GRAPHPILOT: FAITHFUL AND ADAPTIVE REASONING FOR LARGE LANGUAGE MODELS VIA GRAPH-CONSTRAINED EXPLORATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Knowledge graphs (KGs) provide structured knowledge to enhance the reasoning capabilities of large language models (LLMs). However, prevailing methods typically leverage KGs as external evidence or training supervision, while reasoning paths are still generated from LLMs' parametric distribution. Consequently, generated paths may exhibit hallucinations and lack support from KGs. Moreover, fixed reasoning depths and candidate path budgets fail to accommodate varying question requirements, potentially introducing irrelevant noise or missing valid paths. To address these limitations, we propose GraphPilot, which formulates KG reasoning as adaptive exploration within a graph-constrained decoding space. First, it converts KGs into executable token-level decoding constraints and allows LLMs to terminate at valid entity boundaries, enabling graph-faithful reasoning with adaptive depth. Second, it employs divergence-aware policy optimization to encourage adaptive exploration within the constrained space. Finally, it incorporates a relation-chain refinement mechanism, which prunes redundant candidate paths according to relation-chain semantics. Extensive experiments on multiple KGQA benchmarks demonstrate that GraphPilot achieves state-of-the-art performance. Further analysis shows that it maintains graph-faithful reasoning while supporting adaptive exploration.

1 INTRODUCTION

Large language models (LLMs) have demonstrated remarkable capabilities in many complex tasks (Zhang et al., 2026; Wei et al., 2026; Liang et al., 2026). Despite these advances, LLMs still struggle with faithful reasoning due to knowledge gaps and hallucinations (Wu et al., 2026; Fatahi Bayat et al., 2026), which can lead to unsupported intermediate reasoning steps and unreliable conclusions (Franzmeyer et al., 2026; Li et al., 2026). These limitations underscore the need for reasoning mechanisms that can enhance the faithfulness and reliability of LLM reasoning.

Knowledge graphs (KGs) organize factual knowledge into structured entities and relations, providing explicit and verifiable evidence for faithful reasoning (Ao et al., 2025; Lin et al., 2026). In KG-enhanced LLM reasoning, KGs primarily play two roles: *serving as external evidence* and *providing training supervision*, as shown in Figure 1 (a) and (b). In the first paradigm, KGs serve as sources for subgraph retrieval (Mavromatis & Karypis, 2025; Liu et al., 2025; Long et al., 2025; Peng et al., 2026) or environments for iterative interaction (Sun et al., 2024; Chen et al., 2024), providing external knowledge inputs to LLMs. In the second paradigm, KGs provide training data that guide LLMs to learn graph-based reasoning patterns and incorporate structured knowledge into their parameters (Liu et al., 2026; Yan et al., 2026).

Despite the different roles of KGs, both paradigms still rely on LLMs' parametric knowledge to generate reasoning paths, which may contain hallucinations and deviate from KGs even when relevant knowledge is available. This limitation is consistent with findings that even strong LLMs can generate content inconsistent with factual knowledge or the provided context (Ravichander et al., 2025; Li et al., 2026). Beyond faithfulness, fixed reasoning depths and candidate budgets in KG reasoning can limit adaptation to individual questions, leading to missed valid knowledge or unnecessary exploration that introduces noise (Luo et al., 2025; Yan et al., 2026). Therefore, it is essential

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

to ensure that LLM reasoning remains faithful to the available knowledge while making full use of relevant evidence to derive reliable reasoning paths.

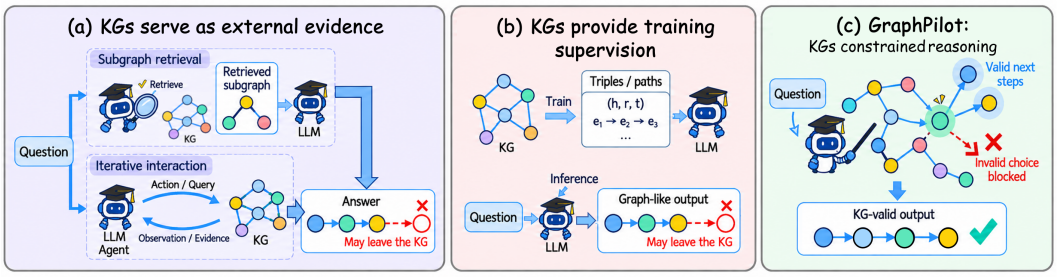


Figure 1: Comparison of existing KG-enhanced LLM reasoning paradigms with GraphPilot.

In this paper, we propose **GraphPilot**, which enables graph-faithful and adaptive reasoning by guiding LLMs to explore within a graph-constrained decoding space. To ensure faithful use of graph knowledge and enable adaptive reasoning depth, we introduce ***Entity-Aware Graph-Constrained Reasoning***, which converts KGs into executable token-level decoding constraints and allows LLMs to terminate at valid entity boundaries. Building on this reasoning mechanism, we design ***Divergence-Aware Policy Optimization*** to enhance path exploration and adaptive termination within the constrained space. To mitigate noise introduced by fixed candidate path budgets, we introduce a ***Relation-Chain Refinement*** mechanism that prunes redundant candidate paths based on their semantic relevance to the question. Extensive experiments on KGQA benchmarks demonstrate that GraphPilot achieves state-of-the-art performance, substantially outperforming a range of baselines while enabling faithful and adaptive reasoning.

2 Related Work

Knowledge graphs organize factual knowledge into structured entities and relations, providing explicit factual grounding to mitigate hallucinations and support more reliable LLM reasoning (Ji et al., 2022; Zhang et al., 2022; Sui et al., 2025). Recent studies on KG-enhanced reasoning primarily use KGs in two ways: as external evidence (Chen et al., 2024; Peng et al., 2026) or as training supervision (Yan et al., 2026).

When KGs serve as external evidence, existing methods typically acquire question-relevant graph knowledge through subgraph retrieval or iterative interaction with the KG, and provide it as external context to support LLM reasoning. For example, EWEK-QA (Dehghan et al., 2024) improves retrieval efficiency by incorporating KG triples without additional LLM calls. To better handle complex multi-hop questions, GNN-RAG (Mavromatis & Karypis, 2025) uses graph neural retrieval to identify relevant candidates and construct reasoning paths. To reduce irrelevant evidence, EPERM (Long et al., 2025) selects and weights reasoning paths from retrieved subgraphs according to their importance. More recently, COSMOS (Peng et al., 2026) jointly optimizes semantic relevance and structural connectivity to retrieve coherent evidence subgraphs. With the rise of LLM agents, recent work has explored iterative interaction with KGs to progressively gather reasoning evidence. ToG (Sun et al., 2024) iteratively explores entities and relations to identify promising paths, while PoG (Chen et al., 2024) introduces planning and reflection for self-correcting graph exploration. KG-Agent (Jiang et al., 2024) further enables autonomous tool use and iterative KG interaction through an agentic framework.

When KGs serve as training supervision, graph-derived structures and reasoning trajectories are used to strengthen LLM reasoning. RoG (Luo et al., 2024) distills KG-grounded relation paths into the model, while KG-SFT (Chen et al., 2025) and LACT (Xia et al., 2025) leverage structured graph supervision for fine-tuning. More recently, PathMind (Liu et al., 2026) introduces path-wise preference alignment, and EoG (Yan et al., 2026) uses reinforcement learning with graph-based rewards for autonomous KG exploration.

In addition, recent studies have increasingly focused on improving the faithfulness of LLM reasoning through symbolic reasoning, rationale alignment, and intermediate-step verification (Xu et al., 2024; Paul et al., 2024; Li et al., 2025; Sadeddine & Suchanek, 2025). In particular, GCR (Luo et al., 2025) introduces graph-constrained decoding to restrict LLM generation to KG-valid reasoning paths, thereby ensuring graph-faithful reasoning.

Different from previous work, GraphPilot goes beyond graph-faithful generation by adaptively exploring the constrained space, enabling question-specific control over both reasoning depth and candidate path selection.

3 METHODOLOGY

In this paper, we introduce GraphPilot, a graph-constrained reasoning framework that enables faithful and adaptive exploration over knowledge graphs. The overall architecture of GraphPilot is illustrated in Figure 2.

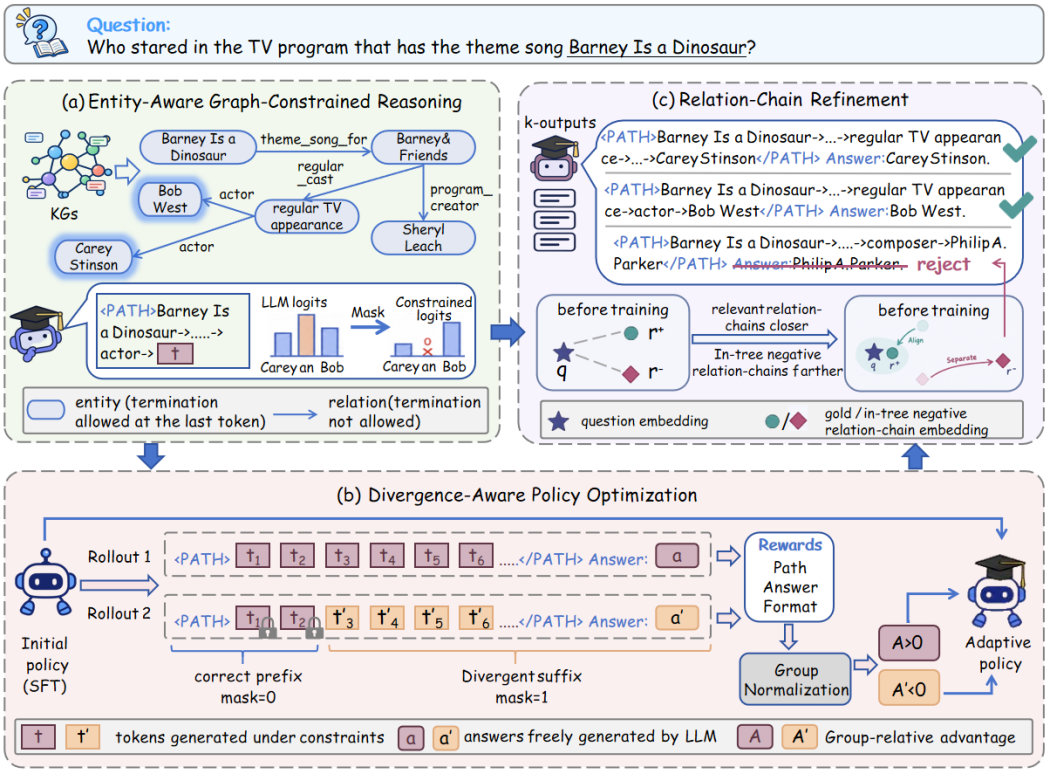


Figure 2: The overall framework of our **GraphPilot**. (a) Entity-Aware Graph-Constrained Reasoning constructs a reasoning space with flexible termination and establishes graph-based constraints over the decoding process. (b) Divergence-Aware Policy Optimization encourages LLMs to adaptively explore this space. (c) Relation-Chain Refinement refines the explored reasoning paths by filtering redundant candidates.

3.1 ENTITY-AWARE GRAPH CONSTRAINED REASONING

Structured Reasoning Format Alignment. We first perform supervised fine-tuning (SFT) to align the LLM with a structured KG reasoning format (Ren et al., 2024). Given a question q, the target output consists of a serialized reasoning path $p = (e_0, r_1, e_1, \ldots, r_h, e_h)$ followed by an answer a:

$$y = \texttt{<PATH>} \oplus \mathrm{Ser}(p) \oplus \texttt{</PATH>} \oplus \mathrm{Ser}(a), \tag{1}$$

where each (e_{j-1}, r_j, e_j) is a valid KG triple, $\mathrm{Ser}(\cdot)$ denotes serialization, and $\oplus$ denotes concatenation. The path delimiters explicitly separate structured reasoning from answer generation, with </PATH> marking termination after a complete entity.

Given the supervised dataset $\mathcal{D}_{\mathrm{SFT}}$ (Luo et al., 2024), we minimize the autoregressive negative log-likelihood:

$$\mathcal{L}_{\mathrm{SFT}}(\theta) = -\mathbb{E}_{(q,y)\sim\mathcal{D}_{\mathrm{SFT}}}\left[\sum_{t=1}^{|y|}\log \pi_\theta(y_t \mid q, y_{<t})\right]. \tag{2}$$

This initialization teaches the model the path serialization and termination format. However, format alignment alone does not guarantee KG-valid generation; we therefore enforce graph-derived decoding constraints in the following step.

Graph-Constrained Space Construction. Given a question q, its topic entity set $\mathcal{E}_q$, and a knowledge graph $\mathcal{G}$, we first construct a question-specific subgraph $\mathcal{G}_q$ through bounded, question-aware layer-wise expansion from the topic entities; details are provided in Appendix B. To reduce storage overhead, we convert $\mathcal{G}_q$ into a prefix tree.

To enable valid and adaptive termination, we attach a termination branch to each eligible entity node before tokenization, ensuring that reasoning can terminate only at complete entity boundaries. We then serialize and tokenize the augmented tree to obtain the token-level constrained decoding space. For a generated prefix u, the admissible next-token set is

$$\mathcal{A}_q(u) = \mathrm{Next}_{\mathcal{G}_q}(u) \cup \begin{cases} \{\texttt{</PATH>}\}, & u \in \mathcal{B}_q, \\ \varnothing, & \text{otherwise}, \end{cases} \tag{3}$$

where $\mathrm{Next}_{\mathcal{G}_q}(u)$ denotes the valid successor tokens represented by the tokenized prefix tree, and $\mathcal{B}_q$ denotes the eligible entity-boundary prefixes at which termination is permitted. This construction restricts generation to graph-valid transitions in $\mathcal{G}_q$ while allowing the model to choose between continuation and termination.

Graph-Constrained Decoding. During reasoning-path generation, we restrict the LLM's next-token distribution to the admissible set $\mathcal{A}_q(y_{<t})$:

$$\pi_\theta^{\mathrm{GC}}(y_t \mid q, y_{<t}) = \frac{\pi_\theta(y_t \mid q, y_{<t})\mathbb{I}[y_t \in \mathcal{A}_q(y_{<t})]}{\sum_{v\in\mathcal{A}_q(y_{<t})}\pi_\theta(v \mid q, y_{<t})}, \tag{4}$$

where $y_{<t}$ denotes the generated path prefix and $\mathbb{I}[\cdot]$ is the indicator function. At eligible entity boundaries, the termination token competes with graph-valid continuation tokens under the same constrained policy. The model therefore jointly determines which path to follow and when to terminate, adapting reasoning depth to the question within the prescribed bounds. The constrained reasoning process described above ensures that the generated evidence remains faithful to the knowledge graph. In contrast, answer generation requires the LLM to integrate the evidence with the question. Therefore, once </PATH> is generated, the graph constraint is released, allowing the model to generate the final answer freely based on the grounded reasoning path.

3.2 Divergence-Aware Policy Optimization

Graph-constrained decoding ensures KG-consistent paths, but SFT does not explicitly optimize path selection or termination. Inspired by recent advances such as Yan et al. (2026); Jiang et al. (2026), we therefore introduce divergence-aware policy optimization, which preserves reference-consistent prefixes when assigning negative policy feedback.

Trajectory-Level Reward. For each question q, we sample G outputs $\{y_i\}_{i=1}^{G}$ from the behavior policy $\tilde{\pi}_{\theta_{\mathrm{old}}}$, each comprising a reasoning path and an answer. We compute a composite reward:

$$R_i = \lambda_{\mathrm{ans}} R_{\mathrm{ans},i} + \lambda_{\mathrm{path}} R_{\mathrm{path},i} + \lambda_{\mathrm{fmt}} R_{\mathrm{fmt},i}, \tag{5}$$

where $R_{\mathrm{ans},i}$ measures answer correctness, $R_{\mathrm{path},i}$ rewards alignment with reference reasoning paths that lead to the correct answer, and $R_{\mathrm{fmt},i}$ measures output-format compliance. Detailed definitions are provided in Appendix C. Following GRPO (Shao et al., 2024), we obtain group-normalized advantages $\hat{A}_i = (R_i - \mu_R)/(\sigma_R + \delta)$, where μ_R and σ_R are the group reward mean and standard deviation, and $\delta > 0$ ensures numerical stability.

Divergence-Aware Credit Assignment. A low-reward trajectory may share a correct prefix with a reference path. Applying its negative advantage to every token would penalize these shared decisions. For each generated path, we select the reference with the longest common token prefix and denote the first divergent position by d_i. The comparison includes the closing path delimiter to account for termination errors; for an exact match, d_i is the first position after this delimiter. We define the token-level advantage as

$$\hat{A}_{i,t}^{\text{div}} = \begin{cases} 0, & \hat{A}_i < 0 \text{ and } t < d_i, \\ \hat{A}_i, & \text{otherwise.} \end{cases} \tag{6}$$

Thus, positive advantages apply to the entire output, while negative advantages apply only from the first divergence onward.

Policy Optimization Objective. We maximize the clipped objective:

$$\mathcal{J}_{\text{div}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\Big(\min\big\{\rho_{i,t}\hat{A}_{i,t}^{\text{div}}, \bar{\rho}_{i,t}\hat{A}_{i,t}^{\text{div}}\big\} - \beta D_{i,t}\Big)\right], \tag{7}$$

where the expectation is over training questions and outputs sampled from the behavior policy, and

$$\rho_{i,t} = \frac{\tilde{\pi}_\theta(y_{i,t} \mid q, y_{i,<t})}{\tilde{\pi}_{\theta_{\text{old}}}(y_{i,t} \mid q, y_{i,<t})}, \qquad \bar{\rho}_{i,t} = \text{clip}(\rho_{i,t}, 1-\epsilon, 1+\epsilon). \tag{8}$$

Here, $\tilde{\pi}$ uses graph-constrained probabilities within the path segment and the original model distribution elsewhere. $D_{i,t}$ is the conditional KL divergence from the current policy to the SFT reference policy under the same constraints; ϵ and β control clipping and regularization. Divergence-aware masking affects only the policy surrogate, leaving KL regularization unchanged.

3.3 Relation-Chain Refinement

Adaptive path selection and termination do not determine how many paths a question requires. With a fixed candidate budget K, generation may produce KG-valid but question-irrelevant trajectories. We therefore introduce a lightweight relevance encoder to filter candidates using their relation-chain semantics.

Relation-Level Contrastive Learning. We represent each path as an ordered relation chain, retaining only the terminal field of each relation identifier to remove namespace-specific redundancy. For each training question, chains ending at gold answers are positives, while those from other graph-valid paths are negatives. These negatives share the same question entities and satisfy KG constraints, encouraging the encoder to distinguish semantic relevance from structural validity.

Given a training triplet (q_i, r_i^+, r_i^-), a shared encoder E_ϕ computes cosine similarities $s_i^\pm = \cos(E_\phi(q_i), E_\phi(r_i^\pm))$. We optimize a calibrated contrastive objective:

$$\mathcal{L}_{\text{rel}} = \frac{1}{B}\sum_{i=1}^{B}\Big[\text{ReLU}(m - s_i^+ + s_i^-) + \sum_{c\in\{+,-\}}\big(\lambda_b d_{\mathcal{I}_c}(s_i^c) + \lambda_c(s_i^c - \mu_c)^2\big)\Big], \tag{9}$$

where B is the batch size, m is the ranking margin, $\mathcal{I}_c$ is the target similarity interval for class c, and μ_c is its batch mean similarity. The interval penalty is

$$d_{[l,u]}(s) = \text{ReLU}(l-s)^2 + \text{ReLU}(s-u)^2. \tag{10}$$

The three terms encourage positive–negative separation, calibrate similarity ranges, and reduce within-class dispersion, respectively; λ_b and λ_c weight the latter two terms.

Adaptive Candidate Filtering. At inference time, given candidates $\mathcal{O}_q = \{(p_i, a_i)\}_{i=1}^K$, we extract the relation chain r_i from each path p_i and retain

$$\mathcal{O}_q^{\text{rel}} = \big\{(p_i, a_i) \in \mathcal{O}_q \mid \cos(E_\phi(q), E_\phi(r_i)) \geq \gamma\big\}, \tag{11}$$

where γ is a relevance threshold. Filtering operates on path–answer pairs, converting the fixed candidate budget into a question-dependent output set.

4 EXPERIMENTS

We conduct experiments to evaluate the effectiveness of GraphPilot and address the following research questions: **RQ1:** How does GraphPilot perform compared to existing competitive methods? **RQ2:** What are the contributions of different components to the overall performance of GraphPilot? **RQ3:** Can GraphPilot achieve faithful and adaptive reasoning?

4.1 EXPERIMENTAL SETTINGS

Datasets. Following prior studies (Luo et al., 2024; Yan et al., 2026), we evaluate GraphPilot on two widely used KGQA benchmarks: WebQSP (Yih et al., 2016) and CWQ (Talmor & Berant, 2018). The details of the datasets are described in Appendix D.

Baselines. We compare GraphPilot with 20 representative baselines grouped into three categories: (1) ***LLM Reasoning***, which evaluates the standalone reasoning capability of LLMs without explicit graph knowledge (DeepSeek-AI et al., 2025; OpenAI, 2025); (2) ***Graph Reasoning***, which performs structured reasoning directly over knowledge graphs without relying on LLMs (Mavromatis & Karypis, 2022; Jiang et al., 2023); and (3) ***KG Enhanced LLM Reasoning***, which leverages KGs as external evidence or training supervision (Luo et al., 2025; Yan et al., 2026). Detailed descriptions of all baselines are provided in Appendix E.

Evaluation Metrics. Following prior work (Luo et al., 2025; Peng et al., 2026), we evaluate model performance using two standard metrics: Hit and F1. Hit measures whether the predicted answer set contains at least one correct answer. F1 balances precision and recall over the predictions to measure the overall coverage of the gold answer set.

Table 1: Performance comparison with different baselines on the two KGQA datasets.

Types	Methods	WebQSP		CWQ	
		Hit	F1	Hit	F1
LLM Reasoning	Llama-3.1-8B (Grattafiori et al., 2024)	55.5	34.8	28.1	22.4
	DeepSeek-v3 (DeepSeek-AI et al., 2025)	64.0	43.9	41.1	33.8
	GPT-4o-mini (OpenAI, 2024)	63.8	40.5	42.2	32.9
	GPT-5 (OpenAI, 2025)	86.1	77.5	74.1	67.6
	Gemini-2.5 Flash (Google, 2025)	91.8	78.2	65.5	59.3
Graph Reasoning	ReaRev (Mavromatis & Karypis, 2022)	76.4	70.9	52.9	47.8
	UniKGQA (Jiang et al., 2023)	77.2	72.2	51.2	49.1
	DECAF (Yu et al., 2023)	82.1	78.8	70.4	–
KG-enhanced LLM Reasoning	EWEK-QA (Dehghan et al., 2024)	71.3	–	52.5	–
	ToG (Sun et al., 2024)	82.6	–	68.5	–
	PoG (Chen et al., 2024)	87.3	–	75.0	–
	RoG (Luo et al., 2024)	85.7	70.8	62.6	56.2
	GNN-RAG (Mavromatis & Karypis, 2025)	85.7	71.3	66.8	59.4
	ORT (Liu et al., 2025)	87.7	71.8	65.4	58.7
	GCR (Luo et al., 2025)	88.3	65.4	68.6	53.2
	EPERM (Long et al., 2025)	88.8	72.4	66.2	58.9
	PathMind (Liu et al., 2026)	89.5	72.8	70.7	61.4
	OntGQA (Shan et al., 2026)	91.5	77.1	74.6	56.8
	COSMOS (Peng et al., 2026)	91.6	78.2	65.7	55.6
	EoG (Yan et al., 2026)	92.8	81.3	86.6	77.9
	GraphPilot (Ours)	**93.7**	**82.6**	**89.8**	**81.1**

Implementations. For GraphPilot, we use `Llama-3-8B` (Grattafiori et al., 2024) as the LLM backbone. For the refinement module, we initialize `all-MiniLM-L6-v2` from Sentence Transformers (Reimers & Gurevych, 2019) as the sentence encoder. We set the maximum reasoning

depth of the LLM under graph constraints to 20 hops, while allowing early termination at valid entity boundaries. At inference time, we adopt group beam search with `num_beams=10` to generate multiple candidate reasoning paths. Detailed hyperparameters and experimental settings are provided in Appendix F.

4.2 **RQ1:** Overall Comparison and Efficiency

Main Results. Table 1 compares GraphPilot with three categories of reasoning methods, exhibitting three notable advantages. First, GraphPilot outperforms the other KG-enhanced methods at a comparable model scale and also surpasses the evaluated closed-source LLMs. These results support the effectiveness of its overall design. Second, a key challenge in multi-answer question answering is to achieve broad coverage of correct answers while avoiding irrelevant predictions. The relatively strong Hit but lower F1 scores of GCR and OntGQA further demonstrate this difficulty. In contrast, GraphPilot improves both Hit and F1, suggesting that it retrieves correct answers more reliably while better balancing answer coverage and noise control. Third, GraphPilot achieves larger gains on the more challenging CWQ dataset, indicating its effectiveness in addressing the reasoning demands of complex multi-hop questions.

Efficiency Analysis. To evaluate the efficiency of LLM inference, we report the average LLM generation time, number of LLM calls, and number of input tokens in Table 2. Specifically, KG-supervised methods benefit from knowledge acquired during training, reducing their reliance on iterative interaction and external evidence input. In contrast, retrieval-based methods require additional input tokens to incorporate retrieved evidence, while agent-based methods incur more LLM calls due to their iterative reasoning process. GraphPilot instead incorporates graph knowledge directly as decoding constraints, achieving a favorable balance between performance and LLM inference efficiency. The graph-constrained space is constructed separately before decoding; even with a maximum graph-expansion depth of 20 hops, this preprocessing takes about 5 seconds per question on average, and the resulting constrained space can be cached and reused for subsequent decoding.

Table 2: Efficiency comparison of GraphPilot.

Type	Method	Hit (%) ↑	Avg. Runtime (s) ↓	Avg. LLM Calls ↓	Avg. LLM Tokens ↓
			Retrieval-Based Methods		
KGs as external evidence	GNN-RAG	85.7	1.5	1	414
	RoG	85.7	2.6	2	521
			Agent-Based Methods		
	ToG	82.6	16.1	11.6	7,069
	PoG	87.3	16.8	9	5,234.8
KGs as training supervision	EoG	92.8	2.3	1	120
Ours	GraphPilot	**93.7**	3.1	1	**101**

4.3 **RQ2:** Ablation Study

We conduct an ablation study to analyze the effectiveness of the Divergence-Aware Policy Optimization (RL) and Relation-Chain Refinement (Refinement). As shown in Table 3, the full GraphPilot achieves the best performance on both datasets. The absence of RL yields inferior performance, indicating the necessity of encouraging effective exploration within the constrained reasoning space. On the other hand, removing the refinement mechanism and relying solely on fixed candidate path budgets leads to a noticeable decrease in precision, due to noisy paths. This highlights the necessity of adapting the number of candidate paths to individual questions. Furthermore, regardless of whether reinforcement learning is applied, the refinement mechanism substantially improves Precision and F1, while causing only slight decreases in Hit and Recall. Therefore, our method achieves the best balance between answer coverage and prediction precision.

Table 3: Ablation results on WebQSP and CWQ. RL denotes Divergence-Aware Policy Optimization, and Refinement denotes Relation-Chain Refinement. P and R denote precision and recall, respectively. The best result in each column is boldfaced.

Variant	WebQSP				CWQ			
	Hit	P	R	F1	Hit	P	R	F1
GraphPilot	93.7	**92.3**	74.8	**82.6**	89.8	**80.7**	81.5	**81.1**
w/o RL	87.2	74.5	69.8	72.1	63.9	51.9	61.7	56.4
w/o Refinement	**96.0**	62.3	**77.6**	69.1	**90.1**	67.0	**82.7**	74.0
w/o RL and Refinement	91.1	43.1	73.3	54.3	78.2	30.0	70.2	42.0

4.4 RQ3: Faithful and Adaptive Reasoning

Faithfulness Study. To evaluate the faithfulness of generated reasoning trajectories, we report Hit and the proportion of KG-supported paths among correct predictions. As shown in Figure 3, both retrieval-based RoG and training-based EoG can still generate hallucinated reasoning paths despite producing correct answers. Both GCR and GraphPilot achieve high graph faithfulness through graph-constrained reasoning. However, GraphPilot consistently delivers stronger performance due to its adaptive exploration capability. We next analyze this adaptivity in greater detail.

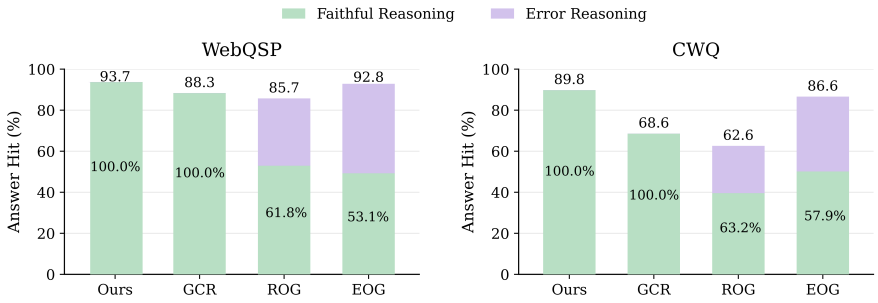


Figure 3: Hit and faithfulness of reasoning paths on WebQSP and CWQ.

Performance Analysis of Adaptive Reasoning. To further analyze the adaptive reasoning capability of GraphPilot, we study its performance under varying maximum reasoning depths and candidate path budgets, as shown in Figure 4.

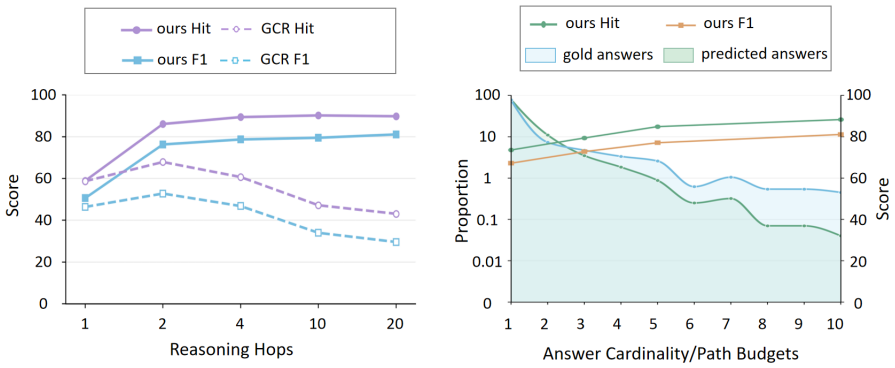


Figure 4: Performance analysis under different reasoning-hop and candidate-path budgets.

First, we compare GraphPilot and GCR under different reasoning depths. As the reasoning hops increase from 1 to 2, both methods achieve improved performance, since 2-hop reasoning provides

access to substantially more relevant evidence. As the reasoning depth increases further, GraphPilot continues to improve, benefiting from the expanded scope of accessible evidence. However, GCR suffers from performance degradation because it is more susceptible to unnecessary continuation under larger depth limits. Second, we analyze the performance of GraphPilot under different candidate path budgets. Benefiting from the relation-chain refinement mechanism, GraphPilot effectively filters out irrelevant paths and maintains strong performance across different budgets. In addition, we compare the distribution of the number of gold answers per question with that of the final predicted answers. The two distributions exhibit similar patterns, suggesting that GraphPilot can maintain broad coverage of multiple correct answers while mitigating the noise introduced by irrelevant paths.

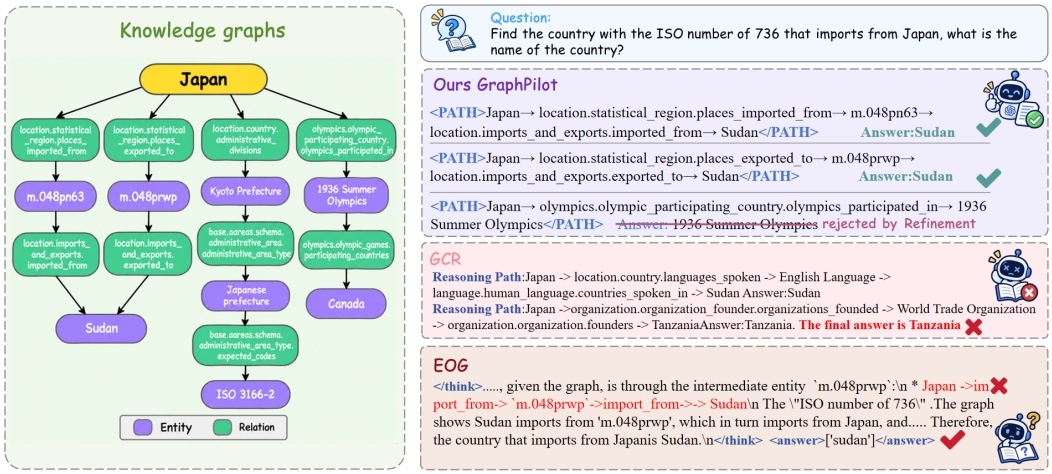


Figure 5: An example of a question and the corresponding outputs of GraphPilot.

Case Study. We further present a complex case to illustrate the effectiveness of GraphPilot. As shown in Figure 5, GraphPilot generates faithful reasoning paths that can be traced in the KG. When the path budget exceeds the number of paths needed to reach the correct answers, GraphPilot can terminate reasoning earlier and leverage the refinement mechanism to remove redundant paths. In contrast, GCR is more susceptible to noisy paths, which can lead to incorrect final answers, while EoG may generate reasoning paths that are not supported by the KG.

5 Conclusion

In this paper, we introduce GraphPilot, a graph-faithful and adaptive framework for KG-enhanced LLM reasoning. The core idea is to formulate KG reasoning as exploration within a graph-constrained decoding space, ensuring that generated reasoning paths are supported by the KG while allowing the reasoning process to adapt in both depth and path cardinality. Our method comprises three key components: entity-aware graph-constrained reasoning, divergence-aware policy optimization, and relation-chain refinement. Extensive results and model analysis demonstrate that GraphPilot consistently outperforms competitive baselines with faithful and adaptive reasoning.

AI Use Statement

In this work, we used generative AI tools to improve the readability of the paper and to assist in generating visualization code for presenting our experimental results as figures. The figures were produced from our actual experimental data rather than generated directly by AI. All AI-assisted visualization code and figures were manually reviewed by the authors and verified against the original experimental records. The remaining disclosure categories are not applicable to this work. We take full responsibility for the final content of this work.

Ethics statement

The data used in this work are drawn exclusively from publicly available, open-source KGQA datasets (see Appendix D). As these datasets contain no personally identifiable information or other sensitive data, our experiments do not raise apparent ethical concerns regarding data privacy or human subjects. We are committed to upholding rigorous standards of scientific integrity and research ethics to ensure the validity and reliability of our findings.

Reproducibility Statement

Our method is formally described in the main text, with additional implementation details provided in the appendices, including dataset descriptions (Appendix D), baseline methods (Appendix E), and experimental settings (Appendix F). We also provide part of the implementation code in the supplementary material to facilitate the reproduction of key components of our method. We are currently organizing the source code and related scripts, which will be released through a public GitHub repository upon acceptance of the paper to support reproducibility.

References

Tu Ao, Yanhua Yu, Yuling Wang, Yang Deng, Zirui Guo, Liang Pang, Pinghui Wang, Tat-Seng Chua, Xiao Zhang, and Zhen Cai. LightPROF: A lightweight reasoning framework for large language model on knowledge graph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 23424–23432, 2025. doi: 10.1609/aaai.v39i22.34510. URL https://doi.org/10.1609/aaai.v39i22.34510.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pp. 1247–1250, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605581026. doi: 10.1145/1376616.1376746. URL https://doi.org/10.1145/1376616.1376746.

Hanzhu Chen, Xu Shen, Jie Wang, Zehao Wang, Qitan Lv, Junjie He, Rong Wu, Feng Wu, and Jieping Ye. Knowledge graph finetuning enhances knowledge manipulation in large language models. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 91387–91409, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/e44337573fcac83f219c8effa4ebf90d-Paper-Conference.pdf.

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

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, et al. Deepseek-v3 technical report, 2025. URL https://arxiv.org/abs/2412.19437.

Mohammad Dehghan, Mohammad Alomrani, Sunyam Bagga, David Alfonso-Hermelo, Khalil Bibi, Abbas Ghaddar, Yingxue Zhang, Xiaoguang Li, Jianye Hao, Qun Liu, Jimmy Lin, Boxing Chen, Prasanna Parthasarathi, Mahdi Biparva, and Mehdi Rezagholizadeh. EWEK-QA: Enhanced web and efficient knowledge graph retrieval for citation-based question answering systems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14169–14187. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.764. URL https://aclanthology.org/2024.acl-long.764/.

Farima Fatahi Bayat, Pouya Pezeshkpour, and Estevam Hruschka. From proof to program: Characterizing tool-induced reasoning hallucinations in large language models. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*,

pp. 42151–42169, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1951. URL https://aclanthology.org/2026.acl-long.1951/.

Tim Franzmeyer, Archie Sravankumar, Lijuan Liu, Yuning Mao, Rui Hou, Sinong Wang, Jakob Foerster, Luke Zettlemoyer, and Madian Khabsa. High accuracy, less talk (halt): Reliable llms through capability-aligned finetuning. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 99495–99512, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/a17f63ce8ea28b396a679633ed00c76b-Paper-Conference.pdf.

Google. We're Expanding Our Gemini 2.5 Family of Models. https://blog.google/products/gemini/gemini-2-5-model-family-expands/, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):494–514, 2022. doi: 10.1109/TNNLS.2021.3070843.

Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, and Ji-Rong Wen. UniKGQA: Unified retrieval and reasoning for solving multi-hop question answering over knowledge graph. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Z63RvyAZ2Vh.

Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, Yang Song, Chen Zhu, Hengshu Zhu, and Ji-Rong Wen. Kg-agent: An efficient autonomous agent framework for complex reasoning over knowledge graph, 2024. URL https://arxiv.org/abs/2402.11163.

Zishang Jiang, Jinyi Han, Tingyun Li, Xinyi Wang, Sihang Jiang, Zhaoqian Dai, Ma Shuguang, Fei Yu, Jiaqing Liang, and Yanghua Xiao. Selective Expert Guidance for Effective and Diverse Exploration in Reinforcement Learning of LLMs. In *International Conference on Learning Representations*, 2026. URL https://iclr.cc/virtual/2026/poster/10008654.

Jiazheng Li, Hanqi Yan, and Yulan He. Drift: Enhancing LLM faithfulness in rationale generation via dual-reward probabilistic inference. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6850–6866, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.340. URL https://aclanthology.org/2025.acl-long.340/.

Shanghao Li, Jinda Han, Yibo Wang, Yuanjie Zhu, Zihe Song, Langzhou He, Kenan Kamel A Alghythee, and Philip S. Yu. Why LLMs hallucinate on structured knowledge: A mechanistic analysis of reasoning over linearized representations. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 19943–19956, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.914. URL https://aclanthology.org/2026.acl-long.914/.

Xiao Liang, Zhong-Zhi Li, Zhenghao Lin, Eric Hanchen Jiang, Hengyuan Zhang, Yelong Shen, Kai-Wei Chang, Ying Nian Wu, Yeyun Gong, and Weizhu Chen. Training LLMs for divide-and-conquer reasoning elevates test-time scalability. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 34402–34427, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1588. URL https://aclanthology.org/2026.acl-long.1588/.

Chenxiao Lin, Ye Luo, KunHong Liu, and Qingqiang Wu. Stepwise contrastive reasoning for retrieval-augmented generation over knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 15243–15251, 2026. doi: 10.1609/aaai.v40i18.38549. URL https://doi.org/10.1609/aaai.v40i18.38549.

Runxuan Liu, Bei Luo, Jiaqi Li, Baoxin Wang, Ming Liu, Dayong Wu, Shijin Wang, and Bing Qin. Ontology-guided reverse thinking makes large language models stronger on knowledge graph question answering. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15269–15284, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.741. URL https://aclanthology.org/2025.acl-long.741/.

Yu Liu, Xixun Lin, Yanmin Shang, Yangxi Li, Shi Wang, and Yanan Cao. Pathmind: a retrieve-prioritize-reason framework for knowledge graph reasoning with large language models. In *Proceedings of the Fortieth AAAI Conference on Artificial Intelligence and Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence and Sixteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'26/IAAI'26/EAAI'26. AAAI Press, 2026. ISBN 978-1-57735-906-7. doi: 10.1609/aaai.v40i18.38565. URL https://doi.org/10.1609/aaai.v40i18.38565.

Xiao Long, Liansheng Zhuang, Aodi Li, Minghong Yao, and Shafei Wang. Eperm: an evidence path enhanced reasoning model for knowledge graph question and answering. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'25/IAAI'25/EAAI'25. AAAI Press, 2025. ISBN 978-1-57735-897-8. doi: 10.1609/aaai.v39i12.33338. URL https://doi.org/10.1609/aaai.v39i12.33338.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. Reasoning on graphs: Faithful and interpretable large language model reasoning, 2024. URL https://arxiv.org/abs/2310.01061.

Linhao Luo, Zicheng Zhao, Gholamreza Haffari, Yuan-Fang Li, Chen Gong, and Shirui Pan. Graph-constrained reasoning: Faithful reasoning on knowledge graphs with large language models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 41540–41565. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/luo25t.html.

Costas Mavromatis and George Karypis. ReaRev: Adaptive reasoning for question answering over knowledge graphs. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 2447–2458, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.181. URL https://aclanthology.org/2022.findings-emnlp.181/.

Costas Mavromatis and George Karypis. GNN-RAG: Graph neural retrieval for efficient large language model reasoning on knowledge graphs. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 16682–16699, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-acl.856. URL https://aclanthology.org/2025.findings-acl.856/.

OpenAI. GPT-4o mini: Advancing Cost-Efficient Intelligence. https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/, 2024.

OpenAI. GPT-5 is Here. https://openai.com/gpt-5/, 2025.

Debjit Paul, Robert West, Antoine Bosselut, and Boi Faltings. Making reasoning matter: Measuring and improving faithfulness of chain-of-thought reasoning. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 15012–15032, Miami, Florida, USA, November 2024. Association for Computational

Linguistics. doi: 10.18653/v1/2024.findings-emnlp.882. URL https://aclanthology.org/2024.findings-emnlp.882/.

Boci Peng, Xiao Liu, Boren Hu, Yun Zhu, Xuanbo Fan, Yanwei Yue, Chunyu Yang, and Yan Zhang. COSMOS: Connectivity-oriented submodular maximization for optimal subgraph retrieval. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 35904–35924, San Diego, California, United States, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.1662. URL https://aclanthology.org/2026.acl-long.1662/.

Abhilasha Ravichander, Shrusti Ghela, David Wadden, and Yejin Choi. HALoGEN: Fantastic LLM hallucinations and where to find them. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1402–1425, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.71. URL https://aclanthology.org/2025.acl-long.71/.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410/.

Mengjie Ren, Boxi Cao, Hongyu Lin, Cao Liu, Xianpei Han, Ke Zeng, Wan Guanglu, Xunliang Cai, and Le Sun. Learning or self-aligning? rethinking instruction fine-tuning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6090–6105, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.330. URL https://aclanthology.org/2024.acl-long.330/.

Zacchary Sadeddine and Fabian M. Suchanek. Verifying the steps of deductive reasoning chains. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 456–475, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.25. URL https://aclanthology.org/2025.findings-acl.25/.

Yongxue Shan, Jie Peng, Zixuan Dong, Fei Hu, and Xiaodong Wang. Reasoning with ontology graph: Toward type-constrained knowledge graph question answering. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10684–10697, San Diego, California, United States, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.489. URL https://aclanthology.org/2026.acl-long.489/.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Yuan Sui, Yufei He, Zifeng Ding, and Bryan Hooi. Can knowledge graphs make large language models more trustworthy? an empirical study over open-ended question answering. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12685–12701, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.622. URL https://aclanthology.org/2025.acl-long.622/.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and

Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 3868–3898, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/10a6bdcabbd5a3d36b760daa295f63c1-Paper-Conference.pdf.

Alon Talmor and Jonathan Berant. The web as a knowledge-base for answering complex questions. In Marilyn Walker, Heng Ji, and Amanda Stent (eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 641–651, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1059. URL https://aclanthology.org/N18-1059/.

Zili Wei, Yilin Wang, Xiaocui Yang, Shi Feng, Weidong Bao, Daling Wang, Zihan Wang, and Yifei Zhang. CIRAG: Construction–integration retrieval and adaptive generation for multi-hop question answering. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 26181–26197, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1203. URL https://aclanthology.org/2026.acl-long.1203/.

Yuhe Wu, Guangyu Wang, Yuran Chen, Jiatong Zhang, Yutong Zhang, Yujie Chen, Jiaming Shang, Guang Zhang, and Zhuang Liu. PRISM: Probing reasoning, instruction, and source memory in LLM hallucinations. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 33619–33656, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1551. URL https://aclanthology.org/2026.acl-long.1551/.

Tianle Xia, Liang Ding, Guojia Wan, Yibing Zhan, Bo Du, and Dacheng Tao. Improving complex reasoning over knowledge graph with logic-aware curriculum tuning. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'25/IAAI'25/EAAI'25. AAAI Press, 2025. ISBN 978-1-57735-897-8. doi: 10.1609/aaai.v39i12.33405. URL https://doi.org/10.1609/aaai.v39i12.33405.

Jundong Xu, Hao Fei, Liangming Pan, Qian Liu, Mong-Li Lee, and Wynne Hsu. Faithful logical reasoning via symbolic chain-of-thought. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13326–13365, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.720. URL https://aclanthology.org/2024.acl-long.720/.

Shiqi Yan, Yubo Chen, Ruiqi Zhou, Zhengxi Yao, Shuai Chen, Tianyi Zhang, Shijie Zhang, Wei-Qiang Zhang, Yongfeng Huang, Haixin Duan, and Yunqi Zhang. Explore-on-graph: Incentivizing autonomous exploration of large language models on knowledge graphs with path-refined reward modeling. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 56761–56789, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/5d2e24df9cfaad3189833b819c40b392-Paper-Conference.pdf.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. The value of semantic parse labeling for knowledge base question answering. In Katrin Erk and Noah A. Smith (eds.), *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 201–206, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-2033. URL https://aclanthology.org/P16-2033/.

Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Yang Wang, Zhiguo Wang, and Bing Xiang. DecAF: Joint decoding of answers and logical forms for question answering over knowledge bases. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://iclr.cc/virtual/2023/poster/11802.

Jing Zhang, Xiaokang Zhang, Jifan Yu, Jian Tang, Jie Tang, Cuiping Li, and Hong Chen. Subgraph retrieval enhanced model for multi-hop knowledge base question answering. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5773–5784, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.396. URL https://aclanthology.org/2022.acl-long.396/.

Yonggang Zhang, Jianqi Gao, and Jie Lu. Enhancing llms for knowledge base question answering by chain-of-decomposition. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 93441–93457, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/96dfe8762de921e7287a7e91b17b3f5d-Paper-Conference.pdf.

A PRELIMINARIES

Knowledge Graphs (KGs). A knowledge graph is represented as a set of relational triples,

$$\mathcal{G} = \{(s, r, o) \mid s, o \in \mathcal{E},\ r \in \mathcal{R}\}, \tag{12}$$

where $\mathcal{E}$ and $\mathcal{R}$ denote the sets of entities and relations. Each triple (s, r, o) represents a factual statement in which the subject entity s is connected to the object entity o through relation r.

Reasoning Path. A reasoning path in a knowledge graph is defined as a sequence of connected relational triples that forms a coherent path from a starting entity to a target entity,

$$\mathcal{P} = e_0 \xrightarrow{r_1} e_1 \xrightarrow{r_2} \cdots \xrightarrow{r_L} e_L, \tag{13}$$

where each transition (e_{i-1}, r_i, e_i) corresponds to a valid triple in $\mathcal{G}$. Here, L denotes the reasoning depth, i.e., the number of hops in the path.

Problem Formulation. In this work, we aim to learn a reasoning function f that, given a question q and an available knowledge graph $\mathcal{G}$, generates multiple graph-valid reasoning paths and predicts their corresponding answers. Formally,

$$(\mathcal{P}_q, \mathcal{A}_q) = f(q, \mathcal{G}), \qquad \mathcal{P}_q \subseteq \mathcal{P}(\mathcal{G}), \tag{14}$$

where $\mathcal{P}(\mathcal{G})$ denotes the set of valid reasoning paths supported by $\mathcal{G}$, $\mathcal{P}_q$ is the set of reasoning paths generated for question q, and $\mathcal{A}_q$ denotes the corresponding predicted answers. Following prior work (Yan et al., 2026; Luo et al., 2025), we assume that each question has at least one valid path from a mentioned entity to a correct answer.

B GRAPH-CONSTRAINED SPACE CONSTRUCTION

Question-Specific Subgraph Construction. Given a question q, its topic entity set $\mathcal{E}_q$, and a knowledge graph $\mathcal{G}$, we construct a question-specific subgraph $\mathcal{G}_q$ through question-aware, layer-wise graph expansion from the Freebase KG (Bollacker et al., 2008). Since exhaustively enumerating all paths up to a large maximum depth would lead to combinatorial growth, we maintain a bounded search frontier and prune candidate paths after each expansion step according to their lexical relevance to the question. The overall procedure is summarized in Algorithm 1.

Algorithm 1 Question-Specific Subgraph Construction

Require: Question q, topic entity set $\mathcal{E}_q$, knowledge graph $\mathcal{G}$, maximum depth $H_{\max}$, frontier size K
Ensure: Question-specific subgraph $\mathcal{G}_q$

```
Initialize F_0 with zero-hop paths from entities in E_q
P_q ← F_0
for d = 0 to H_max − 1 do
    C_{d+1} ← Expand(F_d, G)
    for each p ∈ C_{d+1} do
        S(q, p) ← |T_q ∩ T_p| / |T_q| + 0.05 M(q, p) − 0.01 |p|
    end for
    F_{d+1} ← TopK_{p ∈ C_{d+1}} S(q, p)
    P_q ← P_q ∪ F_{d+1}
end for
Construct G_q from the entities and relations in P_q
return G_q
```

Here, $\mathrm{Expand}(\mathcal{F}_d, \mathcal{G})$ denotes all valid one-hop extensions of the paths in the current frontier. T_q is the set of non-stopword tokens in the question, while T_p contains the tokens extracted from the entities and relations along path p. Relation identifiers are lexically normalized before scoring. For

example, `location.country.languages_spoken` is converted to `location country languages spoken`. $M(q, p)$ denotes the number of question-token matches in the textual representation of path p, and $|p|$ denotes the number of hops in the path.

The first two terms favor paths with stronger lexical correspondence to the question, while the final term introduces only a mild preference for shorter paths. After each expansion step, we retain only the globally highest-scoring K paths as the frontier for further expansion. We set $K = 64$ for WebQSP and $K = 128$ for CWQ.

This procedure can be viewed as a lexically guided beam-style graph search. By bounding the frontier size at each depth, it controls the growth of deep graph expansion while preferentially preserving graph regions whose entities and relations are lexically relevant to the question. The entities and relations appearing in the retained paths are finally merged to form the question-specific subgraph $\mathcal{G}_q$.

Table 4: Statistics of the constructed constrained spaces on the test sets.

Dataset	# Questions	# Trees w/ Answer Path	Distractor Paths (%)
WebQSP	1,628	1,628	72.12
CWQ	3,531	3,531	70.11

The constructed constrained spaces retain answer-relevant evidence together with graph-valid distractors, requiring the model to distinguish relevant trajectories from competing alternatives. We further report statistics of the constructed constrained spaces for each dataset, as shown in Table 4.

Prefix-Tree Construction. Given the resulting question-specific subgraph $\mathcal{G}_q$, we convert it into a prefix-tree, where shared prefixes among valid graph paths are merged to reduce the storage overhead of the constrained space.

Prior work has demonstrated the feasibility of converting a knowledge graph into a prefix tree while maintaining favorable time and space efficiency. See Luo et al. (2025) for more details.

To support valid and adaptive reasoning termination, we augment the prefix tree before tokenization by attaching a termination branch to each eligible entity node. This ensures that reasoning can terminate only at complete entity boundaries. We then tokenize the augmented tree, yielding the token-level graph-constrained decoding space used during reasoning-path generation.

C Detailed Reward Formulation for Divergence-Aware Policy Optimization

Reward Overview. Our reward design jointly considers answer correctness, reasoning-path quality, and output-format compliance under graph-constrained reasoning. For each question q, we sample a group of G outputs $\{y_i\}_{i=1}^{G}$ from the behavior policy $\tilde{\pi}_{\theta_{\text{old}}}$, where each output contains a reasoning path and its corresponding answer. We assign each output a composite reward:

$$R_i = \lambda_{\text{ans}} R_{\text{ans},i} + \lambda_{\text{path}} R_{\text{path},i} + \lambda_{\text{fmt}} R_{\text{fmt},i}, \tag{15}$$

where $R_{\text{ans},i}$ measures answer correctness, $R_{\text{path},i}$ measures alignment with reference reasoning paths leading to correct answers, and $R_{\text{fmt},i}$ evaluates whether the output follows the predefined structured format. The coefficients λ_{ans}, λ_{path}, and λ_{fmt} control the relative contribution of the three reward components. Together, these signals encourage trajectories that produce correct answers, follow appropriate reasoning paths, and maintain the required output structure. In practice, we set $\lambda_{\text{ans}} = 0.3$, $\lambda_{\text{path}} = 0.6$, and $\lambda_{\text{fmt}} = 0.1$ during training.

Answer Reward. To encourage answer correctness, we compare each generated answer with all gold answers and use the highest word-level F1 score as the reward:

$$R_{\text{ans},i} = \max_{a^* \in \mathcal{A}_q} \text{F1}_{\text{word}}(a_i, a^*), \tag{16}$$

where a_i denotes the answer generated in the i-th output, $\mathcal{A}_q$ is the gold answer set for question q, and F1_{word} computes F1 based on word overlap between the generated and gold answers. This reward accommodates multiple valid answers and provides partial credit for incomplete matches.

Path Reward. Graph-constrained decoding guarantees that a generated reasoning path is structurally valid in the KG, but graph validity alone does not ensure that the path is relevant to the question. We therefore introduce a path reward that measures alignment with reference reasoning paths leading to correct answers:

$$R_{\text{path},i} = \max_{p^* \in \mathcal{P}_q^*} \text{F1}_{\text{pos}}(p_i, p^*), \tag{17}$$

where p_i is the generated path and $\mathcal{P}_q^*$ denotes the set of reference reasoning paths for question q.

It is worth noting that, within each question-specific subgraph, we extract valid shortest paths from the topic entities to the correct answers and use them as reference paths. These paths provide a concise supervision signal by capturing sufficient reasoning evidence for reaching the correct answer while avoiding unnecessary graph traversal. In particular, shortest-path supervision helps prevent RL from assigning high rewards to redundant extra hops during the early stages of training. This design is also aligned with adaptive termination: if valid paths of arbitrary length were treated as equally preferred references, the policy could be positively reinforced for unnecessary continuations even after sufficient evidence for answering the question has been obtained.

Each path is represented as an ordered sequence of entities and relations. F1_{pos} computes precision and recall based on exact entity and relation matches at corresponding positions. Consequently, a generated path can still receive partial credit even when it does not exactly reproduce a reference path, as long as part of its reasoning sequence is aligned with the reference. When multiple valid shortest paths exist for the same question, all of them are included in $\mathcal{P}_q^*$, and the maximum score is used so that the generated trajectory is not tied to a single reference reasoning route.

Format Reward. To support reliable parsing and reward computation, we assign a binary reward based on whether the output contains correctly formatted reasoning-path and answer fields:

$$R_{\text{fmt},i} = \begin{cases} 1, & \text{if the output follows the required format,} \\ 0, & \text{otherwise.} \end{cases} \tag{18}$$

This reward encourages structural compliance and discourages malformed outputs during reinforcement learning.

D Dataset Details

KGQA Datasets. For a fair comparison with existing methods, we conduct experiments on WebQSP (Yih et al., 2016) and CWQ (Talmor & Berant, 2018), each consisting of multiple question-answer pairs, where a question may have multiple valid answers. To ensure comparability, we follow the same training and test splits used in prior work (Luo et al., 2024; Liu et al., 2026; Yan et al., 2026). Detailed dataset statistics are provided in Table 5.

Table 5: Statistics of the KGQA datasets.

Dataset	Train	Test
WebQSP	2,826	1,628
CWQ	27,639	3,531

Fine-tuning Datasets. To ensure consistency with prior work, we use the training data provided by RoG (Luo et al., 2024) to fine-tune the LLM. The objective is to align the model with the structured format of path-based reasoning and answer prediction illustrated in Figure 6. Specifically, for each question, Luo et al. (2024) retrieve the shortest path in the Freebase KG that connects the topic entity to the correct answer. In particular, we enclose each reasoning path with the special tokens `<PATH>` and `</PATH>` to explicitly delimit the path generation span. This design enables the LLM to distinguish reasoning-path generation from subsequent answer prediction, while providing an explicit termination signal for later optimization. We then concatenate the tagged reasoning path with its corresponding answer to form the supervised fine-tuning examples.

Question:
Find the country with the ISO number of 736 that imports from Japan, what is the name of the country?

LLM Output:
<PATH>Japan -> location.country.languages_spoken -> English Language -> language.human_language.countries_spoken_in -> Sudan Answer:Sudan</PATH>
Answer:Sudan

Figure 6: An example of the LLM output after fine-tuning.

To further increase reasoning diversity in the training data, we retain all valid shortest paths when multiple shortest paths exist for a question. This provides diverse reasoning trajectories for the same question, helping the model avoid premature convergence to a single path and providing a initialization for subsequent reinforcement learning to explore alternative valid reasoning paths.

Policy Optimization Datasets. For Divergence-Aware Policy Optimization, we use the same WebQSP and CWQ training sets as in the supervised fine-tuning stage. The reasoning paths and answers are used to construct complementary reward signals. Specifically, we use the reference paths to assess the quality of the generated reasoning paths and the reference answers to evaluate the correctness of the predicted answers. Although graph-constrained reasoning guarantees that the generated path consists of valid KG transitions, graph validity alone does not ensure that the path is relevant to the question or leads to the correct answer. Therefore, we use the reference paths to provide an additional supervision signal that encourages the model to explore graph-valid paths that are also consistent with the desired reasoning trajectory.

Algorithm 2 Relation-Chain Refinement Data Construction

Require: Training set $\mathcal{D} = \{(q_i, \mathcal{A}_i)\}_{i=1}^{N}$; question-specific subgraphs $\{\mathcal{G}_{q_i}\}_{i=1}^{N}$
Ensure: Contrastive dataset $\mathcal{D}_{\text{rel}}$

$\mathcal{D}_{\text{rel}} \leftarrow \emptyset$
for each $(q_i, \mathcal{A}_i) \in \mathcal{D}$ **do**
  $\mathcal{P}_i \leftarrow \text{Paths}(\mathcal{G}_{q_i})$
  $\mathcal{P}_i^+ \leftarrow \{p \in \mathcal{P}_i \mid \text{End}(p) \in \mathcal{A}_i\}$
  $\mathcal{P}_i^- \leftarrow \mathcal{P}_i \setminus \mathcal{P}_i^+$
  $\mathcal{R}_i^+ \leftarrow \{\psi(p) \mid p \in \mathcal{P}_i^+\}$
  $\mathcal{R}_i^- \leftarrow \{\psi(p) \mid p \in \mathcal{P}_i^-\}$
  for each $r^+ \in \mathcal{R}_i^+$ **do**
    $r^- \sim \text{Sample}(\mathcal{R}_i^-)$
    $\mathcal{D}_{\text{rel}} \leftarrow \mathcal{D}_{\text{rel}} \cup \{(q_i, r^+, r^-)\}$
  end for
end for
return $\mathcal{D}_{\text{rel}}$

function $\psi(p)$
  Extract the ordered relation sequence $(\rho_1, \ldots, \rho_L)$ from p
  return $(\text{tail}(\rho_1), \ldots, \text{tail}(\rho_L))$
end function

Relation-Chain Refinement Datasets. To construct training data for the relation-chain refinement module, we derive positive and negative reasoning paths from the question-specific subgraph of each training question. Paths reaching gold answers are treated as positives, while the other paths serve as negatives. Each path is then normalized into an ordered relation chain by retaining only the terminal field of each relation identifier. The resulting question–positive–negative triplets are used for contrastive training of the relevance encoder. Algorithm 2 summarizes the construction procedure.

E BASELINES

We compare GraphPilot with 20 representative baselines grouped into three categories: (1) LLM Reasoning, (2) Graph Reasoning, and (3) KG-enhanced LLM Reasoning.

LLM Reasoning. We consider a set of representative open-source and proprietary LLMs that directly answer questions based on their parametric knowledge, without accessing external knowledge graphs or retrieved evidence. The selected models cover different model scales and reasoning paradigms, ranging from general-purpose instruction-following LLMs to models with explicit reasoning capabilities.

- **Llama-3.1-8B** (Grattafiori et al., 2024) is an open-source 8B-parameter model from the Llama 3.1 family, featuring improved reasoning, instruction following, and long-context capabilities.
- **DeepSeek-V3** (DeepSeek-AI et al., 2025) is a large-scale Mixture-of-Experts (MoE) model with 671B total parameters and 37B activated parameters per token, providing strong general-purpose language understanding and reasoning capabilities.
- **GPT-4o-mini** (OpenAI, 2024) is a compact proprietary model developed by OpenAI, designed to provide capable text and vision understanding with low inference cost and latency.
- **GPT-5** (OpenAI, 2025) is a proprietary model from OpenAI with configurable reasoning effort, designed for complex reasoning and general-purpose tasks.
- **Gemini-2.5 Flash** (Google, 2025) is a hybrid reasoning model developed by Google, designed to balance reasoning quality, inference speed, and computational cost.

Graph Reasoning. We consider representative graph-based reasoning methods that exploit structured knowledge from KGs to perform multi-hop question answering. These methods differ in how they retrieve, represent, and reason over graph knowledge, ranging from graph neural reasoning to unified retrieval–reasoning frameworks and retrieval-enhanced generative approaches.

- **ReaRev** (Mavromatis & Karypis, 2022) performs adaptive KG reasoning by iteratively updating question-derived instructions with KG-aware information and executes these instructions using a GNN-based mechanism that emulates breadth-first search.
- **UniKGQA** (Jiang et al., 2023) unifies KG retrieval and reasoning in both model architecture and parameter learning, combining PLM-based question–relation semantic matching with graph-based propagation of matching information.
- **DECAF** (Yu et al., 2023) adopts a retrieval-enhanced sequence-to-sequence framework that jointly generates direct answers and executable logical forms, and combines their predictions to improve robustness and answer accuracy.

KG-Enhanced LLM Reasoning. We further compare GraphPilot with representative KG-enhanced LLM reasoning methods. These approaches integrate structured knowledge from KGs into LLM reasoning through different mechanisms, including external evidence retrieval, iterative graph interaction, and training-based alignment.

- **EWEK-QA** (Dehghan et al., 2024) enriches the evidence provided to LLMs through adaptive web retrieval and efficient integration of KG triples, avoiding additional LLM calls during KG retrieval.
- **ToG** (Sun et al., 2024) treats the LLM as an agent that iteratively explores entities and relations in the KG through beam search to identify promising reasoning paths.
- **PoG** (Chen et al., 2024) introduces a self-correcting planning framework that decomposes questions into sub-objectives and repeatedly performs graph exploration, memory updating, and reflection to refine reasoning paths.
- **RoG** (Luo et al., 2024) adopts a planning–retrieval–reasoning framework that first generates relation paths as reasoning plans, retrieves KG-grounded reasoning paths, and then uses them to support faithful answer generation.

- **GNN-RAG** (Mavromatis & Karypis, 2025) employs a lightweight GNN to retrieve question-relevant answer candidates and the shortest KG paths connecting them to the question entities, which are then provided as context to the LLM.
- **ORT** (Liu et al., 2025) performs ontology-guided reverse reasoning by extracting condition and target labels, constructing label-level reasoning paths from targets back to known conditions, and using these paths to guide KG retrieval.
- **GCR** (Luo et al., 2025) integrates KG structure directly into LLM decoding through a trie-based KG index, constraining generation to KG-valid reasoning paths for faithful graph-grounded reasoning.
- **EPERM** (Long et al., 2025) retrieves a question-relevant subgraph, identifies and scores evidence paths according to their importance, and uses the weighted paths to guide final answer reasoning.
- **PathMind** (Liu et al., 2026) follows a Retrieve–Prioritize–Reason paradigm, prioritizing important reasoning paths with a semantic-aware path priority function and aligning the LLM through instruction tuning and path-wise preference learning.
- **OntGQA** (Shan et al., 2026) performs type-constrained reasoning over a relation-centric ontology graph using a planner–judge architecture, where candidate reasoning paths are guided and verified through entity-type constraints.
- **COSMOS** (Peng et al., 2026) formulates evidence subgraph retrieval as constrained submodular maximization, jointly considering semantic relevance and structural connectivity to construct coherent evidence subgraphs for downstream reasoning.
- **EoG** (Yan et al., 2026) introduces reinforcement learning to encourage autonomous exploration of diverse reasoning paths over KGs, using answer correctness together with path-based reward signals to guide exploration.

F Implementation Details

For constructing the graph-constrained space, we set the maximum reasoning depth to 20 hops, allowing the graph search to cover sufficiently long reasoning paths while bounding the overall exploration space. The setting defines the range of candidate paths available to graph-constrained reasoning, while LLMs can terminate earlier at valid entity boundaries according to the reasoning requirements of each question.

GraphPilot is trained in three stages: supervised fine-tuning, divergence-aware policy optimization, and relation-chain refinement. SFT aligns model outputs with the predefined format, while subsequent reinforcement learning optimizes path selection and termination. The relation-chain encoder is trained separately using contrastive supervision. Table 6 summarizes the configurations for all three stages.

G Templates And Prompts

SFT Prompt Template. We use the training data to align the LLM with the desired output format of reasoning paths and answers. The prompt used for SFT is shown in Figure 7, where the `question` and `entities` are replaced with the corresponding instance-specific inputs.

RL Prompt Template. We employ reinforcement learning to encourage adaptive exploration within the constrained space. The prompt used for reinforcement learning is shown in Figure 8. At this stage, only the question and its topic entities are explicitly provided to the LLM. The KG is instead transformed into a constrained decoding space that governs the generation process, thereby reducing the number of input tokens required by the LLM.

Inference Prompt Template. During inference, we employ beam search to generate diverse reasoning paths and improve answer coverage. The inference prompt is shown in Figure 9. As in the reinforcement learning stage, the knowledge graph is not explicitly provided as input to the LLM; instead, it is used solely to constrain decoding during the reasoning process.

SFT Prompt Template

Given a question and its topic entities, generate one or more KG-grounded reasoning paths that can support answering the question.
#Question:
{question}
#Topic Entities:
{entities}

Figure 7: The prompt template for SFT.

RL Prompt Template

Given a question and its topic entities, generate multiple valid path--answer pairs. Each reasoning path should correspond to a single answer. When multiple answers are possible, prioritize generating paths that lead to different answers; after covering distinct answers, explore alternative valid paths that support the same answer. Each path should terminate at an appropriate entity boundary.
#Question:
{question}
#Topic Entities:
{entities}

Figure 8: The prompt template for RL.

Inference Prompt Template

Given a question and its topic entities, generate a valid reasoning path that supports answering the question. Prefer reasoning trajectories that explore distinct relations and lead to different plausible answers when possible, while avoiding redundant paths that provide equivalent evidence. Each path should terminate at an appropriate entity boundary, followed by the corresponding answer.
#Question:
{question}
#Topic Entities:
{entities}

Figure 9: The prompt template for inference.

Table 6: Training configurations for SFT, RL, and relation-chain refinement.

Configuration	Value
Supervised Fine-Tuning (SFT)	
Training Epochs	3
Batch Size	64
Learning Rate	2×10^{-5}
Warmup Ratio	3%
Max Gradient Norm	1.0
Divergence-Aware Policy Optimization (RL)	
Training Epochs	2
Group Size	10
Learning Rate	3×10^{-6}
Clip Range	0.3
Sampling Temperature	1.0
Top-p	1.0
Maximum Reasoning Hops	20
KL Coefficient	0.1
Max Gradient Norm	1.0
Answer Reward Weight λ_{ans}	0.3
Path Reward Weight λ_{path}	0.6
Format Reward Weight λ_{fmt}	0.1
Relation-Chain Refinement	
Training Epochs	6
Batch Size	128
Learning Rate	3×10^{-4}
Warmup Ratio	10%
Ranking Margin m	0.5
Boundary Loss Weight λ_b	0.5
Compactness Loss Weight λ_c	0.05
Positive Interval	$[0.5, 1.0]$
Negative Interval	$[-1.0, -0.5]$
Relevance Threshold γ	0.5

H Additional Experimental Results

H.1 Construction Efficiency.

We further report the construction cost of the graph-constrained decoding space on the test sets. As shown in Table 7, the constrained space can be constructed within a few seconds per question on average, despite supporting deep graph expansion. The bounded frontier and question-aware pruning effectively control the growth of the search space, making the construction cost manageable even on the more complex CWQ dataset. Moreover, this construction is performed before LLM generation, and the resulting constrained space can be reused throughout the subsequent decoding process for the corresponding question.

Table 7: Construction efficiency of the graph-constrained decoding space on the test sets.

Dataset	Total Construction Time	Avg. Time / Question
WebQSP	1 h 56 min 20 s	4.29 s
CWQ	6 h 20 min 48 s	6.47 s

H.2 Comparison with Standard GRPO.

To isolate the effect of divergence-aware credit assignment, we replace our proposed policy optimization objective with standard GRPO while keeping the reward function, sampling strategy, and other training configurations unchanged. As shown in Table 8, standard GRPO consistently im-

proves over the variant without reinforcement learning, confirming the benefit of policy optimization within the graph-constrained reasoning space.

Table 8: Comparison of Divergence-Aware Policy Optimization and Standard GRPO.

Variant	WebQSP				CWQ			
	Hit	P	R	F1	Hit	P	R	F1
GraphPilot	93.7	92.3	74.8	82.6	89.8	80.7	81.5	81.1
w/o RL	87.2	74.5	69.8	72.1	63.9	51.9	61.7	56.4
with Standard GRPO	91.6	80.5	72.8	76.5	80.3	75.3	71.2	73.2

In addition, our Divergence-Aware Policy Optimization further improves performance on both datasets. The gain is particularly pronounced on CWQ, which contains more complex multi-hop reasoning cases, and is consistent with the motivation of our method. When a low-reward trajectory shares a reference-consistent prefix, standard GRPO applies negative feedback to the entire trajectory, potentially penalizing reasoning decisions that are already correct. In contrast, our divergence-aware strategy preserves the shared prefix and applies negative feedback only after the first divergence, resulting in more targeted token-level credit assignment and reducing unnecessary negative updates to correct reasoning steps.

Inference Prompt Template

You are a helpful assistant skilled in assessing the relevance of reasoning paths.
Given a question and a set of candidate reasoning paths, answer the question using the relevant paths.
Each path starts from a topic entity in the question and consists of an ordered sequence of entities and relations. All candidate paths are valid in the knowledge graph, but graph validity does not necessarily imply relevance to the question.
Evaluate each candidate independently:
- Consider the semantics of the entire path and the order of its relations.
- Select paths that logically connect the topic entity to the information requested by the question.
- Ignore irrelevant information and paths containing unnecessary detours that do not help answer the question.
- Do not select a path solely because individual relation names overlap with words in the question.
- Do not assume that longer or shorter paths are inherently more relevant.
- You may select multiple paths. Do not impose a fixed number of selected paths.
Base your answer only on the supplied paths. Preserve the entities and relations in each selected path; do not invent or modify paths.
For each selected path, output its corresponding answer in the following format, using one line per path:
<PATH>Head Entity -> relation_1 -> Intermediate Entity -> relation_2 -> Answer Entity</PATH> Answer: Answer Entity
Formatting example:
<PATH>Japan -> location.country.languages_spoken -> English Language -> language.human_language.countries_spoken_in -> Sudan</PATH> Answer: Sudan
The example illustrates the output format only; whether a path is relevant depends on the question.
Do not include explanations or any additional text. If none of the supplied paths supports an answer, output:
Answer: Insufficient evidence.
Now answer the question using the candidate reasoning paths below.
#Question:
{question}
#Topic Entities:
{entities}

Figure 10: The prompt template for LLM-based refinement.

H.3 Comparison with LLM-Based Refinement.

To evaluate our refinement design, we compare Relation-Chain Refinement with a prompt-based LLM alternative. Both methods operate on the same candidate paths generated by GraphPilot. For LLM-based refinement, we provide the question and candidate paths and instruct the LLM to identify relevant paths and return their associated answers without modifying the paths or introducing new answers. The prompt template is shown in Figure 10. We keep candidate generation unchanged and compare the resulting answer quality using Hit, precision, recall, and F1.

Figure 11 presents the results of the two refinement methods. Relation-Chain Refinement outperforms LLM-based refinement on both datasets, supporting the effectiveness of task-specific contrastive supervision for learning question–relation-chain relevance. This gap may stem from the difficulty of distinguishing candidate paths: all satisfy graph constraints and may share similar entities, relations, or local semantics, yet not all support an answer to the question. With prompting alone, an LLM may mistake superficial semantic associations for the reasoning connections required to answer the question. In contrast, Relation-Chain Refinement learns these fine-grained distinctions through contrastive training on positive and negative paths for the same question, enabling more effective retention of relevant paths and removal of candidate noise.

Moreover, the generation policy already favors question-relevant paths after reinforcement learning. A fixed candidate budget may nevertheless yield additional paths that are semantically related but do not support valid answers. This can make valid and invalid candidates superficially similar, further complicating prompt-based relevance assessment.

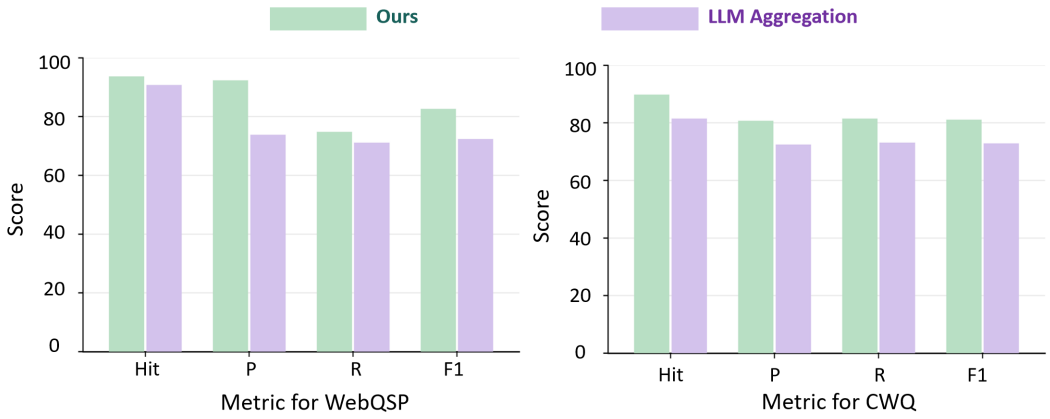


Figure 11: Comparison of Relation-Chain Refinement and LLM-Based Refinement.

H.4 Effect of Contrastive Learning on Relation-Chain Refinement.

To assess the contribution of contrastive learning to relation-chain refinement, we visualize the question–relation-chain cosine similarity distributions produced by all-MiniLM-L6-v2 before and after training on the same test candidates. We label relation chains from paths leading to correct answers as positive and those from the remaining paths as negative.

As shown in Figure 12, the positive and negative distributions overlap substantially before training, particularly on CWQ, where they are nearly indistinguishable. This suggests that the encoder's initial semantic representations do not adequately distinguish answer-supporting paths from semantically related but incorrect candidates. Contrastive learning produces a clearer separation between the two distributions, demonstrating improved discrimination on held-out data. These results support the effectiveness of task-specific contrastive supervision in learning relevance distinctions that general semantic similarity fails to capture, enabling more selective relation-chain refinement.

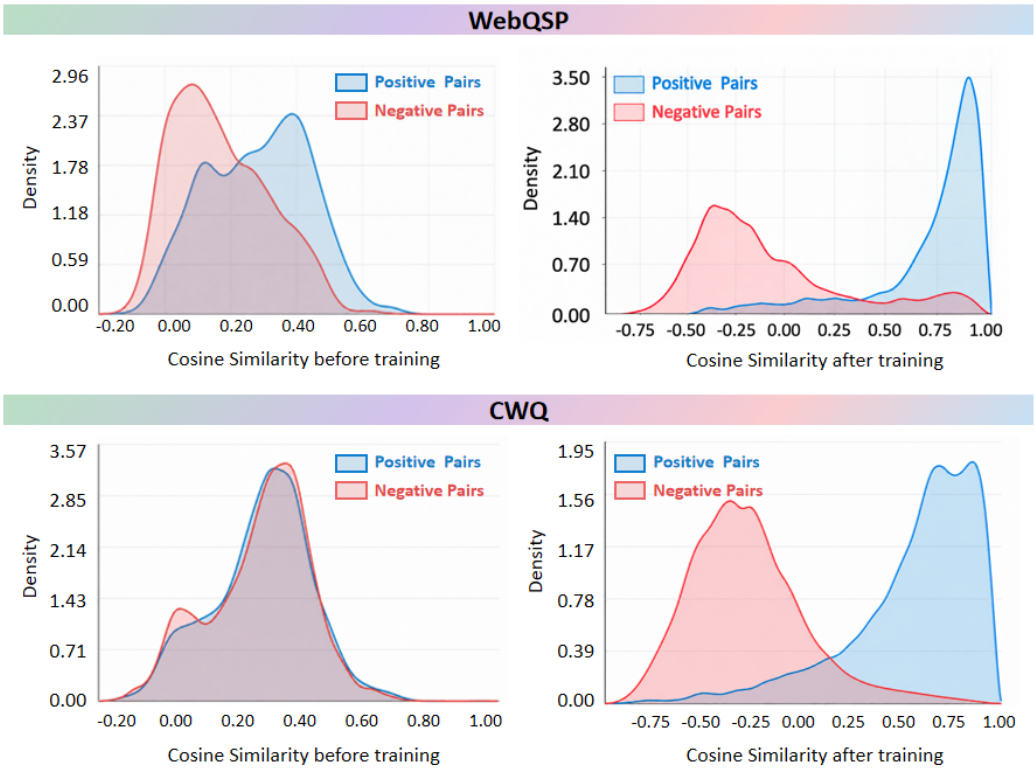


Figure 12: Comparison of Similarity Distributions.

H.5 Case Study on the Adaptivity of GraphPilot

As shown in Table 9, we further provide several case studies to analyze how our method adapts to different questions by retaining different reasoning depths and varying the number of output paths and answers.

From these examples, we observe that GraphPilot can adaptively adjust its reasoning depth within the graph-constrained space according to the requirements of different questions. For questions that can be resolved with short evidence chains, the model terminates at an early entity boundary, whereas for questions requiring multi-hop reasoning, it continues deeper exploration along valid graph transitions. Notably, in Case 1, GraphPilot identifies not only a direct short path to the correct answer, but also a longer valid path supporting the same answer. This suggests that its adaptivity is reflected not only in reasoning depth, but also in its ability to explore diverse valid reasoning trajectories that provide alternative evidence for the same answer. Although we set a fixed output budget , the refinement module effectively retains only candidates that can support answering the question, thereby reducing interference from irrelevant outputs.

Table 9: Examples of adaptive reasoning conducted by GraphPilot.

Case 1.	
Question	Which member of Congress founded New York University?
Answer	**Albert Gallatin**
GraphPilot	**# Output 1:** <PATH>New York University → organization.organization.founders → Albert Gallatin </PATH>Answer: **Albert Gallatin** **# Output 2:** <PATH>Member of Congress → government.government_office_or_title.office_holders → m.04__btv → government.government_position_held. district_represented → California's 50th congressional district → location.location.containedby → California → fictional_universe.fictional_setting.characters_that_have_lived_here → West Rosen → fictional_universe.fictional_character. education → New York University → organization.organization.founders → Albert Gallatin </PATH>Answer: **Albert Gallatin**
Case 2.	
Question	What role did Armie Hammer play in a film crewed by Chad Eshbaugh?
Answer	**Tyler Winklevoss**
GraphPilot	<PATH> Chad Eshbaugh → film.film_crewmember.films_crewed → m.09r7tnh → film.film_crew_gig.film → The Social Network → award.award_nominated_work. award_nominations → m.0z87znf → award.award_nomination. award_nominee → Armie Hammer → film.actor.film → m.0dl5j8l → film.performance.character → Tyler Winklevoss </PATH>Answer: **Tyler Winklevoss**
Case 3.	
Question	What movie with a character named Jane does Taylor Lautner play in?
Answer	**The Twilight Saga: Breaking Dawn - Part 1**; **The Twilight Saga: New Moon**; **Eclipse**
GraphPilot	**# Output 1:** <PATH> Taylor Lautner → film.actor.film → m.0b68zn5 → film.performance.film → The Twilight Saga: Breaking Dawn - Part 1 </PATH>Answer: **The Twilight Saga: Breaking Dawn - Part 1** **# Output 2:** <PATH> Taylor Lautner → film.actor.film → m.05tflhw → film.performance.film → The Twilight Saga: New Moon </PATH>Answer: **The Twilight Saga: New Moon** **# Output 3:** <PATH> Taylor Lautner → film.actor.film → m.04z4zmg → film.performance.character → Jacob Black → film.film_character. portrayed_in_films → m.075wxc2 → film.performance.film → Eclipse</PATH>Answer: **Eclipse**

I LIMITATIONS

Our work has two main limitations. First, we define graph faithfulness as the verifiability of generated reasoning paths within the KG. This ensures consistency with the graph but does not guarantee factual correctness: erroneous facts may lead to incorrect answers, while missing facts or paths may prevent the model from reaching valid answers. Graph faithfulness should therefore be distinguished from factual correctness.Second, GraphPilot mitigates noise introduced by fixed candidate budgets through post-generation relation-chain refinement. Although this step filters irrelevant candidates, it does not avoid the computational cost of generating them. A promising direction is to incorporate relevance assessment into generation, allowing irrelevant paths to be terminated early. Such an approach could reduce candidate noise and inference costs while preserving answer coverage.