A Framework of Knowledge Graph-Enhanced Large Language Model Based on Global Planning

Yading Li, Dandan Song, Yuhang Tian, Hao Wang, Changzhi Zhou, and Shuhao Zhang, *Member, IEEE*

***Abstract*—Knowledge graphs (KGs) can provide structured knowledge to assist large language models (LLMs) in interpretable reasoning. Knowledge graph question answering (KGQA) is a typical benchmark to evaluate KG-enhanced LLM methods. Previous methods of KG-enhanced LLMs for KGQA mainly include: 1) origin question-oriented methods, which perform KG retrieval based solely on the original question without explicitly analyzing multi-step reasoning logic; and 2) stepwise reasoning-oriented methods, which alternate between LLM generating the next reasoning step and targeted KG retrieval but lack systematic planning, leading to poor controllability. To tackle these limitations, we propose KELGoP, a framework of KG-enhanced LLM based on global planning. We propose fine-grained question categorization based on reasoning patterns and corresponding category-driven question decomposition for complex questions, enabling more controllable reasoning and atomic KG retrieval targeted to sub-questions. Furthermore, we propose an adaptive strategy that allows adjusting the reasoning pattern based on the performance of question answering, making the reasoning more flexible and robust. Finally, we introduce several efficient atomic KG retrieval strategies that operate on KG subgraphs to assist the LLM in answering atomic-level questions. A series of experiments on KGQA datasets demonstrate that our proposed framework achieves superior performance compared to existing baselines.**

***Index Terms*—Knowledge graph, large language model, question answering, global planning, fine-grained categorization.**

I. Introduction

ALTHOUGH large language models (LLMs) possess outstanding language understanding and generation capabilities [1], they still suffer from issues such as lack of interpretability and hallucination [2]. To address these issues, some studies attempt to enhance LLM reasoning with knowledge graphs (KGs) [3], [4]. They utilize KGs to explicitly provide accurate structured knowledge in specialized domains to assist LLMs in performing interpretable reasoning, thus mitigating hallucinations. As an important task to evaluate the effectiveness of KG-enhanced LLM methods, the knowledge graph question answering (KGQA) task involves answering natural language questions based on the structured data in KG.

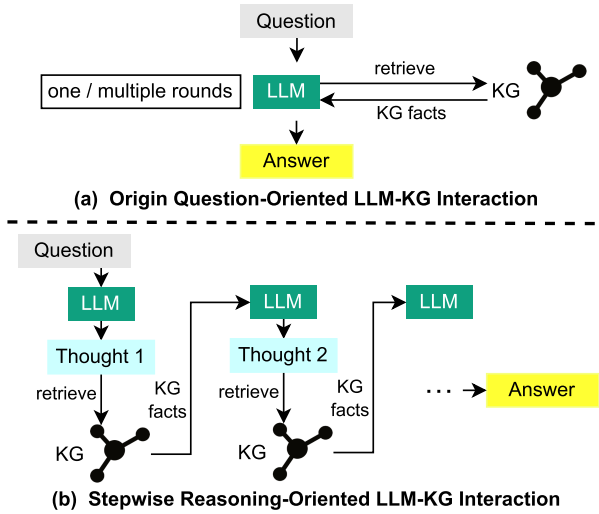


Fig. 1. A comparison diagram of two existing categories of KG-enhanced LLM reasoning methods for KGQA.

In KG-enhanced LLM methods for KGQA, one category of intuitive methods is origin question-oriented LLM-KG interaction, as shown in Fig. 1(a). Some methods [5], [6], [7] perform KG retrieval through a single round of LLM-KG interaction, where the relevant KG facts retrieved are input into the LLM to enhance its reasoning. Other methods [8], [9], [10] involve multiple rounds of LLM-KG interaction for KG retrieval. The LLM starts from the topic entity of the question and searches step by step for the next-hop entity or relation most relevant to the question, continuing until the traversed KG entities and relations are sufficient to answer the question. However, for complex questions that require multi-step reasoning, such methods rely solely on the original complex question during KG retrieval. They do not explicitly analyze the logic of multi-step reasoning or clarify the specific sub-tasks involved in KG retrieval at each sub-reasoning step. This leads to a lack of planning of the reasoning process, and also limits the flexibility of LLMs in reasoning [11].

Another category of methods is stepwise reasoning-oriented LLM-KG interaction [12], [13], as shown in Fig. 1(b). In these methods, the LLM first generates a "thought" of the next reasoning step to solve the question and then conducts a targeted KG retrieval based on the thought. The above two steps are iterated

Received 26 May 2025; revised 29 October 2025; accepted 24 November 2025. Date of publication 3 December 2025; date of current version 30 December 2025. This work was supported by the National Natural Science Foundation of China under Grant 62476025. Recommended for acceptance by S. Shang. *(Corresponding author: Dandan Song.)*

Yading Li, Dandan Song, Yuhang Tian, Hao Wang, and Changzhi Zhou are with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China (e-mail: sdd@bit.edu.cn).

Shuhao Zhang is with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China.

Code and data are available at: https://github.com/LYD369/KELGoP.

Digital Object Identifier 10.1109/TKDE.2025.3639599

to form a process of simultaneous retrieval and reasoning, which ultimately leads to the answer. Compared to the first category, this approach explicitly specifies the specific sub-tasks of KG retrieval at each reasoning step through the "thought" generated by LLM, making KG retrieval more targeted. However, this category still lacks a clear framework to provide a complete plan and systematically guide the entire reasoning flow. On the one hand, errors in LLM reasoning or KG retrieval can propagate, disrupting subsequent reasoning steps. This results in poor controllability and stability in the reasoning process and reduces accuracy. On the other hand, the heuristic approach without systematic planning may lead to irrelevant or missing information, reducing the efficiency of both KG retrieval and LLM-KG interaction.

To address the limitation of existing KG-enhanced LLM methods that lack systematic planning, in the preliminary version of this work [14], we propose an initial framework that includes both question categorization and decomposition in a coarse-grained manner, followed by targeted KG retrieval for each atomic-level question. However, this initial design still has limitations. In the current version, we extend the framework in the following ways, as summarized in the table provided in Appendix A.

1) Using fine-grained handling for complex questions: In our preliminary version, we coarsely categorize the questions into "simple" or "complex" and apply the same few-shot prompt to decompose all complex questions in the preliminary version. However, considering the diversity of complex questions and the different reasoning patterns they require, the decomposition logic needed for each category varies. Thus, the original coarse approach leads to uncontrollable decomposition results. To address this limitation, we come up with the idea of categorizing complex questions based on their reasoning patterns and designing different decomposition templates for each pattern's reasoning logic, enabling controllable question decomposition. This refinement enables more systematic and controllable planning before reasoning begins, which can improve the efficiency and accuracy of KG retrieval and LLM reasoning, while avoiding unnecessary detours or errors in the reasoning process.

2) Reducing the impact of inappropriate categorization: For complex questions, relying solely on a one-time decision to determine the reasoning pattern introduces the risk of selecting an inappropriate decomposition logic, which may lead to suboptimal reasoning plans. To address this issue, a feasible solution is to introduce an adaptive strategy that allows the framework to re-select the reasoning pattern and attempt other decomposition approaches when the current one fails. This offers an opportunity to correct errors and improve the accuracy of question answering.

3) Integrating more advanced KG searching techniques: In the preliminary version, we adopt two KG searching strategies based on the retrieval-then-reranking approach. However, both strategies are relatively simple heuristic methods that filter candidate KG entities or relations merely based on their semantic similarity to the question. Moreover, they lack a global view of the KG structure, which restricts their ability to retrieve deeply connected or implicit facts. Therefore, we consider incorporating a more advanced GNN-based KG searching method inspired by GNN-RAG [6]. Unlike GNN-RAG which performs GNN reasoning directly based on the original question, we integrate GNN reasoning into our framework for the atomic-level sub-questions. By leveraging the structural information of KG, this approach captures richer semantic signals beyond simple surface-level similarities, leading to more relevant and informative facts.

Based on the above ideas, we propose a Framework of **K**G-**E**nhanced **L**LM Based on **Glo**bal **P**lanning (**KELGoP**). Specifically, we propose fine-grained question categorization based on question reasoning patterns and, on this basis, introduce category-driven question decomposition for complex questions requiring multi-step reasoning. This serves as the backbone of LLM-KG interactive reasoning. Meanwhile, we design different decomposition logic and templates for complex questions of various reasoning patterns. These predefined decomposition templates make the question decomposition more accurate and controllable, aligning with the decomposition needs of complex questions with different reasoning patterns. At the same time, question decomposition enables targeted atomic retrieval from the KG at each sub-reasoning step for its corresponding sub-question. Additionally, we propose an adaptive strategy that dynamically adjusts the decomposition logic based on the effectiveness of question answering. When the decomposition logic of a selected reasoning pattern fails to lead to sufficient sub-Q&A pairs to effectively answer the question, we provide an opportunity for reflection and adjustment. The strategy allows for re-selecting the reasoning pattern in order to try other types of decomposition approaches. This makes the question decomposition and reasoning framework more flexible and robust. Furthermore, we introduce several efficient KG retrieval strategies, including methods based on retrieval-then-reranking and a method based on GNN, to perform accurate atomic retrieval within the candidate subgraphs on KG. These strategies work more efficiently in conjunction with the LLM reasoning process.

Our main contributions are as follows:

- We propose a global planning-oriented LLM-KG interaction method, guided by the fine-grained question categorization based on reasoning patterns and the category-driven question decomposition for complex questions. We also design different decomposition logic and templates for complex questions with different reasoning patterns. This approach makes the question decomposition more accurate and controllable, while enabling atomic retrieval for sub-questions on the KG.
- We propose an adaptive strategy for the fine-grained categorization and decomposition of questions, which provides the opportunity to reflect and dynamically adjust the decomposition approach based on the effectiveness of question answering. This makes the question decomposition and reasoning framework more flexible and robust.
- We introduce several efficient KG retrieval strategies, including methods based on retrieval-then-reranking and a method based on GNN. These strategies enable accurate

atomic retrieval within the candidate subgraphs on KG, working efficiently in conjunction with LLM reasoning.
- We evaluate the proposed framework on widely used KGQA benchmarks including WebQSP [15], CWQ [16], and GrailQA [17]. Experimental results demonstrate that different variants of our framework effectively enhance LLM reasoning with KG and consistently outperform existing baselines within their respective types, namely training-based and reasoning-based methods.

II. Related Work

A. LLM Prompting

Various prompting techniques have been developed to boost the reasoning capabilities of LLMs. Early work [18] proposes few-shot prompting for LLMs, allowing LLMs to learn and adapt to specific tasks or patterns through a small number of question-answer samples. Later, to stimulate the reasoning ability of LLMs, chain-of-thought (CoT) [19] is proposed to guide LLMs in generating intermediate reasoning steps and thought processes, thus enabling clearer and more interpretable reasoning. Building on this, several works introduce prompting strategies with different structures such as tree of thoughts (ToT) [20] and graph of thoughts (GoT) [21]. Another work [22] refines the greedy decoding strategy in CoT by generating multiple reasoning paths through the LLM and picking the most consistent response. To tackle more complex problems that require multi-step solutions, some works [23], [24] propose that LLMs first generate a reasoning plan and then solve the sub-tasks step-by-step or in parallel. For issues that errors inevitably occur during LLM reasoning, some works [25], [26] suggest using a series of error checking and correcting processes on the generated content, improving the accuracy of LLM reasoning. However, relying solely on the LLM itself for reasoning introduces hallucination issues. To address this, some works [27], [28] incorporate external knowledge into the LLM prompt to obtain more accurate and reliable reasoning results.

B. Knowledge Graph Question Answering (KGQA)

Existing methods for KGQA primarily include two categories: semantic parsing (SP)-based methods and information retrieval (IR)-based methods. **SP-based methods** [13], [29], [30], [31] first parse the semantics of the input question, converting it into a logical form such as SPARQL or S-expression, and then execute this logical form on the KG to obtain the answer. To address the issue of non-executable logical forms, DecAF [32] takes into account the direct answer generation by LLM to produce the final result.

Our proposed framework belongs to **IR-based methods**, which retrieve relevant facts from the KG and reason based on these facts to generate the answer to the question. In this category, early methods [33], [34], [35], [36] utilize structures such as key-value memory networks and graph neural networks (GNNs) to encode knowledge from the KG, enabling KG retrieval. Later, researchers [37], [38], [39], [40] begin to incorporate pretrained language models (PLMs) for tasks such as question encoding, KG relation retrieval, and semantic matching between questions and relations, to help improve the accuracy of KG retrieval and the overall reasoning process. With the rise of LLMs, recent works start to explore their potential in KG retrieval and question reasoning. Approaches like StructGPT [8] and ToG [9] employ LLMs in entity and relation retrieval on the KG, whereas GCR [41] uses LLM to retrieve reasoning paths over the KG. These approaches, together with GNN-RAG [6], further rely on LLMs for subsequent reasoning and answer generation. Additionally, RoG [5], Readi [42], and Paths-over-Graph [43] first use LLMs to generate relation paths or reasoning indicators, then employ these paths or indicators as reasoning plans for KG retrieval, and finally perform LLM-driven reasoning to generate the final answers. In order to conduct more targeted KG retrieval, PoG [44], GoG [12], and KBQA-o1 [45] explicitly carry out multi-step reasoning, gradually reaching the answer by alternately thinking about the next task with LLM and KG retrieval based on this.

However, due to the lack of a systematic and well-designed reasoning plan, these LLM-based methods suffer from inefficient LLM-KG interaction, which limits their accuracy and efficiency. Among existing methods, StructGPT [8], ToG [9], GNN-RAG [6], GCR [41], RoG [5], Readi [42], and Paths-over-Graph [43] belong to the original question-oriented category mentioned in Section I. Their KG retrieval and reasoning are not explicitly targeted to each reasoning step. GoG [12] and KBQA-o1 [45] fall into the stepwise reasoning-oriented category, which achieves explicit multi-hop reasoning but still lack a complete reasoning plan, resulting in unstable reasoning behavior. In contrast, our proposed KELGoP is a global planning-oriented framework that first decomposes the question as a global reasoning plan, and then conducts systematic targeted reasoning accordingly. Although PoG [44] also performs preliminary task decomposition, it applies a unified decomposition template across questions and performs the decomposition only once without the opportunity to revise the generated sub-objectives. If the initial decomposition is inaccurate, subsequent reasoning may fail to recover. In contrast, KELGoP performs more precise reasoning pattern-specific decomposition and introduces an adaptive strategy as a fault-tolerant mechanism to dynamically adjust the reasoning plan, thereby improving robustness and reliability.

III. Method

Before presenting our method, we first provide the definitions of Knowledge Graph (KG) and Knowledge Graph Question Answering (KGQA).

KG is a structured representation of facts, where each fact can be represented as a triple consisting of a head entity, a relation, and a tail entity. Formally, a KG is denoted as $\mathcal{G} = (e_h, r, e_t) \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$. The sets $\mathcal{E}$ and $\mathcal{R}$ refer to the collections of all entities and relations in the KG, respectively.

KGQA refers to the task of answering specific questions using the factual data within a KG. For a given question q and a KG $\mathcal{G}$, topic entities T^q can be identified from the question, and the corresponding correct answers are represented as $A^q = \{a^q\}$. Both the topics and the answers are entities within the KG,

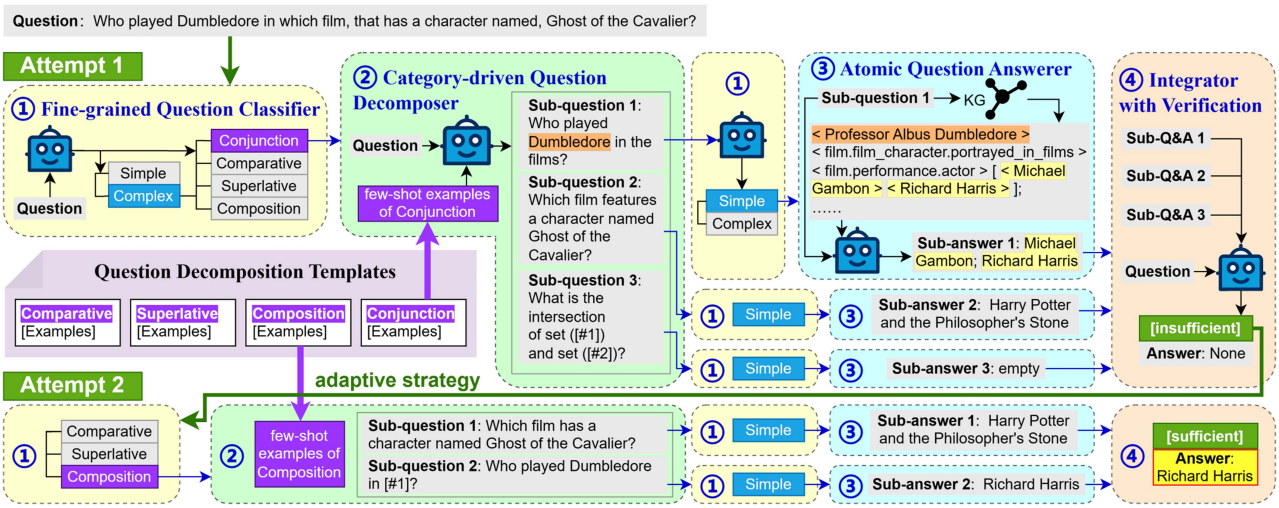


Fig. 2. An example of the workflow of our proposed framework KELGoP, which consists of four main components: *Fine-grained Question Classifier*, *Category-driven Question Decomposer*, *Atomic Question Answerer*, and *Integrator with Verification*.

meaning $T^q, A^q \subseteq \mathcal{E}$. To accomplish the KGQA task, we need to create a model f that predicts the answer a^q for a question q based on the factual knowledge in the KG $\mathcal{G}$, which is mathematically expressed as $a^q = f(q, \mathcal{G})$.

A. Overview

Fig. 2 illustrates the KELGoP framework, showing its workflow for solving a complex question, where it uses the adaptive strategy to explore two decomposition approaches. Given the input question, the *Fine-grained Question Classifier* (see Section III-B) first performs a two-stage categorization to determine its complexity and reasoning pattern. For a simple question, the *Atomic Question Answerer* (see Section III-E) retrieves relevant knowledge from the KG and derives the answer in a single reasoning step. For a complex question, the *Category-driven Question Decomposer* (see Section III-C) selects the decomposition template from our designed templates based on its category of reasoning pattern and decomposes it into a series of sub-questions. Each of them is then fed sequentially into our KELGoP framework to derive its answer. Finally, the *Integrator with Verification* (see Section III-D) integrates the sub-questions and their answers to verify their sufficiency in solving the original complex question and to infer the final answer. The sufficiency determines whether the adaptive strategy is activated to adjust the decomposition approach and re-conduct reasoning. The framework controls question decomposition by setting a maximum depth D. The decomposition process stops once this depth is reached or when a sub-question is categorized as simple. The precise workflow of KELGoP is detailed in Algorithm 1.

B. Fine-Grained Question Classifier

In order to handle questions with different complexities and reasoning patterns using different approaches, this module is designed to employ the few-shot prompted LLM to perform a two-stage fine-grained categorization on the input question q. In the first stage, question q is coarsely categorized based on its complexity, formalized as:

$$c_c^q = \text{CoarseClassifier}(q), \tag{1}$$

where $c_c^q \in \{\text{“}\texttt{Simple}\text{”}, \text{“}\texttt{Complex}\text{”}\}$. The few-shot prompt for LLM employed in the CoarseClassifier is as follows:

Determine the type of the question (simple or complex). Answer “{{Simple}}” or “{{Complex}}”.
Few-Shot Examples
Question: {Question}
Answer:

We aim to enable the LLM to understand the criteria for the first-stage categorization through the few-shot examples. We manually annotate these examples on the basis of the SPARQL provided in the dataset as well as the semantic and logical complexity of the questions. A question is considered complex if it requires multi-step reasoning. Otherwise, it is simple.

Complex questions require fine-grained categorization based on their reasoning pattern in the second stage, formalized as:

$$c_f^q = \text{FineClassifier}(q, C), \tag{2}$$

where $c_f^q \in C, C \subseteq \{\text{“}\texttt{Comparative}\text{”}, \text{“}\texttt{Composition}\text{”}, \text{“}\texttt{Conjunction}\text{”}, \text{“}\texttt{Superlative}\text{”}\}$. C contains the reasoning pattern categories that have not been selected during the adaptive reasoning process and is initialized to include all four categories. The few-shot prompt for LLM employed in the FineClassifier is as follows:

Based on the following examples, learn how to determine the type of a question based on its semantics, rather than relying on keywords in the question. Then, determine the type of the given question based on its semantics. Answer {Candidate Categories}.
{Few-Shot Examples of Candidate Categories}
Question: {Question}
Answer:

Algorithm 1: KELGoP.

Input: question q, golden topic entities T_g^q of question q, KG $\mathcal{G}$, maximum depth D of question decomposition, maximum attempts θ of fine-grained categorization
Output: answer entity a^q

$depth \leftarrow 0$
$a^q \leftarrow$ KELGoP$(q, T_g^q, \mathcal{G}, D, \theta, depth)$
function KELGoP$q, T_g^q, \mathcal{G}, D, \theta, depth$
  if $depth = D$ OR CoarseClassifier$(q) =$ "`Simple`" **then**
    $a^q \leftarrow$ AtomicQuestionAnswerer$(q, \mathcal{G}, T_g^q)$
  else
    $C \leftarrow \{$"`Comparative`", "`Composition`", "`Conjunction`", "`Superlative`"$\}$
    while $len(C) \geq 5 - \theta$ **do**
      $c_f^q \leftarrow$ FineClassifier(q, C)
      $Sq^q \leftarrow$ Decomposer(q, c_f^q)
      $\widetilde{Sq^q}, Sa^q \leftarrow [\,], [\,]$
      for $i = 0 \rightarrow len(Sq^q) - 1$ **do**
        $\widetilde{sq_i^q} \leftarrow replace(Sq^q[i], Sa_{<i}^q)$
        $\widetilde{Sq^q}.append(\widetilde{sq_i^q})$
        $\widetilde{a^{sq_i^q}} \leftarrow$ KELGoP$(\widetilde{sq_i^q}, T_g^q, \mathcal{G}, D, \theta, depth + 1)$
        $Sa^q.append(\widetilde{a^{sq_i^q}})$
      end for
      $(a^q, l^q) \leftarrow$ Integrator$(q, \widetilde{Sq^q}, Sa^q)$
      $C \leftarrow C \setminus \{c_f^q\}$
      if $l^q =$ "`[sufficient]`" **then**
        break
      end if
    end while
  end if
  return a^q
end Function

The prompt includes examples of candidate reasoning pattern categories C, with each question's category derived from the dataset annotations.

C. Category-Driven Question Decomposer

To flexibly handle diverse questions categorized as `Complex` by the CoarseClassifier, this module processes them according to their reasoning pattern assigned by the FineClassifier. It employs the few-shot prompted LLM to decompose a given question q into a series of sub-questions Sq^q according to its category c_f^q, formalized as:

$$Sq^q = \text{Decomposer}(q, c_f^q). \tag{3}$$

where $Sq^q = [sq_1^q, \ldots, sq_d^q]$ is the list of d sub-questions and sq_i^q is the sub-question of the i-th reasoning step. The few-shot prompt for LLM used in the Decomposer is as follows:

> Use multi-step reasoning and decompose the question into several sub-questions. If a subsequent subquestion requires the answer to the previous subquestion, replace the answer to subquestion-k with [#k] in the subsequent subquestion.
> {`Few-Shot Examples of Specific Category`}
> Question: {Question}
> Answer:

Considering the continuity between steps in multi-step reasoning, the sub-questions in later steps are likely to require the answers from preceding sub-questions. Therefore, we manually craft the sub-questions in the examples to help the LLM understand their format requirements. For each sub-question, $sq_i^q =< w_1, \ldots, w_{|sq_i^q|} >$, where $w_k \in \mathcal{W} \cup \mathcal{T}_i$. $\mathcal{W}$ is the set of words and punctuation marks. $\mathcal{T}_i = \{$`[#j]` $|1 \leq j < i, i \in \mathbb{Z}\}$ is the set of tags for answers to preceding sub-questions of sq_i^q, where `[#j]` denotes the answer to sub-question j.

Additionally, the few-shot examples in the prompt guide the LLM to formulate a global plan in the form of sub-questions. As different categories c_f^q require different question decomposition logic, the prompt includes category-specific examples to ensure accurate and controllable decomposition. The sub-questions for each category in these examples are manually crafted based on the SPARQL annotations in the dataset and our understanding of question semantics.

Specifically, we design dedicated decomposition methods for different categories of reasoning patterns. Detailed decomposition examples can be found in Appendix B. A `Composition` question involves two progressive sub-questions. In the decomposition example, we break it down sequentially: solve sub-question 1 first to obtain the intermediate answer, then use it to solve sub-question 2, leading to the final answer. A `Conjunction` question consists of two parallel sub-questions and requires answers satisfying both conditions. So we break it into two sequential sub-questions and take the intersection of their answers in the example. `Comparative` and `Superlative` questions also involve two progressive sub-questions, but their second reasoning steps differ from `Composition` questions. `Comparative` questions compare a specific attribute of the intermediate answer to a given value, while `Superlative` questions compare intermediate answers against each other on a specific attribute to determine the extreme value. After comparison, both categories identify the attribute that meets the condition and select the corresponding intermediate answer as the final answer.

D. Integrator With Verification

Considering the sequential nature of multi-step reasoning, once the *Category-driven Question Decomposer* decomposes a complex question into d sub-questions, they should be solved sequentially. The processing and solving of the i-th sub-question sq_i^q $(1 \leq i \leq d, i \in \mathbb{Z})$ is formalized as:

$$\begin{aligned} \widetilde{sq_i^q} &= replace(sq_i^q, Sa_{<i}^q), \\ \widetilde{a^{sq_i^q}} &= \text{KELGoP}(\widetilde{sq_i^q}, \mathcal{G}), \end{aligned} \tag{4}$$

where $Sa^q_{<i} = [\widetilde{a^{sq^q_1}}, \ldots, \widetilde{a^{sq^q_{i-1}}}]$, storing the answers to the preceding sub-questions in order. Before feeding sq^q_i into our framework KELGoP to derive the answer $\widetilde{a^{sq^q_i}}$, it is necessary to employ the function *replace* to replace any preceding answer tag "`[#j]`" $(1 \leq j < i, i \in \mathbb{Z})$ that may be included in sq^q_i with the corresponding answer text $\widetilde{a^{sq^q_j}} \in Sa^q_{<i}$. It is worth noting that if a sub-question sq^q_i depends on a preceding answer that contains multiple answer entities, the function *replace* substitutes the corresponding preceding answer tag in sq^q_i with a semicolon-joined string of these entities, yielding $\widetilde{sq^q_i}$. In addition, we retain a set of instances in which each of these entities is individually substituted into the tag position, for later use. Specifically, when $\widetilde{sq^q_i}$ is subsequently fed into KELGoP, if it is categorized as `Simple` and topic entity extraction is required (see Section III-E1), each instance in this set of instantiated sub-questions is independently passed to the TopicExtractor. This design avoids the omission of relevant topics that may occur if multiple entities are provided simultaneously within a single input sentence.

After obtaining answers to all d sub-questions, this module uses the few-shot prompted LLM to integrate them. It verifies their sufficiency l^q in addressing the original complex question q and attempts to infer the answer a^q, formalized as:

$$(a^q, l^q) = \text{Integrator}(q, \widetilde{Sq^q}, Sa^q), \tag{5}$$

where $l^q \in \{$"`[sufficient]`", "`[insufficient]`"$\}$. $\widetilde{Sq^q} = [\widetilde{sq^q_1}, \ldots, \widetilde{sq^q_d}]$ and $Sa^q = [\widetilde{a^{sq^q_1}}, \ldots, \widetilde{a^{sq^q_d}}]$ represent the lists of d sub-questions and their corresponding answers, respectively.

The few-shot prompt for LLM in the Integrator is as follows:

> Answer the question "{{answer entity}}" according to the given Q&A references and your knowledge.
> `Few-Shot Examples`
> Question: {Question}
> References: {Pairs of Sub-Q&A}
> Answer:

The "Answer" in the few-shot examples are manually crafted, including the sufficiency label l^q, the reasoning process, and the answer a^q to the original question. If the verification result is `[sufficient]`, the LLM should integrate the sub-Q&A pairs to generate the whole reasoning process. Based on this, the LLM attempts to generate the final answer.

The sufficiency label l^q determines whether our adaptive strategy is activated. If the label l^q is `[insufficient]`, it may indicate that the currently selected reasoning pattern or decomposition method is not suitable for this question and therefore needs to be adjusted. The currently selected category c^q_f is removed from C, and the framework attempts to solve the question by selecting a new category of reasoning pattern using FineClassifier$(q, C \setminus \{c^q_f\})$.

E. Atomic Question Answerer

This module is designed to handle atomic questions categorized as `Simple` by the CoarseClassifier, referring to either original questions or sub-questions that are identified as solvable through one-step reasoning. Therefore, we can perform atomic retrieval for the given question to obtain relevant KG facts. Then with their assistance, we employ the few-shot prompted LLM to derive the answer.

1) Topic Entity Extraction: The KG contains a vast amount of data, most of which is irrelevant to a simple question. To avoid inefficient retrieval and introduction of noise to subsequent answer generation, we locate a relevant subgraph in the large-scale KG. Therefore, it is first necessary to identify the topic entities of the question.

For each atomic question q, we extract entity IDs from the SPARQL annotated for its corresponding original question in the dataset to preliminarily form its topic entity set T^q_g. Since atomic questions may have intermediate topics that are produced during multi-step reasoning and not included in the SPARQL, we further employ an entity linker as the TopicExtractor to identify and link these entities to KG $\mathcal{G}$, which is formalized as:

$$T^q = \text{TopicExtractor}(q, \mathcal{G}) \cup T^q_g, \tag{6}$$

where $T^q = \{t^q\} \subseteq \mathcal{E}$, serving as the source entities for the subsequent search of relevant KG facts.

2) Relevant Fact Searching: Given the question q and its topic set T^q, we need to search for relevant facts Ref^q on KG $\mathcal{G}$, which can be formalized as:

$$Ref^q = \text{Searcher}(q, T^q, \mathcal{G}), \tag{7}$$

The Searcher consists of two main steps: 1) extracting a relevant subgraph in the neighborhood of topic entities as candidates; 2) retrieving relevant facts from the candidates. We introduce several strategies to implement the Searcher, as shown in Fig. 3. Among them, two are designed based on commonly used retrieval-then-reranking approach in RAG, which focus on textual similarity. Another strategy follows the GNN-based approach proposed in [6], which focuses more on the structural information of KG.

Retrieval-then-Reranking-Based Searching: Based on the retrieval-then-reranking approach, we design two searching strategies, which are **triple/quadruple-based** and **relation-based**, respectively. The difference lies in the form of obtained candidates and the focus of retrieval, as shown in Fig. 3(a) and (b).

In the first step of candidate extraction, for each $t^q \in T^q$, we extract candidates $Cand^q_{t^q}$ relevant to the question q within a 2-hop range where t^q serves as either the head or tail entity, for reasons discussed in Appendix C. The ranking and selection of relevant relations are based on the semantic similarity between the question and the candidate first- and second-hop relations, which is computed using a dense retriever. The second hop is only considered when the first-hop entity linked by the selected top-m first-hop relations is a compound value type (CVT), which is an intermediate node representing n-ary relations in KG and does not have natural language names. When constructing candidates, the triple/quadruple-based strategy incorporates linked first- or second-hop entities to form triples or quadruples, while the relation-based strategy does not, as illustrated in Fig. 3(a) and (b). Finally, we merge the candidates extracted for each

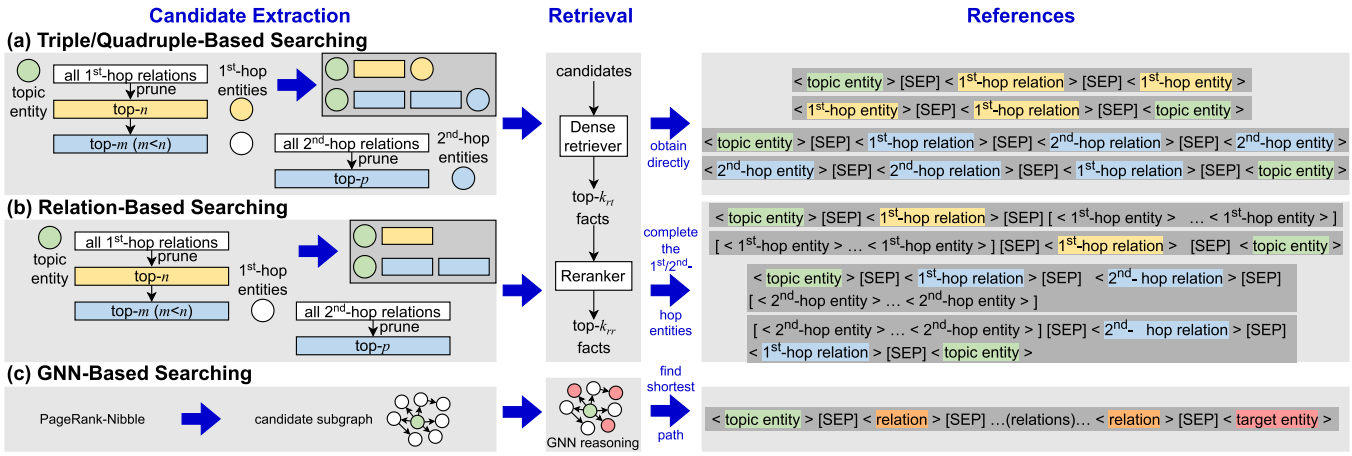


Fig. 3. Illustration of the question-relevant KG fact searching strategies, where (a) and (b) are retrieval-then-reranking-based strategies and (c) is the GNN-based strategy.

topic entity to obtain the candidate subgraph of question q in the form of a fact list, denoted as $Cand^q = \cup Cand^q_{t^q}, \forall t^q \in T^q$.

In the second step, we follow the retrieval-then-reranking process with a dense retriever and a reranker to obtain a small number of facts from candidate set $Cand^q$ as the reference set Ref^q for subsequent answer generation. For the relation-based strategy, it is necessary to further complete the first- or second-hop entities for each $ref^q \in Ref^q$.

GNN-Based Searching: Following [6], the PageRank-Nibble (PRN) [46] is used to extract the candidate subgraph for the original question. We adopt them as candidates for the corresponding atomic questions.

Then, following the GNN reasoning module in [6], the GNN reasoning is performed on the candidate subgraph to obtain nodes with a high likelihood of being the correct answer as retrieved entities. The shortest paths between these entities and the topic entities in the candidate subgraph are then identified to form the relevant facts, as shown in Fig. 3(c), which collectively constitute the reference set Ref^q.

3) Answer Generation: After obtaining the retrieved fact set Ref^q on the KG for the atomic question q, we use them as references and employ the few-shot prompted LLM to reason and generate the answer a^q, which is formalized as:

$$a^q = \text{Answerer}(q, Ref^q). \tag{8}$$

The few-shot prompt for the LLM in Answerer is as follows:

Reason and answer the question in "{{answer entity}}" according to the given retrieved knowledge and your knowledge.
Few-Shot Examples
Question: {Question}
Retrieved knowledge: {Retrieved KG Facts}
Answer:

The manually crafted few-shot examples guide the LLM to organize the reasoning process with the given knowledge and use its inherent knowledge for assistance when necessary.

IV. Experiments

A. Datasets and Evaluation Metric

We evaluate the effectiveness of our framework KELGoP on three KGQA datasets. **WebQuestionsSP** (WebQSP) [15] consists of 2,848, 250 and 1,639 QA pairs for training, validation and testing, respectively, following the split in [6]. **ComplexWebQuestions** (CWQ) [16] is built upon WebQSP by increasing question complexity through approaches such as expanding original entities into sub-questions or adding answer constraints. CWQ contains 27,639, 3,519 and 3,531 QA pairs for training, validation and testing, respectively. **GrailQA** [17] includes three levels of questions, namely i.i.d., compositional and zero-shot. We follow [9] and [44] to use 1,000 QA pairs for testing on GrailQA. All three datasets are based on Freebase KG [47]. We pre-process Freebase following [48], retaining only entities in the form of Freebase IDs, English text, and numerical values, along with their corresponding triples.

We use Exact Match (EM) as the evaluation metric, following previous works [8], [9], [28], [44], [49]. We match the correct answer within the "{}" scope marked in the LLM response. If the marked content is absent, we search for the correct answer within the entire response.

B. Baselines and Our Frameworks

We consider two types of baselines for comparison. **Training-based methods** refer to those that involve training or fine-tuning at least one component, such as PLMs, LLMs or KG retrievers, on the KGQA datasets, including KV-Mem [33], GraftNet [34], PullNet [35], EmbedKGQA [50], TransferNet [37], NSM [36], KGT5 [51], SR+NSM+E2E [38], TIARA [52], KD-CoT [28], UniKGQA [39], ReasoningLM [40], DecAF [32], RoG [5], GNN-RAG [6], and SubgraphRAG [7]. **Reasoning-based methods** refer to those that perform no training or fine-tuning on the KGQA datasets we use, including IO [18], CoT [19], CoT+SC [22], StructGPT [8], KB-BINDER [49], Readi [42], ToG [9], GoG [12], and PoG [44].

Our proposed framework has three variants based on different relevant KG fact searching strategies. **KELGoP-tri** and **KELGoP-rel** use the triple/quadruple-based and relation-based retrieval-then-reranking strategies, respectively, and both are reasoning-based methods. In contrast, **KELGoP-gnn** adopts the GNN-based searching strategy that trains a GNN on the KGQA datasets, thus considered a training-based method, though the LLM remains untrained.

C. Implementation Details

Unless otherwise specified, the default LLM used in our experiments is gpt-3.5-turbo-0125 [53], with a maximum output token limit of 200. The maximum depth D of question decomposition in our framework is set to 1. For our adaptive strategy, the maximum attempts θ of fine-grained categorization is set to 3 by default, meaning that a complex question can attempt up to three different reasoning patterns.

For topic entity extraction, we use the golden topic entities for each question and adopt ELQ [54] as the entity linker. In retrieval-then-reranking-based KG searching, we use DistilBERT[1] as the dense retriever and MiniLM[2] as the reranker. As illustrated in Fig. 3, the parameters for candidate extraction are set as $n = 10$, $m = 5$, and $p = 5$. During retrieval, the number of retained facts k_{rt} is set to 20, while during reranking, the number of retained facts k_{rr} is set to 10 for KELGoP-tri and 13 for KELGoP-rel by default.

All experiments were conducted on a single NVIDIA RTX A6000 GPU (48 GB VRAM), an Intel Xeon Gold 5220R CPU, and 503 GB of system memory, with the actual runtime memory usage peaking at about 72 GB RAM. Further implementation details of the experiments can be found in Appendix D.

D. Main Results

Table I presents a comprehensive comparison between baselines and our KELGoP on WebQSP and CWQ. All three variants of KELGoP show significant improvements over the IO prompting method which represents the few-shot performance of vanilla GPT-3.5. This demonstrates the effectiveness of our method in enhancing LLM reasoning with KG. KELGoP-gnn and KELGoP-rel outperform the previous state-of-the-art baselines in their respective types, training-based and reasoning-based methods, achieving the best known performance. KELGoP-tri also surpasses most reasoning-based baselines. The strong performance of our framework on both WebQSP with relatively simple questions and CWQ with more complex questions indicates its adaptability to different question complexity. Moreover, compared to most training-based methods, KELGoP-rel achieves comparable or superior performance without extra training or fine-tuning. This shows the robustness of our framework in KGQA while reducing training costs.

To further evaluate the generalizability of our framework, we conduct experiments on GrailQA, which also contains complex questions that require multi-step reasoning, but without the annotations of comparative, composition, conjunction and superlative as in CWQ. GrailQA includes questions with more diverse reasoning types beyond these four predefined categories, providing a broader testbed for evaluating our framework's generalization ability. The performance of reasoning-based variants of our framework and baselines is shown in Table II. Both variants achieve significant improvements over LLM prompting methods that lack KG support, including IO, CoT, and CoT+SC, with KELGoP-rel achieving the best performance among all reasoning-based methods. These demonstrate that our framework effectively enhances LLM reasoning with KG on complex questions across datasets, outperforming existing state-of-the-art methods.

TABLE I
PERFORMANCE (EM IN PERCENT) OF OUR FRAMEWORK AND BASELINES ON WEBQSP AND CWQ

Types	Methods	WebQSP	CWQ
Training-based	KV-Mem [33]	46.7	18.4
	GraftNet [34]	66.4	36.8
	PullNet [35]	68.1	45.9
	EmbedKGQA [50]	66.6	-
	TransferNet [37]	71.4	48.6
	NSM [36]	74.3	48.8
	KGT5 [51]	56.1	36.5
	SR+NSM+E2E [38]	69.5	49.3
	TIARA [52]	75.2	-
	KD-CoT [28]	68.6	55.7
	UniKGQA [39]	77.2	51.2
	ReasoningLM [40]	78.5	69.0
	DecAF [32]	82.1	70.4
	RoG [5]	85.7	62.6
	GNN-RAG [6]	85.7	66.8
	SubgraphRAG [7]	83.1	56.3
	KELGoP-gnn	**88.5**	**71.5**
Reasoning-based	StructGPT [8]	72.6	-
	KB-BINDER [49]	74.4	-
	Readi [42]	74.3	55.6
	ToG [9]	76.2	58.9
	GoG [12]	78.7	55.7
	PoG [44]	82.0	63.2
	IO [18] †	64.8	34.4
	CoT [19] †	70.3	40.2
	CoT+SC [22] †	71.4	41.6
	KELGoP-tri	79.6	61.9
	KELGoP-rel	**84.7**	**67.7**

† indicates the results obtained by reproducing the methods.

TABLE II
PERFORMANCE (EM IN PERCENT) OF OUR FRAMEWORK AND BASELINES ON GRAILQA

Methods	GrailQA			
	Overall	I.I.D.	Compositional	Zero-shot
KB-BINDER [49]	53.2	-	-	-
ToG [9]	68.7	70.1	56.7	72.7
PoG [44]	76.5	76.3	62.1	81.7
IO [18] †	29.4	24.6	21.2	34.3
CoT [19] †	33.2	29.6	21.2	39.0
CoT+SC [22] †	34.9	34.6	22.7	39.3
KELGoP-tri	75.5	77.1	**63.1**	79.2
KELGoP-rel	**78.1**	**78.3**	**63.1**	**83.3**

† indicates the resained by reproducing the methods.

[1]https://huggingface.co/sentence-transformers/msmarco-distilbert-base-v3
[2]https://huggingface.co/cross-encoder/ms-marco-MiniLM-L-12-v2

TABLE III
PERFORMANCE (EM IN PERCENT) OF OUR FRAMEWORK AND LLM PROMPTING BASELINES ON QUESTIONS OF DIFFERENT CATEGORIES IN CWQ

Methods	Compa	Compo	Conj	Super	Overall
IO [18] †	42.7	42.6	43.0	21.8	34.4
CoT [19] †	49.3	50.5	45.7	27.4	40.2
CoT+SC [22] †	43.2	50.6	46.7	28.4	41.6
KELGoP-tri	54.0	67.9	60.9	31.0	61.9
KELGoP-rel	59.6	**72.5**	67.4	41.1	67.7
KELGoP-gnn	**71.8**	69.1	**77.3**	**43.1**	**71.5**

Compa, **Compo**, **Conj** and **Super** represents Comparative, Composition, Conjunction and Superlative respectively. † indicates the results obtained by reproducing the methods.

E. Performance on Different Categories

CWQ annotates a gold label for each question's category of reasoning pattern. Table III shows the performance of our framework and LLM prompting baselines across various categories annotated in CWQ.

Among the four categories, our framework performs significantly better on comparative, composition, and conjunction questions than on superlative questions. We consider that this discrepancy arises because superlative questions require comparing many entities by a numeric attribute. However, KELGoP-tri and KELGoP-rel provide limited reference facts in KG searching, often insufficient for such comparisons. And in KELGoP-gnn, the KG searcher we use [6] excludes numeric entities during the PRN-based candidate extraction step. This limitation in the preprocessing stage, rather than in the GNN itself, makes it impossible for subsequent steps to retrieve necessary numeric attributes. Consequently, they result in lower accuracy for superlative questions. A potential enhancement is to incorporate KG schema information into the searching process, enabling the model to identify numeric aggregation or comparison scenarios and to jointly query the same numeric attributes across multiple entities. During retrieval, lightweight aggregation or filtering operations such as MAX, MIN, or range-based constraints can be introduced to ensure that relevant numeric facts are retrieved within the limited number of reference facts.

F. Performance on Different LLMs

We evaluate our proposed framework with different LLMs, including Qwen2.5-7B [55], Llama-3.1-8B [56], gpt-3.5-turbo-0125 [53], and gpt-4o-2024-11-20 [57]. The results are presented in Fig. 4.

Compared with the IO prompting method that represents the original few-shot performance of LLMs, all three variants of our KELGoP show significant improvements with different LLMs. This demonstrates that the proposed framework can stably and substantially enhance the reasoning ability of various LLMs. Comparing the performance of different LLMs on each variant of KELGoP, we find that in most cases, GPT-4 performs the best, followed by GPT-3.5, with Llama-3.1 and Qwen-2.5 showing slightly weaker performance. One possible reason is that, compared to GPT-4 and GPT-3.5, Llama-3.1 and Qwen-2.5 not only have weaker reasoning capabilities, but also attempt fewer reasoning patterns. They are more likely to output the `[sufficient]` label in the Integrator, thus missing opportunities to correct mistakes by switching to new reasoning patterns.

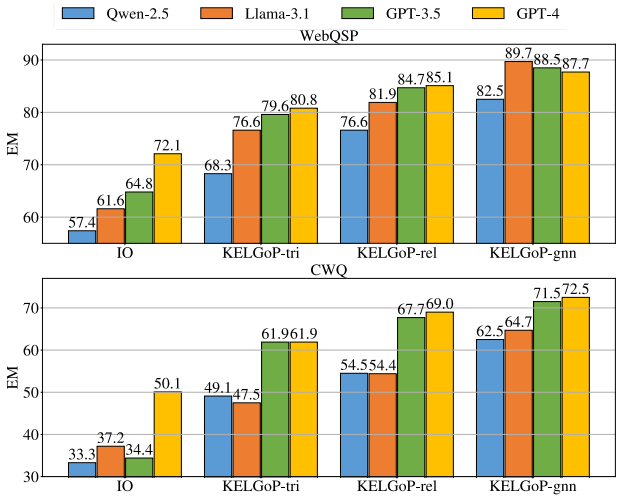


Fig. 4. Performance (EM in percent) of our framework on different LLMs.

TABLE IV
ABLATION STUDY RESULTS (EM IN PERCENT) OF OUR FRAMEWORK

Methods	CWQ	WebQSP	GrailQA
KELGoP-tri	**61.9**	79.6	**75.5**
w/o adaptive strategy	58.7	76.4	73.9
w/o fine-grained categorization	58.2	**80.3**	74.7
w/o decomposition	57.4	79.3	72.1
KELGoP-rel	**67.7**	84.7	**78.1**
w/o adaptive strategy	64.4	82.9	75.2
w/o fine-grained categorization	63.9	84.7	75.4
w/o decomposition	58.7	**85.4**	75.1
KELGoP-gnn	**71.5**	88.5	-
w/o adaptive strategy	68.2	88.5	-
w/o fine-grained categorization	68.1	88.2	-
w/o decomposition	71.4	**89.1**	-

G. Ablation Study

To evaluate the contribution of each individual strategy in KELGoP, we conduct ablation study by removing them one by one. w/o adaptive strategy means there is no opportunity to reselect the reasoning pattern. On this basis, w/o fine-grained categorization removes the second-stage of Classifier, applying the same decomposition examples that cover all four reasoning patterns to all complex questions. w/o decomposition removes the Classifier entirely, treating all questions as simple and solving them solely with the Atomic Question Answerer.

The ablation study results for different variants of KELGoP are shown in Table IV. Considering the results in column "Complex Proportion" of Table V, we observe that on CWQ, where the average complexity of questions is the highest, the overall accuracy tends to decrease as more strategies are removed from the framework, indicating that most strategies contribute positively to performance. An exception is observed with KELGoP-gnn, where coarse-grained decomposition leads to a performance

TABLE V
EFFICIENCY ANALYSIS OF OUR FRAMEWORK

Datasets	Methods	Complex Proportion (%)	LLM Calls	LLM Input	LLM Output	KG Searching Time (s)	Total Time (s)
WebQSP	KELGoP-tri	7.4	2.3	2253.7	89.1	18.9	14.3*
	KELGoP-rel	7.6	2.3	4111.7	99.7	14.2	6.7*
	KELGoP-gnn	7.1	2.3	2522.4	97.1	0.8	10.0
CWQ	KELGoP-tri	85.9	6.1	6957.3	431.4	32.6	33.7*
	KELGoP-rel	85.8	5.9	10921.5	400.1	28.9	19.4*
	KELGoP-gnn	86.0	5.8	8561.6	423.1	8.8	39.7
GrailQA	KELGoP-tri	35.4	3.5	4527.5	192.2	11.5	4.6*
	KELGoP-rel	36.4	3.5	6630.3	211.8	9.9	8.1*

Complex Proportion indicates the proportion of questions in the dataset that are categorized as `Complex` by our framework. **LLM Calls**, **LLM Input**, and **LLM Output** indicate the average number of LLM calls, LLM input and output tokens per question, respectively. **KG Searching Time** represents the average time spent on all topic entity extraction and relevant fact searching steps per question. **Total Time** represents the average time for our framework to solve each question, with * indicating results under a multi-threaded setting.

drop, as its GNN-based KG retriever is trained on original questions and less effective for sub-questions. However, this performance gap is recovered and even surpassed by introducing fine-grained categorization and the adaptive strategy. Our strategies are naturally more suitable for handling complex questions. On WebQSP, where most questions are simple, they may misclassify simple questions as complex and introduce unnecessary steps and potential errors that hurt overall performance. On GrailQA, where question complexity is moderate, most strategies improve the performance noticeably. Although limiting the fine-grained categorization to one attempt causes a performance decline, adding the adaptive strategy substantially enhances the overall performance to a higher level.

H. Effects of Maximum Attempts θ of the Fine-Grained Categorization

In our adaptive strategy, the maximum number of attempts θ determines how many different categories of reasoning patterns are allowed to be explored when solving a complex question, rather than the actual number of attempts used during reasoning. With four categories in total, θ ranges from 1 to 4. We evaluate the performance of our framework under various settings of θ on WebQSP and CWQ, with results in Fig. 5.

As the maximum allowed attempts θ increases, the IO content of LLM naturally grows longer. However, the accuracy does not continuously improve with higher θ, and the overall EM typically peaks at $\theta = 3$. On CWQ, despite its complexity, most questions do not require too many reasoning attempts. When $\theta = 3$, the average actual attempts per complex question in CWQ are only 1.43, 1.35, and 1.30 for KELGoP-tri, KELGoP-rel, and KELGoP-gnn, respectively. Setting θ too high may prevent the framework from effectively curbing the excessive self-doubt of LLM, leading to unnecessary re-selection of less appropriate reasoning patterns and increased errors. To further verify the reliability of the adaptive mechanism itself, we manually examine the sufficiency judgment of LLM based on the results of KELGoP-rel on CWQ. Among 151 sufficiency decisions from 100 randomly sampled complex questions, the correctness rate reaches 90.1%, indicating that the sufficiency assessment of LLM is generally dependable as the core signal of our adaptive strategy.

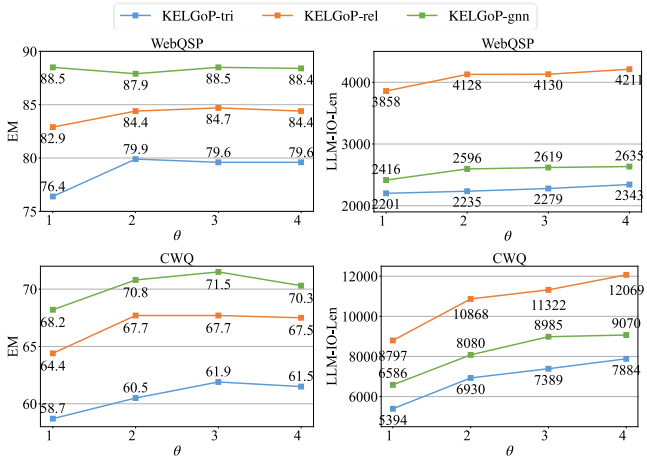


Fig. 5. Experimental results with different maximum numbers of attempts. LLM-IO-Len is the average total length of LLM input and output tokens for all questions.

However, since the questions in WebQSP are relatively simple, the fine-grained categorization and adaptive strategy are rarely triggered. As a result, the accuracies of KELGoP variants on WebQSP do not exhibit a consistent trend with varing θ.

I. Effects of the Retained KG Facts Number k_{rr} in References

For our reasoning-based variants, the number k_{rr} of facts retained after reranking in KG search corresponds to the number of references provided to LLM in the Atomic Question Answerer. The results for different k_{rr} values on WebQSP and CWQ are shown in Fig. 6. As the number of references grows, KG searching accuracy steadily improves. However, this does not result in a continuous increase in overall accuracy. KELGoP-tri and KELGoP-rel achieved the best overall EM on both datasets when k_{rr} was set to 10 and 13, respectively. Further increasing k_{rr} not only leads to longer LLM input prompts but also causes a decrease in overall accuracy.

That is, additional correct references are not utilized by the LLM effectively and may even cause a decline in overall performance. We suspect this is due to the issue of information dilution. Although excessive references increase the probability of including correct KG facts, thereby boosting retrieval

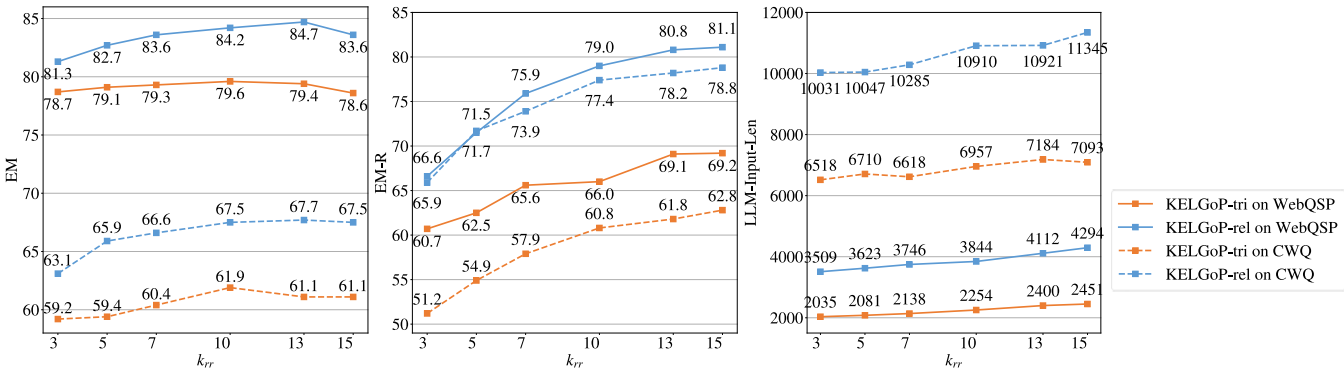


Fig. 6. Experimental results under different facts numbers in references. EM-R represents the accuracy of the KG searching in the framework. LLM-Input-Len is the average length of LLM input tokens for all questions.

accuracy, an excessively long prompt may cause the LLM to unevenly distribute its attention. This can result in reduced focus on the core information of the question. Moreover, an overload of references may cause the LLM to rely more on external information and less on its internal reasoning, thereby limiting its reasoning capability. Ultimately, this leads to a decrease in overall question-answering performance. To further illustrate this effect, we provide an example in Appendix G showing how different values of k_{rr} affect the overall accuracy.

J. Efficiency Analysis

Table V presents an efficiency analysis of the proposed framework. The average time cost per question for running the entire pipeline of different KELGoP variants on each dataset is reported in the "Total Time" column. The reasoning time of KELGoP mainly depends on the complexity of questions, the time spent on KG searching, and the number of LLM calls.

As shown in the "Complex Proportion" column of Table V, CWQ has the highest proportion of questions categorized as complex, followed by GrailQA, while most questions in WebQSP are categorized as simple. This indicates that a larger proportion of questions in CWQ require using modules including the FineClassifier, Decomposer and Integrator, resulting in more requirements of LLM calls and KG searching. This leads to significantly higher values in terms of the average number of LLM calls, LLM input and output tokens, and the average time for KG searching and overall reasoning on CWQ compared to those on other datasets.

For different variants of KELGoP, KELGoP-rel requires significantly more LLM input tokens. This is because its relation-based searching strategy obtains KG facts that include all target entities linked through a relation chain from the topic entity, which leads to longer references in LLM input prompts of the Atomic Question Answerer. However, compared to the triple/quadruple-based strategy, the relation-based strategy reduces the number of candidate facts by retrieving at the relation level. This makes KELGoP-rel faster in KG searching, which in turn shortens its total reasoning time compared to KELGoP-tri. KELGoP-gnn performs fastest in KG searching, as GNN reasoning is more efficient than sequential filtering by the retriever and reranker.

V. Conclusion

This study introduces a framework of KG-enhanced LLM based on global planning, called KELGoP. We propose fine-grained question categorization based on reasoning patterns and corresponding category-driven question decomposition, which make reasoning more controllable and enable atomic KG retrieval. Additionally, we propose an adaptive strategy for reasoning pattern selection, which adjust according to the performance of reasoning to ensure flexibility and robustness. Furthermore, we introduce retrieval-then-reranking-based and GNN-based KG searching strategies to perform efficient atomic-level searching. Experimental results show that our KELGoP outperforms the state-of-the-art baselines.

In future work, we will try to explore more accurate and efficient KG retrieval strategies that better integrate the structural and textual features of the KG to support the reasoning process. Additionally, we plan to investigate methods that can adapt to a wider range of question complexities, in order to further extend and improve our framework.

References

[1] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu, "Unifying large language models and knowledge graphs: A roadmap," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 7, pp. 3580–3599, Jul. 2024.

[2] Z. Ji et al., "Survey of hallucination in natural language generation," *ACM Comput. Surv.*, vol. 55, no. 12, pp. 1–38, Mar. 2023.

[3] X. Zhang et al., "GreaseLM: Graph REASoning enhanced language models," in *Proc. Int. Conf. Learn. Representations*, 2022, pp. 24765–24780.

[4] L. Hu, Z. Liu, Z. Zhao, L. Hou, L. Nie, and J. Li, "A survey of knowledge enhanced pre-trained language models," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 4, pp. 1413–1430, Apr. 2024.

[5] L. Luo, Y.-F. Li, R. Haf, and S. Pan, "Reasoning on graphs: Faithful and interpretable large language model reasoning," in *Proc. Int. Conf. Learn. Representations*, 2024, pp. 34088–34111.

[6] C. Mavromatis and G. Karypis, "GNN-RAG: Graph neural retrieval for efficient large language model reasoning on knowledge graphs," in *Proc. Annu. Meeting Assoc. Comput. Linguistics*, 2025, pp. 16682–16699.

[7] M. Li, S. Miao, and P. Li, "Simple is effective: The roles of graphs and large language models in knowledge-graph-based retrieval-augmented generation," in *Proc. Int. Conf. Learn. Representations*, 2025, pp. 49797–49825.

[8] J. Jiang, K. Zhou, Z. Dong, K. Ye, W. X. Zhao, and J.-R. Wen, "StructGpt: A general framework for large language model to reason over structured data," in *Proc. Conf. Empirical Methods Nat. Lang. Process.*, 2023, pp. 9237–9251.

[9] J. Sun et al., "Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph," in *Proc. Int. Conf. Learn. Representations*, 2024, pp. 14199–14229.

[10] J. Jiang et al., "KG-Agent: An efficient autonomous agent framework for complex reasoning over knowledge graph," in *Proc. Annu. Meeting Assoc. Comput. Linguistics*, 2025, pp. 9505–9523.

[11] L. Yang, H. Chen, Z. Li, X. Ding, and X. Wu, "Give us the facts: Enhancing large language models with knowledge graphs for fact-aware language modeling," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 7, pp. 3091–3110, Jul. 2024.

[12] Y. Xu et al., "Generate-on-graph: Treat LLM as both agent and KG for incomplete knowledge graph question answering," in *Proc. Conf. Empirical Methods Nat. Lang. Process.*, 2024, pp. 18410–18430.

[13] G. Xiong, J. Bao, and W. Zhao, "Interactive-KBQA: Multi-turn interactions for knowledge base question answering with large language models," in *Proc. Annu. Meeting Assoc. Comput. Linguistics*, 2024, pp. 10561–10582.

[14] Y. Li et al., "A framework of knowledge graph-enhanced large language model based on question decomposition and atomic retrieval," in *Proc. Conf. Empirical Methods Natural Lang. Process. Findings*, 2024, pp. 11472–11485.

[15] W.-t. Yih, M. Richardson, C. Meek, M.-W. Chang, and J. Suh, "The value of semantic parse labeling for knowledge base question answering," in *Proc. Annu. Meeting Assoc. Comput Linguistics.*, 2016, pp. 201–206.

[16] A. Talmor and J. Berant, "The web as a knowledge-base for answering complex questions," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguist., Hum. Lang. Technol.*, 2018, pp. 641–651.

[17] Y. Gu et al., "Beyond IID: Three levels of generalization for question answering on knowledge bases," in *Proc. World Wide Web Conf.*, 2021, pp. 3477–3488.

[18] T. Brown, "Language models are few-shot learners," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 1877–1901.

[19] J. Wei et al., "Chain-of-thought prompting elicits reasoning in large language models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, vol. 35, pp. 24824–24837, 2022.

[20] S. Yao et al., "Tree of thoughts: Deliberate problem solving with large language models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, vol. 36, pp. 11809–11822 .

[21] M. Besta et al., "Graph of thoughts: Solving elaborate problems with large language models," in *Proc. AAAI Conf. Artif. Intell.*, Mar. 2024, vol. 38, no. 16, pp. 17682–17690.

[22] X. Wang et al., "Self-consistency improves chain of thought reasoning in language models," in *Proc. Int. Conf. Learn. Representations*, 2023, pp. 12648–12671.

[23] L. Wang et al., "Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models," in *Proc. Annu. Meet. Assoc. Comput Linguistics.*, 2023, pp. 2609–2634.

[24] X. Ning, Z. Lin, Z. Zhou, Z. Wang, H. Yang, and Y. Wang, "Skeleton-of-thought: Prompting LLMs for efficient parallel generation," in *Proc. Int. Conf. Learn. Representations*, 2024, pp. 44032–44082.

[25] A. Madaan et al., "Self-refine: Iterative refinement with self-feedback," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, vol. 36, pp. 46534–46594.

[26] N. Miao, Y. W. Teh, and T. Rainforth, "SelfCheck: Using LLMs to zero-shot check their own step-by-step reasoning," in *Proc. Int. Conf. Learn. Representations*, 2024, pp. 1225–1240.

[27] S. Yao et al., "React: Synergizing reasoning and acting in language models," in *Proc. Int. Conf. Learn. Representations*, 2023, pp. 30084–30116.

[28] K. Wang et al., "Knowledge-driven cot: Exploring faithful reasoning in LLMS for knowledge-intensive question answering," 2023, *arXiv:2308.13259*.

[29] Y. Sun, L. Zhang, G. Cheng, and Y. Qu, "SPARQA: Skeleton-based semantic parsing for complex questions over knowledge bases," in *Proc. AAAI Conf. Artif. Intell.*, Apr. 2020, vol. 34, no. 05, pp. 8952–8959.

[30] L. Zhang et al., "FC-KBQA: A fine-to-coarse composition framework for knowledge base question answering," in *Proc. Annu. Meeting Assoc. Comput Linguistics*, 2023, pp. 1002–1017.

[31] Y. Tian et al., "Augmenting reasoning capabilities of LLMS with graph structures in knowledge base question answering," in *Proc. Conf. Empirical Methods Natural Lang. Process., Findings*, 2024, pp. 11967–11977.

[32] D. Yu et al., "DecAF: Joint decoding of answers and logical forms for question answering over knowledge bases," in *Proc. Int. Conf. Learn. Representations*, 2023, pp. 10518–10535.

[33] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston, "Key-value memory networks for directly reading documents," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2016, pp. 1400–1409.

[34] H. Sun, B. Dhingra, M. Zaheer, K. Mazaitis, R. Salakhutdinov, and W. Cohen, "Open domain question answering using early fusion of knowledge bases and text," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2018, pp. 4231–4242.

[35] H. Sun, T. Bedrax-Weiss, and W. Cohen, "PullNet: Open domain question answering with iterative retrieval on knowledge bases and text," in *Proc. Empirical Methods Natural Lang. Process., 9th Int. Joint Conf. Natural Lang. Process.*, 2019, pp. 2380–2390.

[36] G. He, Y. Lan, J. Jiang, W. X. Zhao, and J.-R. Wen, "Improving multi-hop knowledge base question answering by learning intermediate supervision signals," in *Proc. ACM Int. Conf. Web Search Data Mining*, 2021, pp. 553–561.

[37] J. Shi, S. Cao, L. Hou, J. Li, and H. Zhang, "TransferNet: An effective and transparent framework for multi-hop question answering over relation graph," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2021, pp. 4149–4158.

[38] J. Zhang et al., "Subgraph retrieval enhanced model for multi-hop knowledge base question answering," in *Proc. Annu. Meeting Assoc. Comput. Linguistics.*, 2022, pp. 5773–5784.

[39] J. Jiang, K. Zhou, X. Zhao, and J.-R. Wen, "UniKGQA: Unified retrieval and reasoning for solving multi-hop question answering over knowledge graph," in *Proc. Int. Conf. Learn. Representations*, 2023, pp. 34362–34377.

[40] J. Jiang, K. Zhou, W. X. Zhao, Y. Li, and J.-R. Wen, "Reasoninglm: Enabling structural subgraph reasoning in pre-trained language models for question answering over knowledge graph," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2023, pp. 3721–3735.

[41] L. Luo, Z. Zhao, G. Haffari, Y.-F. Li, C. Gong, and S. Pan, "Graph-constrained reasoning: Faithful reasoning on knowledge graphs with large language models," in *Proc. 42nd Int. Conf. Mach. Learn.*, 2025, pp. 41540–41565.

[42] S. Cheng et al., "Call me when necessary: LLMS can efficiently and faithfully reason over structured environments," in *Proc. Annu. Meeting Assoc. Comput. Linguistics.*, 2024, pp. 4275–4295.

[43] X. Tan, X. Wang, Q. Liu, X. Xu, X. Yuan, and W. Zhang, "Paths-over-graph: Knowledge graph empowered large language model reasoning," in *Proc. World Wide Web Conf.*, 2025, pp. 3505–3522.

[44] L. Chen, P. Tong, Z. Jin, Y. Sun, J. Ye, and H. Xiong, "Plan-on-graph: Self-correcting adaptive planning of large language model on knowledge graphs," in *Proc. Adv. Neural Inf. Process. Syst*, 2024, pp. 37665–37691.

[45] H. Luo et al., "KBQA-o1: Agentic knowledge base question answering with Monte Carlo tree search," in *Proc. 42nd Int. Conf. Mach. Learn.*, 2025, pp. 41177–41199.

[46] R. Andersen, F. Chung, and K. Lang, "Local graph partitioning using Pagerank vectors," in *Proc. Annu. IEEE Symp. Found. Comput. Sci.*, 2006, pp. 475–486.

[47] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2008, pp. 1247–1250.

[48] Y. Lan and J. Jiang, "Query graph generation for answering multi-hop complex questions from knowledge bases," in *Proc. Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 969–974.

[49] T. Li, X. Ma, A. Zhuang, Y. Gu, Y. Su, and W. Chen, "Few-shot in-context learning on knowledge base question answering," in *Proc. Annu. Meeting. Assoc. Comput. Linguistics.*, 2023, pp. 6966–6980.

[50] A. Saxena, A. Tripathi, and P. Talukdar, "Improving multi-hop question answering over knowledge graphs using knowledge base embeddings," in *Proc. Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 4498–4507.

[51] A. Saxena, A. Kochsiek, and R. Gemulla, "Sequence-to-sequence knowledge graph completion and question answering," in *Proc. Annu. Meeting Assoc. Comput. Linguistics*, 2022, pp. 2814–2828.

[52] Y. Shu et al., "Tiara: Multi-grained retrieval for robust question answering over large knowledge base," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2022, pp. 8108–8121.

[53] "OpenAI, Introducing chatGPT," 2022. [Online]. Available: https://openai.com/index/chatgpt

[54] B. Z. Li, S. Min, S. Iyer, Y. Mehdad, and W.-t. Yih, "Efficient one-pass end-to-end entity linking for questions," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2020, pp. 6433–6441.

[55] A. Yang et al., "Qwen2.5 technical report," 2024, *arXiv:2412.15115*.

[56] A. Grattafiori et al., "The llama 3 herd of models," 2024, *arXiv:2407.21783*.

[57] "OpenAI, GPT-4," 2023. [Online]. Available: https://openai.com/index/gpt-4-research

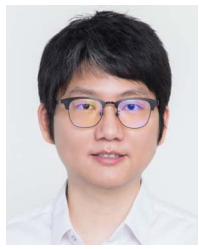
Yading Li received the bachelor's degree in 2024 from the Beijing Institute of Technology, Beijing, China, where she is currently working toward the master's degree with the School of Computer Science and Technology. Her research interests include large language models and knowledge graph question answering.

Dandan Song received the BE and PhD degrees from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2004 and 2009, respectively. She is currently a tenured professor with the School of Computer Science and Technology, Beijing Institute of Technology. Her research interests include knowledge graph, knowledge enhanced large language models, and data mining.

Yuhang Tian received the bachelor's degree in 2021 from the Beijing Institute of Technology, Beijing, China, where he is currently working toward the PhD degree in computer science. His research interests include knowledge base question answering and theory of mind-enhanced dialogue.

Hao Wang is currently working toward the PhD degree with the Beijing Engineering Research Center of High Volume Language Information Processing and Cloud Computing Application, School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China. His research interests include knowledge graph embedding, knowledge graph completion, and graph learning.

Changzhi Zhou received the bachelor' s degree from the Shandong University of Science and Technology, Shandong, China, in 2020. He is currently working toward the PhD degree with the Beijing Institute of Technology, Beijing, China. His research interests include large language models and natural language processing.



Shuhao Zhang (Member, IEEE) received the bachelor's degree from Nanyang Technological University (NTU), and the PhD degree from the National University of Singapore. He was an assistant professor with the College of Computing and Data Science, NTU. From 2020 to 2021, he was a postdoctoral Researcher with Prof. Volker Markl at Technische Universität Berlin. He is currently a professor with the School of Computer Science and Technology, Huazhong University of Science and Technology. His research interests include parallel database systems, large language model inference frameworks, and their integration with stream processing techniques.