeness, which are often overlooked by prior methods due to tight coupling between semantics and structure. By combining a fine-tuned LLM-based relation type generator and a budget-dominance-based KG retriever, Faico generates schema-valid and query-relevant relations for efficient subgraph construction. Experiments on multiple KGQA benchmarks demonstrate consistent improvements in both answer accuracy and reasoning efficiency over existing methods.

Acknowledgments

Zhiwei Zhang is supported by the National Key Research and Development Program of China (Grant No.2024YFE0209000), the NSFC (Grant No.U23B2019).

References

[1] Palaash Agrawal, Shavak Vasania, and Cheston Tan. 2025. Can LLMs Perform Structured Graph Reasoning Tasks?. In *International Conference on Pattern Recognition*. Springer, 287–308.

[2] Rohit Babbar, Ioannis Partalas, Eric Gaussier, and Massih R Amini. 2013. On flat versus hierarchical classification in large-scale taxonomies. *Advances in neural information processing systems* 26 (2013).

[3] Jinheon Baek, Alham Aji, and Amir Saffari. 2023. Knowledge-Augmented Language Model Prompting for Zero-Shot Knowledge Graph Question Answering. In *ACL*.

[4] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. 1247–1250.

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[6] Liyi Chen, Ying Sun, Shengzhe Zhang, Yuyang Ye, Wei Wu, and Hui Xiong. 2024. Tackling uncertain correspondences for multi-modal entity alignment. *Advances in Neural Information Processing Systems* 37 (2024), 119386–119410.

[7] 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 *NeurIPS*.

[8] Nurendra Choudhary, Edward W Huang, Karthik Subbian, and Chandan K Reddy. 2024. An interpretable ensemble of graph and language models for improving search relevance in e-commerce. In *Companion Proceedings of the ACM Web Conference 2024*. 206–215.

[9] Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. 2023. Towards revealing the mystery behind chain of thought: a theoretical perspective. *Advances in Neural Information Processing Systems* 36 (2023), 70757–70798.

[10] Xian Gao, Zongyun Zhang, Mingye Xie, Ting Liu, and Yuzhuo Fu. 2025. Graph of AI Ideas: Leveraging Knowledge Graphs and LLMs for AI Research Idea Generation. *arXiv preprint arXiv:2503.08549* (2025).

[11] Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. [n. d.]. Beyond IID: three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*. ACM, 3477–3488.

[12] Ruixin Hong, Hongming Zhang, Hong Zhao, Dong Yu, and Changshui Zhang. 2023. Faithful Question Answering with Monte-Carlo Planning. In *ACL*. 3944–3965.

[13] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net. https://openreview.net/forum?id=nZeVKeeFYf9

[14] Yujia Hu, Tuan-Phong Nguyen, Shrestha Ghosh, and Simon Razniewski. 2025. Enabling LLM knowledge analysis via extensive materialization. In *ACL*. 16189–16202.

[15] Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023. StructGPT: A General Framework for Large Language Model to Reason over Structured Data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 9237–9251.

[16] Juyeon Kim, Geon Lee, Taeuk Kim, and Kijung Shin. 2025. KGMEL: Knowledge Graph-Enhanced Multimodal Entity Linking. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 3015–3019.

[17] Kun Li, Tianhua Zhang, Xixin Wu, Hongyin Luo, James Glass, and Helen Meng. 2025. Decoding on graphs: Faithful and sound reasoning on knowledge graphs through generation of well-formed chains. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 24349–24364.

[18] 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 *ICLR*.

[19] Muzhi Li, Cehao Yang, Chengjin Xu, Xuhui Jiang, Yiyan Qi, Jian Guo, Ho-fung Leung, and Irwin King. 2025. Retrieval, reasoning, re-ranking: A context-enriched framework for knowledge graph completion. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. 4349–4363.

[20] Xinwei Li, Li Lin, Shuai Wang, and Hanqian Wu. 2025. Seeing Beyond Hallucinations: LLM-based Compositional Information Extraction for Multimodal Reasoning. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1000–1010.

[21] Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Shafiq Joty, Soujanya Poria, and Lidong Bing. 2024. Chain-of-Knowledge: Grounding Large Language Models via Dynamic Knowledge Adapting over Heterogeneous Sources. In *ICLR*.

[22] Xiaoxi Li, Yujia Zhou, and Zhicheng Dou. 2024. Unigen: A unified generative framework for retrieval and question answering with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 8688–8696.

[23] Yuhan Li, Xinni Zhang, Linhao Luo, Heng Chang, Yuxiang Ren, Irwin King, and Jia Li. 2025. G-refer: Graph retrieval-augmented large language model for explainable recommendation. In *Proceedings of the ACM on Web Conference 2025*.

[24] Xujian Liang and Zhaoquan Gu. 2025. Fast think-on-graph: Wider, deeper and faster reasoning of large language model on knowledge graph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 39. 24558–24566.

[25] Runxuan Liu, Luobei Luobei, Jiaqi Li, Baoxin Wang, Ming Liu, Dayong Wu, Shijin Wang, and Bing Qin. 2025. Ontology-guided reverse thinking makes large language models stronger on knowledge graph question answering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 15269–15284.

[26] Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2024. Reasoning on Graphs: Faithful and Interpretable Large Language Model Reasoning. In *ICLR*.

[27] Elan Markowitz, Anil Ramakrishna, Jwala Dhamala, Ninareh Mehrabi, Charith Peris, Rahul Gupta, Kai-Wei Chang, and Aram Galstyan. 2024. Tree-of-Traversals: A Zero-Shot Reasoning Algorithm for Augmenting Black-box Language Models with Knowledge Graphs. In *ACL*. 12302–12319.

[28] OpenLink Software. [n. d.]. Virtuoso Universal Server. https://virtuoso.openlinksw.com/. Accessed: 2025-07-31.

[29] Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering* 36, 7 (2024), 3580–3599.

[30] Pengpeng Qiao, Zhiwei Zhang, Zhetao Li, Yuanxing Zhang, Kaigui Bian, Yanzhou Li, and Guoren Wang. 2022. TAG: Joint triple-hierarchical attention and GCN for review-based social recommender system. *IEEE Transactions on Knowledge and Data Engineering* 35, 10 (2022), 9904–9919.

[31] Parshin Shojaee, Kazem Meidani, Shashank Gupta, Amir Barati Farimani, and Chandan K. Reddy. 2025. LLM-SR: Scientific Equation Discovery via Programming with Large Language Models. In *The Thirteenth International Conference on Learning Representations*.

[32] Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M. Ni, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-Graph: Deep and Responsible Reasoning of Large Language Model on Knowledge Graph. In *ICLR*.

[33] Lei Sun, Zhengwei Tao, Youdi Li, and Hiroshi Arakawa. 2024. ODA: Observation-Driven Agent for integrating LLMs and Knowledge Graphs. In *ACL*. 7417–7431.

[34] Alon Talmor and Jonathan Berant. 2018. The Web as a Knowledge-base for Answering Complex Questions. In *Proceedings of NAACL-HLT*. 641–651.

[35] 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*. 3505–3522.

[36] Qwen Team. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115* (2024).

[37] Shuyao Wang, Yongduo Sui, Chao Wang, and Hui Xiong. 2024. Unleashing the power of knowledge graph for recommendation via invariant learning. In *Proceedings of the ACM Web Conference 2024*. 3745–3755.

[38] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems* 35 (2022), 24824–24837.

[39] Marek Wydmuch, Kalina Jasinska, Mikhail Kuznetsov, Róbert Busa-Fekete, and Krzysztof Dembczynski. 2018. A no-regret generalization of hierarchical softmax to extreme multi-label classification. *Advances in neural information processing systems* 31 (2018).

[40] Yilin Xiao, Chuang Zhou, Qinggang Zhang, Bo Li, Qing Li, and Xiao Huang. 2025. Reliable reasoning path: Distilling effective guidance for llm reasoning with knowledge graphs. *arXiv preprint arXiv:2506.10508* (2025).

[41] Amy Xin, Yunjia Qi, Zijun Yao, Fangwei Zhu, Kaisheng Zeng, Xu Bin, Lei Hou, and Juanzi Li. 2024. LLMAEL: Large Language Models are Good Context Augmenters for Entity Linking.

[42] Yao Xu, Shizhu He, Jiabei Chen, Zihao Wang, Yangqiu Song, Hanghang Tong, Guang Liu, Jun Zhao, and Kang Liu. 2024. Generate-on-Graph: Treat LLM as both Agent and KG for Incomplete Knowledge Graph Question Answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[43] 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)*. 201–206.

[44] Zhangyue Yin, Qiushi Sun, Qipeng Guo, Jiawen Wu, Xipeng Qiu, and Xuan-Jing Huang. 2023. Do Large Language Models Know What They Don't Know?. In *ACL*.

[45] Jinghan Zhang, Xiting Wang, Weijieying Ren, Lu Jiang, Dongjie Wang, and Kunpeng Liu. 2025. Ratt: A thought structure for coherent and correct llm reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 39. 26733–26741.

A Notation

Table 5: Frequently used notations.

Notation	Description
$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{L})$	A knowledge graph with entity set $\mathcal{V}$, relation set $\mathcal{E}$, and a labeling function $\mathcal{L}$ that maps each relation to a semantic type in Σ
$\ell(e)$	Semantic label of edge e assigned by $\mathcal{L}$
$\mathcal{G}_r$	Reasoning subgraph for question q.
$\hat{\mathcal{R}}_q$/ $\mathcal{R}_q$	Ground-truth / predicted relation type set
T_q	topic entities
B	label budget
k	bound of the relation types

B Proof

The soundness of the completeness mainly involves the correctness of the k-BET path and the maximality of the retrieved graph. We prove this by contradiction, establishing that any counterexample would violate path effectiveness and graph maximality.

(1) Path Effectiveness. This property is guaranteed by the algorithmic design. Specifically, the algorithm initializes a per-relation budget of k, which explicitly constrains how many times each relation type may be used in the path during graph traversal. An edge is traversed only if the remaining budget of its corresponding relation is strictly positive (line 9 in algorithm1), and it decrements the budget after each use. As a result, any candidate path constructed by the algorithm is subject to a hard upper bound of k occurrences for each relation. Consequently, no path included in the final retrieved subgraph $\mathcal{G}_r$ can use a single relation more than k times. This directly satisfies the definition of a k-BET path.

(2) Graph Maximality (by contradiction). Assumption: Assume the algorithm is **incomplete** and missed at least one valid k-BET reachable path, which we will call P^*.

If P^* was missed, there must be a **neighbor edge** on this path, denoted as $e = (u, r, v)$, share one vertex u with retrieved reasoning subgraph $\mathcal{G}_r$. This means the algorithm reached node u but did not proceed to v, which could only happen for two cases:

- **Case 1: Budget Exhaustion.** The budget for relation r was already depleted upon reaching u. This would mean r had already been used k times. Traversing edge e would result in $k + 1$ uses, which would make the path P^* invalid. This contradicts our assumption that P^* is a valid path.
- **Case 2: Pruning by Budget Dominance.** This means node v was visited before via a different path that resulted in a superior (or equal) budget. A superior budget has an equal or greater number of remaining uses for all relations. In this case, the current path must be a subset (or a strict subset) of the previously visited path. Since the previously visited path dominates in budget, continuing with the current path would be redundant, leading to a contradiction.

Both scenarios lead to a contradiction, so our initial assumption that the algorithm misses paths must be false.

Having satisfied both path effectiveness and graph maximality, we conclude that the generated subgraph $\mathcal{G}_r$ is a k-BET Subgraph.

C Dataset

Table 6: Profile of datasets.

Dataset	Answer Format	# Train / Test Questions	Hop
CWQ	Entity	27,734 / 3,531	2-4
WebQSP	Entity/Number	3,098 / 1,639	1-2
GrailQA	Entity/Number	44,337 / 1,000	1-4

D Implementation Details

To support efficient and consistent evaluation, we developed a shared in-memory graph class based on the Compressed Sparse Row (CSR) format, which serves as the backend for all experiments. This design ensures uniform efficiency comparisons, as previous methods often relied on Virtuoso [28] and performed graph operations via network calls, introducing potential variability and delays from network fluctuations. We restrict the maximum search depth d ($d = 2$ for WebQSP and $d = 4$ for CWQ and GrailQA) to align with the datasets and existing studies [7, 32, 35]. For other methods, we use their default settings (max width $b = 3$,$d = 3$ for ToG, $d = 3$ for paths-over-graph with fuzzy selection and precise selection, top-200 triples for subgraphrag, $b = 3$ for RoG, $d = 4$ for plan-on-graph, using complete KG for GoG)

E Metrics

The predicted answer set A produced by the LLM is compared with the ground-truth answer set A^*. We report the following metrics: Precision $\mathbf{PRE} = \frac{|\mathcal{A} \cap \mathcal{A}^*|}{|\mathcal{A}|}$, Recall $\mathbf{REC} = \frac{|\mathcal{A} \cap \mathcal{A}^*|}{|\mathcal{A}^*|}$, and $\mathbf{F1} = 2 \times \frac{PRE \times REC}{(PRE + REC)}$. We also report $\mathbf{Hit@1} = \frac{\left|\left\{q | \mathcal{A}_q \cap \mathcal{A}_q^* \neq \emptyset\right\}\right|}{|Q|}$, where Q denotes the set of all queries. To align with prior work, we report a relaxed Hit metric $\mathbf{Hit}^*$, where answer coverage is evaluated based on containment rather than exact set overlap. All effectiveness metrics are computed using *macro averaging*, metrics are first computed for each individual query, then averaged across the entire evaluation set.

F Additional Statistical Analysis.

We report the mean ± standard deviation over three re-runs, where metrics are macro-averaged over queries. All results are rounded to one decimal place. see Tables 7,8 and 9. We observe that methods involving multiple LLM calls or larger reasoning subgraphs tend to exhibit higher variance, such as Paths-over-Graph, Plan-on-Graph, and SubgraphRAG.

G Error Analysis

We analyze the failure cases of Faico to understand the limitations imposed by the budget constraint and upstream predictions. As shown in Figure 8, the errors primarily fall into two categories: budget exhaustion and semantic drift.

- **Limit of** k. Figure 8a illustrates the recall failure when the reasoning depth exceeds the preset budget. In this case, retrieving the answer *Temple University* requires a multi-hop path from

Table 7: Performance comparison on WebQSP dataset.

Method	Hit*	Hit@1	PRE	REC	F1
LLM-direct	66.1 ± 0.4	37.3 ± 0.2	32.2 ± 0.3	26.9 ± 0.2	26.9 ± 0.2
LLM-COT	65.7 ± 0.3	36.9 ± 0.8	30.9 ± 0.7	26.8 ± 0.7	26.2 ± 0.7
ToG	67.2 ± 0.3	51.3 ± 0.5	47.4 ± 0.4	40.3 ± 0.2	40.9 ± 0.2
RoG	78.9 ± 0.1	73.4 ± 0.1	**78.2** ± 0.1	63.7 ± 0.1	61.0 ± 0.0
SubgraphRAG	87.7 ± 0.4	62.7 ± 0.4	58.2 ± 0.3	56.0 ± 0.3	54.4 ± 1.1
Plan-on-Graph	75.3 ± 0.3	58.8 ± 0.4	53.3 ± 0.4	46.8 ± 0.2	47.1 ± 0.3
GoG	80.3 ± 0.4	68.7 ± 0.6	63.1 ± 0.1	55.2 ± 0.1	55.8 ± 0.2
Paths-over-Graph	81.7 ± 0.9	73.6 ± 2.6	66.9 ± 2.5	60.9 ± 1.0	60.1 ± 1.3
Faico(Ours)	**88.6** ± 0.2	**84.0** ± 0.1	77.3 ± 0.1	**75.3** ± 0.1	**73.4** ± 0.0

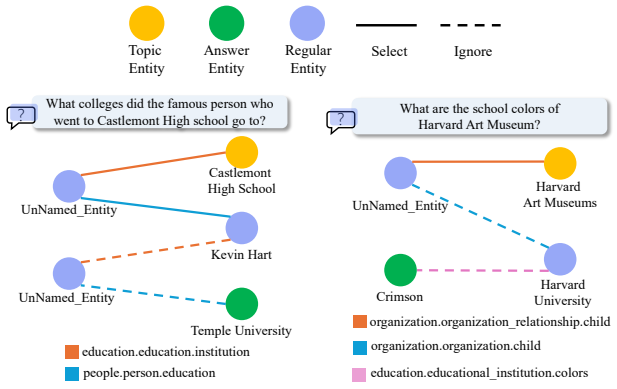
Table 8: Performance comparison on GrailQA dataset.

Method	Hit*	Hit@1	PRE	REC	F1
LLM-direct	31.7 ± 0.2	11.2 ± 0.3	10.6 ± 0.4	8.2 ± 0.2	8.4 ± 0.3
LLM-COT	29.7 ± 0.5	8.4 ± 0.4	7.7 ± 0.5	6.4 ± 0.5	6.5 ± 0.5
ToG	64.3 ± 0.3	51.4 ± 0.6	51.1 ± 0.6	46.0 ± 0.4	46.9 ± 0.4
Plan-on-Graph	76.8 ± 0.6	65.7 ± 0.4	64.0 ± 0.5	59.1 ± 0.4	59.5 ± 0.5
GoG	77.7 ± 0.3	71.3 ± 0.1	68.0 ± 0.4	64.5 ± 0.0	63.8 ± 0.1
Paths-over-Graph	**84.8** ± 0.5	**79.0** ± 1.2	**75.1** ± 1.6	**71.3** ± 0.4	**70.2** ± 0.6
Faico(Ours)	78.4 ± 0.1	68.4 ± 0.2	66.1 ± 0.2	61.8 ± 0.2	62.1 ± 0.2

Table 9: Performance comparison on CWQ dataset.

Method	Hit*	Hit@1	PRE	REC	F1
LLM-direct	40.2 ± 0.2	28.7 ± 0.0	26.8 ± 0.1	25.6 ± 0.0	25.5 ± 0.1
LLM-COT	41.6 ± 0.4	29.1 ± 0.5	26.9 ± 0.4	25.8 ± 0.4	25.6 ± 0.4
ToG	47.1 ± 0.4	36.2 ± 0.5	35.3 ± 0.4	32.4 ± 0.4	33.0 ± 0.3
RoG	47.6 ± 0.3	43.8 ± 0.3	37.4 ± 0.3	40.5 ± 0.2	37.1 ± 0.2
SubgraphRAG	62.9 ± 0.7	48.2 ± 1.1	44.8 ± 1.2	44.4 ± 1.1	43.0 ± 1.2
Plan-on-Graph	50.9 ± 0.5	37.0 ± 0.1	35.2 ± 0.2	32.9 ± 0.2	33.0 ± 0.2
GoG	65.5 ± 0.3	57.3 ± 0.5	53.8 ± 0.2	50.7 ± 0.2	50.7 ± 0.2
Paths-over-Graph	72.1 ± 0.5	69.1 ± 1.4	66.8 ± 1.3	**65.4** ± 0.3	**65.2** ± 0.5
Faico(Ours)	**75.2** ± 0.4	**70.5** ± 0.3	**67.6** ± 0.3	65.3 ± 0.4	64.9 ± 0.4

Castlemont High School via *Kevin Hart*. However, under the setting of $k = 1$, the traversal is forced to stop early because the required path length (> 1) surpasses the allowed budget. Consequently, the correct answer node becomes unreachable within the constrained search space.



(a) Limit of k **(b) False prediction**

Figure 8: Error analysis of cases where the answer entity is missing.

- **False Prediction**. Figure 8b demonstrates a failure where the correct answer is missed due to errors in the upstream relation prediction. The path to the correct entity (*Harvard University*) requires the relation `organization.organization.child`. However, this relation was not included in the predicted set $\mathcal{R}_q$. Since Faico strictly follows $\mathcal{R}_q$ to prune the graph, the edge leading to the correct answer is ignored.

H Additional Experiments

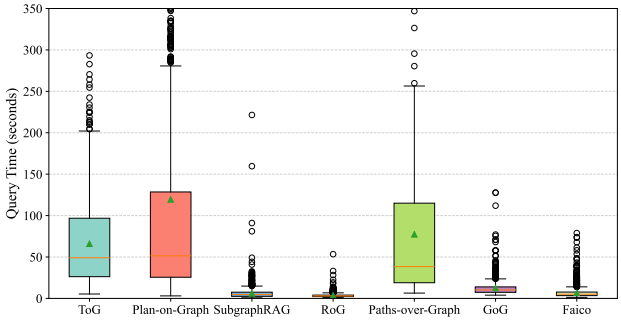


Figure 9: Query time distribution on WebQSP.

Efficiency. We additionally record the query time distribution of all methods on WebQSP. As it shows in Figure 9. Faico achieves the highest Hit@1 accuracy (84%) , while simultaneously maintaining a compact and stable query latency distribution (median 5.13s, Q3 7.68s). Compared to other high-accuracy baselines such as Paths-over-Graph (Hit@1=73.6%, median=38.43s, Q3=114.99s) and Plan-on-Graph (Hit@1=58.8%, median=51.44s, Q3=128.53s). Faico achieves 7.49x and 10.03x speedup in the median of the latency compared to Paths-over-Graph and Plan-on-Graph, respectively. Faico improves accuracy while substantially reducing long-tail high-latency queries, making it well suited for real-time and production-oriented scenarios.

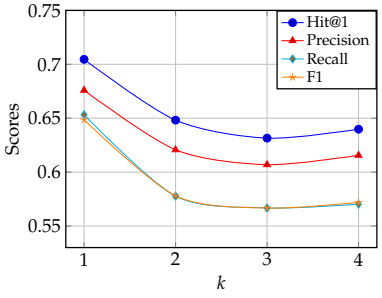


Figure 10: Performance by varying bound k on CWQ.

Robustness. As shown in Figure 10, it suggests that $k = 1$ is generally sufficient for most queries. While increasing k expands graph coverage, it tends to introduce noisy relations that outweigh potential gains, leading to marginal or negative performance changes. This trend aligns well with the empirical k-distribution shown in Figure 3b, where most queries concentrate at relation type that occurs once.