STEM: Structure-Tracing Evidence Mining for Knowledge Graphs-Driven Retrieval-Augmented Generation

Peng Yu, En Xu, Bin Chen*, Haibiao Chen, Yinfei Xu
AI Product Center, Kingsoft Corporation, Beijing, China
{yupeng5,xuen,chenbin,chenhaibiao,xuyinfei}@kingsoft.com

Abstract

Knowledge Graph-based Question Answering (KGQA) plays a pivotal role in complex reasoning tasks but remains constrained by two persistent challenges: the structural heterogeneity of Knowledge Graphs (KGs) often leads to semantic mismatch during retrieval, while existing reasoning path retrieval methods lack a global structural perspective. To address these issues, we propose Structure-Tracing Evidence Mining (STEM), a novel framework that reframes multi-hop reasoning as a schema-guided graph search task. First, we design a Semantic-to-Structural Projection pipeline that leverages KG structural priors to decompose queries into atomic relational assertions and construct an adaptive query schema graph. Subsequently, we execute globally-aware node anchoring and subgraph retrieval to obtain the final evidence reasoning graph from KG. To more effectively integrate global structural information during the graph construction process, we design a Triple-Dependent GNN (Triple-GNN) to generate a Global Guidance Subgraph (Guidance Graph) that guides the construction. STEM significantly improves both the accuracy and evidence completeness of multi-hop reasoning graph retrieval, and achieves State-of-the-Art performance on multiple multi-hop benchmarks. Our source code is available at https://github.com/PennyYu123/STEM_RAG.

1 Introduction

The research and development of large language models (LLMs) have spanned several years (OpenAI, 2023; Anthropic, 2024; Touvron et al., 2023; Chowdhery et al., 2023; Jiang et al., 2023a; Bai et al., 2023), however, LLMs suffer from the issue of hallucination, often leading to inaccuracies in responses to fact-based questions. To address this, Retrieval-Augmented Generation (RAG) was introduced as a promising paradigm to ground model outputs in verifiable external knowledge bases(Lewis et al., 2020; Trivedi et al., 2023; Guu et al., 2020; Borgeaud et al., 2022). By leveraging pre-existing knowledge bases, RAG enables LLMs to reference relevant contextual information when generating answers, thereby improving the accuracy and quality of responses. In recent years, knowledge graph-based question answering systems for LLMs have gained significant research attention (Yasunaga et al., 2021; Zhang et al., 2022; He et al., 2024; Edge et al., 2024). These systems utilize structured knowledge graphs to organize relationships among various real-world entities, providing the LLM with a hierarchical and logically clear knowledge network, thereby enabling more precise guidance for generating accurate answers.

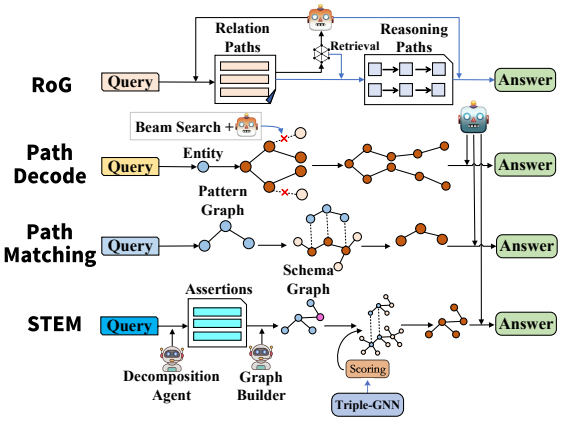

Figure 1: Different KG Retrieval Reasoning Frameworks.

Knowledge graph-based multi-hop reasoning for question answering has also garnered extensive research attention (Yasunaga et al., 2021; Luo et al., 2024; Yu et al., 2023; Sui et al., 2025; Liu et al., 2026; Cai et al., 2025). Existing approaches for KG-based reasoning generally fall into three main paradigms: The first involves generating reasoning plans based on the query to retrieve evidence chains

*Corresponding author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 7252–7285

for answer generation (RoG) (Luo et al., 2024). The second employs step-wise path exploration, such as beam search, utilizing intelligent modules like LLMs to iteratively determine the optimal path (Path Decode) (Sui et al., 2025). The third focuses on structural matching, which involves constructing an aligned schema graph to guide step-by-step searches within the KG to build a minimal distance subgraph (Path Matching) (Cai et al., 2025). The retrieved subgraph is subsequently verbalized and fed into an LLM for final answer generation. An overview of the design of the above method is detailed in Figure 1.

Despite the extensive research in multi-hop reasoning over Knowledge Graphs (KGs), existing methods still face significant impediments that limit their efficacy and robustness. We identify three primary challenges as follows:

First, the semantic-structural gap between LLM-generated plans and KG schemas hinders accurate retrieval. KGs are characterized by inherent organizational complexity, containing millions of entities and diverse relation types. Current approaches typically rely on LLMs to decompose questions or generate reasoning plans in natural language. However, due to the LLM's lack of prior knowledge regarding the specific KG schema, these generated plans often suffer from **schema hallucination**—predicting relations that are semantically plausible but topologically nonexistent in the target KG. For instance, consider the query "Which airport to fly into rome?" While the ground-truth path in the KG follows an hierarchical structure like "Rome $\xrightarrow{\text{location.location.nearby\_airports}}$ Ciampino–G. B. Pastine International Airport", but a schema-agnostic planner is likely to generate a forward-looking formulation such as "[ENT1] $\xrightarrow{\text{fly into}}$ Rome". This creates a fundamental topological mismatch: the semantic implication of "fly into" often fails to align with the knowledge schema, resulting in retrieval failures.

Second, prevailing path-search paradigms suffer from lack of global view and evidence fragmentation. Most existing methods employ stepwise search strategies, which rely on local semantic similarity or LLM-based decision-making to select the next hop. While some incorporate look-ahead scoring to anticipate future steps, they fundamentally lack a global structural blueprint. This leads to three critical issues: (1) Path Deviation: Without global guidance, the search process is highly sensitive to local spurious semantic correlations. (2)Hub Node Problem: The sheer volume of neighboring relations creates an overwhelming candidate space, which significantly increases the error rate of selection. This necessitates a schema graph for effective search space pruning. (3) Information Fragmentation: Since complex questions often require a subgraph structure rather than isolated reasoning paths, path-based methods frequently retrieve fragmented evidence, consequently the question cannot be adequately answered.

Finally, the reliance on interactive reasoning incurs prohibitive computational costs. Many state-of-the-art methods adopt an "interleaved" approach, where the LLM is invoked at every step of the reasoning path to judge validity or refine the search direction, these approaches create a significant bottleneck in inference latency and resource consumption. This bottleneck is particularly exacerbated in multi-answer scenarios, where retrieving multiple answers necessitates traversing parallel reasoning paths, thereby leading to substantially higher LLM overhead.

To address these challenges, we propose Structure-Tracing Evidence Mining (STEM), a novel architecture that reframes multi-hop KG reasoning from sequential path finding to holistic schema-guided subgraph matching. STEM distinguishes itself through two key innovations: First, we bridge the schema gap via building a Semantic-to-Structural Projection pipeline. Different from existing works attempting to convert queries into logic forms (e.g., SPARQL) for KG retrieval (Sun et al., 2020; Lan and Jiang, 2020; Ye et al., 2022), we design two specialized modules: the Schema-Grounded Decomposition Agent (SGDA) and the Symbol-Aligned Graph Builder (SAGB) to ensure the retrieval plan aligns with the KG's inherent topology. The SGDA first decomposes the complex query into a sequence of atomic relational assertions, stripping away linguistic ambiguity and align the semantic narratives with the logic of the KG. Subsequently, the SAGB grounds each assertion into a concrete triple structure, assembling them into a coherent schema graph.

Second, we implement Structure-Tracing Subgraph Retrieval to retrieve related evidence. STEM employs a Triple-Dependent GNN (Triple-GNN) to produce a Global Guidance Subgraph (Guidance Graph). During the traversal phase, the retrieval is guided not only by local transition probabilities (semantic distance) but also by the global structural

scores according to Guidance Graph. This mechanism ensures that search step is globally weighted, effectively correcting potential deviations and ensuring the retrieval of a complete, logically connected evidence subgraph. Based on the comprehensive introduction, the main contributions of this paper are summarized as follows:

- We propose a novel Semantic-to-Structural Projection pipeline to bridge the inherent gap between natural language queries and KG schemas. By employing a Schema-Grounded Decomposition Agent and a Symbol-Aligned Graph Builder, we construct a precise schema graph that serves as a topological blueprint for subsequent retrieval.
- We introduce Structure-Tracing Evidence Mining, a holistic subgraph matching paradigm for KGQA. By leveraging a Triple-Dependent GNN to construct a Global Guidance Subgraph, our method incorporates global structural priors into the search process, effectively ensuring both the accuracy and completeness of the retrieved evidence.
- Our proposed method achieves State-of-the-Art performance on complex multi-hop reasoning benchmarks including WebQSP and CWQ, demonstrating the effectiveness of our proposed method.

2 Preliminaries

Typically, the structure of a knowledge graph can be defined as a tuple: $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{N}, \mathcal{R})$, where $\mathcal{V}$ represents the set of nodes, $\mathcal{E}$ denotes the set of edges, $\mathcal{T} = \{(h, r, t) \mid h, t \in \mathcal{V}, r \in \mathcal{E}\}$ represents the set of triplets. $\mathcal{N}$ corresponds to the textual descriptions of each entity node in $\mathcal{V}$, i.e., for each node $v_i \in \mathcal{V}$, there exists $e_i \in \mathcal{N}$ representing the textual description of v_i. Likewise $\mathcal{R}$ denotes the set of relations, corresponding to the descriptions of edge in $\mathcal{E}$, i.e., for each edge $d_i \in \mathcal{E}$, there exists $r_i \in \mathcal{R}$ representing the textual description of d_i.

The primary objective of KGQA framework is to identify and extract a relevant subgraph $\mathcal{G}' \subseteq \mathcal{G}$ from the entire knowledge graph based on a given natural language query Q, this retrieved subgraph serves as structured evidence to ground the reasoning process of a Large Language Model. We pre-index all entities mapped to nodes in graph $\mathcal{G}$ within a high-speed indexing system for the rapid retrieval.

3 Methodology

We present our methodology as follows. First, we describe how the SGDA module performs question decomposition. Next, we explain how the SAGB module builds a schema graph. We then introduce the Triple-GNN for Guidance Graph building. Subsequently, we present the complete workflow of Structure-Tracing Subgraph Retrieval. Finally, we introduce the answer generation process. The overall design framework of the approach in this paper is shown in Figure 2.

3.1 Schema-Aligned Question Decomposition

The objective of the SGDA is to generate atomic relational assertions from the original query, these assertions will align the semantic narratives with the relations in KG. CoG (Zhao et al., 2025) proposed a "knowledge reciting" task, training an LLM with query-path pairs from datasets to equip with prior knowledge of paths, preventing plausible but irretrievable paths in the KG. Inspired by this, and considering the scale and complexity of nodes and paths in KGs, our SGDA module focuses on learning patterns rather than specific knowledge. For instance, given the training example ("What is the San Francisco Giants' mascot?", "San Francisco Giants' mascot is [ENT1]"), the SGDA can construct assertions following the same pattern for similar queries, such as "What is the X's mascot?", successfully resolving cases associated with this type of problem.

Atomic Relational Assertion. An atomic relational assertion refers to a text describing a single logical relationship—for instance, "Western Sahara contains Smara"—which can be directly mapped to a single triple (Western Sahara, contains, Smara). Formally, Given the prompt $\mathcal{P}_1$ and the multi-hop reasoning question Q, SGDA decompose the original query into a structured sequence $\mathcal{S}$, where each element is represented as an assertion s, the form is described below:

$$\mathcal{S} = \{s_1, s_2, , ..., s_N\} \tag{1}$$

We present the specific content of $\mathcal{P}_1$ and all subsequent prompts in the Appendix G. To maintain logical connectivity across multi-hop reasoning steps, the SGDA employs a unified placeholder mechanism for entity linking. Since the answer to a preceding sub-question often serves as the bridging entity for the subsequent one, we standardize these intermediate variables using shared identifiers (e.g.,

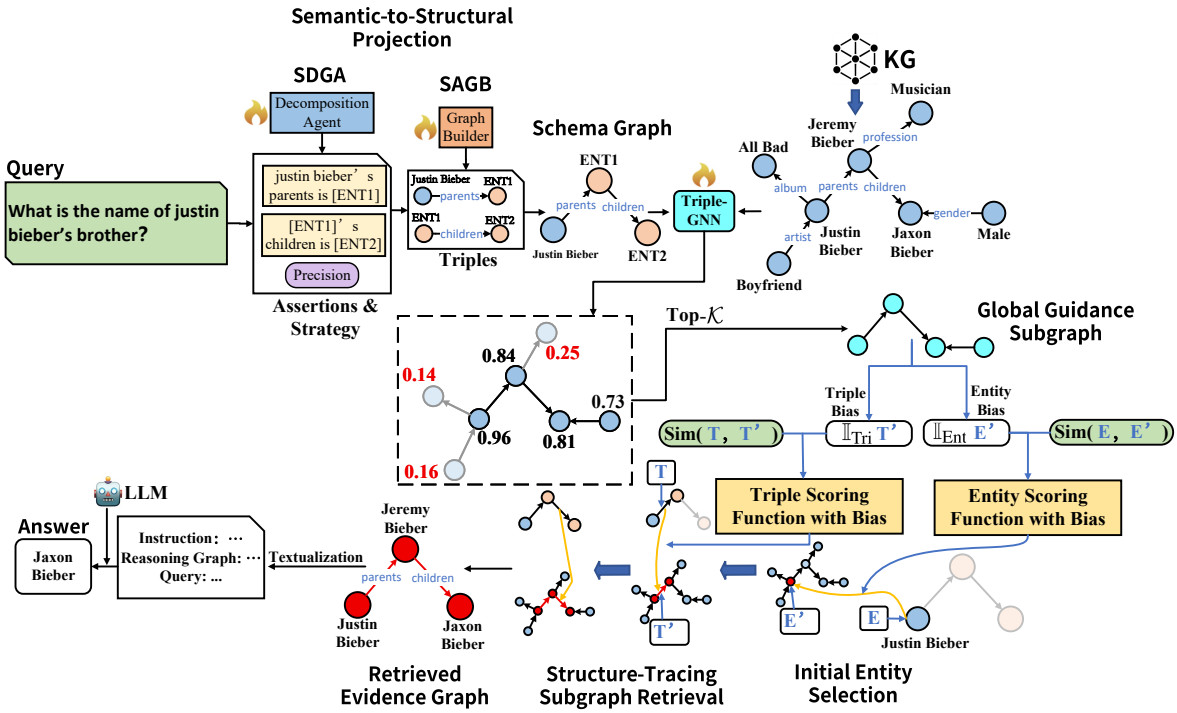

Figure 2: Overview of the STEM Framework.

[ENTX]). For instance, given the multi-hop query "Where is the arena stadium of the team whose mascot is Clutch the Bear?", the SGDA decomposes it into a coherent sequence of assertions sharing the bridging entity [ENT1]:

1.ENT1's mascot is Clutch the Bear

2.ENT1's arena stadium is [ENT2]

Answer Strategy. We consider multi-answer scenarios. For instance, the question "what are the four main languages spoken in Spain" corresponds to multiple potential answers, retrieving only a single evidence graph will result in incomplete answers. To address this, we categorize questions into two types: *Precision* and *Breadth*. The former corresponds to a definitive answer, while the latter involves multiple valid answers. The distinction between these two types will influence the behavior of subgraph matching, as detailed in subsequent sections. Since a single query may correspond to diverse potential KG logical structures, we employ beam search to enable the SGDA to generate $\mathcal{B}$ multiple candidate results, finally constructing a list of candidate assertion-strategy pairs $(\mathcal{S}_Q, \sigma)$ corresponding to different planning results, σ signifies the retrieval strategy assigned to Q,where $\sigma \in \{\text{“}Precision\text{”}, \text{“}Breadth\text{”}\}$.

3.2 Symbol-Aligned Graph Construction

While the SGDA successfully decomposes complex queries into atomic relational assertions, the objective of the SAGB is to perform symbolic grounding: mapping these textual assertions into precise structural triples. For instance, consider the assertion: "Darryl Sutter's hockey position is [ENT1].", A naive keyword match might fail to identify the correct edge due to lexical divergence (e.g., ("Darryl Sutter", "position", "[ENT1]")). However, possessing prior knowledge of the KG's symbolic representation, the SAGB accurately grounds this assertion into the standardized triple: ("Darryl Sutter", "ice_hockey.hockey_player.hockey_position", "[ENT1]").

Formally, for a given set of assertions and the prompt $\mathcal{P}_2$, denoted as $\mathcal{S} = \{s_1, s_2, , ..., s_N\}$, SAGB build a corresponding set of structural triples, represented as

$$\mathcal{T} = \{t_1, t_2, , ..., t_N\} \tag{2}$$

After obtaining the set of triples $\mathcal{T}$, we construct the schema graph $\mathcal{G}_{sch}$ by traversing each member of $\mathcal{T}$.

To develop these two modules, we propose a specialized data construction and training framework.

Furthermore, to enhance the KG knowledge injection and generalization capabilities of the SGDA and SAGB, we introduce Structure-to-Query Reverse Generation method for data augmentation. All details will be described in Appendix C.

In Appendix F, we provide concrete running examples to illustrate the end-to-end processing workflow of the Semantic-to-Structural Projection pipeline and the underlying principles of pattern acquisition during training.

3.3 Global Guidance Subgraph

Graph Neural Networks (GNNs) have been widely adopted for reasoning over KGs (Mavromatis and Karypis, 2025; Yasunaga et al., 2021; Liu et al., 2026). GFM-RAG (Luo et al., 2025) typically employs a Query-Dependent GNN, which incorporates query information to compute interaction-aware representations, enabling the ability to capture relevant entity knowledge within the graph structure. Based on this approach, we propose the Triple-Dependent GNN (Triple-GNN), which leverages the explicit structural relationship inherent in triples.

Formally, let $\mathcal{G}$ denote the query-specific subgraph[1] corresponding to Q, which serves as the knowledge graph for retrieval. $\mathcal{N}$ and $\mathcal{R}$ denote the set of entities and relations within $\mathcal{G}$, respectively, and $\mathcal{N}_Q$ denotes the set of all question entities in Q. After obtaining the structural triples $\mathcal{T}$ and schema graph $\mathcal{G}_{\text{sch}}$. For each triple $t \in \mathcal{T}$,we first employ a Pretrained Embedding Model (PEM) to generate its vector representation $\mathbf{E}_t \in \mathbb{R}^{d_{\text{GNN}}}$. A pooling operation is then applied to integrate all $\mathbf{E}_{t_i}$ $(t_i \in \mathcal{T})$ into a unified representation $\mathbf{E}_Q \in \mathbb{R}^{d_{\text{GNN}}}$. In the message passing stage, $\mathbf{E}_Q$ is fed into the L-layer Triple-GNN to derive the embedding representation $\mathbf{h}_e^L$ for each entity $e \in \mathcal{N}_Q$. We consolidate all entity representations into a single representation $\mathbf{H}_Q^L \in \mathbb{R}^{|\mathcal{N}| \times d_{\text{GNN}}}$. The overall computation process is described as follows:

$$\mathbf{E}_Q = \frac{1}{N} \sum_{i=0}^{N-1} \mathbf{E}_{t_i} \tag{3}$$

$$\mathbf{H}_Q^L = \text{Triple-GNN}(\mathcal{G}, \mathbf{H}_e^0, \mathbf{H}_r^0) \tag{4}$$

$\mathbf{H}_r^0 \in \mathbb{R}^{|\mathcal{R}| \times d_{\text{GNN}}}$ denotes the initialized feature representations of relations in $\mathcal{R}$. $\mathbf{H}_e^0 \in \mathbb{R}^{|\mathcal{N}| \times d_{\text{GNN}}}$ represents the initial feature input of entities, for any $\mathbf{h}_{e_i}^0 \in \mathbf{H}_e^0$, its initialization is defined as follows:

$$\mathbf{h}_{e_i}^0 = \begin{cases} \mathbf{E}_Q, & e_i \in \mathcal{N}_Q, \\ \mathbf{0}, & \text{otherwise.} \end{cases} \tag{5}$$

Further details regarding the initialization of the entity and relation inputs ($\mathbf{H}_e^0$ and $\mathbf{H}_r^0$), as well as the subsequent computations within the Triple-GNN, can be found in Appendix A.

After obtaining entity embeddings $\mathbf{H}_Q^L$ via Triple-GNN, we pass them through a linear projection layer, followed by a Sigmoid operation to derive the node probability distribution[2]:

$$\boldsymbol{P}_Q = \text{Sigmoid}(\text{MLP}\phi_1(\mathbf{H}_Q^L)) \in \mathbb{R}^{|\mathcal{N}|} \tag{6}$$

Based on the probability distribution of entities, we select the Top-$\mathcal{K}$ entities with the highest probabilities and construct a candidate entity list $\mathcal{N}_Q^{'}$:

$$\mathcal{N}_Q^{'} = \textbf{\textit{argTop}}\mathcal{K}_{\boldsymbol{p} \in \boldsymbol{P}_Q}(\boldsymbol{p}) \tag{7}$$

where the value of $\mathcal{K}$ is set to $\mid \mathcal{T} \mid *4$, where $\mid \mathcal{T} \mid$ is the number of triples in $\mathcal{T}$.

Upon obtaining the candidate entity list $\mathcal{N}_Q^{'}$, we anchor each entity $e_i^{'} \in \mathcal{N}_Q^{'}$ within $\mathcal{G}$ and construct a subgraph connected by their existing relations $\mathcal{R}$, yielding the final Global Guidance Subgraph denoted as $\mathcal{G}_Q^{'}$. The training details of Triple-GNN will be described in Appendix C.

3.4 Structure-Tracing Subgraph Retrieval

In this section, we introduce the Structure-Tracing Subgraph Retrieval module, the primary objective of this module is to identify a specific subgraph within $\mathcal{G}$ that exhibits high structural and semantic isomorphism[3] (Cai et al., 2025) to the query schema graph $\mathcal{G}_{\text{sch}}$. For each question entity $\hat{e} \in \mathcal{N}_Q$, we first retrieve the Top-N (N=50) most similar entity nodes $R_{\hat{e}}$ from $\mathcal{G}$, along with their corresponding cosine similarity scores $\boldsymbol{S}_{\hat{e}}$:

$$\mathcal{N}_{\hat{e}} = \textbf{\textit{argTopN}}_{e' \in \mathcal{N}}(\text{Sim}(\hat{e}, e')) \tag{8}$$

$$\boldsymbol{S}_{\hat{e}} = \{\text{Sim}(\hat{e}, e') | e' \in \mathcal{N}_{\hat{e}}\} \tag{9}$$

[1]We utilize the subgraph structures extracted by RoG (Luo et al., 2024). In their publicly available dataset, each query corresponds to a specific KG and a list of question entities, all of which are derived from Freebase.

[2]The subscript ϕ_x indicates that this module contains trainable parameters proposed in this paper.

[3]Following the approach of SimGRAG, we say that P and S are isomorphic if there exists a bijective mapping f : $V_P \rightarrow V_S$ s.t. an edge $\langle u, v \rangle$ exists in P if and only if the edge $\langle f(u), f(v) \rangle$ exists in S.

Global Structural Consistency Bias. Since **Sim** is a shallow entity-to-entity semantic similarity function, we design a global-prior score rectification mechanism by incorporating the structural priors from Guidance Graph $\mathcal{G}_Q^{'}$. Specifically, for the aforementioned score $\boldsymbol{S}_{\hat{e}}$, the score rectification is applied as follows:

$$\begin{aligned}\boldsymbol{S}_{\hat{e}}^{*} &= \boldsymbol{S}_{\hat{e}} * \mathbb{I}_{\text{Ent}}(\mathcal{N}_{\hat{e}}) \\ &= \{\text{Sim}(\hat{e}, e') * \mathbb{I}_{\text{Ent}}(e') | e' \in \mathcal{N}_{\hat{e}}\} \end{aligned} \tag{10}$$

where $\mathbb{I}_{\text{Ent}}$ is **Entity-level Global Structural Consistency Bias**, and $\mathbb{I}_{\text{Ent}}$ is defined as follows:

$$\mathbb{I}_{\text{Ent}}(e) = \begin{cases} \mathbf{\frac{3}{2}}, & e \in \mathcal{N}_Q^{'}, \\ \mathbf{1}, & \text{otherwise.} \end{cases} \tag{11}$$

This means that if an entity e exists in the Global Guidance Subgraph $\mathcal{G}_Q^{'}$, it is considered more important for the query reasoning structure.

To initiate the subgraph matching search, we first establish a starting anchor mapping between the schema graph $\mathcal{G}_{\text{sch}}$ and the knowledge graph $\mathcal{G}$. Specifically, for the question entity $\hat{e}$, we identify its counterpart node e^* in $\mathcal{G}$ by selecting the highest-scoring candidate from $\mathcal{N}_{\hat{e}}$ based on $\boldsymbol{S}_{\hat{e}}^{*}$. Simultaneously, we locate the corresponding node e within $\mathcal{G}_{\text{sch}}$ corresponding to the entity $\hat{e}$ via fuzzy string matching. This establishes the starting pair (e, e^*). Proceeding from this anchor, we execute the structure-tracing matching: for a specific edge (e, r, e')[4] defined in $\mathcal{G}_{\text{sch}}$, we seek a structurally and semantically matching edge $(e^*, r^*, e^{*'})$ within $\mathcal{G}$. This matching process is guided by calculating the globally-aware triple score T-Score defined as follows:

$$\begin{aligned}&\text{T-Score}((e, r, e'), (e^*, r^*, e^{*'})) = \\ &\text{Sim}(\text{Encoder}(e, r, e'), \text{Encoder}(e^*, r^*, e^{*'})) \\ &+\mathbb{I}_{\text{Tri}}((e^*, r^*, e^{*'}))\end{aligned} \tag{12}$$

where $\text{Sim}(t_i, t_j) = \frac{\mathbf{E}_{t_i} \cdot \mathbf{E}_{t_j}}{\|\mathbf{E}_{t_i}\| \|\mathbf{E}_{t_j}\|}$. Similar to entity nodes retrieval, we incorporate a structural constraint based on the Global Guidance Subgraph $\mathcal{G}_Q^{'}$ by integrating with a **Triple-level Global Structural Consistency Bias** $\mathbb{I}_{\text{Tri}}$. The definition of $\mathbb{I}_{\text{Tri}}$ denotes an Triple-level structural constraint, defined as follows:

$$\mathbb{I}_{\text{Tri}}(t) = \begin{cases} \mathbf{\frac{1}{2}}, & t \in \mathcal{G}_Q^{\prime}, \\ \mathbf{0}, & \text{otherwise.} \end{cases} \tag{13}$$

The above describes the method for obtaining edge mappings through single-step reasoning using T-Score. For the entire graph reasoning process, recursive matching is performed until a concrete subgraph $\mathcal{G}_{\text{sch}}^{*}$ that is structurally isomorphic to $\mathcal{G}_{\text{sch}}$. Specifically, at step i of the matching process, the cumulative score being $\mathcal{S}_i$, the score for the next step $\mathcal{S}_{i+1}$ is computed as follows:

$$\mathcal{S}_{i+1} = \mathcal{S}_i + \text{T-Score}(t_{i+1}, t_{i+1}^{*}) \tag{14}$$

$$\mathcal{S}_0 = \boldsymbol{S}_e^{*}[e_0^{*}] \tag{15}$$

It is important to note that although the KG is directed, the matching process operates in an undirected context. In other words, we do not distinguish between incoming and outgoing edges in the matching selection. We will provide a detailed description of this algorithm in the Appendix H.

3.5 Retrieval Behaviors of Different Strategies

As discussed in Section 3.1, multi-hop questions can be categorized into *Precision* and *Breadth* types. To accommodate these distinct reasoning requirements, STEM adopts an adaptive search strategy that dynamically adjusts the edge selection behavior during the Structure-Tracing Subgraph Retrieval process. Specifically, for *Precision* strategy, we employ a Greedy Selection mechanism by selecting strictly the edge with the maximum score $\mathcal{S}$. For *Breadth* strategy, we employ a Threshold-Based Selection mechanism by retaining all candidate edges whose scores $\mathcal{S}$ exceed a pre-defined confidence threshold θ. In fact, Threshold-Based Selection allows the structure tracing to branch out, transforming the linear search path into a search tree.

We present detailed illustrations of both selection strategies and experiments to analyze the their impact on performance in the Appendix D.1.

3.6 Generation

Upon completing the subgraph retrieval, we obtain a query-specific evidence subgraph $\mathcal{G}_{\text{reason}}$ by integrating the resulting subgraphs from all search processes. To linearize this graph structure into a format compatible with LLM prompting, we perform Depth-First Search starting from the question

[4]**Note:** For brevity and to explicitly delineate the constituent nodes of each edge, we uniformly use (e, r, e') to denote both the triple t and the edge corresponding to relation r.

Model	WebQSP		CWQ	
	Hit@1	F_1 Score	Hit@1	F_1 Score
GPT-4o				
GPT-4o	61.8	43.6	38.2	32.9
GPT-4o+Fewshot	71.68	63.7	57.59	44.72
GPT-4o+CoT	74.12	64.25	59.36	48.24
With Finetuning				
NSM	74.31	-	53.92	-
DeCAF$_{FiD-3B}$	82.1	-	70.42	-
KD-CoT$_{T5-large}$	73.7	50.2	50.5	-
RoG$_{Llama2-Chat-7B}$	83.15	69.81	61.39	56.17
RoG$_{Llama-3.1-70B}$	86.1	68.87	67.43	60.3
RoG$_{GPT-4o}$	88.09	70.12	69.61	61.97
LightProf$_{Llama3-8B}$	83.8	-	59.3	-
GRAG$_{LLaMA2-7B}$	72.75	50.41	-	-
GNN-RAG$_{Llama2-7B}$	86.4	69.0	67.3	59.1
With Prompting				
Kaping$_{gpt-3.5-turbo}$	72.42	65.12	53.42	50.32
ToG$_{gpt-3.5-turbo}$	75.08	72.32	57.59	56.64
G-Ret$_{Llama2-7B}$	70.16	50.23	-	-
PoG$_{gpt-3.5}$	82.0	-	63.2	-
ReKnoS$_{gpt-3.5}$	81.1	-	58.5	-
MFC$_{gpt-4o-mini}$	78.9	-	62.8	-
SubgraphRAG$_{gpt-4o}$	83.1	-	56.3	-
FiDeLiS$_{gpt-4-turbo}$	84.39	**78.32**	71.47	64.32
ProgRAG$_{gpt-4o-mini}$	90.4	-	73.3	-
Our Proposed Method				
STEM$_{Llama-3.1-8B}$	86.63	71.05	68.76	60.81
STEM$_{Llama-3.1-70B}$	88.08	74.62	72.53	62.09
STEM$_{GPT-4o}$	**90.94**	76.18	**74.09**	**65.33**

Table 1: Comparison of different models on WebQSP and CWQ datasets.

entity nodes within $\mathcal{G}_{\text{reason}}$ to flatten the subgraph into a set of coherent reasoning chains, finally obtain $\mathcal{C}_{\text{reason}}$. We apply prompt $\mathcal{P}_3$ as an LLM instruction and take $\mathcal{P}_3$, Q and $\mathcal{C}_{\text{reason}}$ as input, to infer the final answer:

$$\mathcal{A} \leftarrow \text{LLM}_\theta(\mathcal{P}_3, Q, \mathcal{C}_{\text{reason}}) \tag{16}$$

4 Experiments

In this section, we present the experimental results and analysis. We conduct the experimental process from the following aspects: (1) How does STEM perform on multi-hop reasoning tasks compared to existing classical and state-of-the-art methods? (2) A fine-grained performance analysis across varying answer numbers, reasoning depths, and underlying reasoning models. (3) Ablation studies on the Semantic-to-Structural Projection pipeline and the Global Structural Consistency Bias. Detailed implementation regarding the training of SGDA, SAGB, and Triple-GNN—including data construction processes and training configurations—is provided in Appendix C.

Furthermore, we will describe the details of the experimental setup, test datasets, evaluation metrics, baselines, and other experiments and analyses in the Appendix B and Appendix D. Additionally, a systematic analysis of failure modes and error propagation across the STEM pipeline is detailed in Appendix E.

4.1 Main Results

As described in Table 1, the comparative experimental results demonstrates that STEM achieves a significant performance improvement over other models across both datasets. First, compared to methods relying solely on GPT-4o (OpenAI, 2024), STEM exhibits an increase of over 10% in both Hit@1 and F1 Score on both datasets. When compared to fine-tuned models, STEM + Llama-3.1-8B outperforms other baselines with similar parameter scales, including RoG, LightProf and GNN-RAG. The lead is particularly notable on CWQ, where Hit@1 improves by approximately 6% compared to RoG. Remarkably, it even surpasses the RoG model utilizing the larger Llama-3.1-70B backbone. Meanwhile, STEM + Llama-3.1-70B achieves even greater margins, boosting the F1 score on WebQSP by about 6% and Hit@1 on CWQ by 5% compared to RoG + Llama-3.1-70B. When compared with prompting-based methods, STEM + GPT-4o demonstrates a significant advantage over other approaches utilizing the GPT series. Specifically, on WebQSP, it improves Hit@1 by approximately 7% over the highly competitive baseline, FiDeLiS, although its F1 score is slightly lower. On CWQ, STEM yields distinct improvements across all metrics. Ultimately, STEM + GPT-4o achieves SOTA performance on three out of the four evaluated metrics, with the exception of the F1 score on WebQSP.

4.2 Performance Analysis

We partition the test set based on the number of answers and evaluate performance on each subset, with the results shown in Table 2. It is evident that STEM significantly outperforms both RoG and GNN-RAG across all answer count categories. Notably, STEM achieves an improvement of approximately 4% on the WebQSP subset with answers ≥ 10 and a 9% gain on the CWQ subset with answers in [2, 4]. These results demonstrate STEM's superior performance in ensuring comprehensive coverage for multi-answer queries.

We stratify the performance by reasoning hop number, with the results presented in Table 3. It

Method	Dataset	Ans = 1	Ans ∈ [2,4]	Ans ∈ [5,9]	Ans ≥ 10
RoG	**WebQSP**	67.89	79.39	75.04	58.33
	CWQ	56.90	53.73	58.36	43.62
GNN-RAG	**WebQSP**	71.24	76.30	74.06	56.28
	CWQ	60.40	55.52	61.49	50.08
STEM+GPT-4o	**WebQSP**	75.26	81.87	78.38	62.46
	CWQ	65.32	64.35	66.37	53.86

Table 2: Detailed results (F1) grouped by the number of answers.

Method	Dataset	Hop = 1	Hop = 2	Hop ≥ 3
RoG	**WebQSP**	77.03	64.86	-
	CWQ	62.88	58.46	37.82
GNN-RAG	**WebQSP**	72.0	69.8	-
	CWQ	47.4	69.4	51.8
STEM+GPT-4o	**WebQSP**	81.49	75.35	-
	CWQ	67.46	66.73	52.15

Table 3: Detailed results (F1) grouped by the maximum reasoning hop number.

Method	Reasoning Model	WebQSP		CWQ	
		Hit@1	F_1 Score	Hit@1	F_1 Score
RoG	LLaMA2-Chat-7B	83.15	69.81	61.39	56.17
	Llama-3.1-70B-Ins	86.1	68.87	67.43	60.3
	GPT-4o	88.09	70.12	69.61	61.97
STEM	Llama-3.1-70B-Ins	88.08	74.62	72.53	62.09
	GPT-4o	90.94	76.18	74.09	65.33

Table 4: Impact of reasoning models on performance.

is evident that STEM generally maintains a strong competitive edge. However, a notable exception is observed on the CWQ (hop=2), where our method lags significantly behind GNN-RAG.

We conducted a comparative analysis of RoG and STEM using different reasoning models while keeping the retrieval pipeline constant, the results are presented in Table 4. As the results demonstrate, although replacing the original LLaMA2-Chat-7B used in RoG with Llama-3.1-70B-Ins or GPT-4o yields performance gains, STEM maintains a competitive edge under identical reasoning model configurations.

However, we acknowledge a potential ambiguity in this analysis: the performance improvements might stem partially from the superior parametric knowledge of these advanced models, rather than solely from enhancements in reasoning capability. To investigate this further, we conducted an additional experiment to assess the coverage rate of evidence subgraph retrieval. Specifically, we calculated the proportion of the ground-truth reasoning path covered by the evidence subgraph obtained via Structure-Tracing Subgraph Retrieval. Specifically, given a question Q, let $\mathcal{R}$ denote the ground-truth reasoning path of Q, and $\mathcal{G}_{\text{reason}}$ denote the evidence subgraph. Coverage rate is defined as:

$$\text{Coverage\_rate} = \frac{|\ \mathcal{R} \cap \mathcal{G}_{\text{reason}}\ |}{|\ \mathcal{R}\ |} \tag{17}$$

The results are presented in Table 5, indicating that the coverage rate of the evidence subgraph gradually decreases as the number of answers increases, yet it remains at a relatively high level. Furthermore, the coverage on CWQ is consistently lower than on WebQSP due to the question complexity.

4.3 Ablation study

Semantic-to-Structural Projection Pipeline. We conduct ablation studies by comparing our proposed Semantic-to-Structural Projection pipeline against powerful off-the-shelf LLMs to evaluate its effectiveness[5]. The comparative results are presented in Table 6. Our method demonstrates a significant performance advantage over the baseline models, surpassing the strongest competitor by over 23% on the CWQ dataset. Among the baselines, GPT-4o consistently outperforms Llama-3.1-70B-Ins, reflecting its superior few-shot KG alignment. Our results validate the critical role of logic-aware projection in KG reasoning.

Global Structural Consistency Bias. As the Global Guidance Subgraph serves as a critical structural prior, we conducted an ablation study by selectively removing its bias terms. We evaluated three variants: **w/o Entity Bias** (removing the calculation of the indicator term $\mathbb{I}_{\text{Ent}}$ in Equation 10), **w/o Triple Bias** (removing the calculation of $\mathbb{I}_{\text{Tri}}$ from Equation 12), and **w/o Both**.

The ablation results are presented in Table 7. We observe that incorporating both Entity-level and Triple-level Biases yields significant performance gains. Conversely, removing Triple-level Bias leads to a marked decline, particularly in the Hit@1 metric on WebQSP and across all metrics on CWQ, with performance drops reaching up to 10% on CWQ. Removing both biases causes further degradation, with the Hit@1 score on CWQ dropping by an additional 3% compared to the w/o Entity Bias setting, while the F1 score on WebQSP plummets by nearly 5%. Furthermore, the w/o Triple-level setting yields inferior perfor-

[5]For baseline LLMs, we implement the pipeline using a few-shot prompting approach, adopting the identical prompt templates employed by the SGDA and SAGB. Regarding the answer strategy, if a valid strategy cannot be successfully extracted after 5 retries, we default to *Precision* for the current query.

Dataset	Ans = 1	Ans ∈ [2,4]	Ans ∈ [5,9]	Ans ≥ 10
WebQSP	81.9	76.64	71.45	58.23
CWQ	74.28	66.57	62.71	51.89

Table 5: Coverage Rate (%) of ground-truth reasoning paths within evidence subgraphs.

Pipelines	WebQSP		CWQ	
	Hit@1	F_1 Score	Hit@1	F_1 Score
Llama-3.1-70B-Ins	77.74	61.21	46.68	41.83
GPT-4o	83.14	65.77	50.43	43.2
Our Pipeline	90.94	76.18	74.09	65.33

Table 6: Comparison of different Question Planing Pipelines. **Llama-3.1-70B-Ins** and **GPT-4o** denote baselines where the corresponding models are employed to perform decomposition and schema graph construction. The reasoning model consistently uses GPT-4o.

mance compared to the w/o Entity-level, indicating that the Triple-level bias plays a more critical role. These results revealing the limitations of relying solely on local semantic matching. Appendix D.3 details its error-correction analysis.

We conducted a comprehensive set of additional experiments that are equally critical to validating the robustness of STEM; however, due to space constraints, these results are detailed in Appendix D. The supplementary evaluations encompass fine-grained performance analyses and sensitivity studies on key hyperparameters, including the Answer Strategy σ, Initial Entity Count $\mathcal{K}$, and SGDA Beam Size $\mathcal{B}$, as well as parameter tuning for the Global Structural Consistency Bias $\mathbb{I}_{\text{Ent}}$ and $\mathbb{I}_{\text{Tri}}$. Furthermore, we provide in-depth Case Studies, Efficiency Analysis, and Interpretability Analysis.

5 Related Work

5.1 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) has emerged as a dominant paradigm to mitigate the hallucination issues of Large Language Models (LLMs) (Lewis et al., 2020; Guu et al., 2020; Karpukhin et al., 2020; Izacard et al., 2022; Asai et al., 2024). Adaptive-RAG (Jeong et al., 2024) dynamically selects retrieval strategies based on query complexity. Similarly, Corrective RAG (CRAG) (Yan et al., 2024) incorporates a lightweight evaluator to filter irrelevant documents and trigger web searches as a fallback.

5.2 Multi-hop Reasoning in RAG

For multi-step deduction, research has evolved from single-step retrieval to Iterative and Chain-of-Thought (CoT) Retrieval (Trivedi et al., 2023). ReAct (Yao et al., 2023) further generalizes this by modeling LLMs as agents that can perform search actions. Chain-of-Note (Yu et al., 2024) generates sequential reading notes to evaluate document relevance before aggregation. Demonstrate-Search-Predict (DSP) (Khattab et al., 2022) uses frozen LMs to orchestrate sophisticated retrieval pipelines via natural language programs. Tree of Clarifications (Kim et al., 2023) constructs a tree of disambiguations to handle ambiguous questions recursively.

Scoring Bias	WebQSP		CWQ	
	Hit@1	F_1 Score	Hit@1	F_1 Score
STEM$_{GPT-4o}$	90.94	76.18	74.09	65.33
w/o $\mathbb{I}_{\text{Ent}}$ & $\mathbb{I}_{\text{Tri}}$	86.31	70.80	63.91	55.59
w/o $\mathbb{I}_{\text{Ent}}$	86.45	75.81	66.35	57.35
w/o $\mathbb{I}_{\text{Tri}}$	86.95	73.45	64.90	56.42

Table 7: Ablation Study on Entity-level and Triple-level Scoring Biases.

5.3 Knowledge Graph-based RAG

Knowledge Graphs offer a promising solution to the reasoning disconnection problem to encode graph structures (Yasunaga et al., 2021; Zhang et al., 2022), but often struggled to scale or integrate flexibly with LLMs. GraphRAG (Edge et al., 2024) introduces a community-detection-based approach to generate hierarchical summaries of the graph for global query understanding. For multi-hop reasoning, GNN-RAG (Mavromatis and Karypis, 2025) combines GNN-based retrieval with LLM reasoning to handle complex graph topology. Other works like StructGPT (Jiang et al., 2023b) and KAPING (Baek et al., 2023) explore zero-shot prompting strategies to interface LLMs with structured data.

6 Conclusion

We presented Structure-Tracing Evidence Mining, a novel framework that shifts multi-hop KG-RAG from sequential path finding to holistic subgraph matching. By synergizing a fine-tuned Semantic-to-Structural Projection pipeline with a Triple-Dependent GNN, STEM effectively bridges the gap between natural language and KG schemas, retrieving logically connected evidence subgraphs and align with the knowledge structure. Extensive experiments on WebQSP and CWQ demonstrate that STEM significantly outperforms existing baselines, achieving new State-of-the-Art results.

Limitations

Although STEM effectively bridges the semantic-structural gap, challenges persist due to the inherent diversity of KG topologies. First, in highly complex reasoning tasks, planning deviations may still occur, leading to scenarios where all generated candidate schema graphs fail to match the factual structure, thereby resulting in retrieval failure. Second, STEM relies on domain-specific fine-tuning and access to the target KG's structure. While achieving strong performance on Freebase-based benchmarks (WebQSP, CWQ), we acknowledge it is not a general-purpose zero-shot method; this dependency limits its transferability to unseen KGs or novel schema types. Finally, regarding efficiency, the threshold-based expansion in the *Breadth* strategy increases computational latency, we consider this a necessary trade-off for answer exhaustiveness. Moreover, this expansion is selective—occurring not at every step, but exclusively when the retrieval of multiple simultaneous answer paths is required.

Ethical Considerations

We address the ethical considerations and potential risks as follows:

Data Provenance and Licensing. The knowledge graph utilized in this study, is a widely adopted, publicly available database distributed under the CC-BY license. Our usage of WebQSP and CWQ datasets strictly adheres to their respective data usage policies and licenses. These datasets are standard benchmarks in the research community and do not contain private, personally identifiable information (PII), or offensive content that would require special redaction for this study.

Bias and Fairness. We acknowledge that KGQA systems are susceptible to propagating biases inherent in their underlying knowledge sources. Specifically, the Freebase KG is known to exhibit significant demographic, cultural, and geographical skews. Since STEM is designed to be faithful to the retrieved subgraph, it inevitably reflects the distributional properties of the source KG. Therefore, users should interpret the outputs of STEM as a reflection of the facts stored in the specific Knowledge Graph, rather than as an unbiased representation of real-world truth.

Acknowledgments

We would like to thank the Action Editors and the anonymous reviewers for their constructive feedback and insightful comments, which helped improve the quality of this paper.

References

Anthropic. 2024. The Claude 3 model family: Opus, Sonnet, and Haiku. *Anthropic Technical Report*.

Tu Ao, Yanhua Yu, Yuling Wang, Yang Deng, Zirui Guo, Liang Pang, Pinghui Wang, Tat-Seng Chua, Xiao Zhang, and Zhen Cai. 2025. LightPROF: A lightweight reasoning framework for large language model on knowledge graph. In *Thirty-Ninth AAAI Conference on Artificial Intelligence, AAAI 2025*, pages 23424–23432. AAAI Press.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations, ICLR 2024*.

Jinheon Baek, Alham Fikri Aji, and Amir Saffari. 2023. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. *CoRR*, abs/2306.04136.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, and 29 others. 2023. Qwen technical report. *CoRR*, abs/2309.16609.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, and 9 others. 2022. Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning, ICML 2022*, pages 2206–2240. PMLR.

Yuzheng Cai, Zhenyue Guo, YiWen Pei, WanRui Bian, and Weiguo Zheng. 2025. SimGRAG: Leveraging similar subgraphs for knowledge graphs driven retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 3139–3158, Vienna, Austria. Association for Computational Linguistics.

Liyi Chen, Panrong Tong, Zhongming Jin, Ying Sun, Jieping Ye, and Hui Xiong. 2024. Plan-on-Graph: Self-correcting adaptive planning of large language model on knowledge graphs. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, and 48 others. 2023. PaLM: Scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24:240:1–240:113.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From Local to Global: A graph RAG approach to query-focused summarization. *CoRR*, abs/2404.16130.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: retrieval-augmented language model pre-training. *CoRR*, abs/2002.08909.

Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In *WSDM '21, The Fourteenth ACM International Conference on Web Search and Data Mining*, pages 553–561. ACM.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-Retriever: Retrieval-augmented generation for textual graph understanding and question answering. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. 2025. GRAG: Graph retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 4145–4157, Albuquerque, New Mexico. Association for Computational Linguistics.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Trans. Mach. Learn. Res.*, 2022.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C. Park. 2024. Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. *CoRR*, abs/2403.14403.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023a. Mistral 7B. *CoRR*, abs/2310.06825.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Xin Zhao, and Ji-Rong Wen. 2023b. 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*, pages 9237–9251, Singapore. Association for Computational Linguistics.

Dhiraj D. Kalamkar, Dheevatsa Mudigere, Naveen Mellempudi, Dipankar Das, Kunal Banerjee, Sasikanth Avancha, Dharma Teja Vooturi, Nataraj Jammalamadaka, Jianyu Huang, Hector Yuen, Jiyan Yang, Jongsoo Park, Alexander Heinecke, Evangelos Georganas, Sudarshan Srinivasan, Abhisek Kundu, Misha Smelyanskiy, Bharat Kaul, and Pradeep Dubey. 2019. A study of BFLOAT16 for deep learning training. *CoRR*, abs/1905.12322.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. 2022. Demonstrate-Search-Predict: Composing retrieval and language models for knowledge-intensive NLP. *CoRR*, abs/2212.14024.

Gangwoo Kim, Sungdong Kim, Byeongguk Jeon, Joonsuk Park, and Jaewoo Kang. 2023. Tree of Clarifications: Answering ambiguous questions with retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 996–1009, Singapore. Association for Computational Linguistics.

Yunshi Lan and Jing Jiang. 2020. Query graph generation for answering multi-hop complex questions from knowledge bases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 969–974, Online. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*.

Mufei Li, Siqi Miao, and Pan Li. 2025. Simple is Effective: The roles of graphs and large language models in knowledge-graph-based retrieval-augmented generation. In *The Thirteenth International Conference on Learning Representations, ICLR 2025*.

Yu Liu, Xixun Lin, Yanmin Shang, Yangxi Li, Shi Wang, and Yanan Cao. 2026. PathMind: A retrieve-prioritize-reason framework for knowledge graph reasoning with large language models. In *Fortieth AAAI Conference on Artificial Intelligence, AAAI 2026*, pages 15386–15393. AAAI Press.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019*.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2024. Reasoning on Graphs: Faithful and interpretable large language model reasoning. In *The Twelfth International Conference on Learning Representations, ICLR 2024*.

Linhao Luo, Zicheng Zhao, Gholamreza Haffari, Dinh Q. Phung, Chen Gong, and Shirui Pan. 2025. GFM-RAG: graph foundation model for retrieval augmented generation. *CoRR*, abs/2502.01113.

Costas Mavromatis and George Karypis. 2025. GNN-RAG: Graph neural retrieval for efficient large language model reasoning on knowledge graphs. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 16682–16699, Vienna, Austria. Association for Computational Linguistics.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

OpenAI. 2024. GPT-4o system card. *CoRR*, abs/2410.21276.

Minbae Park, Hyemin Yang, Jeonghyun Kim, Kunsoo Park, and Hyunjoon Kim. 2026. ProgRAG: Hallucination-resistant progressive retrieval and reasoning over knowledge graphs. In *Fortieth AAAI Conference on Artificial Intelligence, AAAI 2026*, pages 32674–32682. AAAI Press.

Yuan Sui, Yufei He, Nian Liu, Xiaoxin He, Kun Wang, and Bryan Hooi. 2025. FiDeLiS: Faithful reasoning in large language models for knowledge graph question answering. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 8315–8330, Vienna, Austria. Association for Computational Linguistics.

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 *The Twelfth International Conference on Learning Representations, ICLR 2024*.

Yawei Sun, Lingling Zhang, Gong Cheng, and Yuzhong Qu. 2020. SPARQA: Skeleton-based semantic parsing for complex questions over knowledge bases. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, pages 8952–8959. AAAI Press.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 641–651, New Orleans, Louisiana. Association for Computational Linguistics.

Gemini Team. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *CoRR*, abs/2507.06261.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.

Junhong Wan, Tao Yu, Kunyu Jiang, Yao Fu, Weihao Jiang, and Jiang Zhu. 2025. Digest the knowledge: Large language models empowered message passing for knowledge graph question answering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15426–15442, Vienna, Austria. Association for Computational Linguistics.

Keheng Wang, Feiyu Duan, Sirui Wang, Peiguang Li, Yunsen Xian, Chuantao Yin, Wenge Rong, and Zhang Xiong. 2023. Knowledge-Driven CoT: Exploring faithful reasoning in LLMs for knowledge-intensive question answering. *CoRR*, abs/2308.13259.

Song Wang, Junhong Lin, Xiaojie Guo, Julian Shun, Jundong Li, and Yada Zhu. 2025. Reasoning of large language models over knowledge graphs with super-relations. In *The Thirteenth International Conference on Learning Representations, ICLR 2025*.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. 2024. Corrective retrieval augmented generation. *CoRR*, abs/2401.15884.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *3rd International Conference on Learning Representations, ICLR 2015*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023*.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. QA-GNN: Reasoning with language models and knowledge graphs for question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 535–546, Online. Association for Computational Linguistics.

Xi Ye, Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, and Caiming Xiong. 2022. RNG-KBQA: Generation augmented iterative ranking for knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6032–6043, Dublin, Ireland. Association for Computational Linguistics.

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, ACL 2016*. The Association for Computer Linguistics.

Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Yang Wang, Zhiguo Wang, and Bing Xiang. 2023. DecAF: Joint decoding of answers and logical forms for question answering over knowledge bases. In *The Eleventh International Conference on Learning Representations, ICLR 2023*.

Wenhao Yu, Hongming Zhang, Xiaoman Pan, Peixin Cao, Kaixin Ma, Jian Li, Hongwei Wang, and Dong Yu. 2024. Chain-of-Note: Enhancing robustness in retrieval-augmented language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14672–14685, Miami, Florida, USA. Association for Computational Linguistics.

Bo Zhang, Jianghua Zhu, Chaozhuo Li, Hao Yu, Li Kong, Zhan Wang, Dezhuang Miao, Xiaoming Zhang, and Junsheng Zhou. 2025a. What is a good question? assessing question quality via meta-fact checking. In *Thirty-Ninth AAAI Conference on Artificial Intelligence, AAAI 2025*, pages 15248–15256. AAAI Press.

Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D. Manning, and Jure Leskovec. 2022. GreaseLM: Graph reasoning enhanced language models. In *The Tenth International Conference on Learning Representations, ICLR 2022*.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025b. Qwen3 Embedding: Advancing text embedding and reranking through foundation models. *CoRR*, abs/2506.05176.

Ruilin Zhao, Feng Zhao, and Hong Zhang. 2025. Correcting on graph: Faithful semantic parsing over knowledge graphs with large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 5364–5376, Vienna, Austria. Association for Computational Linguistics.

A Triple-GNN Implementation

In this section, we describe the overall execution process of the Triple-GNN. Specifically, we first employ an Pretrained Embedding Model (PEM) to obtain vector representations for both the triples and the entities:

$$\mathbf{E}_x = \text{Encoder}(x) \in \mathbb{R}^{d_{\text{PEM}}} \tag{18}$$

$$\mathbf{E}_t = \text{MLP}_{\phi_2}([\mathbf{E}_e; \mathbf{E}_r; \mathbf{E}_{e'}]) \in \mathbb{R}^{d_{\text{GNN}}} \tag{19}$$

for all $x \in \{e, r, e'\}$, where the terms e, r and e' represent the head entity, relation, and tail entity of the triple t, respectively. The [;] operation denotes the concatenation of the embedding vectors, transforming the dimensionality from $\mathbb{R}^{d_{\text{PEM}}} \rightarrow \mathbb{R}^{3d_{\text{PEM}}}$.

Subsequently, the MLP operation refers to a linear transformation that maps the concatenated vector from $\mathbb{R}^{3d_{\text{PEM}}} \rightarrow \mathbb{R}^{d_{\text{GNN}}}$. We finally obtain the embedding representation $\mathbf{E}_t$ of the triple t.

Given the structured triples $\mathcal{T} = \{t_1, t_2, ..., t_N\}$ obtained from the Semantic-to-Structural Projection pipeline based on query Q, with the corresponding knowledge graph $\mathcal{G}$, entity set $\mathcal{N}$ and relation set $\mathcal{R}$, we first compute the embedding for each triple to yield $\mathbf{E}_{\mathcal{T}} = \{\mathbf{E}_{t_1}, \mathbf{E}_{t_2}, ..., \mathbf{E}_{t_N}\}$, then aggregate the embedding set into a single representation $\mathbf{E}_Q \in \mathbb{R}^{d_{\text{GNN}}}$ via average pooling, and finally process it through the L-layer Triple-GNN to obtain $\mathbf{H}_Q^L \in \mathbb{R}^{|\mathcal{N}| \times d_{\text{GNN}}}$:

$$\mathbf{E}_Q = \frac{1}{N} \sum_{i=1}^{N} \mathbf{E}_{t_i} \tag{20}$$

$$\mathbf{H}_Q^L = \text{Triple-GNN}_{\phi_3}(\mathcal{G}, \mathbf{H}_e^0, \mathbf{H}_r^0) \tag{21}$$

$\mathbf{H}_e^0$ represents the initial feature input of entities and $\mathbf{H}_e^0 \in \mathbb{R}^{|\mathcal{N}| \times d_{\text{GNN}}}$, for any $\mathbf{h}_{e_i}^0 \in \mathbf{H}_e^0$, its initialization method is defined as follows:

$$\mathbf{h}_{e_i}^0 = \begin{cases} \mathbf{E}_Q, & e_i \in \mathcal{N}_Q, \\ \mathbf{0}, & \text{otherwise.} \end{cases} \tag{22}$$

where $\mathcal{N}_Q$ denotes the set of question entities for question Q.

$\mathbf{H}_r^0 \in \mathbb{R}^{|\mathcal{R}| \times d_{\text{GNN}}}$ denotes the initialized feature representations of relations in $\mathcal{R}$. For each $\mathbf{h}_{r_i}^0 \in \mathbf{H}_r^0$, we initialize it using the same Encoder employed for the triple embeddings, followed by an MLP for linear projection, as defined below:

$$\mathbf{z}_{r_i} = \text{Encoder}(r_i) \in \mathbb{R}^{d_{\text{PEM}}}, \tag{23}$$

$$\mathbf{h}_{r_i}^0 = \text{MLP}_{\phi_4}(\mathbf{z}_{r_i}) \in \mathbb{R}^{d_{\text{GNN}}}, \quad \forall r_i \in \mathcal{R} \tag{24}$$

the MLP projects the encoded relation representation from $\mathbb{R}^{d_{\text{PEM}}}$ into the $\mathbb{R}^{d_{\text{GNN}}}$ space.

We now describe the message passing computational flow of a single layer of Triple-GNN. Building upon the Query-Dependent GNN design proposed by GFM-RAG (Luo et al., 2025), for the embedding representation $\mathbf{h}_e^{L-1}$ of node e obtained at the (L-1)-th GNN layer, the representation at the L-th layer is computed as follows:

$$\mathbf{m}_e^L = \text{MSG}(\mathbf{h}_e^{L-1}, g^L(\mathbf{h}_r^{L-1}), \mathbf{h}_{e'}^{L-1}), \quad (25)$$

$$\mathbf{h}_e^L = \text{Update}(\mathbf{h}_e^{L-1}, \text{Agg}(\mathcal{M}_e^L)) \quad (26)$$

where $(e, r, e') \in \mathcal{G}$, and $\mathcal{M}_e^L = \{\mathbf{m}_{e'}^L | e' \in \text{N}_r(e), r \in \mathcal{R}\}$. The set $\text{N}_r(e)$ denotes the collection of all neighboring nodes of entity e under relation $r \in \mathcal{R}$.

In the context of a triple (e, r, e') associated with the entity e, the notations $\mathbf{h}_e^{L-1}$, $\mathbf{h}_r^{L-1}$, and $\mathbf{h}_{e'}^{L-1}$ represent the embedding representations of the head entity e, the relation r, and the tail entity e' at layer L-1, respectively. The operation denoted as MSG employs a DistMult (Yang et al., 2015) function to process the triple. The function g^L constitutes a 2-layer MLP operation at the subsequent layer L. The Agg operation collects the states $\mathbf{m}_{e'}^L$ of all neighbor nodes e' of e and performs a mean reduction:

$$\text{Agg}(\mathcal{M}_e^L) = \frac{1}{|\mathcal{M}_e^L|} \sum_{\mathbf{m}_{e'}^L \in \mathcal{M}_e^L} \mathbf{m}_{e'}^L \quad (27)$$

The Update function performs the node update operation, achieved by fusing the aggregated neighbor features $\mathcal{M}_e^L$ into the current node $\mathbf{h}_e^{L-1}$. Specifically, the expression for Update is defined as follows:

$$\text{Update}(\mathbf{h}_e^{L-1}, \text{Agg}(\mathcal{M}_e^L)) = \text{MLP}([\mathbf{h}_e^{L-1}; \text{Agg}(\mathcal{M}_e^L)]) \quad (28)$$

the MLP is designed to map the concatenated $\mathbb{R}^{2d_{\text{GNN}}}$ intermediate state back to an $\mathbb{R}^{d_{\text{GNN}}}$ representation, acting as an effective fusing mechanism of the neighbor features.

B Experiment Details

B.1 Datasets and Base KG

We conduct experiments on two publicly available datasets for multi-hop reasoning, including WebQuestionsSP (WebQSP) (Yih et al., 2016) and ComplexWebQuestions (CWQ) (Talmor and Berant, 2018). WebQSP is a large-scale multi-hop question-answering dataset, which comes with a knowledge graph in the Freebase[6] format. CWQ is a more difficult and challenging version of such datasets. Specifically, we utilized the open-source data format provided by RoG (Luo et al., 2024)[7][8], as it contains complete queries paired with their corresponding subgraphs extracted from Freebase. To ensure fairness and consistency across experiments, we partitioned all datasets according to RoG, obtaining separate training and test splits. We present the statistics for all datasets in the Table 8. The distribution of answer counts in the dataset is presented in Table 9.

B.2 Implementation Details

STEM involves three LLM-based modules: SGDA, SAGB, and the LLM reasoning model. For the first two modules, we fine-tune Qwen3-8B[9] respectively, and for reasoning model, we select Llama-3.1-8B-Instruct[10], Llama-3.1-70B-Instruct[11], and GPT-4o[12] (OpenAI, 2024). To ensure experimental reproducibility, we set the temperature of the reasoning model to 0. Additionally, we configured the SGDA with a beam size $\mathcal{B}$ of 4 and the SAGB with a temperature of 0. For feature embedding, we use the Qwen3-Embedding-0.6B[13] (Zhang et al., 2025b) model as the pretrained embedding model(i.e. d_{PEM}=1024), the Triple-GNN is configured with L=6 layers, other configurations are consistent with the settings described in GFM-RAG (Luo et al., 2025)(i.e. d_{GNN}=512). To determine the threshold θ for the threshold-based search employed in the *Breadth* strategy, we conducted a parameter search on the validation sets of WebQSP and CWQ, ultimately setting θ=0.6. We performed three independent runs of the full experimental pipeline—encompassing both module training and retrieval-inference testing—and report the average values across all metrics. All models employed in this study were used in strict accordance with their respective licenses and terms of

[6] https://github.com/microsoft/FastRDFStore
[7] https://huggingface.co/datasets/rmanluo/RoG-webqsp
[8] https://huggingface.co/datasets/rmanluo/RoG-cwq
[9] https://huggingface.co/Qwen/Qwen3-8B
[10] https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct
[11] https://huggingface.co/meta-llama/Llama-3.1-70B-Instruct
[12] https://chatgpt.com/
[13] https://huggingface.co/Qwen/Qwen3-Embedding-0.6B

Dataset	Train	Dev	Test	Hops	1 Hop	2 Hops	≥3 Hops
WebQSP	2,826	239	1,628	{1,2}	65.5%	34.5%	-
CWQ	27,639	3297	3,531	{1,2,3,4}	41%	38.3%	20.7%

Table 8: Statistics of the number of WebQSP and CWQ dataset splits along with the question hops.

Dataset	**Ans** = 1	**Ans** ∈ [2,4]	**Ans** ∈ [5,9]	**Ans** ≥ 10
WebQSP	51.8%	27.1%	8.1%	13.0%
CWQ	71.4%	19.0%	5.9%	3.7%

Table 9: Distribution statistics of answer counts in the two datasets.

use: OpenAI Terms of Use for GPT-4o, Llama 3.1 Community License for Llama models, and Apache 2.0 License for Qwen models.

B.3 Evaluation Metrics

Following established evaluation protocols, we assess model performance using two standard metrics: **Hits@1** and **F1 score**. Hits@1 quantifies the accuracy of the top-ranked answer prediction, while the F1 score provides a balanced assessment of answer coverage.

B.4 Baselines

Our experimental baselines are categorized into four groups: **Pure LLM Reasoning**, referring to methods that utilize only the large language model's inherent capabilities, **With Finetuning**, referring to methods that involve fine-tuning the reasoning model, **With Prompting**[14], referring to methods that control the reasoning and answering behavior of large language models through prompting, and **Our Proposed Method**. Notably, our proposed STEM inherently belongs to the fine-tuning category, as it relies on fine-tuned upstream modules for retrieval. We now introduce each category as follows:

Pure LLM Reasoning. We do not employ any retrieval components, but instead rely solely on the inherent reasoning capabilities, the model selected is GPT-4o, and experiments are conducted using three distinct settings: pure reasoning, few-shot learning, and CoT prompting.

With Finetuning. We select the following methods for comparison: **NSM** (He et al., 2021) proposes a teacher network to supervise the intermediate reasoning process. **DeCAF** (Yu et al., 2023) performs question-relevant retrieval at the document level and constructs logic forms for answers to optimize responses. **KD-CoT** (Wang et al., 2023) proposes KG-guided intermediate reasoning verification to ensure a more reliable reasoning process. **RoG** (Luo et al., 2024) leverages LLM-generated reasoning paths and continuously refines them with KG before performing retrieval. **GRAG** (Hu et al., 2025) optimizes subgraph retrieval complexity and employs both text view and graph view to enhance question comprehension, and **LightProf** (Ao et al., 2025) retrieves the reasoning path, then integrate KG factual and structural information into embeddings for improved answering.

With Prompting. We adopt the following approaches as baselines for comparison: **G-Ret** (G-Retriever) (He et al., 2024) proposes a novel RAG framework that formulates subgraph retrieval as a Prize-Collecting Steiner Tree (PCST) problem. **ToG** (Sun et al., 2024) introduces a framework where an LLM acts as an agent to iteratively explore reasoning paths on a knowledge graph via beam search. **Kaping** (Baek et al., 2023) is a zero-shot framework that retrieves relevant facts from a knowledge graph and prepends them to the input prompt. **FiDeLiS** (Sui et al., 2025) proposes a training-free framework that combines step-wise beam search with a deductive scoring function and Path-RAG module. **PoG** (Chen et al., 2024) decomposes questions into sub-objectives and iteratively adapts reasoning paths through guidance, memory, and reflection mechanisms. **ReKnoS** (Wang et al., 2025) introduces a novel framework that enhances LLM reasoning by incorporating super-relations in knowledge graphs. **MFC** (Zhang et al., 2025a) transforms questions into knowledge graph triples using LLMs and quantifies question quality based on cognitive metrics. **SubgraphRAG** (Li et al.,

[14]Notably, comparing STEM against prompting baselines does not imply zero-shot parity. Rather, it demonstrates that when training is feasible, structural alignment provides substantial gains in factual precision and hallucination mitigation.

2025) decouples the roles of knowledge graphs and LLMs in RAG systems. **GNN-RAG** (Mavromatis and Karypis, 2025) leverages lightweight GNNs for efficient graph retrieval. **ProgRAG** (Park et al., 2026) introduces feedback-aware and evidence-aware mechanisms to progressively align LLM reasoning with factual knowledge from graphs.

C Training Setup

C.1 Basic Training Configuration

Our work involves the training of three modules: Schema-Grounded Decomposition Agent, Symbol-Aligned Graph Builder, and Triple-GNN[15]. We will sequentially introduce the data construction and training methods for each of these modules.

First, we introduce the training data construction method, we utilize the training splits of the WebQSP and CWQ datasets. For both datasets, we take a question Q from the training split, along with its corresponding answer $\mathcal{A}$, question entities set $\mathcal{N}_Q$ and KG $\mathcal{G}$. We first extract reasoning chain $\mathcal{R}$ from $\mathcal{G}$ employing the method proposed in CoG (Zhao et al., 2025), then decompose the reasoning chain into individual triples $\mathcal{T}$:

$$\mathcal{T} = \{t_1, t_2, ..., t_N\} \tag{29}$$

Then, based on the question Q, we mark all entities in $\mathcal{T}$ that do not appear in $\mathcal{N}_Q$ using placeholders, this is because these entities are not question entities, but rather intermediate answer entities in the multi-hop reasoning process. This marking complies with the following principles: (1) when two triples share the same answer entity (indicating that they are connected in the graph), the same identifier "[ENTX]" is used; (2) different entities are distinguished by different identifiers ("[ENTX]" and "[ENTY]"). After formatting process, we obtain a new masked T:

$$\mathcal{T}^{'} = \{t_1^{'}, t_2^{'}, ..., t_N^{'}\} \tag{30}$$

Based on the obtained $\mathcal{T}^{'}$, we generate an assertion for each $t^{'} \in \mathcal{T}^{'}$ by using a prompt $\mathcal{P}_4$ to instruct a large language model (for all prompt-based data generation tasks in this study, we consistently utilized **Gemini 2.5 Pro**(Team, 2025) API, with the temperature set to 1.0), thereby obtaining a set $\mathcal{S}$ containing assertions corresponding to each triple in $\mathcal{T}^{'}$:

$$\mathcal{S} \leftarrow \text{LLM}(\mathcal{P}_4, \mathcal{T}^{'}, Q) \tag{31}$$

For **Symbol-Aligned Graph Builder Training**, we treat the generated assertions $\mathcal{S}_j$ as the source input and the original structured triples $\mathcal{T}_j^{'}$ as the target output to optimize the SAGB model:

$$\mathcal{L}_{\text{SAGB}} = -\sum_{j=1}^{|\mathcal{D}|}\sum_{k=1}^{|\mathcal{T}_j^{'}|} \log P_{\phi_5}(y_{j,k} \mid \mathcal{P}_2, \mathcal{S}_j, y_{j,<k}) \tag{32}$$

where $\mathcal{D}$ represents the training dataset, $\mathcal{P}_2$ denotes the instruction prompt used by the SAGB for schema graph building as introduced in Section 3.2.

For question-answering strategy generation, we determine based on the number of ground-truth answers for Q: if there is a single answer, σ is set to "Precision"; if there are multiple answers, σ is set to "Breadth". Thus for **Schema-Grounded Decomposition Agent** training, we use Q_j as the source input and $(\mathcal{S}_j, \sigma_j)$ as the target output to optimize the SGDA model:

$$\mathcal{L}_{\text{SGDA}} = -\sum_{j=1}^{|\mathcal{D}|}\sum_{k=1}^{|(\mathcal{S}_j,\sigma_j)|} \log P_{\phi_6}(y_{j,k} \mid \mathcal{P}_1, Q_j, y_{j,<k}) \tag{33}$$

For Triple-GNN Training, we apply the method described in Appendix A to obtain the pooled embedding representation $\mathbf{E}_Q \in \mathbb{R}^{d_{\text{GNN}}}$ of $\mathcal{T}^{'}$, then fed into the L-layer Triple-GNN to produce $\mathbf{H}_Q$, and finally transformed via a mapping function and activated to obtain the entity probability:

$$\mathbf{H}_Q^L = \text{Triple-GNN}_{\phi_3}(\mathcal{G}, \mathbf{H}_e^0, \mathbf{H}_r^0) \tag{34}$$

$$\boldsymbol{P}_Q = \text{Sigmoid}(\text{MLP}_{\phi_1}(\mathbf{H}_Q^L)) \tag{35}$$

The definition of $\mathbf{H}_e^0 \in \mathbb{R}^{|\mathcal{N}|\times d_{\text{GNN}}}$ and $\mathbf{H}_r^0 \in \mathbb{R}^{|\mathcal{R}|\times d_{\text{GNN}}}$ is the same as in Appendix A, $\mathcal{N}$ and $\mathcal{R}$ denote the set of entity nodes and relations in graph $\mathcal{G}$ respectively. For the label of each entity $e_i \in \mathcal{N}$, we construct it as follows:

$$\boldsymbol{y}_{e_i} = \begin{cases} \mathbf{1}, & e_i \in \mathcal{T}, \\ \mathbf{0}, & \text{otherwise.} \end{cases} \tag{36}$$

The final training loss employs the BCE loss (Luo et al., 2025), and is formulated as follows:

$$\mathcal{L}_{\text{Triple-GNN}} = -\frac{1}{|\mathcal{N}|}\sum_{e_i \in \mathcal{N}} \boldsymbol{y}_{e_i} \log \boldsymbol{p}_{e_i} \tag{37}$$

[15]Here, the Triple-GNN encompasses not only the parameters of the GNN module itself but also the parameters of various associated projection layers: $\mathbf{MLP}_{\phi_1}$, $\mathbf{MLP}_{\phi_2}$ and $\mathbf{MLP}_{\phi_4}$, which have been denoted in their respective formula descriptions.

where $\boldsymbol{p}_{e_i} \in \boldsymbol{P}_Q$ denotes the probability of entity e_i, the training details for each component are as follows:

- For the SGDA, we utilize a training set of approximately 25k entries in the format (Q, ($\mathcal{S}$, σ)). The training configuration includes a learning rate of 1e-4, a batch size of 32, and 2 training epochs, we employing Bfloat16 (Kalamkar et al., 2019) precision and the AdamW optimizer (Loshchilov and Hutter, 2019). The maximum input length for the SGDA is set to 2048 tokens.

- For the SAGB, the training set consists of about 29k entries in the format ($\mathcal{S}$, $\mathcal{T}^{'}$). It is trained for 2 epochs with a learning rate of 1e-5[16], using Bfloat16 precision and the AdamW optimizer. The maximum input length is set to 2048 tokens.

- The Triple-GNN is trained for 2 epochs with a learning rate of 1e-5.

- All training processes were conducted on a single machine with 4 × NVIDIA H100 GPUs.

Statistics of Trainable Parameters. The trainable components in our framework primarily include the SGDA and SAGB models, while each possesses a nominal size of 8B parameters. Additionally, the framework incorporates the Triple-GNN module ($\boldsymbol{W}_{\phi_3}$ ≈8M parameters) and several auxiliary projection layers ($\boldsymbol{W}_{\phi_1} \in \mathbb{R}^{512}$ = 512, $\boldsymbol{W}_{\phi_2} \in \mathbb{R}^{3072\times512}$ ≈1.5M and $\boldsymbol{W}_{\phi_4} \in \mathbb{R}^{1024\times512}$ ≈0.5M), the aggregate number of above trainable parameters is about 10M parameters.

C.2 Structure-to-Query Reverse Generation

The training datasets provide high-quality supervision, however they are limited in scale, which consequently restricts its coverage of the logic and schema diversification encapsulated in the KG. To equip our Semantic-to-Structural Projection pipeline with broader generalization capabilities and cover long-tail relations, we propose a novel Structure-to-Query Reverse Generation strategy, which constructs a large-scale, synthetic instruction-tuning dataset directly from the KG topology. We elaborate on the procedure in two distinct phases.

Phase 1: Reasoning-Path Subgraph Sampling. For each graph $\mathcal{G}$ in the WebQSP and CWQ datasets, we employ a random walk strategy to obtain a corresponding subgraph $\mathcal{G}_{sub}$ and its associated entity set $\mathcal{N}_{sub}$. Specifically, for subgraph $\mathcal{G}_{sub}$, we randomly mask a subset of nodes to serve as both the target answers and intermediate reasoning entities, replacing them with unified "[ENTX]" placeholders consistent with Section C.1, designating the remaining structure, denoted as $\mathcal{G}^*_{sub}$, as the evidence subgraph required for reasoning.

Phase 2: LLM-Driven Reverse Generation. We leverage a powerful LLM to generate natural language queries from the sampled subgraphs. Formally, sampled and masked graph $\mathcal{G}^*_{\text{sub}} = \{(e_1, r_1, e_2), (e_2, r_2, \text{"[ENT1]"})\}$ (assuming e_3 is masked to serve as the answer entity), we instruct the large language model to generate multi-hop question Q and declarative statements $\mathcal{S}_{\text{sub}}$ using prompt $\mathcal{P}_6$, $\mathcal{G}_{\text{sub}}$ and designated answer entity "[ENT1]":

$$(Q, \mathcal{S}_{\text{sub}}) \leftarrow \text{LLM}(\mathcal{P}_6, \mathcal{G}^*_{\text{sub}}, \text{"[ENT1]"}) \quad (38)$$

where $\mathcal{S}_{\text{sub}} = \{s_1, s_2, ..., s_n\}$. Considering the complexity of this prompt, which entails a two-step task execution, we enabled the "thinking mode" throughout the LLM inference process. We then employ prompt $\mathcal{P}_5$, Q and $\mathcal{S}_{\text{sub}}$ to generate corresponding answering strategy σ, following the method described in Appendix C.1.

Through the LLM-driven Reverse Generation method, we ultimately construct the knowledge graph reverse-generation dataset $\mathcal{D}_{\text{syn}}$, for every $d_i \in \mathcal{D}_{\text{syn}}$:

$$d_i = (Q^i, \mathcal{A}^i, \mathcal{G}^i, \mathcal{N}^i_{sub}, \mathcal{G}^{*i}_{\text{sub}}, \mathcal{S}^i_{\text{sub}}, \sigma^i) \quad (39)$$

where $\mathcal{A}^i$ denotes the answer to Q^i, corresponding to the masked entity e_3 in the example of $\mathcal{G}^*_{\text{sub}}$.

For the training of the SGDA, the objective function is defined as follows:

$$\mathcal{L}_{\text{SGDA}} = -\sum_{i=1}^{|\mathcal{D}|} \sum_{k=1}^{|(S^i_{sub}, \sigma^i)|} \log P_{\phi_6}(y_{i,k} \mid \mathcal{P}_1, Q^i, y_{i,<k}) \quad (40)$$

For the training of the SAGB, the objective func-

[16]The generation of triples constitutes a format-constrained task, which relies minimally on the prior knowledge of the text-based LLM, so we adopt a smaller learning rate to ensure more stable model training.

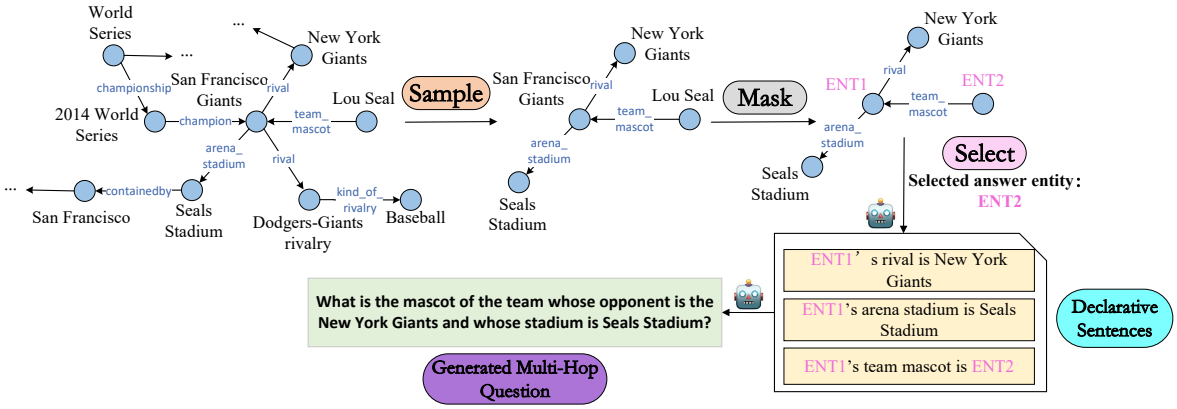

Figure 3: An illustrative example of the Structure-to-Query Reverse Generation pipeline.

tion is defined as follows:

$$\mathcal{L}_{\text{SAGB}} = -\sum_{i=1}^{|\mathcal{D}|}\sum_{k=1}^{|\mathcal{G}_{sub}^{*i}|}\log P_{\phi_5}(y_{i,k} \mid \mathcal{P}_2, \mathcal{S}_{sub}^{i}, y_{i,<k}) \quad (41)$$

The training objective of the Triple-GNN is consistent with that described in C.1. Regarding labeling, we perform positive and negative sampling within the entity set $\mathcal{N}_{sub}^{i}$, adhering to the following labeling principles:

$$\boldsymbol{y}_e = \begin{cases} \mathbf{1}, & e \in \mathcal{N}_{sub}^{i}, \\ \mathbf{0}, & \text{otherwise.} \end{cases} \quad (42)$$

For clarity, we present a complete example of the construction process as shown in Figure 3. Ultimately, we constructed dataset $\mathcal{D}_{\text{syn}}$ comprising about 210k entries. Detailed statistical information regarding $\mathcal{D}_{\text{syn}}$ is presented in Table 10. This dataset is utilized to train the SGDA, SAGB, and Triple-GNN modules, following the same logic as previously described. We finally incorporate $\mathcal{D}_{\text{syn}}$ into the the original training set in Appendix C.1. We will validate the impact of the $\mathcal{D}_{\text{syn}}$ dataset by comparing performance with and without it in the Appendix D. The synthetic dataset $\mathcal{D}_{\text{syn}}$ will be released alongside the source code.

C.3 Data ethics

We employ LLMs to generate intermediate assertions and multi-hop queries, a potential risk in such synthetic data generation is hallucination. To mitigate this, our Structure-to-Query Reverse Generation method is strictly grounded in sampled subgraphs from the KG. The reasoning paths are predetermined by the graph structure, ensuring that

Statistic	Value
Total Count	214,733
Avg Question Length	49
Hop=1	84,465
Hop=2	65,810
Hop=3	34,210
Hop$\geq$ 4	30,248
Precision	132,391
Breadth	82,342

Table 10: Statistics of the synthetic dataset $\mathcal{D}_{\text{syn}}$, including total size, hop-count distribution, and strategy distribution.

the generated questions and assertions are logically consistent with the underlying facts. While we have conducted manual quality checks on random samples, we acknowledge that minor semantic noise may persist. Furthermore, given that Knowledge Graphs and LLMs are known to exhibit inherent biases (e.g., geographical, gender, or cultural), our synthetic data—being derived from these sources—may inadvertently propagate such pre-existing biases. Furthermore, the synthetic data constructed via our reverse generation method is derived exclusively from public knowledge graph and standard LLM outputs, these public resources have already been anonymized and sanitized to remove personally identifiable information, thus all training datasets contain no PII or private data.

D Additional Experimental Results and Analysis

D.1 The Impact of the Answer Strategy σ on Performance

In this section, we investigate the impact of answer strategy differentiation on the experimental results.

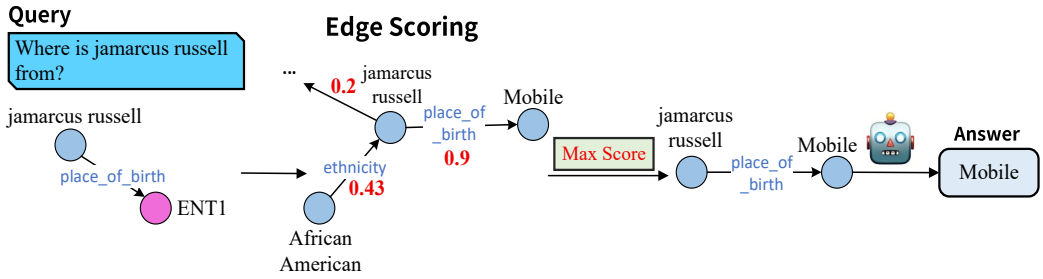

Figure 4: An illustrative example of Greedy Selection, corresponding to the *Precision* answering strategy.

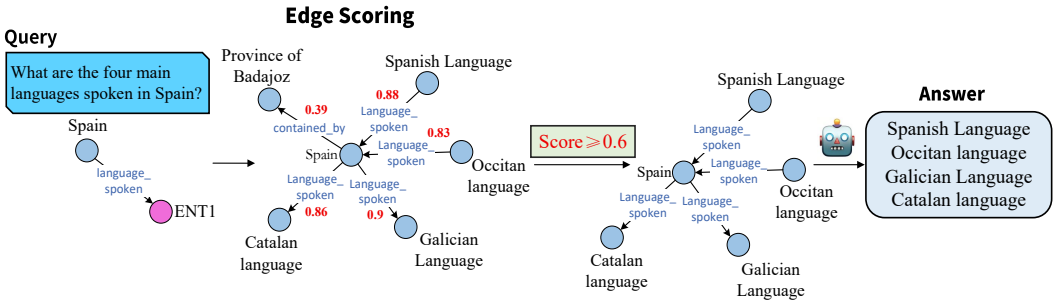

Figure 5: An illustrative example of Threshold-based Selection, corresponding to the *Breadth* answering strategy.

To illustrate the workflows and distinct behaviors of the search modes under the *Precision* and *Breadth* strategies, we first provide a representative example for Greedy Selection in Figure 4 and one for Threshold-based Selection in Figure 5.

We organize the experiments into three groups: the original Adaptive STEM Strategy, a purely Greedy Selection strategy, and a purely Threshold-based Selection strategy. The experimental results are presented in Table 11.

As shown in the results, the Only Greedy setting exhibits a significant performance decline in the F1 score, this is primarily because greedy search suffers from insufficient evidence recall in multi-answer scenarios, leading to incomplete answers. In contrast, the Only Threshold-based setting does not show a drastic drop and even outperforms the original STEM configuration on both F1 metrics. This is because Threshold-based Selection ensures answer coverage. However, it still underperforms STEM in other metrics. This is attributed to the retrieval of excessive irrelevant evidence, which induces hallucinations. We discuss the efficiency of the two search modes in Appendix D.8.

D.2 Initial Entity Count $\mathcal{K}$ on Performance

During the construction of the Global Guidance Subgraph, we perform an initial retrieval of $\mathcal{K}$ entities. In this section, we conduct experiments to evaluate the impact of different $\mathcal{K}$ values, to illustrate this more clearly, we assume $\mathcal{K} = \mid \mathcal{T} \mid * \mathcal{K}^{'}$ and define $\mathcal{K}^{'}$ as the **Guidance Graph Scale Factor**, we investigate the impact of different values of $\mathcal{K}^{'}$ on performance.

Strategy	WebQSP		CWQ	
	Hit@1	F_1 Score	Hit@1	F_1 Score
Only Greedy	88.50	60.75	72.45	44.29
Only Threshold-based	89.36	78.54	74.53	67.18
STEM	90.94	76.18	74.09	65.33

Table 11: Performance comparison with different response strategies.

The comparison results of all parameters $\mathcal{K}^{'}$ across different metrics are shown in Figure 6 (Hit@1) and Figure 7 (F1). From the results in the tables, it can be seen that values of $\mathcal{K}^{'}$ smaller or larger than 4 both affect the performance. Particularly when $\mathcal{K}^{'}$=1, there is a significant decline in metrics across both datasets. In terms of Hit@1 on both datasets, performance generally declines when $\mathcal{K}^{'}$ is less than 4, and gradually improves as $\mathcal{K}^{'}$ increases, with the CWQ dataset even showing a slightly better result at $\mathcal{K}^{'}$=3 compared to the reported results with $\mathcal{K}^{'}$=4. However, performance drops again when $\mathcal{K}^{'}$ exceeds 4. Regarding F1, a

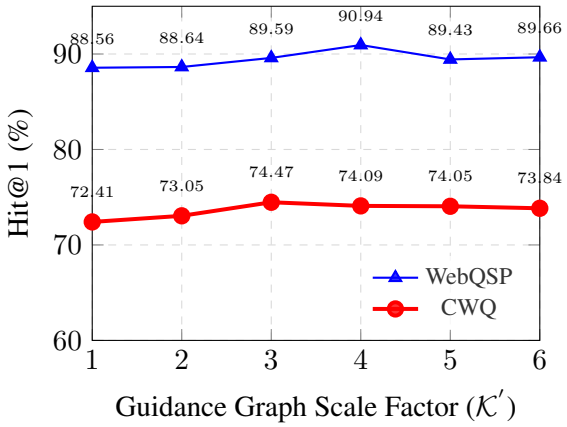

Figure 6: Impact of Guidance Graph construction scale on performance.

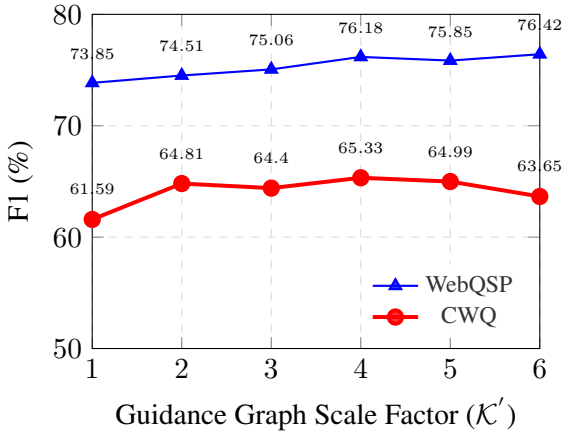

Figure 7: Impact of Guidance Graph construction scale on performance.

similar declining trend is observed when $\mathcal{K}'$ is less than 4, and simultaneously a moderate decline is observed when $\mathcal{K}'$ exceeds 4, with the exception of the WebQSP dataset. Therefore, $\mathcal{K}'$=4 is selected as our final configuration. Our analysis suggests that a smaller $\mathcal{K}'$ results in a smaller Guidance Graph, which risks missing key entities or question entities, while a larger $\mathcal{K}'$ may introduce low-value entities and mislead the evidence search.

D.3 Guidance Graph Correction Analysis

While Table 7 demonstrates that ablating the Guidance Graph significantly degrades overall QA performance, these end-to-end metrics are inevitably confounded by LLM generation stochasticity. To explicitly quantify the Guidance Graph's error-correction capability independent of the LLM, we conduct a pure retrieval evaluation. Specifically, we measure the coverage of ground-truth reasoning paths within the retrieved subgraphs with and without Guidance Graph guidance (using Equation 17). Results are presented in Table 12.

Scoring Bias	WebQSP	CWQ
STEM$_{GPT-4o}$	73.68	70.39
w/o $\mathbb{I}_{\text{Ent}}$ & $\mathbb{I}_{\text{Tri}}$	65.07	59.8
w/o $\mathbb{I}_{\text{Ent}}$	70.25	66.68
w/o $\mathbb{I}_{\text{Tri}}$	68.49	65.04

Table 12: Coverage rate of ground-truth reasoning paths within retrieved evidence subgraphs. We compare the pure retrieval quality under settings with and without the Guidance Graph. All values are reported as percentages (%).

As shown in Table 12, the full Guidance Graph configuration (incorporating both Entity-level and Triple-level biases) achieves the highest retrieval coverage. Quantitatively, the inclusion of Guidance Graph increases coverage from 65.07% to 73.68% on WebQSP (an absolute improvement of over 8%) and from 59.8% to 70.39% on CWQ (an increase exceeding 10%). Conversely, removing both constraints yields the lowest coverage, particularly on the more complex CWQ dataset. This significant drop indicates that without global structural guidance, the subgraph search is highly susceptible to being misled by local semantic features, leading to severe error propagation. Furthermore, ablating either bias individually outperforms the unconstrained variant but remains inferior to the fully constrained variant, confirming their complementary roles.

D.4 Impact of Global Structural Consistency Bias Values on Performance

In this section, we conduct experiments on bias constraints ($\mathbb{I}_{\text{Ent}}$ and $\mathbb{I}_{\text{Tri}}$). Specifically, we apply a multiplicative factor of $3/2$ to the scores during initial nodes selection in Equation 10 and 11, and an additive boost of $1/2$ to the scores during the edge search in Equation 12 and 13. Given that this involves the joint adjustment of two variables, we adopt a grid search strategy. Let λ denote the multiplicative factor for the Entity-level bias, and τ denote the boosting factor for the Triple-level bias. We define the search ranges for λ and τ as follows:

$$\lambda \in \{1.2, 1.5, 1.8, 2.1, 2.4, 2.7\} \quad (43)$$

$$\tau \in \{0.2, 0.5, 0.8, 1.1, 1.4, 1.7, 2.0\} \quad (44)$$

Since grid search involves a large number of experimental iterations, we randomly sample 200 examples from WebQSP and CWQ dataset to construct WebQSP$_{\text{sub}}$ and CWQ$_{\text{sub}}$, and conduct experiments on them. We fix one variable and search for

the optimal setting of the other, all experimental results are presented in Figure 8 and Figure 9.

From the experimental results, we can conclude that when both λ and τ are relatively small, performance drops significantly. In particular, with λ = 1.2, the WebQSP score decreases by about 3%, as λ increases from 1.5 onward, scores generally improve and remain stable thereafter. A similar trend is observed for τ: with τ = 0.2, the performance also deteriorates. Starting from τ = 0.5, the scores improve and stay stable. Since larger parameters do not bring significant further improvements, we select the relatively low values λ = 1.5 and τ = 0.5 as our final configuration.

Our Analysis The results suggests that excessively high parameter values lead to an over-reliance on the Guidance Graph. We posit that during query-based Guidance Graph building, the GNN may inadvertently incorporate edges with low relevance. These edges, while structurally connected, often contribute little to the actual reasoning process—a phenomenon we term "**Structural Over-Interpretation**". This limitation elucidates why the Guidance Graph cannot serve as the final reasoning subgraph in isolation and necessitates a subsequent refinement via semantic search. Fundamentally, STEM represents a synergy between logical reasoning (structure) and semantic matching (content). It achieves an optimal equilibrium, avoiding over-reliance on either modality while leveraging the indispensable strengths of both.

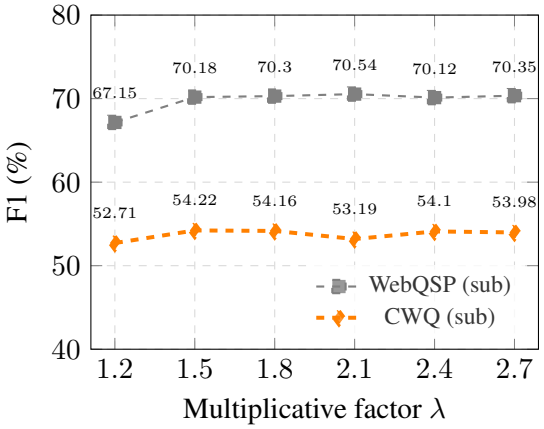

Figure 8: Performance comparison with different λ. Due to the constraints of the controlled variable method, the value of τ is set to 0.2 for all experiments.

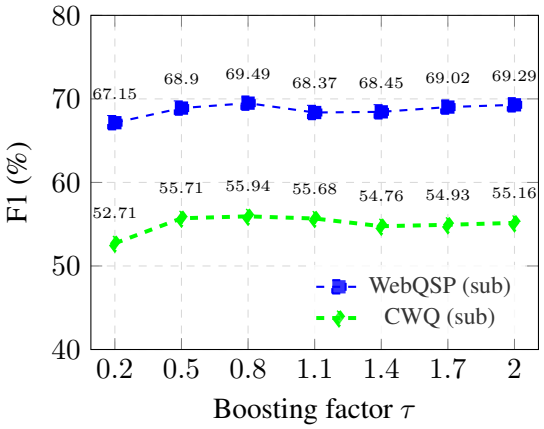

Figure 9: Performance comparison with different τ. Due to the constraints of the controlled variable method, the value of λ is set to 1.2 for all experiments.

D.5 Impact of Reverse Generation Data

We evaluate the impact of Structure-to-Query training set reverse generation on model performance. We first designed two comparative settings: a standard dataset, denoted as $\mathcal{D}_{\text{std}}$, constructed solely from the training splits of WebQSP and CWQ; and an augmented dataset, denoted as $\mathcal{D}_{\text{aug}}$, which combines the $\mathcal{D}_{\text{std}}$ with the synthetic data $\mathcal{D}_{\text{syn}}$ produced in C.2. We trained two separate sets of SGDA, SAGB, and Triple-GNN modules using these respective datasets and conducted comparative experiments. Two specific experiments were designed: **1. Schema Generation Quality:** We employed the SGDA and SAGB modules to construct schema graphs for queries in the WebQSP and CWQ test sets. The quality of these graphs was then evaluated by measuring precision and recall against the ground-truth reasoning paths. **2. End-to-End QA Performance:** We integrated SGDA, SAGB, and Triple-GNN into the complete STEM retrieval framework and evaluated the overall question-answering performance on both test sets.

D.5.1 Graph Evaluation

To evaluate schema graph generation quality, we first define the Precision and Recall metrics for the generated schema graphs. Given a question Q, let $\mathcal{R}$ denote the ground-truth reasoning path provided in the test set, and $\mathcal{G}_{sch}$ denote the schema graph generated by our trained Semantic-to-Structural Projection pipeline. We calculate Precision, Recall, and F1 score as follows:

$$\text{Precision} = \frac{|\mathcal{R} \cap \mathcal{G}_{sch}|}{|\mathcal{G}_{sch}|}, \tag{45}$$

$$\text{Recall} = \frac{|\mathcal{R} \cap \mathcal{G}_{sch}|}{|\mathcal{R}|}, \tag{46}$$

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{47}$$

Dataset	WebQSP		CWQ	
	Hit@1	F_1 Score	Hit@1	F_1 Score
D_{std}	86.35	74.17	67.03	58.78
D_{aug}	90.94	76.18	74.09	65.33

Table 13: Ablation study on reverse generation data: comparison of multi-hop QA results.

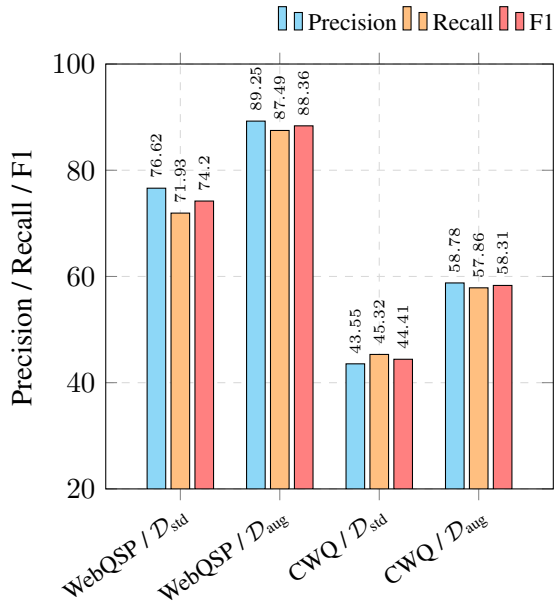

Figure 10: Ablation study on reverse generation data: comparison of schema graph P/R/F1 metrics.

The experimental results are presented in Figure 10. It is evident that incorporating the $\mathcal{D}_{\text{aug}}$ data leads to significant improvements in schema generation Precision, Recall, and F1 scores across both test sets. Notably, on WebQSP, the inclusion of $\mathcal{D}_{\text{aug}}$ yields a Recall increase of approximately 15% and an F1 improvement exceeding 14%. Similarly, the CWQ dataset witnesses a marked 15% rise in Precision and a 12% gain in Recall. Collectively, these results demonstrate that the incorporation of $\mathcal{D}_{\text{aug}}$ significantly bolsters the model's capability to perform logical perception for complex queries.

D.5.2 End-to-End QA Performance

We constructed complete STEM retrieval-QA pipelines using modules trained on the two respective datasets and conducted a comparative evaluation, with results shown in Table 13. The end-to-end tests reveal that the pipeline trained with $\mathcal{D}_{\text{aug}}$ dataset consistently achieves higher overall scores than the one trained with $\mathcal{D}_{\text{aug}}$ dataset. Notably, the improvement margin on CWQ is more significant, with Hit@1 increasing by over 7%. This indicates that the performance benefits yielded by the Structure-to-Query Reverse Generation method are amplified in complex reasoning scenarios involving a higher number of hops, underscoring the critical importance of query planning in multi-hop reasoning tasks.

Dataset	Ans = 1	Ans $\in$ [2,4]	Ans $\in$ [5,9]	Ans $\geq$ 10
WebQSP	3.1	4.03	5.59	8.81
CWQ	3.41	3.68	6.13	9.03

Table 14: Detailed average reasoning time (s) partitioned by answer count intervals.

Dataset	Ans = 1	Ans $\geq$ 2
WebQSP	96.5	93.31
CWQ	93.91	91.62

Table 15: Strategy generation accuracy (%) of SGDA across different test set splits. We categorize questions into two groups based on answer cardinality: those with a single answer (count $=$ 1) and those with answer counts ≥ 2. For the former, accuracy is measured by the proportion of generated *Precision* strategies, while for the latter, it is measured by the proportion of *Breadth* strategies.

D.6 Impact of SGDA Beam Size $\mathcal{B}$ on Performance

Given the inherent structural complexity of KG schemas, the SGDA employs beam search to generate multiple candidate sequences of atomic relational assertions simultaneously. In this section, we investigate the impact of the beam size $\mathcal{B}$ on model performance. The experimental results are illustrated in Figure 11.

It is evident that with a small beam size, retrieval failures often arise from the insufficiency of generated assertions. Specifically, at $\mathcal{B} = 1$, the model yields only the single most probable assertion; however, since queries of the same semantic type do not invariably map to a single identical pattern, retrieving a diverse set of multi-pattern assertions significantly enhances the structural hit rate of the evidence subgraph. Consequently, performance improves significantly as $\mathcal{B} = 1$ gradually increases. This trend is particularly pronounced on CWQ. Conversely, increasing $\mathcal{B}$ beyond 4 yields only marginal performance gains. Consequently, to balance retrieval accuracy with the computational complexity induced by processing excessive assertions, we adopt $\mathcal{B} = 4$ for this work.

Dataset	*Precision*	*Breadth*
WebQSP	4.91	8.42
CWQ	4.35	7.98

Table 16: Latency comparison (s) of different answering strategies.

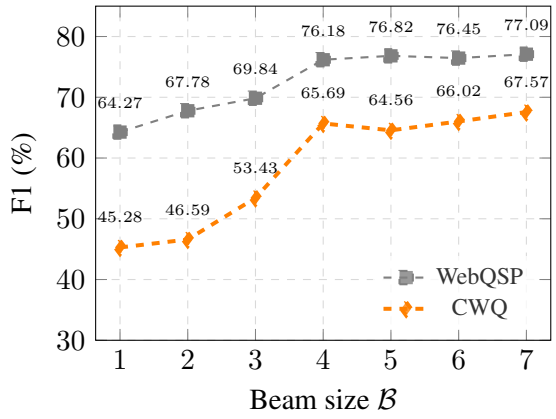

Figure 11: Performance comparison with different beam size $\mathcal{B}$.

D.7 Typical Case Study

In this section, we present a qualitative analysis of STEM's capabilities in query understanding, logical planning, schema graph construction, and retrieval through representative case studies. We selected diverse examples from both the WebQSP and CWQ datasets to illustrate various performance characteristics.

We begin with the WebQSP dataset, examining cases C1 (Table 17), C2 (Table 18), and C3 (Table 19). Case C1 (Table 17) exemplifies the "schema hallucination" challenge discussed in Section 1. As shown in the table, the SGDA module successfully mapped the query semantic "airport to fly into" to the concept of "nearby airport", guiding the SAGB to construct a schema graph consistent with KG facts. We also observe that all four sets of assertions were mapped to an identical schema graph, demonstrating the SAGB's structural consistency in handling such queries. Furthermore, the SGDA correctly predicted the *Breadth* strategy, facilitating successful retrieval of all subgraphs and comprehensive recall of the two answers.

Regarding Case C2 (Table 18), based on diverse assertions, the SAGB constructed multiple candidate schema graphs, retrieving answer-irrelevant yet structurally similar evidence subgraphs (e.g., ("Beech Street Historic District", "location.location.containedby", "Texarkana, Arkansas")). However, leveraging the LLM's precise discriminative capability, the system successfully identified and selected the correct evidence graph: ("Texarkana, Arkansas", "location.hud_county_place.county", "Miller County").

Finally, in Case C3 (Table 19), the SGDA successfully bridged the semantic gap by transforming the phrase "style of music" into the assertion term "music genre". This alignment enabled the accurate construction of the schema graph and the subsequent retrieval of supporting evidence.

Turning to the CWQ dataset, we analyze cases C4 (Table 20), C5 (Table 21), C6 (Table 22) and C7 (Table 23). The increased hop-count complexity inherent in CWQ inevitably exacerbates the potential for reasoning errors. In Case C4 (Table 20), although structurally similar schema graphs were constructed from different assertions, the relation names exhibited significant diversity, and the SAGB incorrectly mapped the relation to symbol "sports.school_sports_team.team". As indicated in the "Retrieved" row, the ground-truth relation symbol is "sports.school_sports_team.school". However, due to the high semantic similarity retained between the predicted and actual relations, the system still achieved accurate evidence retrieval. Simultaneously, the reasoning model successfully derived the final answer from the retrieved evidence subgraph.

Case C5 (Table 21) demonstrates the SGDA's robust divergent reasoning capability when decomposing multi-hop queries. While assertions (1) represents a generic logical form, the structural complexity of assertions (3) stems from the SGDA's comprehensive grasp of KG schemas, exemplifying high-quality factual alignment. Consequently, the schema graph derived from assertions (3) accurately retrieved the target evidence subgraph. In contrast, the retrieval results for assertions (1) and (2) were rejected by the reasoning model due to the absence of critical information regarding the movie "Forrest Gump" (the question entity given in the test set) thereby ensuring the output of the correct evidence.

Case C6 (Table 22) demonstrates a failure case of the SGDA. The assertions (1) ("Corfu's official language is [ENT1]") resulted in a retrieval failure due to the absence of corresponding relation within the KG. In contrast, assertions (3) ("Corfu is an administrative division of [ENT1]", "[ENT1]'s official language is [ENT2]") successfully aligned accurately with the underlying facts,

ultimately retrieving all correct answers. Meanwhile, assertions (2), utilizing the relation "location.location.containedby", retrieved an irrelevant subgraph.

Finally, Case C7 (Table 23) presents an instance of assertion conversion error, where the relationship between "Brussels" and the "European Union" was misidentified as "location.location.containedby" instead of the correct "organization.organization.founders". Nevertheless, the Triple-GNN successfully identified the "European Union" as a critical node during the construction of the Guidance Graph. Consequently, the correct triple ("European Union", "organization.organization.founders", "Belgium") was included in the Guidance Graph, which applied a positive bias to this edge during the search phase. This structural prior effectively corrected the semantic deviation, ensuring the successful recall of the evidence subgraph[17]. Although the relation "organization.membership_organization.members" triggered the retrieval of an irrelevant subgraph, the presence of the entity "Brussels" successfully prevented the model from being misled.

Collectively, these findings demonstrate STEM's robust resilience against schema inconsistency, manifested in three key aspects:

- **Multiple Planning Hypotheses**: Faced with diverse KG structures, the SGDA generates multiple candidate decomposition plans, significantly increasing the hit rate for the correct knowledge structure.
- **Fuzzy Semantic Matching**: In cases of relation symbol mismatch caused by SAGB, the search mechanism compensates via semantic similarity, ensuring successful evidence recall provided that the semantics remain proximate.
- **Global Structural Guidance**: For semantically distant relations between schema graph and true logic in KG, the Guidance Graph-derived consistency bias incorporates global structural priors to prioritize potentially optimal edges, thereby safeguarding the correctness of each search step.

[17]The correct triple ("European Union", "organization.organization.founders", "Belgium") would not have been selected as the top-ranked edge under a pure semantic matching regime due to the significant semantic divergence between the relations "founders" and "containedby", which yielded a low similarity score of 0.64. However, by incorporating the consistency bias, the score of this valid edge was successfully boosted to 1.14 and become the final selected edge.

D.8 Efficiency Analysis

The STEM framework is designed to minimize reliance on heavy, iterative LLM calls. For a single query, the process involves exactly three distinct LLM inference steps:

- **Projection**: For both the SGDA and SAGB module, we deploy a locally hosted model respectively. So the projection pipeline necessitates a single inference call of the 8B SGDA model to generate $\mathcal{B}$ planning candidates, followed by $\mathcal{B}$ forward passes of the 8B SAGB model, the total number of model invocations is $(2 \times \mathcal{B})$. Compared to FiDeLiS, which necessitates $(\mathcal{B}L + L + 1)$ LLM calls (where L denotes reasoning depth), our projection operation scales linearly with the beam size while maintaining optimal performance. This efficiency is comparable to existing high-efficiency methods such as LMP $(2L + 1)$ (Wan et al., 2025) and RoG $(N + 1)$, where N represents the number of generated relation paths. Moreover, due to the significantly shorter generation length of our assertions and schema graphs, the actual inference latency is further reduced.
- **Triple-Dependent GNN**: We execute a single forward pass of a 6-layer Triple-Dependent GNN to compute the Guidance Graph. Since node and triple embeddings can be pre-indexed, the online computational cost is restricted to the propagation of query-specific interaction scores.
- **Parallel Structure-Tracing Retrieval**: Since the entity selection phase yields multiple entry entities simultaneously, we employ a parallel execution strategy, where search threads originating from all the anchored nodes are conducted concurrently. By parallelizing the retrieval process and merging the resulting subgraphs, we significantly reduce the wall-clock time compared to sequential search methods. Consequently, the overall time cost of subgraph retrieval takes approximately 1.5 to 2 times longer than retrieval from a single entry node.
- **Answer Generation**: Once the evidence subgraph is retrieved and linearized, the generator LLM is invoked once to produce the final response.

Question	which airport to fly into rome
Assertions	1. ("rome's nearby airport is [ENT1]",) 2. ("the airport near rome is [ENT1].",) 3. ("rome is served by a nearby airport, [ENT1].",) 4. ("[ENT1] is a nearby airport for rome.",)
Strategy	Breadth
Schema Graphs	1. [("rome", "location.location.nearby_airports", "[ENT1]")]
Retrieved	1. [("Rome", "location.location.nearby_airports", "Ciampino–G. B. Pastine International Airport")] 2. [("Rome", "location.location.nearby_airports", "Leonardo da Vinci–Fiumicino Airport")]
Ground Truth(2 items)	Ciampino–G. B. Pastine International Airport, Leonardo da Vinci–Fiumicino Airport
Output Answer	Ciampino - G. B. Pastine International Airport and Leonardo da Vinci – Fiumicino Airport.

Table 17: Case study C1: Interpretability analysis on the WebQSP dataset.

Question	what county is texarkana arkansas in
Assertions	1. ("texarkana arkansas is a country of [ENT1]",) 2. ("texarkana, arkansas is a country within [ENT1].",) 3. ("texarkana arkansas is part of the country [ENT1].",) 4. ("the country to which texarkana arkansas belongs is [ENT1].",)
Strategy	Precision
Schema Graphs	1. [("texarkana arkansas", "location.location.containedby", "[ENT1]")] 2. [("texarkana arkansas", "location.hud_county_place.county", "[ENT1]")] 3. [("texarkana arkansas", "location.administrative_division", "[ENT1]")]
Retrieved	1. [("Beech Street Historic District", "location.location.containedby", "Texarkana, Arkansas")] 2. [("texarkana, arkansas", "location.hud_county_place.county", "Miller County")] 3. [("Arkansas","location.administrative_division.country","United States of America")]
Ground Truth	Miller County
Output Answer	Miller County

Table 18: Case study C2: Interpretability analysis on the WebQSP dataset.

Question	what style of music did bessie smith perform
Assertions	1. ("bessie smith's music genre is [ENT1]",) 2. ("the music genre of bessie smith is [ENT1].",) 3. ("bessie smith's genre of music is [ENT1].",) 4. ("[ENT1] is the music genre associated with bessie smith.",)
Strategy	Precision
Schema Graphs	1. [("bessie smith", "music.artist.genre", "[ENT1]")]
Retrieved	1. [("Bessie Smith", "music.artist.genre", "Jazz")]
Ground Truth	Jazz
Output Answer	Jazz

Table 19: Case study C3: Interpretability analysis on the WebQSP dataset.

Question	What educational institution with men's sports team named Wisconsin Badgers did Russell Wilson go to?
Assertions	1. ("Wisconsin Badgers is a school sports team of [ENT1].", "Russell Wilson's educational institution is [ENT1].") 2. ("The school sports team known as the Wisconsin Badgers belongs to [ENT1].", "The educational institution that Russell Wilson attended is [ENT1].") 3. ("[ENT1]'s official school sports team is called the Wisconsin Badgers.", "Russell Wilson's educational institution is [ENT1].") 4. ("[ENT1] is the institution that fields the Wisconsin Badgers sports team.", "Russell Wilson received his education at [ENT1].")
Strategy	Precision
Schema Graphs	1.[("Wisconsin Badgers", "sports.sports_league.teams", "[ENT1]"), ("Russell Wilson", "education.education.institution", "[ENT1]")] 2.[("Wisconsin Badgers", "sports.school_sports_team.team", "[ENT1]"), ("Russell Wilson", "education.education.institution", "[ENT1]")] 3.[("Wisconsin Badgers", "sports.sports_league.teams", "[ENT1]"), ("[ENT1]", "education.education.student", "Russell Wilson")]
Retrieved	1.[("Wisconsin Badgers men's basketball", "sports.school_sports_team.school", "University of Wisconsin-Madison"), ("Russell Wilson", "education.education.institution", "University of Wisconsin-Madison")] 2.[("Wisconsin Badgers", "education.athletics_brand.teams", "Wisconsin Badgers women's ice hockey") , ("University of Wisconsin-Madison", "education.educational_institution.sports_teams", "Wisconsin Badgers women's ice hockey")] 3.[("m.0hpny0z", "education.education.student", "Russell Wilson"), ("m.0hpny0z","education.education.degree","Bachelor of Arts")]
Ground Truth	University of Wisconsin-Madison
Output Answer	University of Wisconsin-Madison

Table 20: Case study C4: Interpretability analysis on the CWQ dataset.

Question	What actor played the a kid in the movie with a character named Jenny's Father?
Assertions	1. ("Jenny's father is a movie character in [ENT1].", "[ENT2] performs a role in the production [ENT1].") 2. ("Jenny's father is a character in [ENT1].", "[ENT2] appears as an actor in [ENT1].") 3. ("Jenny's father is a character in movie [ENT1].", "[ENT2] is a character in [ENT1].", "[ENT3] portrayed [ENT2] in the film.")
Strategy	Precision
Schema Graphs	1.[("Jenny's Father", "film.performance.character", "[ENT1]"), ("[ENT2]", "film.performance.actor", "[ENT1]")] 2.[("Jenny's Father", "film.film_character.portrayed_in_films", "[ENT1]"), ("[ENT2]", "film.film_character.portrayed_in_films", "[ENT1]"), ("[ENT2]", "film.performance.actor", "[ENT3]")]
Retrieved	1.[("m.0y54dnx", "film.performance.character", "Jenny's Father"), ("m.0y54dnx","film.performance.actor","Kevin Mangan")] 2.[("Jenny's Father", "film.film_character.portrayed_in_films", "Forrest Gump"), ("Forrest Gump", "film.film_character.portrayed_in_films", "m.02xgww5"), ("m.02xgww5", "film.performance.actor", "Michael Connor Humphreys")]
Ground Truth	Michael Connor Humphreys
Output Answer	Michael Connor Humphreys

Table 21: Case study C5: Interpretability analysis on the CWQ dataset.

Question	People from the country that contains Corfu speak what language?
Assertions	1. ("Corfu's official language is [ENT1].",) 2. ("Corfu is belong to [ENT1].", "[ENT1]'s official language is [ENT2].") 3. ("Corfu is an administrative division of [ENT1].", "[ENT1]'s official language is [ENT2].")
Strategy	Breadth
Schema Graphs	1.[("Corfu", "location.country.official_language", "[ENT1]")] 2.[("Corfu", "location.location.containedby", "[ENT1]"), ("[ENT1]", "location.country.official_language", "[ENT2]")] 3.[("Corfu", "location.administrative_division.country", "[ENT1]"), ("[ENT1]", "location.country.official_language", "[ENT2]")]
Retrieved	1.[("Corfu", "location.administrative_division.country", "Greece"), ("Greece", "location.country.languages_spoken", "Albanian language")] 2.[("Corfu", "location.administrative_division.country", "Greece"), ("Greece", "location.country.official_language", "Greek Language")] 3.[("Corfu", "location.location.containedby", "Corfu Island"), ("Corfu Island", "common.topic.article", "m.0cc3p")]
Ground Truth(2 items)	Albanian language, Greek Language
Output Answer	Albanian and Greek language

Table 22: Case study C6: Interpretability analysis on the CWQ dataset.

Question	What European Union country is home to the capital city of Brussels?
Assertions	1. ("[ENT1]'s capital city is Brussels", "European Union contains [ENT1].") 2. ("The capital cities of [ENT1] are Brussels.", "The European Union is composed of [ENT1].") 3. ("Brussels serves as the capital city for [ENT1].", "The member states of the European Union are [ENT1].") 4. ("Brussels is the capital city of [ENT1]", "European Union contains [ENT1].")
Strategy	Precision
Schema Graphs	1. [("Brussels", "location.administrative_division.capital", "[ENT1]"]), ("[ENT1]", "location.location.containedby", "European Union")] 2. [("Brussels", "location.location.containedby", "[ENT1]"]), ("[ENT1]", "location.location.containedby", "European Union")] 3. [("Brussels", "location.administrative_division.capital", "[ENT1]"]), ("[ENT1]", "organization.membership_organization.members", "European Union")] 4. [("Brussels", "location.administrative_division.capital", "[ENT1]"]), ("[ENT1]", "location.location.containedby", "European Union")]
Retrieved	1. [("European Union", "organization.organization.founders", "Belgium"), ("Brussels", "location.administrative_division.capital", "Belgium")] 2. [("European Union", "organization.membership_organization.members", "France"), ("Paris", "location.administrative_division.capital", "France")]
Ground Truth	Belgium
Output Answer	Belgium

Table 23: Case study C7: Interpretability analysis on the CWQ dataset.

A critical factor influencing the execution efficiency of STEM is the subgraph search mode, which is determined by the answer strategy. Consequently, we conduct a focused inference analysis of different search mode. All experiments were conducted on a single NVIDIA H100 GPU.

Given the close correlation between answer number and search modes, we stratify the test samples by answer number and evaluate the inference efficiency for each group independently. We first present the strategy generation accuracy of the SGDA module in Table 15. The results demonstrate that the SGDA module exhibits strong performance in strategy discrimination, achieving an accuracy of over 90% across all dataset splits.

Subsequently, we present a comparison of inference efficiency grouped by answer counts in Table 14, which indicates that inference latency increases significantly as the number of answers grows. Notably, when the answer count exceeds 10, the inference time rises to over 9 seconds. This is attributed to the fact that a higher volume of answers corresponds to a greater number of matched KG edges aligning with the schema graph.

We further evaluated the test set by stratifying samples based on different answer strategies, with results shown in Table 16. Although inference latency is significantly higher in *Breadth* mode than in *Precision* mode, the substantial improvement in F1 score with *Breadth* mode (Section D.1) justifies this trade-off for more comprehensive answers. We consider the moderate latency increase acceptable, especially given that real-world deployment efficiency is expected to improve with advancing technologies.

Pruning Strategy. The threshold-based search mode incorporates a pruning mechanism to prevent computational explosion. Taking Case C6 (Table 22) as an example, the reasoning chain requires traversing from "Corfu" to "[ENT1]" and subsequently to "[ENT2]". The retrieval results reveal that during the transition from "Corfu" to "[ENT1]", the threshold filtering mechanism retained only a single unique edge, effectively converging to one path. Parallel branching emerged only during the subsequent expansion from "[ENT1]" to "[ENT2]" to capture all valid answers. This demonstrates that parallel paths are triggered specifically when necessary to cover multiple answers, thereby significantly preserving computational efficiency.

Efficiency Comparison with Baselines. We compare our proposed method against existing baselines to demonstrate its advantages and superiority. To better contextualize the inference latency, we first categorize the existing approaches into three paradigms: (1) Interactive methods involve iterative LLM calls during the retrieval process. (2) Generation-based methods require constant number of LLM calls upfront for path or plan generation. (3) Retrieval-based methods rely entirely on GNNs for graph context encoding. The results are shown in Table 24.

Method	Type	WebQSP	CWQ
FiDeLiS	Interactive	34.97	40.16
PoG	Interactive	14.28	14.08
RoG	Generation	3.75	4.31
GNN-RAG	Retrieval	2.64	3.81
STEM (Ours)	Generation	5.77	5.40

Table 24: Comparison of average inference latency across different methods. Values denote the average wall-clock time (s) required to process a single query.

Empirical results demonstrate that STEM is substantially faster than interactive baselines. Specifically, it achieves a nearly a 6-fold speedup over FiDeLiS (5.77s vs. 34.97s) and operates 2.5 times faster than PoG (5.77s vs. 14.28s). This validates our claim that the offline Structure-Tracing paradigm effectively circumvents the computational bottleneck of repetitive LLM reasoning. While STEM is marginally slower than RoG (which generates linear relation paths) and GNN-RAG (which relies on node-level scoring), it delivers vastly superior retrieval accuracy, representing a highly favorable trade-off between efficiency and effectiveness.

D.9 Interpretability Analysis

STEM offers a transparent, interpretable workflow rooted in two key structural mechanisms:

- **Explicit Logical Blueprinting**: By leveraging the linguistic and reasoning capabilities of LLMs within a KG-constrained framework, STEM explicitly visualizes the reasoning process. The SGDA transforms the ambiguity of natural language into a sequence of logical atomic relational assertions, which the SAGB then projects into a concrete schema graph. The resulting blueprint is not merely a semantic abstraction but a topologically valid plan aligned with the latent knowledge structures of the target KG. This allows us to directly

inspect and verify the model's decomposition logic before any retrieval occurs.

- **Guided Structural Tracing**: Building on this blueprint, the Structure-Tracing Retrieval mechanism enables global matching based on the logical subgraph structure, directly mapping the query's structural features onto the KG knowledge. The schema graph acts as a navigational chart, while the Guidance Graph derived from the Triple-GNN serves as a soft structural bias. This dual-guidance system ensures that the search process is not driven solely by one-sided semantic matching—which is prone to deviation—but is anchored by global structural hypotheses.

E Analysis of Failure Modes and Error Propagation

Given the sequential architecture of STEM—where the outputs of the SGDA and SAGB modules serve as inputs for the Triple-GNN and subsequent retrieval—inaccuracies at any upstream stage can inevitably propagate downstream. To better understand this cascading effect, this section presents a systematic analysis of failure modes and error accumulation across the STEM pipeline.

Specifically, we construct two error attribution matrices on WebQSP to reveal how errors at different stages affect the final QA performance. These matrices capture the cascading effects of two critical intermediate phases: (1) Schema Generation Correctness (SGDA & SAGB) versus Final QA Accuracy, and (2) Retrieved Evidence Subgraph Correctness versus Final QA Accuracy. We provide a detailed analysis of each dimension below.

Schema Generation Correctness vs. QA Accuracy: We constructed an error attribution matrix to evaluate the correlation between Schema Generation Correctness (produced by the SGDA and SAGB pipeline) and the final QA Accuracy. The results are presented in Table 25.

The results show that the highest proportion (85.24%) occurs when both planning and QA are correct, demonstrating the effectiveness of the query planning modules. The 8.67% where planning is correct but QA fails likely stems from retrieval errors or the LLM not utilizing the correct evidence. In 5.91% of cases, planning fails yet QA succeeds—almost entirely attributable to the LLM's parametric knowledge. The matrix reveals that incorrect schema graphs account for only 16.4% of total QA failures (0.017/0.1037). This indicates that the vast majority of errors originate in downstream stages—namely, during subgraph retrieval and the LLM's final answer generation—rather than from upstream planning.

$\mathcal{G}_{sch}$ **match**	**Final QA Output**	
	Correct	Incorrect
Valid	85.24	8.67
Invalid	5.91	1.7

Table 25: Schema Generation Correctness vs. QA Accuracy Matrix. Rows indicate whether the schema graph produced by the SGDA and SAGB pipeline successfully matches the ground-truth reasoning graph, while columns represent whether the final generated answer is correct. All values are reported as percentages (%).

Retrieved Evidence Subgraph Correctness vs. QA Accuracy: We analyze the correlation between retrieval correctness and final answer correctness. The results are shown in Table 26.

The proportion of cases where retrieval is correct and the answer is correct reaches 83.01%, reflecting the effectiveness of the retrieval algorithm and its contribution to question answering. In 4.12% of cases, retrieval is correct but the answer is incorrect, which is primarily attributable to the LLM's failure in contextual answer extraction. Cases where retrieval is incorrect yet the answer is correct account for 7.34%, largely due to the LLM's parametric knowledge. Finally, instances where both retrieval and QA fail constitute 5.53%. From this analysis, we consider the 83.01%—where both retrieval and QA are correct—as the true reflection of STEM's capability, demonstrating the robustness and accuracy of its retrieval mechanism and its positive impact on final answer generation. Further analysis reveals that retrieval failures account for 57.3% (0.0553/0.0965) of all incorrect answers. This indicates that flawed evidence retrieval is the primary source of downstream errors, with the remaining 42.7% attributable to inherent LLM hallucinations during the final generation phase.

F Detailed Mechanism of the Semantic-to-Structural Projection and Structural Pattern Acquisition

To address potential ambiguities regarding how our method maps a natural language text to schema graph, we provide a detailed walkthrough of the Semantic-to-Structural Projection pipeline. This

$\mathcal{G}_{\text{reason}}$ **match**	**Final QA Output**	
	Correct	Incorrect
Valid	83.01	4.12
Invalid	7.34	5.53

Table 26: Retrieved Evidence Subgraph Correctness vs. QA Accuracy. The rows correspond to the correctness of the retrieved evidence subgraph, while the columns represent the correctness of the final answer. All values are reported as percentages (%).

pipeline, comprising the SGDA and SAGB modules, operates as an end-to-end generative translation process, converting complex raw queries into a set of triples that constitute the schema graph.

To illustrate this workflow concretely, consider the query: "where is the fukushima daiichi nuclear plant located", it's processed through the following two stages:

Stage 1: SGDA Decomposition The SGDA module first decomposes the raw text query into a set of atomic relational assertions, ultimately yielding:

"The fukushima daiichi nuclear power plant is contained by [ENT1]."

Here, [ENT1] serves as a structural placeholder for the unknown answer or intermediate entity, simplifying the subsequent mapping task.

Stage 2: SAGB Alignment To perform the symbolic mapping, the SAGB module autoregressively translates the input atomic assertions into the corresponding valid KG triples:

("The fukushima daiichi nuclear power plant", "location.location.containedby", "[ENT1]")

This process successfully translates highly variable natural language into strict KG-oriented structures by leveraging the LLM's reasoning capabilities. Since this structural generalization capability is explicitly instilled through our training regimen, we next provide concrete examples to illustrate how it is achieved.

Generalization over Training Importantly, this Semantic-to-Structural Projection learns underlying alignment patterns rather than merely memorizing specific facts. During the fine-tuning phase, the training data for SGDA and SAGB contain structurally similar alignments, such as those illustrated in Figures 12 and 13.

Example (1)

Query: In which country is Kagoshima Prefecture located?
Atomic Relational Assertions: ("Kagoshima Prefecture is contained by [ENT1]",)
Schema Graph: [("Kagoshima Prefecture", "location.location.containedby", "[ENT1]")]
Answer: Japan

Figure 12: SDGA & SAGB Training Data Example (1).

Example (2)

Query: Which city in Aomori Prefecture was affected by the 2011 Tohoku earthquake?
Atomic Relational Assertions: ("[ENT1] is contained by Aomori Prefecture.", "[ENT1] experienced the event of the 2011 Tōhoku earthquake and tsunami")
Schema Graph: [("[ENT1]", "location.location.containedby", "Aomori Prefecture"), ("[ENT1]", "location.location.events", "2011 Tōhoku earthquake and tsunami")]
Answer: Tohoku

Figure 13: SDGA & SAGB Training Data Example (2).

After capturing these schema patterns, the pipeline can effectively generalize to structurally similar assertions (e.g., "X is located in Y") across different entities. This generative design enables STEM to perform robust, structure-aware schema alignment, circumventing the rigidity and out-of-vocabulary issues typical of traditional step-wise path search or dictionary-based matching methods.

G Prompt List

To ensure the robustness and reproducibility of STEM, we detail the core prompts utilized across the different stages of our pipeline. These encompass the processing prompts for the SGDA and SAGB modules, the generation prompt for the QA model, the assertion synthesis prompt for training data construction, the prompt for determining the response strategy, and notably, the core feature of our work: the Structure-to-Query Reverse Generation data synthesis prompt, which significantly enhances the model's structural generalization capabilities. We present the complete instructional content of each prompt in Figures 14 through 19.

G.1 Schema-Aligned Question Decomposition Prompt $\mathcal{P}_1$

Schema-Aligned Question Decomposition Prompt ($\mathcal{P}_1$)

You are a multi-hop question decomposition expert specialized in knowledge graph-based question answering. Your task is to decompose an input multi-hop question into a sequence of single-hop assertions, and returns the answer strategy required to answer the query. The specific requirements are as follows:

- 1. Decomposition must be entity-centric. Each single-hop assertion should correspond to a pair of entities and describe the relationship between them.
- 2. Every single-hop assertion generated should contribute to answering the original multi-hop question.
- 3. For entity references—including the final answer or any intermediate answer entities—you must label them with [ENT1], [ENT2], etc., and ensure that:
  · The same entity is consistently referred to with the same label across all single-hop assertions.
  · Different entities are assigned distinct labels.
- 4. Answering Strategy: If you determine that the answer to the current question is exclusive and deterministic, please return the strategy "Precision". If you assess that the question involves multiple distinct answers (including final or intermediate answers), please return the strategy "Breadth". You must select your strategy strictly from "Precision" and "Breadth".

Your response should be a single result consisting of the planned single-hop assertion (s) along with the strategy.
Your output format:
(Assertion_1, Assertion_2, Assertion_3), Strategy)

Example:
[EXAMPLE_1]
[EXAMPLE_2]
[EXAMPLE_3]

Input multi-hop question:
[Query]
Please now plan the decomposition for the given question. You must strictly follow the requirements above.

Figure 14: The prompt template for Schema-Aligned Question Decomposition ($\mathcal{P}_1$).

G.2 Schema Graph Construction Prompt $\mathcal{P}_2$

Schema Graph Construction Prompt ($\mathcal{P}_2$)

You are an entity-relationship construction expert who has memorized a rich and professional knowledge graph-oriented semantic and logical structure. Based on your mastered graph structure data, you can construct appropriate entity-relationship triples for given single-hop assertions.
Since these assertions are decomposed from a multi-hop query, you must fully consider their interdependencies when returning the triples.
The specific requirements are as follows:

- 1. Based on the given assertions and combined with your understanding of knowledge graph data, you must generate structural triples that best match the meaning expressed. The entities and relationship descriptions in the triples must be consistent with the meaning of the assertions.
- 2. If an assertion contains an entity placeholder like [ENTX], you must copy it exactly as is when converting it into a triple. Do not alter the content of the placeholder. If there is no [ENTX] label in an assertion, generate the most relevant triple based on the described entities and relationship.

Your output format:
[
(Entity_1, Relation_1, Entity_2),
(Entity_2, Relation_2, Entity_3),
...
]

Example:
[EXAMPLE_1]
[EXAMPLE_2]

Input single-hop assertions:
[Assertion_1, Assertion_2, ...]

Figure 15: The prompt template for Schema Graph Construction ($\mathcal{P}_2$).

G.3 Generation Prompt $\mathcal{P}_3$

Generation Prompt ($\mathcal{P}_3$)

Based on the knowledge structure graph, please answer the given question. Please keep the answer as simple as possible and return all the possible answers as a list.

Knowledge Structure Graph: [Knowledge Structure Graph]
Question: [Question]
Answer: [Answer]

Figure 16: The prompt template for Generation ($\mathcal{P}_3$).

G.4 Path-based Assertions Generation Prompt $\mathcal{P}_4$

Path-based Assertions Generation Prompt ($\mathcal{P}_4$)

Please construct appropriate declarative assertions for each given triple based on the provided logical triple list and original query. The requirements are as follows:

- 1. The format of each given triple is ("entity1", "relation", "entity2"). You need to generate a suitable declarative assertion based on the meanings of entity1 and entity2 and the relationship between them.
- 2. If an entity is in the label format such as [ENTX], the corresponding entity in the generated sentence should also be written in the same label format. If multiple triples contain the same label, for example, all are [ENT1], you must ensure consistency in the generated assertions and avoid modifying the label content arbitrarily.
- 3. The number of assertions you return must match the number of triples in the given list, and they should correspond one-to-one.
- 4. The assertions you generate must serve as important evidence for answering the given original query, meaning that answering the query requires referencing these assertions.

Your output format:
[
Assertion_1,
Assertion_2,
...
]
Example:
[EXAMPLE_1]
[EXAMPLE_2]
[EXAMPLE_3]

Original Query:
[Query]
Given Triple List:
[Triple_1, Triple_2, ...]

Figure 17: The prompt template for Path-based Assertions Generation ($\mathcal{P}_4$).

G.5 Response Strategy Generation Prompt $\mathcal{P}_5$

Response Strategy Generation Prompt ($\mathcal{P}_5$)

Please determine the appropriate retrieval strategy for the given question when used for multi-hop retrieval-augmented generation in a knowledge graph context. There are two available strategies: "Precision" and "Breadth", described as follows:

- 1. "Precision" is primarily used for answering questions that have a exclusive and deterministic answer, such as "In which year was Edison born?" Questions of this type typically have only one correct answer, and contradictions may arise if more than two answers are provided.
- 2. "Breadth" is mainly used for answering questions involving multiple distinct answers, such as "What country is Russia close to?", this type of question requires retrieving all qualifying answers to ensure comprehensive answer recall.
- 3. You will be given a list of assertions representing the decomposed declarative statements derived from the given question. These assertions constitute the multi-hop logical decomposition of the question. You may reference them to get more in-depth guidance.

Please remember, your response should contain only the word "Precision" or "Breadth", with no additional explanatory content!

Example:
[EXAMPLE_1]
[EXAMPLE_2]
[EXAMPLE_3]

Question: [Question]
Assertions: [Assertion_1, Assertion_2, ...]

Figure 18: The prompt template for Response Strategy Generation ($\mathcal{P}_5$).

G.6 Query & Assertions Generation based on Sampled-Graph Prompt $\mathcal{P}_6$

Query & Assertions Generation based on Sampled-Graph Prompt ($\mathcal{P}_6$)

Please construct appropriate declarative sentences for each given triple based on the provided triple list. The requirements are as follows:

- 1. The format of each given triple is ("entity1", "relation", "entity2"). You need to generate a suitable declarative sentence based on the meanings of entity1 and entity2 and the relationship between them.
- 2. If an entity is in the label format such as [ENTX], the corresponding entity in the generated sentence should also be written in the same label format. If multiple triples contain the same label, you must ensure consistency in the generated sentences and avoid modifying the label content arbitrarily.
- 3. The number of sentences you return must match the number of triples in the given list, and they should correspond one-to-one.

After constructing the declarative sentences, proceed to generate a multi-hop question based on them. You must adhere to the following requirements:

- 1. Given the specific placeholder [ENTX] designated as the answer entity, your generated question must target [ENTX] as its final answer.
- 2. If the sentences contain multiple distinct placeholders, treat all placeholders other than the target [ENTX] as intermediate answers. You must integrate them into the multi-hop question as modifiers, relative clauses, or nested constraints.
- 3. The generated multi-hop question must be constructed such that answering it requires referencing all the provided declarative sentences.
- 4. If you determine that the declarative sentences cannot yield an unambiguous multi-hop question that strictly relies on every sentence, simply output: "No Solution".

Your output format:
[(
Sentence_1,
Sentence_2,
...), Multi-Hop Question]

Example:
[EXAMPLE_1]
[EXAMPLE_2]
[EXAMPLE_3]

Given Triple List: [Triple_1, Triple_2, ...]
Answer Entity: [ENTX]

Figure 19: The prompt template for Query & Assertions Generation based on Sampled-Graph ($\mathcal{P}_6$).

H Structure-Tracing Subgraph Retrieval

Algorithm 1 and 2 illustrate the overall execution and helper functions of Structure-Tracing Subgraph Retrieval, encompassing the execution procedures for two search modes. Descriptions of the key helper functions are provided below:

- CONTRADICT: Determines whether a specific triple t already exists within the list of currently matched triples.
- GET_TAIL: Given a triple t and one of its constituent entities e, retrieves the complementary entity node.
- GET_N: Retrieves all incident edges (not differentiate edge direction) of entity e within and $\mathcal{G}$ returns them as a list of triples.
- BUILD_GRAPH: Constructs a graph structure from a list of triples.

Algorithm 1 Structure-Tracing Subgraph Retrieval

Input: Pattern graph $\mathcal{G}_{sch} = (\mathcal{N}_Q, \mathcal{R}_Q)$; KG $\mathcal{G}$; the matching entity e in $\mathcal{G}_{sch}$ corresponding to the question entity; globally-aware entity score S_e^*; retrieval strategy σ; confidence threshold θ for *Breadth* Strategy;
Output: Matched subgraph $\mathcal{G}_{sch}^*$;

$e^* \leftarrow$ Identify the matching entity node in $\mathcal{G}$ corresponding to entity node e in $\mathcal{G}_{sch}$.
$S_0 = S_e^*[e^*]$
$final\_list \leftarrow \emptyset$
$last\_visit \leftarrow$ **None**
$\mathcal{T}_{all} \leftarrow$ MATCH$(\mathcal{G}, \mathcal{G}_{sch}, S, e, e^*, final_list, last_visit, \sigma, \theta)$
$\mathcal{G}_{sch}^* \leftarrow$ BUILD_GRAPH$(\mathcal{T}_{all})$
return $\mathcal{G}_{sch}^*$

function CONTRADICT(t, $final\_list$)
  if t in $final\_list$ **then**
    return TRUE
  else
    return FALSE

function GET_N(e, $\mathcal{G}$)
  triples ←Fetch all incident edges for entity e in $\mathcal{G}$ and output them as a triple list.
  return triples

function MATCH($\mathcal{G}, \mathcal{G}_{sch}, S, e, e^*, final\_list, last\_visit, \sigma, \theta$)
  for each t in GET_N($e, \mathcal{G}_{sch}$) **do**
    if $last\_visit$=t **then**
      continue
    if σ = "Precision" **then**
      $step\_list \leftarrow$ STEP_PRECISION$(\mathcal{G}, \mathcal{G}_{sch}, S, e, e^*, t, final_list, \sigma)$
      $final\_list \leftarrow final\_list \cup step\_list$
    if σ = "Breadth" **then**
      $step\_list \leftarrow$ STEP_BREADTH$(\mathcal{G}, \mathcal{G}_{sch}, S, e, e^*, t, final_list, \sigma, \theta)$
      $final\_list \leftarrow final\_list \cup step\_list$
  return $final\_list$

Algorithm 2 Subgraph Search Single Step Functions

function STEP_PRECISION($\mathcal{G}, \mathcal{G}_{sch}, S, e, e^*, t,$ $final\_list, \sigma$)
  $N_{e^*} \leftarrow$ GET_N($e^*, \mathcal{G}$)
  $max\_score \leftarrow -1$
  $max\_t \leftarrow$ *None*
  for each t^* in N_{e^*} **do**
    if CONTRADICT (t^*, $final\_list$) **then**
      continue
    $S^{'} \leftarrow S +$ T-SCORE(t^*, t)
    if $S^{'} > max\_score$ **then**
      $max\_score \leftarrow S^{'}$
      $max\_t \leftarrow t^*$
  $final\_list \leftarrow final\_list \cup \{max\_t\}$
  $e_t \leftarrow$ GET_TAIL(t, e)
  if $max\_t \neq$ *None* **then**
    $e_t^* \leftarrow$ GET_TAIL(max_t, e^*)
    $step\_list \leftarrow$ MATCH($\mathcal{G}, \mathcal{G}_{sch}, max\_score, e_t,$ $e_t^*, final\_list, t, \sigma,$ *None*)
    $final\_list \leftarrow final\_list \cup step\_list$
  return $final\_list$

function STEP_BREADTH($\mathcal{G}, \mathcal{G}_{sch}, S, e, e^*, t,$ $final\_list, \sigma, \theta$)
  $N_{e^*} \leftarrow$ GET_N($e^*, \mathcal{G}$)
  $candidates \leftarrow \emptyset$
  $list\_t \leftarrow \emptyset$
  for each t^* in N_{e^*} **do**
    if CONTRADICT (t^*, $final\_list$) **then**
      continue
    if T-SCORE(t^*, t) $>= \theta$ **then**
      $S^{'} \leftarrow S +$ T-SCORE(t^*, t)
      $candidates \leftarrow candidates \cup \{(S^{'}, t^*)\}$
      $final\_list \leftarrow final\_list \cup \{t^*\}$
  $e_t \leftarrow$ GET_TAIL(t, e)
  for each ($score, t^*$) in $candidates$ **do**
    $e_t^* \leftarrow$ GET_TAIL(t^*, e^*)
    $step\_list \leftarrow$ MATCH($\mathcal{G}, \mathcal{G}_{sch}, score, e_t, e_t^*,$ $final\_list, t, \sigma, \theta$)
    $final\_list \leftarrow final\_list \cup step\_list$
  return $final\_list$