StruProKGR: A Structural and Probabilistic Framework for Sparse Knowledge Graph Reasoning

Yucan Guo, Saiping Guan[†], Miao Su, Jiyao Wei,
Xiaolong Jin[†], Jiafeng Guo, Xueqi Cheng
[1]State Key Laboratory of AI Safety
[2]Institute of Computing Technology, Chinese Academy of Sciences
[3]School of Computer Science and Technology, University of Chinese Academy of Sciences
{guoyucan23z, guansaiping, jinxiaolong}@ict.ac.cn

Abstract

Sparse Knowledge Graphs (KGs) are commonly encountered in real-world applications, where knowledge is often incomplete or limited. Sparse KG reasoning, the task of inferring missing knowledge over sparse KGs, is inherently challenging due to the scarcity of knowledge and the difficulty of capturing relational patterns in sparse scenarios. Among all sparse KG reasoning methods, path-based ones have attracted plenty of attention due to their interpretability. Existing path-based methods typically rely on computationally intensive random walks to collect paths, producing paths of variable quality. Additionally, these methods fail to leverage the structured nature of graphs by treating paths independently. To address these shortcomings, we propose a Structural and Probabilistic framework named StruProKGR, tailored for efficient and interpretable reasoning on sparse KGs. StruProKGR utilizes a distance-guided path collection mechanism to significantly reduce computational costs while exploring more relevant paths. It further enhances the reasoning process by incorporating structural information through probabilistic path aggregation, which prioritizes paths that reinforce each other. Extensive experiments on five sparse KG reasoning benchmarks reveal that StruProKGR surpasses existing path-based methods in both effectiveness and efficiency, providing an effective, efficient, and interpretable solution for sparse KG reasoning.[1]

1 Introduction

Knowledge Graphs (KGs) contain facts in the form of triples $(head\ entity, relation, tail\ entity)$, denoted as (h, r, t). They support a variety of downstream applications, including question answering (Agarwal et al., 2024; Liu et al., 2025), recommender systems (Wang et al., 2024, 2025), and information retrieval (Gutiérrez et al., 2024; Cai et al., 2025). In real-world situations, KGs often exhibit sparsity, as many triples are missing due to incomplete knowledge collection. For example, considering Freebase (Bollacker et al., 2008), the well-known open-source KG, 71% of individuals in Freebase have no recorded place of birth, and 75% have no identified nationality (Dong et al., 2014). Sparse KG reasoning, the task of inferring missing knowledge over sparse KGs, is crucial for uncovering valuable insights in knowledge-scarce scenarios.

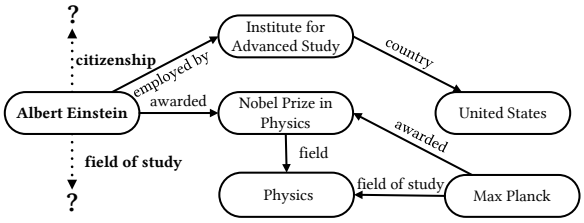


Figure 1: An example illustrating sparse KG reasoning.

Figure 1 shows an example of sparse KG reasoning over a highly incomplete KG with six entities and five relations. Solid arrows denote facts that are observed, while the dotted arrows highlight the relations that should be completed, i.e., $(Albert\ Einstein,\ citizenship,\ ?)$ and $(Albert\ Einstein,\ field\ of\ study,\ ?)$. These two relations do not appear explicitly in the observed graph but can be inferred from other information within it. The $citizenship$ of $Albert\ Einstein$ is likely to be deduced from the institution he employed by, while the $field\ of\ study$ can often be inferred by propagating the $field$ from the prize to its laureate. However, the scarcity of knowledge and the complexity of understanding intricate relational patterns render this task exceptionally challenging. Existing KG reasoning methods often fall short in sparsity scenarios (Pujara et al., 2017), highlighting

[†]Corresponding authors.
[1]The code is publicly available at https://github.com/YucanGuo/StruProKGR.

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 8402–8416
October 24-29, 2026 ©2026 Association for Computational Linguistics

the need for approaches specifically designed for sparse KGs.

Sparse KG reasoning methods fall into three major categories: embedding-based, rule-based, and path-based methods. Embedding-based methods (Zhang et al., 2022; Tan et al., 2023; Chen et al., 2024) encode entities and relations into continuous spaces and yield strong predictive performance, but they are often opaque. Rule-based methods (Meilicke et al., 2020; Sun et al., 2023) mine symbolic rules from KGs and provide interpretability, yet they suffer from scalability issues and costly rule mining. Path-based methods (Lao et al., 2011; Lv et al., 2020; Guan et al., 2024), by contrast, trace explicit relational paths without requiring meticulously designed rules, offering transparency that is crucial for trustworthy knowledge inference. Existing path-based methods typically rely on either random walk-based (Lao et al., 2011; Gardner and Mitchell, 2015; Guan et al., 2024) or reinforcement learning (RL)-based strategies (Das et al., 2018; Lv et al., 2020) for path collection, followed by path reasoning over the collected paths. However, both path collection and path reasoning stages remain challenging. Random walk-based approaches are computationally expensive on large KGs and often generate low-relevance paths due to stochastic exploration. RL-based approaches guide path selection via learned policies but compromise interpretability. Moreover, most path-based methods reason over paths independently, ignoring the structural dependencies among paths in sparse KGs. This assumption prevents models from capturing collective relational patterns and limits reasoning accuracy.

To overcome these challenges, we present StruProKGR, a novel path-based framework meticulously designed for effective, efficient, and interpretable reasoning over sparse KGs. StruProKGR introduces a distance-guided path collection mechanism that markedly reduces computational overhead compared to random walk-based methods. This approach leverages distances to the tail entity to prioritize paths that are most likely to contribute to accurate reasoning outcomes, thereby optimizing the exploration process of sparse KGs. By prioritizing paths with high relevance to target relations, this approach ensures both effectiveness and efficiency, addressing the scalability concerns of prior random walk-based techniques. Additionally, StruProKGR utilizes the structural properties of sparse KGs through a probabilistic path aggregation strategy during path reasoning. This approach considers the correlations among paths as a whole, resulting in more accurate inferences of missing knowledge while preserving the interpretability of path-based methods.

In summary, the contributions of this paper are as follows:

- We present StruProKGR, a training-free path-based framework for effective, efficient, and interpretable reasoning over sparse KGs.
- We design a distance-guided path collection and a probabilistic path aggregation mechanism that reduce computational overhead while leveraging graph structure to enhance reasoning accuracy.
- Extensive experiments on five sparse KG benchmarks demonstrate the effectiveness and efficiency of StruProKGR.

2 Related Work

In this section, we review the related work of embedding-based, rule-based, and path-based sparse KG reasoning methods.

Embedding-based Methods. Embedding-based methods learn vector representations for entities and relations in a KG, using these to score the plausibility of potential triples. Early methods like TransE (Bordes et al., 2013) interpret relations as translations in vector space, while DistMult (Yang et al., 2015) employs the bilinear objective to learn relational semantics. Several studies enhance embeddings by graph walks or relation paths to capture richer semantic information. RDF2Vec (Ristoski and Paulheim, 2016) learns entity embeddings from sequences collected by graph walks. PTransE (Lin et al., 2015) incorporates multi-step relation paths into embedding learning to improve reasoning, while RPJE (Niu et al., 2020) integrates automatically mined Horn rules and paths into representation learning. More advanced methods, such as ConvE (Dettmers et al., 2018) and TuckER (Balažević et al., 2019), leverage convolutional neural networks and tensor factorization to model complex interactions. Recent studies (Tan et al., 2023; Chen et al., 2024) integrate graph context into their models to tackle the sparsity issue. These methods often achieve strong prediction performance but suffer from limited interpretability and high computational costs due to representation learning.

Rule-based Methods. Rule-based methods mine logical rules from KGs to infer new knowledge, offering clear explanations for predictions. AMIE (Galárraga et al., 2013) and AnyBURL (Meilicke et al., 2020) extract horn clauses to capture relational patterns, with AnyBURL incorporating RL to enhance rule mining. NTP (Rocktäschel and Riedel, 2017) integrates differentiable proving with subsymbolic representations, enabling logical rule induction through gradient-based optimization. RLvLR (Omran et al., 2018) presents a method that combines representation learning with closed path rule mining, using embeddings and sampling to handle large KGs. However, these methods face challenges in sparse KGs, where limited facts reduce rule coverage and reliability. Additionally, the rule mining process can become computationally intensive as the complexity of the rules increases.

Path-based Methods. Path-based KG reasoning methods collect and traverse relational paths to infer missing knowledge, typically using either random walk-based or RL-based strategies. Random walk-based methods are pioneered by PRA (Lao et al., 2011), with later extensions such as ProbCBR (Das et al., 2020) introducing probabilistic case-based reasoning and LoGRe (Guan et al., 2024) constructing a global relation-path schema to mitigate sparsity. However, as the KG complexity increases, the computational cost of random walks escalates exponentially, rendering these approaches impractical for large-scale KGs. RL-based methods instead learn to navigate paths. DacKGR (Lv et al., 2020) expands the search space with dynamically added edges, SparKGR (Xia et al., 2022) integrates rule-guided iterative refinement, and recent systems such as DT4KGR (Xia et al., 2024) and Hi-KnowE (Xie et al., 2024) incorporate decision Transformers or hierarchical RL. Despite these advances, RL-based methods commonly require handcrafted rewards or external resources, which add complexity and undermine the interpretability central to path-based reasoning.

3 Problem Statement

We first introduce key concepts of sparse KGs, then formally define the sparse KG reasoning task.

3.1 Preliminaries

Definition 3.1 (KG) *Given an entity set $\mathcal{E}$ and a relation set $\mathcal{R}$, a KG is a directed graph $\mathcal{G} = \{(h, r, t) | h, t \in \mathcal{E}, r \in \mathcal{R}\}$, where each entity $e \in \mathcal{E}$ belongs to an entity type $c \in \mathcal{C}$, each triple (h, r, t) indicates that there is a relation r from the head entity h to the tail entity t.*

A sparse KG (Lv et al., 2020) refers to a KG that is highly incomplete, meaning that many potential links and facts between entities are missing. This form of sparsity should be distinguished from the statistical rarity of specific relations. In practice, sparsity manifests as low triple density and weak connectivity between entities, which significantly increases the difficulty of reasoning. To address sparsity, path-based methods often leverage relational sequences, i.e., relation paths, which describe multi-hop connections between entities. Relation paths can be defined at different granularities (Guan et al., 2024): (1) **Type-specific relation paths** capture connections between entities of a given type and a target relation; (2) **Relation paths** generalize these by aggregating type-specific relation paths across multiple entity types.

3.2 Problem Definition

Definition 3.2 (Sparse KG reasoning) *Given a sparse KG $\mathcal{G}_s$ and a query $(h, r, ?)$, where $h \in \mathcal{E}$ is a head entity and $r \in \mathcal{R}$ is a relation, the task is to predict the missing tail entity $t \in \mathcal{E}$ such that (h, r, t) is likely to hold in $\mathcal{G}_s$.*

Queries of the form $(?, r, t)$ can be equivalently handled by introducing the inverse relation r^{-1} and reformulating the query as $(t, r^{-1}, ?)$. Hence, it suffices to study the $(h, r, ?)$ case.

4 Methodology

In this section, we introduce the details of the proposed StruProKGR framework, and Figure 2 illustrates its architecture. Specifically, StruProKGR is designed with three main phases: distance-guided path collection (§4.1), the calculation of path probability and joint probability (§4.2), and path structure-based reasoning (§4.3).

4.1 Distance-Guided Path Collection

In path-based sparse KG reasoning methods, efficiently collecting relevant paths between entities is critical for accurate reasoning. We introduce a distance-guided path collection mechanism that enhances computational efficiency and path relevance. Short paths, which are more likely to be rules in path-based methods, are prioritized by leveraging distance information to guide and prune the path

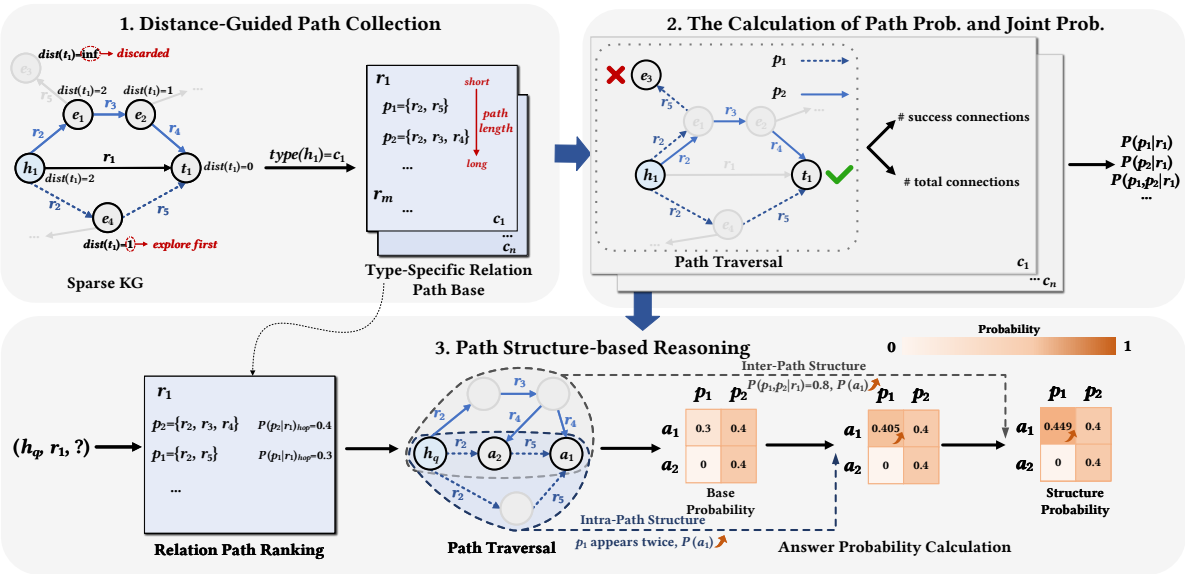


Figure 2: The proposed StruProKGR framework.

collection process. The distance-guided path collection phase adopts a Depth-First Search (DFS) procedure to collect type-specific relation paths, and leverages precomputed shortest-path information to prune the search space aggressively. Specifically, this phase consists of two steps, i.e., (1) distances precomputation, and (2) distance-guided path collection.

Distances Precomputation. We perform a truncated Breadth-First Search (BFS) from each entity $u \in \mathcal{E}$, up to depth l_{max}, to fill the matrix $dist[u][v]$, where l_{max} is the maximum path length. This truncated BFS records the minimum number of hops from u to every reachable v with $dist[u][v] \leq l_{max}$. By doing so, once at initialization, we can quickly check whether a partial path still has a chance to reach the target, thereby avoiding unnecessary visits to paths that are guaranteed to fail. Distances are stored in a sparse matrix structure containing only nodes within the bounded radius, so both time and space complexity scale with the local neighborhood size rather than the total number of entities.

Distance-Guided Path Collection. For each training triple (h, r, t), we perform a DFS starting from the head entity h to reach the tail entity t. The search incrementally builds a candidate type-specific relation path $path = [r_1, r_2, \ldots, r_m]$ for entity type $c \in C$ of h, which records the sequence of relations traversed so far. At each expansion step, a current entity u can only move to a neighbor v if

$$dist[v][t] \leq l_{max} - len(path) - 1, \qquad (1)$$

where $len(path)$ denotes the length of the current path. This guarantees that v can still reach t within the remaining length budget, eliminating futile branches. To further narrow down the search, at each step, only the top-k neighbors ranked by $dist[v][t]$ are retained, focusing the search on the most promising candidates and avoiding the redundancy of exploring many low-quality paths. The two-stage pruning above ensures that only those branches that can feasibly reach the target within the allotted steps are considered, and concentrates efforts on the nearest neighbors.

Formally, the procedure outputs the set of all collected paths $\mathcal{P}$, where each type-specific subset $\mathcal{P}(c, r)$ contains paths for entity type c and relation r. A complete algorithmic description is provided in Appx. §A.1.

4.2 The Calculation of Path Probability and Joint Probability

To prepare for the path structure-based reasoning phase, we need to calculate the *path probability* and *joint path probability* of paths in a sparse KG. The path probability aggregates type-specific relation paths across entity types to form a unified measure for relation paths, while the joint probability evaluates path pairs.

Path Probability. The probability of a relation path p for a relation r, denoted $P(p|r)$, quantifies

the precision of p in connecting a head entity h to a tail entity t. Since relation paths are aggregated from type-specific relation paths defined for specific entity types, we traverse the collected type-specific relation paths and aggregate them to form $P(p|r)$. It is defined as:

$$P(p|r) = \frac{\sum_{c \in \mathcal{C}} S_c(r,p)}{\sum_{c \in \mathcal{C}} T_c(r,p)}, \tag{2}$$

where $S_c(r,p)$ is the number of occurrences for path $p \in \mathcal{P}(c,r)$ that successfully reach the correct tail entities, and $T_c(r,p)$ is the total number of entities reached. This metric prioritizes paths that consistently yield accurate inferences, forming the foundation for reliable reasoning.

Joint Path Probability. The joint probability of two paths p_i and p_j for a relation r, denoted $P(p_i, p_j|r)$, measures the likelihood that both paths collaboratively infer the relation r correctly and reflects the combined reliability of path pairs. It is defined as:

$$P(p_i, p_j|r) = \frac{JS(r, p_i, p_j)}{JT(r, p_i, p_j)}, \tag{3}$$

where $JS(r, p_i, p_j)$ is the number of joint correct occurrences for path pair (p_i, p_j), and $JT(r, p_i, p_j)$ is the total number of joint occurrences.

To calculate $P(p|r)$ and $P(p_i, p_j|r)$ efficiently, we propose a batch search-based path traversal algorithm, which is detailed in Appx. §A.2.

4.3 Path Structure-based Reasoning

Sparse KGs inherently contain complex structural properties that need to be taken into account. For example, a relation may be more likely to hold true when certain paths occur together, while it may be less likely to be true when specific combinations of paths coexist. Thus, in the final phase, StruProKGR models structural properties of paths in a probabilistic manner to conduct reasoning effectively. We categorize structural properties in sparse KGs into two main groups: *intra-path structures*, which relate to the internal characteristics of individual path types, and *inter-path structures*, which focus on the relationships between different types of paths.

Relation Path Ranking. Before reasoning, collected paths are ranked by incorporating a hop decay factor $\alpha^{\text{len}(p)-1}$ into their probability (Guan et al., 2024), where $\alpha \in (0,1)$ and $\text{len}(p)$ is the path length. The adjusted probability is calculated by

$$P(p|r)_{hop} = P(p|r) \cdot \alpha^{\text{len}(p)-1}, \tag{4}$$

which provides a measurable assessment of path relevance by balancing informativeness and conciseness, and serves as the base probability for the subsequent path structure-based probability update.

Intra-Path Structure Modeling. Intra-path structure focuses on the repetitive occurrence of a single path type that reaches the same entity during reasoning, and we model the contribution of repetition in a diminishing way. Specifically, each additional occurrence of the same path contributes less to the overall probability, which is defined as follows:

$$P(p|r)_k = \beta^{k-1} \cdot P(p|r)_{hop}, \tag{5}$$

where k means the k-th occurrence of path p, and the diminishing factor $\beta \in (0,1)$.

Consequently, the probability of a path, taking into account the intra-path structure, can be expressed as follows:

$$P(p|r)_{intra} = 1 - \prod_{i=1}^{T_p} \big(1 - P(p|r)_i\big), \tag{6}$$

where T_p represents the total occurrences of path p from the same head entity to the same tail entity.

Inter-Path Structure Modeling. Inter-path structure examines the relationships between different types of paths, which often exhibit complex interactions that influence the accuracy of inferring missing knowledge. To address this, we propose a probabilistic framework that models inter-path structures using path and joint probabilities.

Likelihood Ratio Calculation. Bayes' theorem provides a principled way of updating prior beliefs in light of new evidence, where the strength of evidence is captured by a likelihood ratio (Joyce, 2021). Following this, we introduce a scalable approximation of the likelihood ratio that aggregates evidence from multiple paths:

$$LR(p_i, \mathcal{P}(r) \setminus \{p_i\}) = \frac{\sum_{p_j} P(p_i, p_j|r)}{\sum_{p_j} [P(p_i|r) + P(p_j|r) - P(p_i|r) \cdot P(p_j|r)]}, \tag{7}$$

where $p_j \in \mathcal{P}(r) \setminus \{p_i\}$ and subject to the condition $P(p_j|r)_{hop} > P(p_i|r)_{hop}$. The ratio compares the observed joint correctness to the expected correctness under independence. A value greater than 1 suggests that the paths are more likely to be correct

together than independently, indicating collaboration, while a value less than 1 suggests inhibition. A mathematical proof for the approximation is provided in Appx. §A.3.1.

Updating Inter-Path Probabilities. After calculating the likelihood ratio, path probabilities considering the inter-path structure can be determined using the odds form of Bayes' theorem. The prior odds for path p_i are calculated as:

$$O(p_i) = \frac{P(p_i|r)_{intra}}{1 - P(p_i|r)_{intra}}, \tag{8}$$

reflecting the initial confidence in p_i before accounting for interactions. The posterior odds, adjusted based on evidence from other paths, are then:

$$O(p_i|\mathcal{P}(r) \setminus \{p_i\}) = O(p_i) \cdot LR(p_i, \mathcal{P}(r) \setminus \{p_i\}). \tag{9}$$

This step updates our confidence in p_i by incorporating the influence of other paths in $\mathcal{P}(r)$. To obtain the updated probability, we convert the posterior odds back to a probability:

$$P(p_i|r)_{inter} = \frac{O(p_i|\mathcal{P}(r) \setminus \{p_i\})}{1 + O(p_i|\mathcal{P}(r) \setminus \{p_i\})}. \tag{10}$$

Reasoning Workflow. Bringing together intra- and inter-path structures, StruProKGR executes reasoning in three steps. First, for a given query $(h, r, ?)$, it selects the top-N_{top} relation paths from $\mathcal{P}(r)$ and traverses them starting from the head entity h, thereby gathering a set of candidate answers. Second, for each candidate, path probabilities are adjusted by incorporating both intra-path repetition effects and inter-path interactions, yielding $P(p|r)_{inter}$ for all contributing paths. Third, the probability of each candidate a is aggregated from the set of paths $\mathcal{P}_a$ as

$$P(a) = 1 - \prod_{p \in \mathcal{P}_a} \left(1 - P(p|r)_{inter}\right). \tag{11}$$

By doing so, evidence from multiple supporting paths is combined, thereby jointly increasing the confidence in candidate a. The candidates are then ranked by their final scores, producing the reasoning output. The algorithmic details and pseudocode are provided in Appx. §A.3.2.

5 Experiments

In this section, we conduct extensive experiments to verify the effectiveness, efficiency, and interpretability of StruProKGR across five widely recognized benchmark datasets for sparse KGs. The empirical findings are aimed at addressing the following key research questions: **RQ1.** How does StruProKGR perform against existing state-of-the-art methods in sparse KG reasoning? **RQ2.** To what extent does the distance-guided path collection phase enhance both the effectiveness of the overall reasoning process and the efficiency of the path collection process? **RQ3.** What are the impacts of different components in the path structure-based reasoning phase, including intra-path structure and inter-path structure? **RQ4.** Despite its theoretical interpretability, does StruProKGR demonstrate interpretability in the practical reasoning process?

Table 1: Statistics of the datasets used in experiments.

Dataset	# Ent.	# Rel.	# Train	# Valid	# Test
FB15K-237-10%	11,512	237	27,211	15,624	18,150
FB15K-237-20%	13,166	237	54,423	16,963	19,776
FB15K-237-50%	14,149	237	136,057	17,449	20,324
NELL23K	22,925	200	25,445	4,961	4,952
WD-singer	10,282	131	15,906	2,084	2,134

5.1 Experimental Setup

5.1.1 Datasets

We utilize five benchmark datasets that are widely used for sparse KG reasoning tasks: FB15K-237-10%, FB15K-237-20%, FB15K-237-50%, NELL23K, and WD-singer (Lv et al., 2020). These datasets vary in sparsity and domain characteristics, making them well-suited to test the ability of StruProKGR to leverage structural path information under knowledge-limited conditions. **FB15K-237-10%**, **FB15K-237-20%**, and **FB15K-237-50%** are subsampled versions of FB15K-237 (Toutanova et al., 2015), retaining 10%, 20%, and 50% of the original triples, respectively. **NELL23K** is a randomly sampled dataset from the NELL (Carlson et al., 2010) knowledge base. **WD-singer** is a domain-specific Wikidata (Vrandečić and Krötzsch, 2014) subset focused on singer-related entities. Detailed statistics on the benchmark datasets can be found in Table 1.

5.1.2 Implementation Details

We conduct a grid search to determine the optimal hyperparameter for the maximum branch number, $k \in \{3, 5, 10, 15, 20, 30\}$. The applied settings are: $k = 15$ for FB15K-237-10%, $k = 5$ for FB15K-237-20%, $k = 3$ for FB15K-237-50%, $k = 30$ for NELL23K and WD-singer. The diminishing factor

Table 2: Experimental results presented in terms of MRR and Hits@{3, 10} (%). The best scores for rule-based methods (the second block) and path-based methods (the third block) are in bold, while the best scores for embedding-based methods (the first block) are underlined.[2]

Method	FB15K-237-10%			FB15K-237-20%			FB15K-237-50%			NELL23K			WD-singer		
	MRR	H@3	H@10	MRR	H@3	H@10	MRR	H@3	H@10	MRR	H@3	H@10	MRR	H@3	H@10
TransE	0.105	15.9	27.9	0.123	18.0	31.3	0.177	23.4	40.4	0.084	10.9	24.7	0.210	32.1	44.6
TuckER	0.252	27.2	40.4	0.268	28.9	42.8	0.314	34.2	50.1	0.276	30.2	46.7	0.421	47.1	57.1
ConvE	0.245	26.2	39.1	0.261	28.3	41.8	0.313	34.2	50.1	0.276	30.1	46.4	0.448	47.8	56.9
NBFNet	0.241	26.3	38.8	0.260	27.8	41.7	0.316	34.1	50.3	0.274	28.9	46.9	0.453	49.3	58.9
KRACL	0.164	17.0	21.2	0.170	16.9	19.8	0.222	26.8	44.4	0.158	15.8	27.6	0.142	13.4	20.7
HoGRN	0.257	27.5	41.2	-	-	-	-	-	-	0.292	32.2	49.1	0.470	51.0	57.8
NTP	0.083	11.4	16.9	0.173	16.1	21.7	0.222	23.1	30.7	0.132	14.9	24.1	0.292	31.1	44.2
RLvLR	0.107	12.2	20.6	0.132	15.2	27.1	0.199	20.8	32.4	0.152	17.3	25.0	0.374	32.0	47.6
AnyBURL	0.149	15.5	26.7	0.164	16.7	29.3	0.198	21.3	35.1	0.176	18.5	25.2	0.392	34.1	48.6
DacKGR	0.218	23.9	33.7	0.242	27.2	38.9	0.293	32.0	45.7	0.197	20.0	31.6	0.377	42.1	48.5
SparKGR	0.228	24.5	35.0	0.252	27.7	39.1	0.292	32.0	46.2	0.203	22.2	33.9	0.393	43.7	50.7
DT4KGR	-	-	-	0.254	-	40.1	0.297	-	46.2	-	-	-	-	-	
Hi-KnowE	0.224	**25.5**	34.1	0.247	27.7	38.1	-	-	-	-	-	-	-	-	-
LoGRe	0.228	24.5	36.2	0.261	28.0	41.3	0.297	32.7	46.4	0.259	27.9	41.7	0.459	48.9	54.5
StruProKGR	**0.234**	25.2	**37.3**	**0.267**	**28.8**	**42.1**	**0.304**	**33.3**	**47.6**	**0.262**	**28.5**	**42.7**	**0.461**	**49.8**	**55.6**

Table 3: Effectiveness analysis of path collection algorithms presented in terms of MRR and Hits@{3, 10} (%).

Method	FB15K-237-10%			FB15K-237-20%			FB15K-237-50%			NELL23K			WD-singer		
	MRR	H@3	H@10	MRR	H@3	H@10	MRR	H@3	H@10	MRR	H@3	H@10	MRR	H@3	H@10
$\text{StruProKGR}_{\text{RW}}$	0.226	24.3	36.2	0.261	28.2	41.5	0.298	32.9	46.7	0.260	28.1	**42.8**	0.459	49.4	55.0
StruProKGR	**0.234**	**25.2**	**37.3**	**0.267**	**28.8**	**42.1**	**0.304**	**33.3**	**47.6**	**0.262**	**28.5**	42.7	**0.461**	**49.8**	**55.6**

β is set to 0.5, and the inter-path structure is only taken into account for the top 200 paths to save time. For the other hyperparameters, we adhere to the settings in LoGRe (Guan et al., 2024).

5.1.3 Baselines and Evaluation Metrics

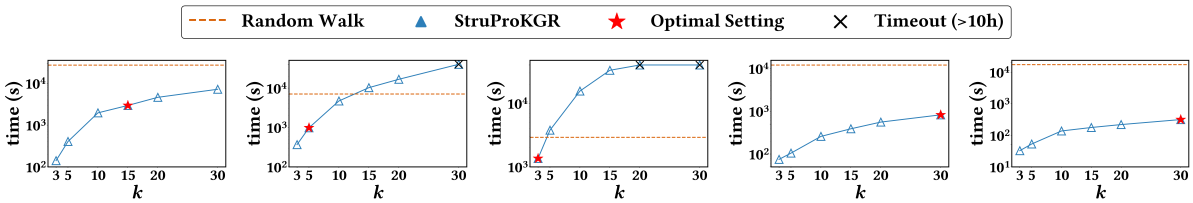
Baselines. We compare StruProKGR against a diverse set of state-of-the-art methods. For path-based methods, we compared with DacKGR (Lv et al., 2020), SparKGR (Xia et al., 2022), DT4KGR (Xia et al., 2024), Hi-KnowE (Xie et al., 2024), and LoGRe (Guan et al., 2024). For rule-based methods, we compared with NTP (Rocktäschel and Riedel, 2017), RLvLR (Omran et al., 2018), and AnyBURL (Meilicke et al., 2020). For embedding-based methods, we compared with TransE (Bordes et al., 2013), TuckER (Balažević et al., 2019), ConvE (Dettmers et al., 2018), NBFNet (Zhu et al., 2021), KRACL (Tan et al., 2023), and HoGRN (Chen et al., 2024).

Evaluation Metrics. We adopt standard metrics for sparse KG reasoning, i.e., Mean Reciprocal Rank (MRR) and Hits@K (K=3,10). Higher scores indicate a better ranking of correct answers.

5.2 Overall Performance (RQ1)

To address **RQ1**, we evaluate the performance of StruProKGR against state-of-the-art methods across five benchmark datasets. As shown in Table 2, StruProKGR consistently outperforms all rule-based and path-based baselines. Notably, it achieves clear gains in MRR and Hits@{3,10}, with relative improvements on Hits@10 of 1.9% to 3.0% across all datasets. These improvements highlight the benefits of combining distance-guided path collection with probabilistic modeling of both intra-path and inter-path structures. While strong path-based methods such as LoGRe and SparKGR also exploit relational paths, they lack explicit modeling of structural dependencies, limiting their effectiveness in sparse settings. Rule-based approaches remain less competitive than the other two categories of methods, as their reliance on strict logical rules restricts generalization under sparsity. Compared with embedding-based methods, StruProKGR achieves competitive performance despite not relying on dense representations or model training. Its MRR scores fall within 0.1% to 3%

[2]Empty entries indicate that the method did not report results on the dataset. The code of DT4KGR and Hi-KnowE is also not publicly available.



(a) FB15K-237-10% (b) FB15K-237-20% (c) FB15K-237-50% (d) NELL23K (e) WD-singer

Figure 3: Path collection running time for $\text{StruProKGR}_{\text{RW}}$ and StruProKGR on five datasets with varying k.

Table 4: Path collection running time (s) comparison of random walk and StruProKGR with optimal setting.

Method	FB15K-237-10%	FB15K-237-20%	FB15K-237-50%	NELL23K	WD-singer
Random Walk	27131.76	7059.71	2917.02	12054.34	17274.81
StruProKGR	2939.74	981.22	1357.07	824.14	314.47
Speedup	↑ 9.22×	↑ 7.19×	↑ 2.14×	↑ 14.62×	↑ 54.93×

of the strongest embedding-based baselines, underscoring its effectiveness as a training-free and interpretable alternative.

5.3 Path Collection Analysis (RQ2)

To address **RQ2** that investigates the extent to which the distance-guided path collection enhances both the effectiveness of the overall reasoning process and the efficiency of the path collection process, a comparative analysis is conducted.

5.3.1 Effectiveness Analysis

The effectiveness of the distance-guided path collection phase is assessed by comparing StruProKGR with a random walk-based variant, namely $\text{StruProKGR}_{\text{RW}}$. As reported in Table 3, StruProKGR consistently outperforms $\text{StruProKGR}_{\text{RW}}$ across all datasets, with relative MRR gains ranging from 0.4% on WD-singer to 3.5% on FB15K-237-10%. Similar trends hold for Hits@3 and Hits@10. These improvements demonstrate that distance guidance effectively mitigates the randomness of random walks, ensuring that collected paths are more relevant for reasoning.

5.3.2 Efficiency Analysis

The efficiency of the distance-guided path collection is evaluated by comparing the path collection running time of StruProKGR against random walk across varying values of the maximum branch number k, with results presented in Figure 3. As expected, the running time increases with larger k, and the optimal k also grows with sparsity. However, it remains efficient, with timeouts (exceeding 10 hours) only occurring at $k = 30$ on FB15K-237-20% and at $k \geq 20$ on FB15K-237-50%. With optimal k settings, Table 4 shows that StruProKGR demonstrates efficiency improvements over the random walk approach across all datasets. Specifically, StruProKGR achieves up to 54.93× speedup compared to random walk, substantially reducing computational overhead while preserving accuracy.

Table 5: Ablation study results.

Method	NELL23K		WD-singer	
	MRR	Hits@3	MRR	Hits@3
StruProKGR	**0.262**	**28.5**	**0.461**	**49.8**
w/o structure	0.260	28.2	0.459	49.3
w/o intra-path	0.261	28.1	0.459	49.3
w/o inter-path	0.261	28.2	0.460	49.7

5.4 Ablation Study (RQ3)

To address **RQ3** and examine the impacts of different components in the path structure-based reasoning phase, including the intra-path structure and inter-path structure, we conduct an ablation study on NELL23K and WD-singer, the two most sparse datasets. As shown in Table 5, removing both structures leads to the largest performance drop, highlighting their complementary importance. Nevertheless, even without structural modeling, StruProKGR remains competitive and surpasses prior methods such as DacKGR and SparKGR. Between the two, intra-path modeling contributes slightly more than inter-path, but the small performance gaps across variants indicate the overall robustness of StruProKGR in sparse scenarios.

Table 6: Top 10 relation paths for query $(Kathy\ Cash, father, ?)$.

Query: $(Kathy\ Cash, father, ?)$		
Path	Base Rank	Struc. Rank
$(child^{-1})$	1	1 –
$(sibling^{-1},\ father)$	2	2 –
$(sibling,\ father)$	3	3 –
$(sibling^{-1},\ child^{-1})$	5	4 ↑
$(mother,\ spouse)$	4	5 ↓
$(sibling,\ child^{-1})$	6	6 –
$(child^{-1},\ child,\ father)$	8	7 ↑
$(sibling,\ sibling,\ father)$	11	8 ↑
$(sibling^{-1},\ sibling^{-1},\ father)$	12	9 ↑
$(child^{-1},\ spouse)$	7	10 ↓

5.5 Case Study (RQ4)

To address RQ4 and provide deeper insights into the reasoning process of the StruProKGR, a case study is conducted with the query $(Kathy\ Cash, father, ?)$ in WD-singer, for which the correct answer is $Johnny\ Cash$. The case study examines the top 10 paths that lead to the correct answer, focusing on how the incorporation of intra-path and inter-path structures affects path rankings. The paths, along with their base and structured rankings, are presented in Table 6. The base rank reflects the initial ordering of $P(p|r)_{hop}$ without structural adjustments, while the structural rank accounts for the combined effects of intra-path and inter-path interactions, i.e., ranked by $P(p|r)_{inter}$.

The analysis of Table 6 yields several key insights. Directly relevant paths, such as $(child^{-1})$, $(sibling^{-1}, father)$, and $(sibling, father)$, consistently occupy the top ranks, showing the robustness of structural adjustments in preserving correct evidence. Less coherent paths, e.g., $(child^{-1}, spouse)$, are demoted (rank 7 → 10), while multi-hop variants like $(sibling, sibling, father)$ and $(sibling^{-1}, sibling^{-1}, father)$ are promoted (11 → 8, 12 → 9), indicating that structural modeling enhances the visibility of contextually relevant but indirect paths. Overall, StruProKGR effectively prioritizes the most plausible reasoning chains, improving both accuracy and interpretability.

6 Conclusions

In this paper, we presented StruProKGR, a structural and probabilistic framework for sparse KG reasoning. Unlike previous path-based methods that rely on random walks to collect paths, it employs a distance-guided strategy to facilitate path collection, significantly reducing computational costs while improving the relevance of collected paths. Additionally, it incorporates probabilistic path aggregation to evaluate path reliability and utilizes the structural properties of the graph for accurate knowledge inference. Extensive experiments across five benchmark sparse KG reasoning datasets demonstrate its superior performance over existing path-based methods, offering a solution that is simultaneously effective, efficient, and interpretable for sparse KG reasoning.

Limitations

Although StruProKGR adopts a probabilistic formulation for modeling path structures, the underlying estimates rely on empirical frequency statistics rather than true probability distributions, a constraint shared with prior works. Pure probabilistic modeling remains infeasible in sparse KGs, and frequency-based estimates may violate probability bounds. StruProKGR alleviates this issue through an odds-form Bayesian update, which improves numerical stability in practice. Besides, similar to many sparse KG reasoning approaches, StruProKGR is primarily designed for static KGs. Updates to a dynamic KG would require recomputing path statistics and structural probabilities. However, the paths collected by StruProKGR carry semantic meaning. As a result, once a relation has been observed previously, the model can still make reasonable predictions for the tail entity even in a dynamically evolving KG. Developing efficient mechanisms for supporting graph updates remains an important direction for future work.

Acknowledgments

This work is partially funded by the National Key Research and Development Program of China under Grant No. 2024YFC3308200, the Strategic Priority Research Program of the CAS under Grant No. XDB0680102, the National Natural Science Foundation of China under Grants 62306299, 62441229 and 62406308, and the Innovation Funding of ICT, CAS under Grant No. E561010.

References

Prerna Agarwal, Nishant Kumar, and Srikanta Bedathur. 2024. Symkgqa: few-shot knowledge graph question answering via symbolic program generation and

execution. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10119–10140.

Ivana Balažević, Carl Allen, and Timothy Hospedales. 2019. Tucker: Tensor factorization for knowledge graph completion. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5185–5194.

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

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

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.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam Hruschka, and Tom Mitchell. 2010. Toward an architecture for never-ending language learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 24, pages 1306–1313.

Weijian Chen, Yixin Cao, Fuli Feng, Xiangnan He, and Yongdong Zhang. 2024. Hogrn: Explainable sparse knowledge graph completion via high-order graph reasoning network. *IEEE Transactions on Knowledge and Data Engineering*, 36(12):8462–8475.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *International Conference on Learning Representations*.

Rajarshi Das, Ameya Godbole, Nicholas Monath, Manzil Zaheer, and Andrew Mccallum. 2020. Probabilistic case-based reasoning for open-world knowledge graph completion. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4752–4765.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610.

Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. 2013. Amie: association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd international conference on World Wide Web*, pages 413–422.

Matt Gardner and Tom Mitchell. 2015. Efficient and expressive knowledge base completion using subgraph feature extraction. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1488–1498.

Saiping Guan, Jiyao Wei, Xiaolong Jin, Jiafeng Guo, and Xueqi Cheng. 2024. Look globally and reason: Two-stage path reasoning over sparse knowledge graphs. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 695–705.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. Hipporag: Neurobiologically inspired long-term memory for large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

James Joyce. 2021. Bayes' Theorem. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*, Fall 2021 edition. Metaphysics Research Lab, Stanford University.

Ni Lao, Tom Mitchell, and William Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the 2011 conference on empirical methods in natural language processing*, pages 529–539.

Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015. Modeling relation paths for representation learning of knowledge bases. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 705–714.

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

Xin Lv, Xu Han, Lei Hou, Juanzi Li, Zhiyuan Liu, Wei Zhang, Yichi Zhang, Hao Kong, and Suhui Wu. 2020. Dynamic anticipation and completion for multi-hop reasoning over sparse knowledge graph. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 5694–5703.

Christian Meilicke, Melisachew Wudage Chekol, Manuel Fink, and Heiner Stuckenschmidt. 2020. Reinforced anytime bottom up rule learning for knowledge graph completion. *arXiv preprint arXiv:2004.04412*.

Guanglin Niu, Yongfei Zhang, Bo Li, Peng Cui, Si Liu, Jingyang Li, and Xiaowei Zhang. 2020. Rule-guided compositional representation learning on knowledge graphs. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 2950–2958.

Pouya Ghiasnezhad Omran, Kewen Wang, and Zhe Wang. 2018. Scalable rule learning via learning representation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 2149–2155.

Jay Pujara, Eriq Augustine, and Lise Getoor. 2017. Sparsity and noise: Where knowledge graph embeddings fall short. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 1751–1756.

Petar Ristoski and Heiko Paulheim. 2016. Rdf2vec: Rdf graph embeddings for data mining. In *International semantic web conference*, pages 498–514. Springer.

Tim Rocktäschel and Sebastian Riedel. 2017. End-to-end differentiable proving. *Advances in neural information processing systems*, 30.

Yongjiao Sun, Jiancheng Guo, Boyang Li, and Nur Al Hasan Haldar. 2023. Effective rule mining of sparse data based on transfer learning. *World Wide Web*, 26(1):461–480.

Zhaoxuan Tan, Zilong Chen, Shangbin Feng, Qingyue Zhang, Qinghua Zheng, Jundong Li, and Minnan Luo. 2023. Kracl: Contrastive learning with graph context modeling for sparse knowledge graph completion. In *Proceedings of the ACM web conference 2023*, pages 2548–2559.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1499–1509.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Shijie Wang, Wenqi Fan, Yue Feng, Lin Shanru, Xinyu Ma, Shuaiqiang Wang, and Dawei Yin. 2025. Knowledge graph retrieval-augmented generation for llm-based recommendation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 27152–27168.

Shuyao Wang, Yongduo Sui, Chao Wang, and Hui Xiong. 2024. Unleashing the power of knowledge graph for recommendation via invariant learning. In *Proceedings of the ACM Web Conference 2024*, WWW '24, page 3745–3755, New York, NY, USA. Association for Computing Machinery.

Yi Xia, Mingjing Lan, Junyong Luo, Xiaohui Chen, and Gang Zhou. 2022. Iterative rule-guided reasoning over sparse knowledge graphs with deep reinforcement learning. *Information Processing & Management*, 59(5):103040.

Yi Xia, Junyong Luo, Gang Zhou, Mingjing Lan, Xiaohui Chen, and Jing Chen. 2024. Dt4kgr: Decision transformer for fast and effective multi-hop reasoning over knowledge graphs. *Information Processing & Management*, 61(3):103648.

Shaorong Xie, Ruishen Liu, Xinzhi Wang, Xiangfeng Luo, Vijayan Sugumaran, and Hang Yu. 2024. Hierarchical knowledge-enhancement framework for multi-hop knowledge graph reasoning. *Neurocomputing*, 588:127673.

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

Zhanqiu Zhang, Jie Wang, Jieping Ye, and Feng Wu. 2022. Rethinking graph convolutional networks in knowledge graph completion. In *Proceedings of the ACM web conference 2022*, pages 798–807.

Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. 2021. Neural bellman-ford networks: A general graph neural network framework for link prediction. *Advances in neural information processing systems*, 34:29476–29490.

Algorithm 1: Distance-Guided Path Collection

Input: Sparse KG $\mathcal{G}_s = \{(h, r, t) | h, t \in \mathcal{E}, r \in \mathcal{R}\}$, entity type mapping function $\psi : \mathcal{E} \rightarrow \mathcal{C}$, maximum path length l_{max}, maximum branch number k.

Output: Set of collected type-specific relation paths $\mathcal{P}$.

```
/* Step 1. Precompute shortest distances */
1  foreach u ∈ E do
2      Compute distance matrix dist[u][v] for all v with dist[u][v] ≤ l_max;
/* Step 2. Distance-guided path collection */
3  P[c][r] ← ∅ for all c ∈ C, r ∈ R;
4  foreach (h, r, t) ∈ G_s do
5      Initialize stack S ← [(h, [], {h})];
6      while S is not empty do
7          (u, path, visited) ← S.pop();
8          if u = t then
9              P[ψ(h)][r] ← P[ψ(h)][r] ∪ {path};
10             continue;
11         nextEntities ← [];
12         foreach (rel, v) in adjacency list of u do
13             if v ∉ visited and dist[v][t] ≤ l_max − len(path) − 1 then
14                 nextEntities.append((rel, v));
15         if |nextEntities| > k then
16             sort nextEntities by dist[v][t] in ascending order;
17             nextEntities ← nextEntities[: k];
18         foreach (rel, v) ∈ reverse(nextEntities) do
19             S.push((v, path + [rel], visited ∪ {v}));
20 return P;
```

A Methodology Details

A.1 Distance-Guided Path Collection

Algorithm 1 shows the process of distance-guided path collection, which integrates global distance information into local DFS expansion, enabling efficient pruning and effective prioritization of short and relevant paths. Algorithm 1 consists of two steps: *distances precomputation* (lines 1-2) and *distance-guided path collection* (lines 3-19). In the first step, all pairwise distances up to the maximum path length l_{max} are computed. In the second step, a bounded DFS from each triple (h, r, t) is performed, maintaining at most k most promising next-hop expansions at every step. For each training triple (h, r, t), we push the initial state $(h, [], \{h\})$ onto a stack S (line 5) and then repeatedly pop a state $(u, path, visited)$, where u represents the current entity being explored, $path$ denotes the list of relations traversed, and $visited$ tracks the set of entities already encountered to avoid cycles. (lines 6-7). If $u = t$, which indicates that the current path can reach the tail entity, then the path will be recorded in $\mathcal{P}[\psi(h)][r]$ as a type-specific relation path of type $\psi(h)$ and relation r (lines 8-9). Subsequently, any further expansion will be skipped (line 10). Otherwise, we enumerate all outgoing edges (rel, v) of u and include in $nextEntities$, the list of candidate neighbor nodes for extending the path, only those neighbors $v \notin visited$ whose precomputed distance to t satisfies

$$dist[v][t] \leq l_{max} - len(path) - 1. \quad (12)$$

This criterion (line 13) ensures that no loops are present in the path, and each candidate v has at least one path to the target within the remaining length budget. To further narrow down the search, if the size of $nextEntities$ exceeds k, the maximum branch number, the candidates will be sorted by $dist[v][t]$ in ascending order and only the top k will be kept (lines 15-17). Finally, for each (rel, v) in this pruned set, we push the updated

state $(v, path \cup [rel], visited \cup \{v\})$ onto S in reverse order, ensuring that entities closer to t are explored first (lines 18-19).

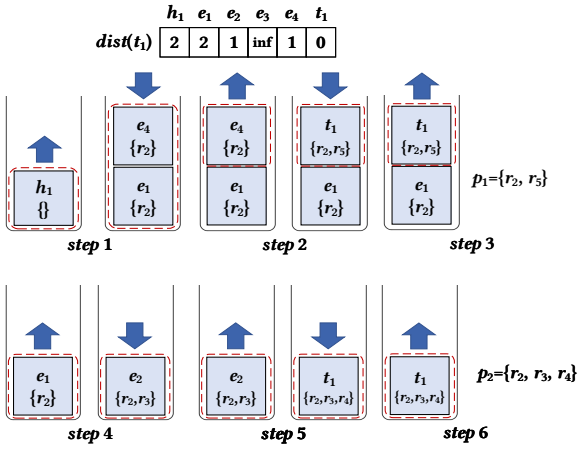


Figure 4: Illustration of distance-guided path collection.

Example A.1 *Figure 4 illustrates the distance-guided path collection process for the toy sparse KG depicted in Figure 2. Given the triple (h_1, r_1, t_1) and the precomputed distance to t_1, as shown by the $dist(t_1)$ with values 2, 2, 1, inf, 1, and 0 for h_1, e_1, e_2, e_3, e_4, and t_1 respectively. The process unfolds in six steps, and the $visited$ set in each state is omitted for simplicity. In the first step, h_1 is paired with an empty relation set $\{\}$. Its neighbors, e_1 and e_4, are then pushed onto the stack in descending order of their distance to t_1. Consequently, e_4 is explored first, as it is closer to t_1 than e_1. Subsequent steps (2-3) explore paths from e_4 to t_1, accumulating relations $\{r_2\}$ and $\{r_2, r_5\}$ respectively, forming path $p_1 = \{r_2, r_5\}$. Steps 4-6 extend the search from e_1 and e_2 to t_1, resulting in path $p_2 = \{r_2, r_3, r_4\}$, while e_3 is pruned during exploration as it is unreachable to t_1.*

Lemma A.1 *The time complexity of Algorithm 1 is $O\big(|\mathcal{E}| \cdot (|\mathcal{E}| + |\mathcal{G}_s|) + k^{l_{max}} \cdot |\mathcal{G}_s|\big)$, where $|\mathcal{G}_s|$ is the number of triples in $\mathcal{G}_s$.*

Proof A.1 *The initialization of distances involves $O(|\mathcal{E}|)$ BFS operations, each taking $O(|\mathcal{E}| + |\mathcal{G}_s|)$ time in the worst case, i.e., each distances $\leq l_{max}$. Thus, the total precomputation cost is $O\big(|\mathcal{E}| \cdot (|\mathcal{E}| + |\mathcal{G}_s|)\big)$. During the DFS phase, each popped state expands at most k neighbors, and the maximum stack depth is l_{max}. In the worst case, the number of DFS states is bounded by $O(k^{l_{max}})$, giving $O\big(k^{l_{max}} \cdot |\mathcal{G}_s|\big)$ time in theory. In practice, the BFS operates up to a maximum depth of l_{max}, and the distance-based pruning in line 13 significantly reduces the number of explored branches, often resulting in a much lower time complexity.*

A.2 The Calculation of Path Probability and Joint Probability

Algorithm 2 shows the batch search-based path traversal process when calculating path probability and joint probability. It begins by initializing a set of current entities with the head entity h and an associated count of 1 (line 1). For each relation r_i in the path $p = [r_1, r_2, \ldots, r_n]$, it iterates through the current entities and identifies all triples in the sparse KG $\mathcal{G}_s$ that match the relation r_i, collecting the next set of reachable entities (lines 2-6). The counts of these entities are updated by adding the count of their predecessor entities, effectively tracking the number of ways to reach each entity (line 6). After processing all relations in the path, the algorithm returns the final set of reachable entities along with their counts (line 8), enabling efficient computation of the reachable entities that are needed for probability calculations.

Algorithm 2: Path Traversal

Input: Sparse KG $\mathcal{G}_s$, entity h, path $p = [r_1, r_2, \ldots, r_n]$.
Output: Set of reachable entities ans.

1 $curEntities \leftarrow \{h : 1\}$;
2 **foreach** $r_i \in p$ **do**
3 $\quad nextEntities \leftarrow \{\}$;
4 $\quad$ **foreach** $e_0 \in curEntities$ **do**
5 $\quad\quad$ **foreach** $(e_0, r_i, e_1) \in \mathcal{G}_s$ **do**
6 $\quad\quad\quad nextEntities[e_1] \leftarrow nextEntities[e_1] + curEntities[e_0]$;
7 $\quad curEntities \leftarrow nextEntities$;
8 **return** $curEntities$;

Lemma A.2 *The time complexity for calculating path probabilities $P(p|r)$ and joint probabilities $P(p_i, p_j|r)$ is $O(|\mathcal{G}_s| \cdot N_{paths} \cdot l_{max} + N_{pairs})$, where N_{paths} is the average number of paths per relation, and N_{pairs} is the number of path pairs.*

Proof A.2 *When calculating path probability, processing each triple in $\mathcal{G}_s$ involves executing up to N_{paths} paths of length l_{max}, yielding a complexity of $O(|\mathcal{G}_s| \cdot N_{paths} \cdot l_{max})$. For joint probability, although the results from the path traversals can be reused, an intersection operation is necessary for N_{pairs} path pairs, contributing a complexity*

Algorithm 3: Path Structure-based Reasoning

Input: Sparse KG $\mathcal{G}_s$, Query $(h, r, ?)$, set of relation paths $\mathcal{P}(r)$ with path probabilities $P(p|r)$, adjusted probabilities $P(p|r)_{hop}$ and joint probabilities $P(p_i, p_j|r)$, the number of explored top paths N_{top}.

Output: Ranked list of candidate answers $\mathcal{A}$.

```
/* Step 1. Execute top paths to gather candidates */
A ← {}, executedPaths ← 0;
foreach p ∈ P(r) do
    A' ← PathTraversal(G_s, h, p);
    if A' ≠ {} then
        foreach (a, times) ∈ A' do
            A(a).append((p, times));
        executedPaths ← executedPaths + 1;
        if executedPaths == N_top then
            break;
/* Step 2. Calculate the probability for each candidate */
foreach a ∈ A do
    Initialize probability P(a) ← 0;
    foreach (p, times) ∈ a do
        P(p|r)_intra ← calculate using the intra-path structure.;
        P(p|r)_inter ← calculate using the inter-path structure;
        P(a) ← P(a) + P(p|r)_inter − P(a) · P(p|r)_inter;
Sort A by P(a) in descending order;
return A;
```

of $O(N_{pairs})$. Therefore, the total time cost for computing probabilities is $O(|\mathcal{G}_s| \cdot N_{paths} \cdot l_{max} + N_{pairs})$.

A.3 Path Structure-based Reasoning

A.3.1 Mathematical Proof for Inter-path Structure Modeling

The Odds Form of Bayes' Theorem. The standard form of Bayes' theorem is unsuitable for updating path probabilities, as these probabilities are calculated based on statistical data. This can lead to updated probabilities $P(p|r)_{inter} > 1$, which is not valid. To overcome this issue, we utilize the odds form of Bayes' theorem, which ensures that the updated probability remains within the valid range $[0, 1]$ (Joyce, 2021). The odds form is defined as follows:

$$\frac{P(A|B)}{P(\neg A|B)} = \frac{P(A)}{P(\neg A)} \times \frac{P(B|A)}{P(B|\neg A)}, \tag{13}$$

where $\frac{P(A)}{P(\neg A)}$ and $\frac{P(A|B)}{P(\neg A|B)}$ are known as the prior odds and the posterior odds, respectively, while $\frac{P(B|A)}{P(B|\neg A)}$ is called the likelihood ratio. In this context, A represents the event that a specific path p_i correctly infers the relation r, while B denotes the evidence obtained from other paths.

We utilize the path probability and the joint probability to form the basis of our model. Strictly speaking, the interaction between paths should be captured through the likelihood ratio:

$$LR(p_i, p_j) = \frac{P(p_j|p_i, r)}{P(p_j|\neg p_i, r)}, \tag{14}$$

where $P(p_j|p_i, r)$ is the conditional probability that p_j is correct given p_i is correct, and $P(p_j|\neg p_i, r)$ is the probability that p_j is correct given p_i is incorrect. However, computing this ratio requires conditioning on the unobserved correctness of p_i, which is computationally intensive and impractical in large KGs with many paths. Pairwise computation across all paths in $\mathcal{P}(r)$ further exacerbates the scalability issue.

A Scalable Approximation. To address these challenges, we propose an approximation that aggregates evidence from multiple paths while avoiding

Table 7: Proportion of running time (%) for each stage across datasets.

Stage	FB15K-237-10%	FB15K-237-20%	FB15K-237-50%	NELL23K	WD-singer
Distance-Guided Path Collection	13.2	6.2	7.3	29.3	18.3
The Cal. of Path Prob. and Joint Prob.	69.3	72.6	83.1	57.8	69.0
Path Structure-based Reasoning	17.5	21.2	9.6	12.9	12.7

the need for exact conditional probabilities:

$$LR(p_i, \mathcal{P}(r) \setminus \{p_i\}) = \frac{\sum_{p_j} P(p_i, p_j|r)}{\sum_{p_j} [P(p_i|r) + P(p_j|r) - P(p_i|r) \cdot P(p_j|r)]}, \tag{15}$$

where $p_j \in \mathcal{P}(r) \setminus \{p_i\}$ and subject to the condition $P(p_j|r)_{hop} > P(p_i|r)_{hop}$. The ratio compares the observed joint correctness to the expected correctness under independence. A value greater than 1 suggests that the paths are more likely to be correct together than independently, indicating collaboration, while a value less than 1 suggests inhibition. By aggregating over multiple paths rather than computing pairwise conditionals, the approach scales linearly with the number of relevant paths, making it feasible for large KGs. The choice to include only paths p_j where $P(p_j|r)_{hop} > P(p_i|r)_{hop}$ leverages the higher reliability of p_j to provide more compelling evidence for updating the probability of p_i. By prioritizing these stronger, more trustworthy paths, the influence of noisy or less dependable signals is reduced, thereby improving the accuracy of the inference. Additionally, this selective focus reduces computational overhead by limiting the number of paths considered, making the process both more efficient and effective.

A.3.2 Path Structure-based Reasoning Algorithm

Algorithm 3 shows the detailed process of path structure-based reasoning. Given a query $(h, r, ?)$, it first traverses the top N_{top} relation paths in $\mathcal{P}(r)$ from the head entity h using `PathTraversal`, collecting candidate answers $\mathcal{A}$ (lines 1-9). Next, for each candidate answer $a \in \mathcal{A}$, it initializes $P(a) = 0$ and updates $P(a)$ by aggregating path probabilities $P(p|r)_{inter}$, which is calculated by considering the intra-path structure and the inter-path structure, sequentially (lines 10-15). Finally, the candidate answers in $\mathcal{A}$ are ranked based on $P(a)$, and the sorted list is returned (lines 16-17).

Lemma A.3 *The time complexity for path structure-based reasoning is $O(\frac{|\mathcal{A}| \cdot N_{top}^2}{2})$, where $|\mathcal{A}|$ is the number of candidate answers, and N_{top} is the number of explored top paths.*

Proof A.3 *For each candidate $a \in \mathcal{A}$, the algorithm evaluates each path p reaching a, takes at most $O(l_{max} \cdot N_{top})$ time for path traversal. For each path, it adjusts probabilities by considering paths with higher probability than it, adding $O(\frac{N_{top}^2}{2})$ complexity per candidate. Thus, the total time complexity is $O(|\mathcal{A}| \cdot (l_{max} \cdot N_{top} + \frac{N_{top}^2}{2})) = O(\frac{|\mathcal{A}| \cdot N_{top}^2}{2})$.*

B Computational Cost Analysis

We provide a breakdown of the computational cost across different stages of our framework. As shown in Table 7, compared with prior path-based methods, where path collection typically dominates the runtime, our distance-guided path collection (including distance precomputation) reduces the cost of this stage by restricting exploration within a bounded radius. As a result, the primary computational overhead shifts to the calculation of path probability and joint probability, while previous works also take a similar time to conduct this process.