Mixture of Length and Pruning Experts for Knowledge Graphs Reasoning

Enjun Du[1,2], Siyi Liu[1], Yongqi Zhang[1]*
[1]The Hong Kong University of Science and Technology (Guangzhou)
[2]Beijing Institute of Technology
{EnjunDu.cs, ssui.liu1022}@gmail.com, yongqizhang@hkust-gz.edu.cn

Abstract

Knowledge Graph (KG) reasoning, which aims to infer new facts from structured knowledge repositories, plays a vital role in Natural Language Processing (NLP) systems. Its effectiveness critically depends on constructing informative and contextually relevant reasoning paths. However, existing graph neural networks (GNNs) often adopt rigid, query-agnostic path-exploration strategies, limiting their ability to adapt to diverse linguistic contexts and semantic nuances. To address these limitations, we propose **MoKGR**, a mixture-of-experts framework that personalizes path exploration through two complementary components: (1) a mixture of length experts that adaptively selects and weights candidate path lengths according to query complexity, providing query-specific reasoning depth; and (2) a mixture of pruning experts that evaluates candidate paths from a complementary perspective, retaining the most informative paths for each query. Through comprehensive experiments on diverse benchmark, MoKGR demonstrates superior performance in both transductive and inductive settings, validating the effectiveness of personalized path exploration in KGs reasoning.[1]

1 Introduction

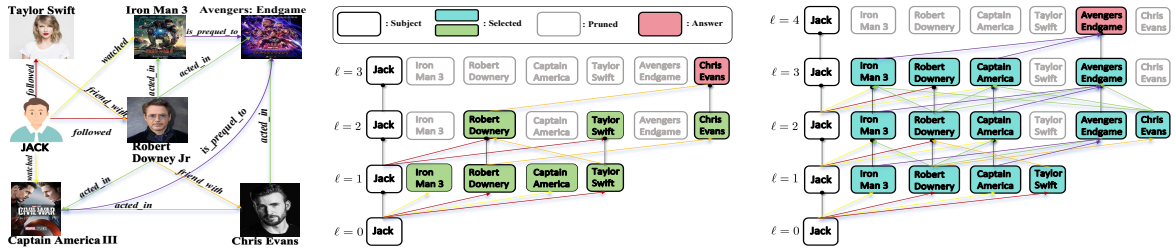
Knowledge Graphs (KGs) are integral to Natural Language Processing (NLP) (Liu et al., 2025a, 2024; Siyue et al., 2025; Shen et al., 2025; Miao et al., 2025; Wang et al., 2025a; Li et al., 2025), offering structured knowledge representations crucial for various language understanding and generation tasks (Ji et al., 2021; Liang et al., 2024; Du et al., 2025a; Zheng et al., 2025b; Liu et al., 2025b; Zheng et al., 2025a). In KGs, entities and their semantic associations are systematically encoded as relational triples (*subject, relation, object*) (Ali et al., 2022; Sun et al., 2021; Zhang et al., 2022, 2019b, 2024, 2020, 2023c), often derived from or used to interpret textual data. These triples form semantic networks that capture intricate connectivity and meaning (Nickel et al., 2015; Ji et al., 2022; Wang et al., 2023b; Zhang et al., 2023b; Yue et al., 2023; Di et al., 2021), thereby enabling advanced NLP applications like sophisticated question answering and semantic search. A KG query can be formulated via a function $\mathcal{Q}$ as $\mathcal{Q}(e_q, r_q) = e_a$, where e_q, r_q, and e_a represent the query entity, query relation, and answer entity, respectively.

Various approaches have been developed to conduct effective and efficient KG reasoning (Dettmers et al., 2017a; Zhang et al., 2020; Du et al., 2025b). A primary focus of this research is the generation and encoding of effective reasoning paths. Included methods that learn logical rules for path generation (Cheng et al., 2022; Qu et al., 2021; Sadeghian et al., 2019; Qiu et al., 2024), or employ reinforcement learning to discover paths based on query conditions (Das et al., 2017). With the advent of Graph Neural Networks (GNNs), recent studies like NBFNet (Zhu et al., 2021) and RED-GNN (Zhang and Yao, 2022) iteratively aggregate and encode all reasoning paths of a certain length ℓ. To reduce computational complexity, subsequent approaches introduce path pruning strategies (Zhu et al., 2023; Zhang et al., 2023d). However, most current GNN-based approaches utilize rigid, query-agnostic path encoding and pruning strategies, resulting in two primary limitations:

- **Disregarding Dynamic Query Requirements.** Existing methods often employ a fixed hop count for path construction, failing to adapt to the dynamic requirements of individual queries. This uniform approach extends paths to the same depth for every query, overlooking the fact that optimal reasoning paths inherently vary. For instance, as illustrated in Fig. 1, resolving the query

* Corresponding author
[1]Code is available on https://github.com/EnjunDu/MoKGR.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 433–454
November 4-9, 2025



(a) Knowledge Graph (b) *(Jack, followed, ?)* (c) *(Jack, watched, ?)*

Figure 1: (a) A complex knowledge graph with two queries—*(JACK, followed, ?)* and *(JACK, watched, ?)*—and their respective answers *Chris Evans* and *Avengers: Endgame*. (b) and (c) visualize MoKGR's personalized path exploration for each query, highlighting its adaptive path length selection and expert-guided pruning, which result in distinct retained paths and entities during reasoning.

(JACK, followed, ?) to find *Chirs Evans* naturally concludes within three hops. Conversely, addressing the query *(JACK, watched, ?)* to identify *Avengers: Endgame* might require exploration beyond three hops to capture critical relationships. These scenarios underscore the need for hop-level personalization, tailoring path exploration to query complexity to improve reasoning efficiency and accuracy.

- **Oversimplified Path Exploration Strategies.** Existing methods often rely on overly simplified path exploration strategies, treating all paths equivalently. Although pruning advancements like AdaProp (Zhang et al., 2023d) and A*Net (Zhu et al., 2023) enhance efficiency, their pruning criteria remain largely uniform, neglecting the distinct significance of individual paths. Effective path exploration should integrate two complementary aspects: (1) **structural patterns**, capturing entity importance through multi-dimensional assessments to accurately identify high-quality paths; and (2) **semantic relevance**, assessing the degree of entity-query association, with paths featuring highly relevant entities more likely yielding correct answers. Proper consideration of these aspects can significantly enhance reasoning path quality.

To address these limitations, we propose a novel framework called the **M**ixture **o**f Length and Pruning Experts for **K**nowledge **G**raph **R**easoning (**MoKGR**). As illustrated in Fig. 1b and Fig. 1c, MoKGR introduces personalization into path exploration via two complementary innovations. First, it employs an **adaptive length-level selection mechanism**, which functions as a mixture of length experts, dynamically assigning importance weights to various path lengths based on individual queries. This allows shorter paths to be selected when adequate, avoiding unnecessary exploration depth. Second, MoKGR utilizes **specialized pruning experts** that analyze diverse properties: global importance through prediction scores, local structural patterns via attention mechanisms, and semantic relationships through entity-query similarity. Thus, MoKGR comprehensively incorporates structural and semantic considerations into path pruning, ensuring robust and query-specific reasoning paths. The contributions can be summarized as follows:

- We propose a personalized path exploration strategy for knowledge graph reasoning that adapts to query-specific requirements and entity characteristics, thereby enabling tailor-made reasoning paths without relying on predefined or static relation selection strategies.
- We introduce a novel mixture-of-experts framework that facilitates personalization in knowledge graph reasoning. The incorporating of both adaptive length-level weighting and personalized pruning strategies effectively addresses the critical limitations of fixed path length and uniform path exploration.
- Experimental results on transductive and inductive datasets highlight MoKGR's achievement of superior reasoning accuracy and computational efficiency, enabling it to consistently outperform existing state-of-the-art methods.

2 RELATED WORKS

2.1 Path-based Methods for KG Reasoning

Path-based reasoning methods aim to construct effective reasoning paths for predicting answer entities through the query function $\mathcal{Q}(e_q, r_q) = e_a$.

These methods can be broadly categorized into traditional path reasoning approaches and more recent GNN-based methods.

Traditional Path Reasoning. Early path reasoning methods primarily rely on reinforcement learning and rule-based approaches. MINERVA (Das et al., 2017) pioneered the use of reinforcement learning, training an agent to autonomously traverse the graph from query entity e_q to potential answers. However, these RL-based approaches often face challenges due to the inherent sparsity of KGs. As an alternative direction, rule-based methods focus on learning logical rules for path generation. DRUM (Sadeghian et al., 2019) employs bidirectional LSTM to capture sequential patterns and enable end-to-end rule learning, while RLogic (Cheng et al., 2022) combines deductive reasoning with representation learning through recursive path decomposition. Despite their contributions, these methods typically focus on sequential pattern extraction without considering personalized path exploration requirements.

GNN-based Path Reasoning. More recently, GNN-based methods have achieved superior performance by effectively leveraging the rich structural information preserved in graphs. Methods such as NBFNet (Zhu et al., 2021) and RED-GNN (Zhang and Yao, 2022) construct reasoning paths by iteratively aggregating information from the ℓ-length neighborhood of the query entity e_q. To enhance path quality, various optimization techniques have been proposed. A*Net (Zhu et al., 2023) and AdaProp (Zhang et al., 2023d) introduce pruning mechanisms based on priority functions and score-based filtering respectively. One-Shot-Subgraph (Zhou et al., 2024) improves efficiency by utilizing PPR scores for preliminary path exploration and pruning. However, these approaches typically employ rigid path exploration strategies with fixed length distances and uniform pruning criteria, limiting their adaptability to query-specific requirements in practical scenarios.

2.2 Mixture of Experts

The Mixture-of-Experts (MoE) paradigm represents a divide-and-conquer learning strategy where multiple specialized expert models collaborate to solve complex tasks, with a gating mechanism dynamically routing inputs to the most suitable experts. The foundational concept of MoE can be traced back to (Jordan and Jacobs, 1994) and has been widely adopted in vision (Riquelme et al., 2021), multi-modal learning (Mustafa et al., 2022), and multi-task learning (Zhu et al., 2022).

In graph-related applications, MoE has demonstrated significant advantages by leveraging diverse graph properties. MoKGE (Yu et al., 2022) integrates experts specializing in different subspaces and relational structures of commonsense KGs, achieving diverse outputs in generative commonsense reasoning. MoG (Zhang et al., 2023a) incorporates pruning experts with complementary sparsification strategies, where each expert executes a unique pruning method to customize pruning decisions for individual nodes. Meanwhile, GMoE (Wang et al., 2023a) deploys message-passing experts that specialize in different hop distances and aggregation patterns, enabling nodes to adaptively select suitable experts for information propagation based on their local topologies.

3 THE PROPOSED METHOD

3.1 Preliminary

As introduced, the reasoning task in KGs is to find the answer entity e_a given a query $(e_q, r_q, ?)$, which we denote as $q = (e_q, r_q)$. To solve this task, GNN-based path reasoning methods, such as NBFNet and RED-GNN, encode all paths up to length L between e_q and e_a into a query-specific representation $\boldsymbol{h}^L_{e_a|q} \in \mathbb{R}^d$, and use it to compute the score of candidate entity e_a. The representation $\boldsymbol{h}^\ell_{e_y|q}$ at iteration ℓ is recursively computed via the following message passing function:

$$\boldsymbol{h}^\ell_{e_y|q} = \bigoplus_{(e_x,r,e_y)\in\mathcal{N}_e(e_y)} \left(\boldsymbol{h}^{\ell-1}_{e_x|q} \otimes \boldsymbol{w}^\ell_q(e_x, r, e_y)\right), \quad (1)$$

where $\boldsymbol{h}^{\ell-1}_{e_x|q}$ encodes all paths of length up to $\ell-1$ from e_q to e_x, and $\boldsymbol{w}^\ell_q(e_x, r, e_y)$ is an edge-specific weight conditioned on the query q. The operator $\otimes$ combines the path representation with the current edge encoding to form a new path of length ℓ, and $\bigoplus$ aggregates multiple such paths reaching e_y. We initialize all representations with $\boldsymbol{h}^0_{e_y|q} = \mathbf{0}$ for any entity $e_y \in \mathcal{V}$, and entities e_y that are further than ℓ steps from e_q will have $\boldsymbol{h}^\ell_{e_y|q} = \mathbf{0}$. After L iterations of Eq. (1), the final score of any entity $e_a \in \mathcal{V}$ is computed by

$$s_L(q, e_a) = (\boldsymbol{w}^L)^\top \boldsymbol{h}^L_{e_a|q}, \quad (2)$$

where $\boldsymbol{w}^L \in \mathbb{R}^d$ is a learnable scoring vector. More details of this path encoding process are provided in Appendix B.1.

3.2 Mixture of Length Experts for Adaptive Path Selection

Traditional KGs reasoning methods employ fixed-length path exploration strategies, which fail to capture the varying complexity of different queries and waste computation cost. To address this limitation, we introduce a mixture of length experts that adaptively selects paths with different lengths and a layer-wise binary gating function to encourage shorter paths.

Mixture of length experts. For a given query $(e_q, r_q, ?)$, we presuppose the minimum and maximum path lengths $L_{\min}$ and L, respectively, and specify the number of selected path length experts as k_1 ($< L - L_{\min}$). Instead of processing all the queries with paths up to length L in Eq. (2), we introduce a mixture of length experts to score entities with a set of path lengths in intermediate $\ell \in [L_{\min}, L]$.

We enable personalized selection of different path lengths. Denote $\boldsymbol{c}_q$ as the contextual embedding of query $(e_q, r_q, ?)$ (details are given in Appendix B.2.1) and $\boldsymbol{E}_1 \in \mathbb{R}^{(L-L_{\min})\times d}$ as the learnable expert embedding of paths with lengths from $L_{\min}$ to L. Then, we can measure the compatibility of each path length expert with

$$\boldsymbol{Q}(\boldsymbol{c}_q) = \boldsymbol{E}_1 \boldsymbol{c}_q + \epsilon \cdot \text{Softplus}(\boldsymbol{W}_n \boldsymbol{c}_q) \in \mathbb{R}^{L-L_{\min}}, \quad (3)$$

where $\epsilon \sim \mathcal{N}(0,1)$ is a Gaussian noise works with Softplus (Dugas et al., 2001) to encourage diverse expert selection and $\boldsymbol{W}_n \in \mathbb{R}^{(L-L_{\min})\times d}$ is a trainable parameter that learns noise scores. Consequently, we obtain the set $\mathcal{A} := \text{Top}_{k_1}(\boldsymbol{Q}(\boldsymbol{c}_q))$ as the indices of selected path lengths. Then the importance of layer $\ell \in \mathcal{A}$ can be computed with softmax function

$$g_q(\ell) = \frac{\exp([\boldsymbol{Q}(\boldsymbol{c}_q)]_\ell / \tau)}{\sum_{\ell' \in \mathcal{A}} \exp([\boldsymbol{Q}(\boldsymbol{c}_q)]_{\ell'} / \tau)}. \quad (4)$$

We then compute the score of an answer entity e_a with the gated outputs of selected experts with different path lengths

$$\Psi(e_a) = \sum\nolimits_{\ell \in A} g_q(\ell) \cdot s_l(q, e_a), \quad (5)$$

where the score $s_l(q, e_a) = (\boldsymbol{w}^\ell)^\top \boldsymbol{h}^\ell_{e_a|q}$ at different ℓ is defined similarly with Eq. (2).

Layer-wise binary gating function. Even though Eq. (5) can adaptively control the importance of different path length ℓ, a limitation still exists that the paths with length from 1 to L should be explored and encoded. This can lead to significant computation costs at large layers. To address this issue, we introduce a layer-wise binary gating function to encourage the model to explore shorter paths. Specifically, during training, we employ a differentiable statistics-based binary gating function $g_b(\ell) \in (0, 1)$ calculated by the Gumbel-Sigmoid (Jang et al., 2017) transformation that evaluates path quality based on layer-wise feature distributions to learn a natural bias towards shorter paths while maintaining differentiability. (details are given in Appendix B.2.2). We use $g_b(\ell)$ to control the update of the message function in Eq. (1) with $\boldsymbol{h}^\ell_{e_y|q} \leftarrow g_b(\ell) \cdot \boldsymbol{h}^\ell_{e_y|q}$. During inference, we further strengthen this preference through a deterministic truncation strategy, where $g_b(\ell) = 1$, if the paths should continue grow, otherwise $g_b(\ell) = 0$.

The iterative message passing process will immediately stop if $g_b(\ell) = 0$. This length control mechanism enables the model to systematically prefer shorter paths when they provide sufficient evidence for reasoning, improving inference efficiency.

3.3 Mixture of Pruning Experts for Personalized Path Exploration

Apart from selecting path length adaptively, we propose to encourage personalized path pruning, which incorporates both structural patterns and semantic relevance in KGs reasoning.

We build upon the node-wise pruning mechanism established in AdaProp (Zhang et al., 2023d). Denote $\mathcal{V}^\ell$ as the set of entities that are covered by the message function Eq. (1) at step ℓ. $\mathcal{V}^\ell$ contains all the ending entities of paths with lengths up to ℓ. When expanding from $\mathcal{V}^{\ell-1}$ to $\mathcal{V}^\ell$, we select Top-K^ℓ entities from $\mathcal{V}^\ell$ as an approach to control the number of selected paths. To implement this fine-grained personalization of path exploration, we propose three specialized pruning experts with different scoring function $\phi^\ell_i(\cdot)$ that analyzes the importance of entities $e_a \in \mathcal{V}^\ell$ from complementary perspectives.

- The Scoring Pruning Expert evaluates the overall contribution to reasoning with the layer-wise score: $\phi^\ell_{\text{Sco}}(e_a) = s_l(q, e_a) = (\boldsymbol{w}^\ell)^\top \boldsymbol{h}^\ell_{e_a|q}$.
- The Attention Pruning Expert specifically ad-

dresses the structural patterns by examining relation combinations and connectivity patterns through an attention mechanism. This expert identifies entities that are important to at least one path connected. As defined in Eq. (1), at each length $\ell-1$, for each entity $e_y \in \mathcal{V}^\ell$, we calculate the attention scores α (detailed computation process is provided in Appendix B.1) for all neighboring edges $(e_x, r, e_y) \in \mathcal{N}_e(e_y)$ where $e_x \in \mathcal{V}^{\ell-1}$, and assign the maximum attention score among all edges connected to e_y as the attention score of entity e_y: $\phi^\ell_{\text{Att}}(e_a) = \text{Max}(\alpha(e_x, r, e_a)|(e_x, r, e_a) \in \mathcal{N}_e(e_a))$.

- The Semantic Pruning Expert focuses on semantic relevance by computing the semantic alignment between entities and query relations, ensuring that the selected paths contain thematically coherent concepts that are meaningfully related to the query context. For instance, when reasoning about movie preferences, this expert would favor paths containing entertainment-related entities and relations. We use cosine similarity to measure the coherence: $\phi^\ell_{\text{Sem}}(e_a) = \cos\left(\boldsymbol{h}^\ell_{e_a|q}, \boldsymbol{w}^\ell_{r_q}\right)$.

To adaptively combine insights from these path evaluation experts, at each layer ℓ, similar as the lengths experts defined in Section 3.2, denote $\boldsymbol{c}^\ell_v$ as the contextual embedding and $\boldsymbol{E}^\ell_2 \in \mathbb{R}^{3\times d}$ as the learnable embedding of pruning experts at length ℓ, we can similarly get $\boldsymbol{Q}^\ell(\boldsymbol{c}^\ell_v) \in \mathbb{R}^3$ as defined in Eq. (3). Let $\mathcal{V}^\ell_{\phi_i}$ denote the set of entities retained by expert i and k_2 denote the predefined number of retained pruning experts, the entities retained in the ℓ-th layer are the union of the entities retained by each selected pruning experts as:

$$\mathcal{V}^\ell_\phi = \{\cup_{i\in \text{TopK}_{k_2}(\boldsymbol{Q}^\ell(\boldsymbol{c}^\ell_v))} \mathcal{V}^\ell_{\phi_i} | \mathcal{V}^\ell_{\phi_i} = \text{TopK}_{K^\ell}(\phi^\ell_i(e_a))\}. \quad (6)$$

To further enhance path quality, we introduce an adaptive path exploration strategy that dynamically controls the exploration breadth. Our strategy allows K^ℓ to increase with path exploration depth in early stages, while decreasing at larger depth (Detailed description in Appendix B.4). This strategy enables thorough exploration of promising path regions while preventing noise accumulation from overextended paths.

3.4 Training Details

Task Loss To enable effective personalized path exploration, we formulate a task loss that jointly optimizes the GNN parameters and expert model parameters. The task loss is defined as:

$$\mathcal{L}_{\text{task}} = \sum_{(e_q, r_q, e_a)\in \mathcal{Q}_{tra}} \left[-\Psi(e_a) + \log \sum_{e_o \in \mathcal{V}} \exp(\Psi(e_o))\right], \quad (7)$$

where the first part is the total score of the positive triple (e_q, r_q, e_a) in the set of training queries, and the second part contains the total scores of all triple with the same query $(e_q, r_q, ?)$.

Experts Balance Loss To achieve balanced and effective path exploration and address the potential "winner-takes-all" problem (Lepikhin et al., 2020), we follow (Wang et al., 2023a) by introducing several regularization terms. These terms prevent the model from overly relying on specific exploration strategies or experts (Details are provided in Appendix D.2). The importance loss is defined as:

$$\begin{aligned} \text{Importance}(\mathcal{C}) &= \sum_{\boldsymbol{c}\in\mathcal{C}} \sum_{g\in\mathcal{G}(\boldsymbol{c})} g, \\ \mathcal{L}_{\text{Importance}}(\mathcal{C}) &= \text{CV}(\text{Importance}(\mathcal{C}))^2, \end{aligned} \quad (8)$$

where $g \in \mathcal{G}(\boldsymbol{c})$ denotes the output of the experts' gating mechanism as calculated in Eq. (4), and $\text{CV}(\boldsymbol{X}) = \sigma(\boldsymbol{X})/\mu(\boldsymbol{X})$ represents the coefficient of variation of input $\boldsymbol{X}$. This formulation yields the length expert importance loss $\mathcal{L}_l$ and pruning importance loss $\mathcal{L}_p$. Furthermore, we introduce a load balancing loss for length experts:

$$\mathcal{L}_{\text{load}} = \text{CV}(\sum_{\boldsymbol{c}_q\in\mathcal{C}} \sum_{p\in P(\boldsymbol{c}_q,\ell)} p)^2, \quad (9)$$

where p represents the node-wise probability in the batch.

The final training objective combines these balance terms with the main reasoning task:

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda_1(\mathcal{L}_l + \mathcal{L}_p) + \lambda_2\mathcal{L}_{\text{load}}, \quad (10)$$

where λ_1, λ_2 are hand-tuned scaling factors.

The full algorithm of MoKGR is shown in Algorithm 1. For each layer's message passing, we first compute the selected pruning experts and their corresponding weights $\boldsymbol{Q}^\ell(\boldsymbol{c}^\ell_v)$ in line 6. Then we obtain the final set of preserved entities $\mathcal{V}^\ell_\phi$ by combining the selected experts in line 7. Subsequently, we perform message passing only on the preserved entity set in line 8. When our message passing reaches layer $L_{\min}$ as shown in line 3, we first calculate the selected experts and their corresponding weights $\boldsymbol{Q}(\boldsymbol{c}_q)$ through the length expert gating

Algorithm 1 MoKGR Algorithm Analysis

Require: Number of length and pruning experts k_1 and k_2, range of paths length $[L_{\min}, L]$.

Ensure: Optimized GNN model parameters Θ and expert model parameters $\mathbb{W}$.

1: **while** not converged **do**
2: **for** Each batch of queries $\{(e_q, r_q, e_a)\}$ from $\mathcal{Q}_{tra}$, $\ell \in [1, L]$ **do**
3: **if** $\ell == L_{\min}$ **then**
4: Compute context representation $\boldsymbol{c}_q$;
5: From $\ell \in [L_{\min}, L]$ select Top-k_1 length experts via $\boldsymbol{Q}(\boldsymbol{c}_q)$;
6: Select Top-k_2 active pruning experts via $\boldsymbol{Q}^{\ell}(\boldsymbol{c}_v^{\ell})$;
7: Union selected experts to get entities $\mathcal{V}_{\phi}^{\ell}$;
8: Update entities in $\mathcal{V}_{\phi}^{\ell}$ via Eq. (1);
9: **if** ℓ is the selected length expert **then**
10: Update the entity scores $\Psi(e_a)$ for $e_a \in \mathcal{V}_{\phi}^{\ell}$;
11: **break if** early stopping condition meet: $g_b(\ell) = 0$;
12: Compute total loss $\mathcal{L}$ combining task and balance losses;
13: Update Θ and $\mathbb{W}$ using gradient of $\mathcal{L}$;
14: **return** Optimized parameters Θ, $\mathbb{W}$.

mechanism. In the subsequent layers $\ell \in [L_{\min}, L]$, if ℓ is a selected length expert, we compute and update the scores of entities selected by pruning experts at that layer in line 10. If our layer-wise binary gating function is activated, early stopping is performed in line 11. After message passing ends or early stopping is triggered, we use the highest-scoring candidate answer entity e_a from all candidates in $\Psi(e_a)$ as the final predicted answer entity.

4 Experiments

In this section, we conduct extensive experiments to answer the following research questions: (**RQ1** 4.2) How effective is MoKGR in improving reasoning performance and efficiency compared to existing methods? (**RQ2** 4.3) How does our expert selection mechanism perform? (**RQ3** 4.4) How does MoKGR achieve personalized path exploration in practice? (**RQ4** 4.5) How do different components and hyperparameters affect the model's performance?

4.1 Experimental Setup

We compares MoKGR with general KGs reasoning methods in both transductive and inductive settings. (The other implementation details and inductive setting is given in Appendix A.) We use filtered ranking-based metrics for evaluation, namely mean reciprocal ranking (MRR) and Hit@k. Higher values for these metrics indicate better performance.

Datasets. We use six widely used KGs reasoning benchmarks: Family (Kok and Domingos, 2007), UMLS (Kok and Domingos, 2007), WN18RR (Dettmers et al., 2017b), FB15k-237 (Toutanova and Chen, 2015), NELL-995 (Xiong et al., 2017), and YAGO3-10 (Suchanek et al., 2007).

Baselines. We compare the proposed MoKGR with (i) non-GNN methods: ConvE (Dettmers et al., 2017a), QuatE (Zhang et al., 2019a), RotatE (Sun et al., 2019), MINERVA (Das et al., 2017), DRUM (Sadeghian et al., 2019), AnyBURL (Meilicke et al., 2020), RNNLogic (Qu et al., 2021), RLogic (Cheng et al., 2022), DuASE (Li et al., 2024) and GraphRulRL (Mai et al., 2025); and (ii) GNN-based methods CompGCN (Vashishth et al., 2019), NBFNet (Zhu et al., 2021), RED-GNN (Zhang and Yao, 2022), A*Net (Zhu et al., 2023), Adaprop (Zhang et al., 2023d), ULTRA (Galkin et al., 2024) and one-shot-subgraph (Zhou et al., 2024).

4.2 Overall Performance (RQ1)

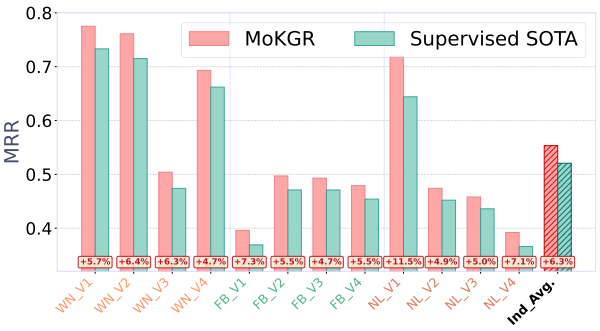
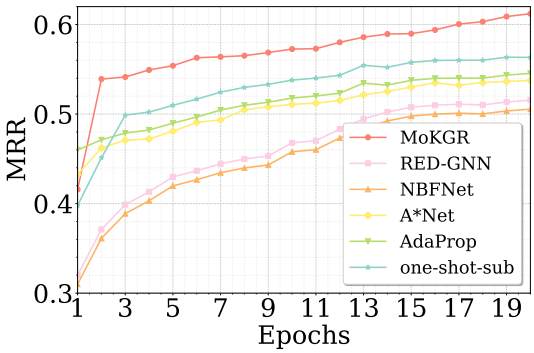


Figure 2: Comparison of MoKGR with supervised state-of-the-art baselines under the inductive setting.

Results. Tab. 1 and Fig. 2 show that our proposed MoKGR exhibits exceptional performance across all benchmark datasets in both transductive and inductive reasoning (Detail implementation and results for inductive setting are given in Appendix A.4). The experimental results validate several key advantages of our approach. First, the adoption of GNN-based message passing proves to be more effective than non-GNN methods for KGs reasoning, as evidenced by consistent perfor-

Type	Model	Family			UMLS			WN18RR			FB15k237			NELL-995			YAGO3-10		
		MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10
Non-GNN	ConvE	0.912	83.7	98.2	0.937	92.2	96.7	0.427	39.2	49.8	0.325	23.7	50.1	0.511	44.6	61.9	0.520	45.0	66.0
	QuatE	0.941	89.6	99.1	0.944	90.5	99.3	0.480	44.0	55.1	0.350	25.6	53.8	0.533	46.6	64.3	0.379	30.1	53.4
	RotatE	0.921	86.6	98.8	0.925	86.3	99.3	0.477	42.8	57.1	0.337	24.1	53.3	0.508	44.8	60.8	0.495	40.2	67.0
	MINERVA	0.885	82.5	96.1	0.825	72.8	96.8	0.448	41.3	51.3	0.293	21.7	45.6	0.513	41.3	63.7	-	-	-
	DRUM	0.934	88.1	99.6	0.813	67.4	97.6	0.486	42.5	58.6	0.343	25.5	51.6	0.532	46.0	66.2	0.531	45.3	67.6
	AnyBURL	0.861	87.4	89.2	0.828	68.9	95.8	0.471	44.1	55.2	0.301	20.9	47.3	0.398	27.6	45.4	0.542	47.7	67.3
	RNNLogic	0.881	85.7	90.7	0.842	77.2	96.5	0.483	44.6	55.8	0.344	25.2	53.0	0.416	36.3	47.8	0.554	50.9	62.2
	RLogic	-	-	-	-	-	-	0.477	44.3	53.7	0.310	20.3	50.1	0.416	25.2	50.4	0.36	25.2	50.4
	DuASE	0.861	81.2	90.8	0.845	72.5	85.5	0.489	44.8	56.9	0.329	23.5	51.9	0.423	37.2	59.2	0.473	38.7	62.8
	GraphRulRL	0.928	87.4	95.1	0.869	84.5	97.1	0.483	44.6	54.1	0.385	31.4	57.5	0.425	27.8	52.7	0.432	35.4	51.7
GNNs	CompGCN	0.933	88.3	99.1	0.927	86.7	99.4	0.479	44.3	54.6	0.355	26.4	53.5	0.463	38.3	59.6	0.421	39.2	57.7
	NBFNet	0.989	98.8	98.9	0.948	92.0	99.5	0.551	49.7	66.6	0.415	32.1	59.9	0.525	45.1	63.9	0.550	47.9	68.6
	RED-GNN	0.992	98.8	**99.7**	0.964	94.6	99.0	0.533	48.5	62.4	0.374	28.3	55.8	0.543	47.6	65.1	0.559	48.3	68.9
	A*Net	0.987	98.4	98.7	0.967	94.8	99.1	0.549	49.5	65.9	0.411	32.1	58.6	0.549	48.6	65.2	0.563	49.8	68.6
	AdaProp	0.988	98.6	99.0	0.969	95.6	99.5	0.562	49.9	67.1	0.417	33.1	58.5	0.554	49.3	65.5	0.573	51.0	68.5
	ULTRA	0.913	86.6	97.2	0.915	89.6	98.4	0.480	47.9	61.4	0.368	33.9	56.4	0.509	46.2	66.0	0.557	53.1	71.0
	one-shot-subgraph	0.988	98.7	99.0	0.972	95.5	99.4	0.567	51.4	66.6	0.304	22.3	45.4	0.547	48.5	65.1	0.606	54.0	72.1
	MoKGR	**0.993**	**99.1**	99.3	**0.978**	**96.5**	**99.6**	**0.611**	**53.9**	**70.2**	**0.443**	**36.8**	**60.7**	**0.584**	**50.7**	**67.9**	**0.657**	**57.7**	**75.8**

Table 1: Comparison of MoKGR with other KG reasoning methods in the transductive setting. Best performance is indicated by the **bold** face numbers, and the underline means the second best. "-" means unavailable results. "H@1" and "H@10" are short for Hit@1 and Hit@10 (in percentage), respectively.



(a) MRR over epochs

Min/Epoch	Training	Inference
MoKGR	111.73	**58.3**
RED-GNN	1382.9	802.2
NBFNet	493.8	291.4
A*Net	112.3	92.4
Adaprop	**108.6**	84.2
one-shot-subgraph	147.9	71.9

(b) Average time per epoch

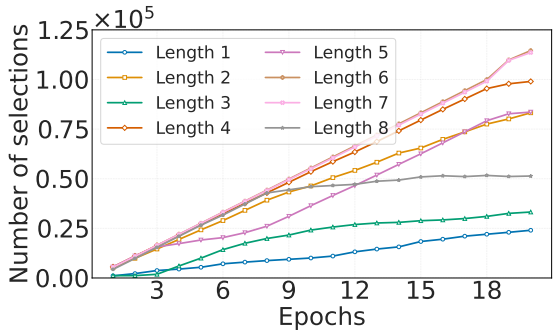
Figure 3: Comparison between MoKGR and current state-of-the-art methods in YAGO3-10 dataset.

mance improvements across all metrics. Furthermore, compared to full-exploration (Zhang and Yao, 2022; Zhu et al., 2021) and fixed pruning strategies (Zhu et al., 2023; Zhang et al., 2023d; Zhou et al., 2024), our MoE system achieves personalization in both pruning and reasoning path length, while implementing a mixture and personalized approach that significantly enhances reasoning accuracy. Notably, our method performs particularly well on the largest dataset YAGO3-10, solving the out-of-memory problem of full-exploration methods, and greatly improving the accuracy compared with other pruning methods.

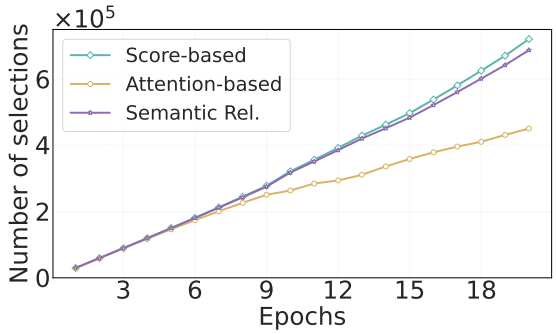
Learning Process Comparison. To comprehensively evaluate the effectiveness of MoKGR, we analyze a few SOTA methods on the YAGO3-10 dataset. We tracked both the MRR performance and computational time (training/inference) over 20 epochs. As shown in Fig. 3, MoKGR exhibits two distinctive advantages: 1) rapid convergence during early training phases, particularly evident in the steep curves within the first five epochs, and 2) stable performance growth throughout the training process. In contrast, other methods show slower convergence and more erratic progression, particularly in later epochs. The computational efficiency analysis presented in Table 3b demonstrates that MoKGR achieves significantly faster inference times compared to other approaches, while its training time is substantially lower than full-exploration methods and comparable to other pruning approaches.

4.3 Expert Selection Analysis (RQ2)

Selection Patterns of Length and Pruning Experts. To validate the necessity and effectiveness of our multi-expert approach, we conducted comprehensive experiments analyzing the selection patterns of both length experts and pruning experts on the YAGO3-10 dataset. The length expert selection patterns in Fig. 4a reveal that the model demonstrates a preference for mixing medium-length paths. For paths of length 8, we observe



(a) Length experts selection

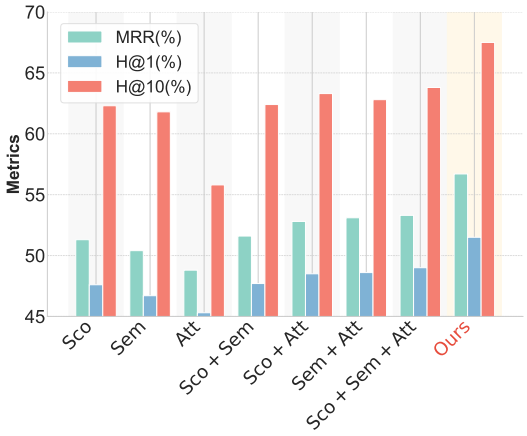


(b) Pruning experts selection

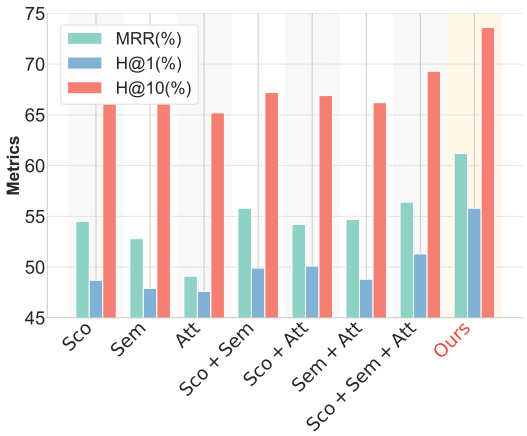
Figure 4: Selection Curves of Two Experts.

a turning point at Epoch 8, which we attribute to the effect of the gating function described in Section 3.2. This function suppresses the utilization of longer paths, causing the model to favor shorter path lengths. The expert selection pattern in pruning, illustrated in Fig. 4b, demonstrates the complementarity of expert selections. Despite variations in their selection frequencies, the overall trend maintains a steady upward progression.

Effectiveness of Adaptive Expert Weighting. We evaluate the effectiveness of adaptive weighting by comparing three approaches on WN18RR and YAGO3-10 datasets: single experts (weight 1.0), fixed-weight expert combinations (equal weights), and our adaptive weighting mechanism. Our experiments compare all possible expert combinations ($C_3^1+C_3^2+C_3^3$). The results shown in Fig. 5 demonstrate that compared to single experts, mixed experts perform better, reflecting the complementary nature of different pruning experts. Moreover, compared to fixed expert weights, our dynamic expert weighting shows superior performance, highlighting the necessity of weight personalization for pruning experts. These results validate that adaptive expert weighting is essential for optimal KGs reasoning, as it enables dynamic adjustment of pruning strategies based on query-specific requirements.



(a) WN18RR

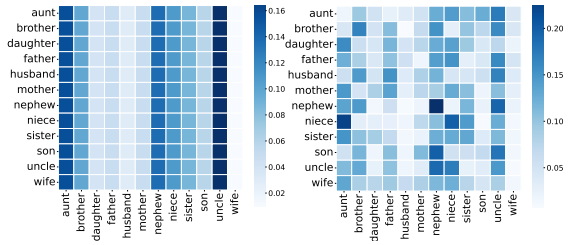


(b) YAGO3-10

Figure 5: Comparison of Pruning Selection Strategy.

4.4 Case Study (RQ3)

To validate MoKGR's personalized path exploration capabilities, we analyze the reasoning paths selected by the full-propagation method RED-GNN (which passes messages to all neighboring nodes at each propagation step) and compare them with those selected by MoKGR. In detail, for each relation r_i, we track its occurrence count $N_{r_i}^{\ell}$ at each path length ℓ, and calculate its aggregated importance N_{ri} by combining these counts with length experts' weights: $N_{r_i} = \sum_{\ell=L_{\min}}^{L} N_{r_i}^{\ell} \times g_q(\ell)$. The resulting normalized heatmaps reveal MoKGR's query-adaptive behavior. While RED-GNN show uniform distribution (Fig. 6a), MoKGR (Fig. 6b) identifies semantically relevant relations, such as emphasizing *aunt* for *niece* queries and *brother*, *nephew*, and *uncle* for *brother* queries. These case studies confirm that MoKGR effectively adapts its path selection to query semantics, highlighting relevant relations while avoiding redundant exploration, which behavior contrasts sharply with full-propagation methods. Further experiments for this analysis are provided in Appendix C.



(a) RED-GNN (b) MoKGR

Figure 6: Heatmaps of relation type ratios in the reasoning path on Family dataset. Rows represent different query relations, and columns correspond to relation types in the selected edges.

4.5 Ablation Study (RQ4)

Dataset		WN18RR			YAGO3-10		
Metrics		MRR	H@1	H@10	MRR	H@1	H@10
Ours		**0.611**	**53.9**	**70.2**	**0.657**	**57.7**	**75.8**
Balancing term	$\lambda_1 = 0$	0.558	50.1	66.9	0.582	53.6	71.7
	$\lambda_2 = 0$	0.560	50.6	67.0	0.596	54.1	72.4
Noise term	$\epsilon = 0$	0.560	50.8	67.1	0.586	53.7	68.3
	$\epsilon = 0.2$	0.561	50.8	67.2	0.602	55.1	70.3
Expert	$\phi_{\text{Sco}^\diamond}$	0.556	50.3	66.7	0.555	49.7	67.9
	$\phi_{\text{Att}^\diamond}$	0.549	49.9	66.4	0.501	48.6	66.2
	$\phi_{\text{Sem}^\diamond}$	0.553	50.2	66.5	0.538	48.9	67.2
	ϕ_{L^*}	0.552	50.5	66.5	0.569	51.0	68.4

Table 2: Ablation study on WN18RR and YAGO3-10 datasets. $\phi_{(.)^\diamond}$ means only customizing this pruning expert (with original length experts) and ϕ_{L^*} means only customizing L-th layer length expert (with original pruning experts).

To evaluate the contribution of each component in the MoKGR framework, we conducted a comprehensive ablation study on the WN18RR and YAGO3-10 datasets, as shown in Table 2. The results reveal several key findings: First, removing any balancing term (λ_1 or λ_2) leads to decreased performance, highlighting the importance of expert coordination. Furthermore, for the noise parameter ϵ given in Eq. (3), dynamic sampling ($\epsilon \sim \mathcal{N}(0, 1)$) consistently outperforms both no noise ($\epsilon = 0$) and fixed noise ($\epsilon = 0.2$) settings, demonstrating the benefits of incorporating controlled randomness in expert selection. Finally, whether in pruning strategies or path length strategies, restricting the model to one fixed strategy significantly weakens its reasoning capability.

5 Conclusion

In this paper, we introduced **MoKGR**, a novel framework that advances KGs reasoning through personalized path exploration. Our approach addresses two critical challenges in KGs reasoning: adaptive path length selection and comprehensive path evaluation. The adaptive length selection mechanism dynamically adjusts path exploration depths based on query complexity, while the mixture-of-pruning experts framework incorporates structural patterns, semantic relevance, and global importance to evaluate path quality. Through extensive experiments across diverse benchmark datasets, MoKGR demonstrates significant improvements in both reasoning accuracy and computational efficiency. The success of our personalized approach opens promising directions for future research, particularly in developing more sophisticated expert collaboration mechanisms and extending the framework to other graph learning tasks where path-based reasoning plays a crucial role. The framework's ability to balance thorough path exploration with computational efficiency makes it particularly valuable for large-scale knowledge graph applications.

6 Limitations

While MoKGR demonstrates significant improvements in knowledge graph reasoning performance, several limitations should be acknowledged: First, the computational complexity of our method, though significantly reduced compared to full-exploration approaches, still grows with the scale of knowledge graphs. For extremely large-scale knowledge graphs beyond those tested in our experiments, additional optimization techniques may be required. Moreover, while MoKGR shows strong empirical performance across diverse datasets, developing theoretical guarantees for the optimality of the selected paths remains challenging due to the complex interplay between different expert components. Finally, our evaluation focused primarily on standard knowledge graph benchmarks. Future work could explore the application of personalized path exploration in more specialized domains such as biomedical knowledge graphs or temporal knowledge graphs, which may exhibit different structural properties. These limitations present promising directions for future research to further enhance personalized path exploration in knowledge graph reasoning.

7 Acknowledgements

This work is supported by Guangdong Basic and Applied Basic Research Foundation 2025A1515010304, Guangzhou Science and Technology Planning Project 2025A03J4491.

References

Muhammad Ali, Abubakar Abid, and Parisa Kordjamshidi. 2022. Knowledge graphs: A comprehensive survey. *IEEE Transactions on Knowledge and Data Engineering*, 34(1):123–145.

Ke Cheng, Jie Liu, Wei Wang, and Yizhou Sun. 2022. Rlogic: Recursive logical rule learning from knowledge graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 179–189.

R. Das, S. Dhuliawala, M. Zaheer, L. Vilnis, I. Durugkar, A. Krishnamurthy, A. Smola, and A. McCallum. 2017. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *International Conference on Learning Representations (ICLR)*.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2017a. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2017b. Convolutional 2d knowledge graph embeddings. In *AAAI*.

Shimin Di, Quanming Yao, Yongqi Zhang, and Lei Chen. 2021. Efficient relation-aware scoring function search for knowledge graph embedding. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 1104–1115. IEEE.

Enjun Du, Xunkai Li, Tian Jin, Zhihan Zhang, Rong-Hua Li, and Guoren Wang. 2025a. Graphmaster: Automated graph synthesis via llm agents in data-limited environments. ArXiv preprint arXiv:2504.00711v2.

Enjun Du, Siyu Liu, and Yongqi Zhang. 2025b. GraphOracle: A foundation model for knowledge graph reasoning. *Preprint*, arXiv:2505.11125.

Charles Dugas, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia. 2001. Incorporating second-order functional knowledge for better option pricing. *Advances in Neural Information Processing Systems (NeurIPS)*, 13:472–478.

Mikhail Galkin, Xinyu Yuan, Hesham Mostafa, Jian Tang, and Zhaocheng Zhu. 2024. Towards foundation models for knowledge graph reasoning. In *Proceedings of the International Conference on Learning Representations (ICLR)*. Published as a conference paper at ICLR 2024.

Jinyang Huang, Xiachong Feng, Qiguang Chen, Hanjie Zhao, Zihui Cheng, Jiesong Bai, Jingxuan Zhou, Min Li, and Libo Qin. 2025. Mldebugging: Towards benchmarking code debugging across multi-library scenarios. *ACL Findings*.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations (ICLR)*.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S Yu. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE transactions on neural networks and learning systems*, 33(2):494–514.

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

Michael I Jordan and Robert A Jacobs. 1994. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6(2):181–214.

Stanley Kok and Pedro Domingos. 2007. Statistical predicate invention. In *ICML*, pages 433–440.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*.

Jiang Li, Xiangdong Su, Fujun Zhang, and Guanglai Gao. 2024. Learning low-dimensional multi-domain knowledge graph embedding via dual archimedean spirals. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1982–1994. Association for Computational Linguistics.

Zihao Li, Xu Wang, Yuzhe Yang, Ziyu Yao, Haoyi Xiong, and Mengnan Du. 2025. Feature extraction and steering for enhanced chain-of-thought reasoning in language models. *arXiv preprint arXiv:2505.15634*.

Ke Liang, Lingyuan Meng, Meng Liu, Yue Liu, Wenxuan Tu, Siwei Wang, Sihang Zhou, Xinwang Liu, Fuchun Sun, and Kunlun He. 2024. A survey of knowledge graph reasoning on graph types: Static, dynamic, and multi-modal. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Guangyi Liu, Yongqi Zhang, Yong Li, and Quanming Yao. 2025a. Dual reasoning: A gnn-llm collaborative framework for knowledge graph question answering. In *Proceedings of the 2nd Conference on Parsimony and Learning (CPAL)*. Proceedings Track, Poster.

Wanlong Liu, Junying Chen, Ke Ji, Li Zhou, Wenyu Chen, and Benyou Wang. 2024. Rag-instruct: Boosting llms with diverse retrieval-augmented instructions. *arXiv preprint arXiv:2501.00353*.

Wanlong Liu, Junxiao Xu, Fei Yu, Yukang Lin, Ke Ji, Wenyu Chen, Yan Xu, Yasheng Wang, Lifeng Shang, and Benyou Wang. 2025b. Qfft, question-free fine-tuning for adaptive reasoning. *arXiv preprint arXiv:2506.12860*.

Shengchao Mai, Shen Zheng, Yunhao Yang, and Hongxia Hu. 2021. Communicative message passing for inductive relation reasoning. In *Proceedings of*

the AAAI Conference on Artificial Intelligence, volume 35, pages 4294–4302.

Zhenzhen Mai, Wenjun Wang, Xueli Liu, Xiaoyang Feng, Jun Wang, and Wenzhi Fu. 2025. A reinforcement learning approach for graph rule learning. *Big Data Mining and Analytics*, 8(1):31–44.

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*.

Christian Meilicke, Mathias Fink, Yanjie Wang, Daniel Ruffinelli, Rainer Gemulla, and Heiner Stuckenschmidt. 2018. Fine-grained evaluation of rule- and embedding-based systems for knowledge graph completion. In *International Semantic Web Conference*, pages 3–20. Springer.

Rui Miao, Yixin Liu, Yili Wang, Xu Shen, Yue Tan, Yiwei Dai, Shirui Pan, and Xin Wang. 2025. Blindguard: Safeguarding llm-based multi-agent systems under unknown attacks. *arXiv preprint arXiv:2508.08127*.

Basil Mustafa, Carlos Riquelme, Joan Puigcerver, Rodolphe Jenatton, and Neil Houlsby. 2022. Multimodal contrastive learning with limoe: The language-image mixture of experts. *arXiv preprint arXiv:2206.02770*.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2015. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33.

Haiquan Qiu, Yongqi Zhang, Yong Li, and Quanming Yao. 2024. Understanding expressivity of gnn in rule learning. In *International Conference on Learning Representations (ICLR)*. Poster.

M. Qu, J. Chen, L. Xhonneux, Y. Bengio, and J. Tang. 2021. Rnnlogic: Learning logic rules for reasoning on knowledge graphs. In *International Conference on Learning Representations (ICLR)*.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. 2021. Scaling vision with sparse mixture of experts. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 8583–8595.

Amirmohammad Sadeghian, Mohammadreza Armandpour, Pasquale Ding, and D Wang. 2019. Drum: End-to-end differentiable rule mining on knowledge graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 15347–15357.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations (ICLR)*. Under review.

Xu Shen, Yixin Liu, Yiwei Dai, Yili Wang, Rui Miao, Yue Tan, Shirui Pan, and Xin Wang. 2025. Understanding the information propagation effects of communication topologies in llm-based multi-agent systems. *arXiv preprint arXiv:2505.23352*.

Zhang Siyue, Xue Yuxiang, Zhang Yiming, Wu Xiaobao, Luu Anh Tuan, and Zhao Chen. 2025. Mrag: A modular retrieval framework for time-sensitive question answering. *Preprint*, arXiv:2412.15540.

Fabian Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A core of semantic knowledge. In *The WebConf*, pages 697–706.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations (ICLR)*.

Zhiqing Sun, Chao Huang, Jianyu Chen, Qianhan Wang, Xiang Wang, Yiyang Li, and Liqiang Nie. 2021. Rotate: Knowledge graph embedding by relational rotations. In *Advances in Neural Information Processing Systems*, pages 10057–10068.

K. K Teru, E. Denis, and W. L Hamilton. 2020. Inductive relation prediction by subgraph reasoning. *arXiv preprint arXiv:1911.06962*.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *PWCVSMC*, pages 57–66.

Shikhar Vashishth, Soumya Sanyal, V. Nitin, and Partha Talukdar. 2019. Composition-based multi-relational graph convolutional networks.

Haotao Wang, Ziyu Jiang, Yuning You, Yan Han, Gaowen Liu, Jayanth Srinivasa, Ramana Rao Kompella, and Zhangyang Wang. 2023a. Graph mixture of experts: Learning on large-scale graphs with explicit diversity modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*. The first two authors contributed equally.

Li Wang, Xiaohui Yan, and Yansong Feng. 2023b. Enhancing knowledge graph embeddings with graph neural networks. *Journal of Artificial Intelligence Research*, 68:789–805.

Xu Wang, Yan Hu, Wenyu Du, Reynold Cheng, Benyou Wang, and Difan Zou. 2025a. Towards understanding fine-tuning mechanisms of llms via circuit analysis. *arXiv preprint arXiv:2502.11812*.

Xu Wang, Zihao Li, Benyou Wang, Yan Hu, and Difan Zou. 2025b. Model unlearning via sparse autoencoder subspace guided projections. *Preprint*, arXiv:2505.24428.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 564–573.

Fan Yang, Zhilin Yang, and William W Cohen. 2017. Differentiable learning of logical rules for knowledge base reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2319–2328.

Wenhao Yu, Chenguang Zhu, Lianhui Qin, Zhihan Zhang, Tong Zhao, and Meng Jiang. 2022. Diversifying content generation for commonsense reasoning with mixture of knowledge graph experts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ling Yue, Yongqi Zhang, Quanming Yao, Yong Li, Xian Wu, Ziheng Zhang, Zhenxi Lin, and Yefeng Zheng. 2023. Relation-aware ensemble learning for knowledge graph embedding. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 16620–16631, Singapore. Association for Computational Linguistics.

Guibin Zhang, Xiangguo Sun, Yanwei Yue, Chonghe Jiang, Kun Wang, Tianlong Chen, and Shirui Pan. 2023a. Graph sparsification via mixture of graphs. *arXiv preprint*.

Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. 2019a. Quaternion knowledge graph embeddings. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Siyue Zhang, Anh Tuan Luu, and Chen Zhao. 2024. Syntqa: Synergistic table-based question answering via mixture of text-to-sql and e2e tqa. *Preprint*, arXiv:2409.16682.

Siyue Zhang, Yilun Zhao, Liyuan Geng, Arman Cohan, Anh Tuan Luu, and Chen Zhao. 2025. Diffusion vs. autoregressive language models: A text embedding perspective. *Preprint*, arXiv:2505.15045.

Yongqi Zhang, Quanming Yao, Wenyuan Dai, and Lei Chen. 2020. Autosf: Searching scoring functions for knowledge graph embedding. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 433–444. IEEE.

Yongqi Zhang, Quanming Yao, and James T. Kwok. 2023b. Bilinear scoring function search for knowledge graph learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):1458–1473.

Yongqi Zhang, Quanming Yao, Yingxia Shao, and Lei Chen. 2019b. Nscaching: Simple and efficient negative sampling for knowledge graph embedding. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 614–625, Macao, China. IEEE.

Yongqi Zhang, Quanming Yao, Ling Yue, Xian Wu, Ziheng Zhang, Zhenxi Lin, and Yefeng Zheng. 2023c. Emerging drug interaction prediction enabled by a flow-based graph neural network with biomedical network. *Nature Computational Science*, 3(12):1023–1033.

Yongqi Zhang, Zhanke Zhou, Quanming Yao, and Yong Li. 2022. Efficient hyper-parameter search for knowledge graph embedding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2715–2735, Dublin, Ireland. Association for Computational Linguistics.

Yuning Zhang and Quanming Yao. 2022. Knowledge graph reasoning with relational directed graph. In *Proceedings of TheWebConf*.

Yuning Zhang, Zhen Zhou, Quanming Yao, Xia Chu, and Bo Han. 2023d. Adaprop: Learning adaptive propagation for knowledge graph reasoning. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*.

Leqi Zheng, Chaokun Wang, Canzhi Chen, Jiajun Zhang, Cheng Wu, Zixin Song, Shannan Yan, Ziyang Liu, and Hongwei Li. 2025a. LAGCL4rec: When LLMs activate interactions potential in graph contrastive learning for recommendation. In *The 2025 Conference on Empirical Methods in Natural Language Processing*.

Leqi Zheng, Chaokun Wang, Ziyang Liu, Canzhi Chen, Cheng Wu, and Hongwei Li. 2025b. Balancing self-presentation and self-hiding for exposure-aware recommendation based on graph contrastive learning. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '25, page 2027–2037, New York, NY, USA. Association for Computing Machinery.

Zhanke Zhou, Yongqi Zhang, Jiangchao Yao, Quanming Yao, and Bo Han. 2024. Less is more: One-shot-subgraph link prediction on large-scale knowledge graphs. In *International Conference on Learning Representations (ICLR)*.

Jinguo Zhu, Xizhou Zhu, Wenhai Wang, Xiaohua Wang, Hongsheng Li, Xiaogang Wang, and Jifeng Dai. 2022. Uni-perceiver-moe: Learning sparse generalist models with conditional moes. *arXiv preprint arXiv:2206.04674*.

Zhaocheng Zhu, Xinyu Yuan, Mikhail Galkin, Sophie Xhonneux, Ming Zhang, Maxime Gazeau, and Jian Tang. 2023. A*net: A scalable path-based reasoning approach for knowledge graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Ziniu Zhu, Zhaocheng Zhang, Louis Xhonneux, and Jian Tang. 2021. Neural bellman-ford networks: A general graph neural network framework for link prediction. In *Advances in Neural Information Processing Systems (NeurIPS)*.

A Experimental Detai and Supplementary Results

A.1 Training Details

All experiments were conducted on an NVIDIA A6000 GPU (48GB), with peak memory usage remaining under 45GB even for the largest datasets. Smaller datasets such as Family and UMLS can also be efficiently run on consumer GPUs like the 3060Ti (8GB). As for the hyper-parameters, we tune the $L_{\min}$ from 1 to $L-2$, the temperature value τ in (0.5, 2.5), the number of length experts k_1 in $(3, L - L_{\min})$ and set $k_2 = 2$, λ_1 in $(10^{-2}, 10^{-4})$ and λ_2 in $(10^{-3}, 10^{-5})$. The other hyperparameters are kept the same as Adaprop (Zhang et al., 2023d).

A.2 Length Distributions

distance	1	2	3	4	5	>5
WN18RR	34.9	9.3	21.5	7.5	8.9	17.9
FB15k237	0.0	73.4	25.8	0.2	0.1	0.5
NELL-995	40.9	17.2	36.5	2.5	1.3	1.6
YAGO3-10	56.0	12.9	30.1	0.5	0.1	0.4

Table 3: Length distribution (%) of queries in $\mathcal{Q}_{tst}$

To validate our claim that shorter paths typically contain enough stable correct answer information, we tracked the four largest datasets in our selection and analyzed the length distribution of shortest paths connecting e_q and e_a in each query. The results, converted to percentages, are shown in Table 3. We observe that for the vast majority of datasets, paths of length 3 already contain most of the information, while paths of length 5 encompass almost all information. This substantiates our argument that answer entities are typically in close proximity to query entities, making the introduction of excessive path lengths usually unnecessary in knowledge graph reasoning. Therefore, it is important to encourage the model to explore shorter paths preferentially.

A.3 Statistics of Datasets

Dataset	#Entity	#relation	$\lvert\mathcal{E}\rvert$	$\lvert Q_{\text{tra}}\rvert$	$\lvert Q_{\text{val}}\rvert$	$\lvert Q_{\text{tst}}\rvert$
Family	3.0k	12	23.4k	5.9k	2.0k	2.8k
UMLS	135	46	5.3k	1.3k	569	633
WN18RR	40.9k	11	65.1k	21.7k	3.0k	3.1k
FB15k237	14.5k	237	204.1k	68.0k	17.5k	20.4k
NELL-995	74.5k	200	112.2k	37.4k	543	2.8k
YAGO3-10	123.1k	37	809.2k	269.7k	5.0k	5.0k

Table 4: Statistics of the transductive KGs datasets. Q_{tra}, Q_{val}, Q_{tst} are the query triplets used for reasoning.

Version	Split	#relations	#nodes	#links
WN18RR_V1	train	9	2746	6678
	test	9	922	1991
WN18RR_V2	train	10	6954	18968
	test	10	2923	4863
WN18RR_V3	train	11	12078	32150
	test	11	5084	7470
WN18RR_V4	train	9	3861	9842
	test	9	7208	15157
FB15k-237_V1	train	183	2000	5226
	test	146	1500	2404
FB15k-237_V2	train	203	3000	12085
	test	176	2000	5092
FB15k-237_V3	train	218	4000	22394
	test	187	3000	9137
FB15k-237_V4	train	222	5000	33916
	test	204	3500	14554
NELL-995_V1	train	14	10915	5540
	test	14	225	1034
NELL-995_V2	train	88	2564	10109
	test	79	4937	5521
NELL-995_V3	train	142	4647	20117
	test	122	4921	9668
NELL-995_V4	train	77	2092	9289
	test	61	3294	8520

Table 5: Statistics of the Inductive KGs datasets.

We evaluate our model on six widely-used knowledge graph datasets of varying scales, their specific data parameters are shown in Table 4 and Table 5. Here $\mathcal{E}$ represents the edge set of the KG. Following the previous GNN-based knowledge graph reasoning method, we add an inverse relationship to each triple. Specifically, if $(e_x, r, e_y) \in \mathcal{E}$, we add an inverse relationship so that $(e_y, r, e_x) \in \mathcal{E}$.

A.4 Implementation Details for Inductive Setting

Inductive reasoning emphasizes the importance of drawing inferences about unseen entities, i.e., those not directly observed during the learning phase. As an illustrative example, consider a scenario where Fig. 1a reveals Jack's most eagerly anticipated movie. In this context, inductive reasoning could be employed to predict Mary's most desired cinematic experience. This methodology necessitates that the model captures semantic information and localized evidence while simultaneously discounting the specific identities of the entities under consideration.

Datasets. Following the approach outlined in (Teru et al., 2020), we utilize the same subsets of the WN18RR, FB15k237, and NELL-995 datasets. Specifically, we will work with 4 versions of each

	models	WN18RR V1	WN18RR V2	WN18RR V3	WN18RR V4	FB15k-237 V1	FB15k-237 V2	FB15k-237 V3	FB15k-237 V4	NELL-995 V1	NELL-995 V2	NELL-995 V3	NELL-995 V4
MRR	RuleN	0.668	0.645	0.368	0.624	0.363	0.433	0.439	0.429	0.615	0.385	0.381	0.333
	Neural LP	0.649	0.635	0.361	0.628	0.325	0.389	0.400	0.396	0.610	0.361	0.367	0.261
	DRUM	0.666	0.646	0.380	0.627	0.333	0.395	0.402	0.410	0.628	0.365	0.375	0.273
	GraIL	0.627	0.625	0.323	0.553	0.279	0.276	0.251	0.227	0.481	0.297	0.322	0.262
	CoMPILE	0.577	0.578	0.308	0.548	0.287	0.276	0.262	0.213	0.330	0.248	0.319	0.229
	NBFNet	0.684	0.652	0.425	0.604	0.307	0.369	0.331	0.305	0.584	0.410	0.425	0.287
	RED-GNN	0.701	0.690	0.427	0.651	0.369	0.469	0.445	0.442	0.637	0.419	0.436	0.363
	AdaProp	0.733	0.715	0.474	0.662	0.310	0.471	0.471	0.454	0.644	0.452	0.435	0.366
	MoKGR	**0.775**	**0.761**	**0.504**	**0.693**	**0.396**	**0.497**	**0.493**	**0.479**	**0.718**	**0.474**	**0.458**	**0.392**
Hit@1 (%)	RuleN	63.5	61.1	34.7	59.2	**30.9**	34.7	34.5	33.8	54.5	30.4	30.3	24.8
	Neural LP	59.2	57.5	30.4	58.3	24.3	28.6	30.9	28.9	50.0	24.9	26.7	13.7
	DRUM	61.3	59.5	33.0	58.6	24.7	28.4	30.8	30.9	50.0	27.1	26.2	16.3
	GraIL	55.4	54.2	27.8	44.3	20.5	20.2	16.5	14.3	42.5	19.9	22.4	15.3
	CoMPILE	47.3	48.5	25.8	47.3	20.8	17.8	16.6	13.4	10.5	15.6	22.6	15.9
	NBFNet	59.2	57.5	30.4	57.4	19.0	22.9	20.6	18.5	50.0	27.1	26.2	23.3
	RED-GNN	65.3	63.3	36.8	60.6	30.2	**38.1**	35.1	34.0	52.5	31.9	34.5	25.9
	AdaProp	66.8	64.2	39.6	61.1	19.1	37.2	37.7	35.3	52.2	34.4	33.7	24.7
	MoKGR	**66.9**	**67.8**	**40.0**	**62.3**	23.1	**38.1**	**38.9**	**36.4**	**64.4**	**35.8**	**35.2**	**27.3**
Hit@10 (%)	RuleN	73.0	69.4	40.7	68.1	44.6	59.9	60.0	60.5	76.0	51.4	53.1	48.4
	Neural LP	77.2	74.9	47.6	70.6	46.8	58.6	57.1	59.3	87.1	56.4	57.6	53.9
	DRUM	77.7	74.7	47.7	70.2	47.4	59.5	57.1	59.3	87.3	54.0	57.7	53.1
	GraIL	76.0	77.6	40.9	68.7	42.9	42.4	42.4	38.9	56.5	49.6	51.8	50.6
	CoMPILE	74.7	74.3	40.6	67.0	43.9	45.7	44.9	35.8	57.5	44.6	51.5	42.1
	NBFNet	82.7	79.9	56.3	70.2	51.7	63.9	58.8	55.9	79.5	63.5	60.6	59.1
	RED-GNN	79.9	78.0	52.4	72.1	48.3	62.9	60.3	62.1	86.6	60.1	59.4	55.6
	AdaProp	86.6	83.6	62.6	75.5	55.1	65.9	63.7	63.8	88.6	65.2	61.8	60.7
	MoKGR	**87.1**	**94.1**	**63.5**	**76.6**	**56.0**	**66.6**	**64.2**	**64.3**	**89.2**	**66.1**	**64.5**	**62.0**

Table 6: Comparison of MoKGR with other reasoning methods in the inductive setting. Best performance is indicated by the **bold** face numbers, and the underline means the second best.

dataset, resulting in a total of 12 subsets. Each of these 12 subsets has a different split between the training and test sets (Due to page limitations, we abbreviate WN18RR, FB15k-237 and NELL-995 as WN, FB and NL respectively in the pictures in Fig 2).

Baselines. Given that the training and test sets of the datasets contain disjoint sets of entities, methods that require entity embeddings, such as ConvE and CompGCN, cannot be applied in this context. Consequently, for non-GNN-based methods, we will compare our proposed AdaProp approach against non-GNN-methods that learn rules without the need for entity embeddings; we also selected some GNN based models for comparison. The final baselines are RuleN (Meilicke et al., 2018), NeuralLP (Yang et al., 2017), DRUM (Sadeghian et al., 2019), GraIL (Teru et al., 2020), CoMPILE (Mai et al., 2021), NBFNet (Zhu et al., 2021), RED-GNN (Zhang and Yao, 2022) and AdaProp (Zhang et al., 2023d).

Results. As demonstrated in Table 6, our proposed MoKGR framework exhibits exceptional performance across all evaluation metrics. This further validates the effectiveness of our mixture-of-experts model in inductive reasoning settings.

B Additional Theoretical Details

B.1 Path Encoding Process

In this subsection, we provide a detailed description of the path encoding process for GNN-based path reasoning methods, focusing on how message passing is performed for reasoning paths in the knowledge graph. Given a knowledge graph $\mathcal{K} = (\mathcal{V}, \mathcal{R}, \mathcal{F})$ with entity set $\mathcal{V}$, relation set $\mathcal{R}$, and fact triple set $\mathcal{F}$, we aim to encode all the paths between the query entity e_q and potential answer entity e_a for reasoning the query triple $(e_q, r_q, ?)$. The path encoding process consists of three main components: representation initialization, iterative message propagation, and score computation.

Representation Initialization. As introduced in Preliminary, let $q = (e_q, r_q)$ denote $(e_q, r_q, ?)$. For

each entity pair (e_q, e_y), we initialize their pair-wise representation at layer 0 as: $\boldsymbol{h}^0_{e_y|q} = 0$.

Message Propagation. The path representation is recursively computed through L layers of message propagation. At layer ℓ, for each edge $(e_x, r, e_y) \in \mathcal{F}$, the message function first combines the path information from the previous layer:

$$\boldsymbol{m}^{\ell}_{e_y|q} = \texttt{MESSAGE}(\boldsymbol{h}^{\ell}_{e_x|q}, \boldsymbol{w}_q(e_x, r, e_y)), \quad (11)$$

where $\boldsymbol{w}_q(e_x, r, e_y)$ is the learnable edge representation for edge $e = (e_x, r, e_y)$.

Furthermore, we define the MESSAGE transfer formula according to the RED-GNN (Zhang and Yao, 2022) method as follows:

$$\begin{aligned}&\texttt{MESSAGE}(\boldsymbol{h}^{\ell-1}_{e_x|q}, \boldsymbol{w}_q(e_x, r, e_y)) \\ &\quad = \alpha^{\ell}_q(e_x, r, e_y)\{+, *, \circ\}(\boldsymbol{h}^{\ell-1}_{e_x|q}, \boldsymbol{w}_q(e_x, r, e_y)),\end{aligned} \quad (12)$$

where $\alpha^{\ell}_q(e_x, r, e_y)$ is the attention weight calculated as:

$$\alpha^{\ell}_q(e_x, r, e_y) = \sigma\left((\boldsymbol{w}^{\ell}_{\alpha})^{\top}\text{ReLU}\left(\boldsymbol{W}^{\ell}_{\alpha} \cdot (\boldsymbol{h}^{\ell-1}_{e_x|q} \| \boldsymbol{w}^{\ell}_{r} \| \boldsymbol{w}^{\ell}_{r_q})\right)\right). \quad (13)$$

In this formulation, $\boldsymbol{w}^{\ell}_{r}$ and $\boldsymbol{w}^{\ell}_{r_q}$ represent the relation representation and query relation representation in the ℓ-th layer, respectively. $\boldsymbol{w}^{\ell}_{\alpha} \in \mathbb{R}^{d_\alpha}$ and $\boldsymbol{W}^{\ell}_{\alpha} \in \mathbb{R}^{d_\alpha \times 3d}$ are learnable parameters that enable the attention mechanism to adapt to different structural patterns.

Then, for each entity pair (e_q, e_a), we aggregate messages from all incoming edges of e_a to update its pair-wise representation:

$$\boldsymbol{h}^{\ell}_{e_y|q} = \texttt{AGGREGATE}(\boldsymbol{h}^{\ell-1}_{e_x|q} \cup \{\boldsymbol{m}^{\ell}_{e_x,r,e_y} | (e_x, r, e_y) \in \mathcal{N}(e_y)\}), \quad (14)$$

where $\mathcal{N}_e(e_y)$ denotes the neighboring edge of e_y. We specify the AGGREGATE function to be sum, mean, or max.

Overall Path Encoding. After L layers of the above message propagation, we obtain the final pair-wise representation $\boldsymbol{h}^{L}_{r_q}(e_q, e_a)$ for each entity pair (e_q, e_a). This collect all paths of up to length L connecting e_q to e_a under query relation r_q, thus encoding the reasoning paths from the query entity e_q to any candidate e_a. The pair-wise representation can then be used for downstream scoring functions, such as

$$s_L(q, e_a) = \boldsymbol{w}_L^{\top}\, \boldsymbol{h}^{L}_{e_a|q}, \quad (15)$$

where $\boldsymbol{w}_L$ is a trainable parameter vector. By comparing $s_L(q, e_a)$ among different candidate entities $e_a \in \mathcal{V}$, the model predicts which entity is the correct answer for the query $(e_q, r_q, ?)$.

Overall, this recursive path encoding process effectively captures the structural and query-relevant information from multiple-length paths, leveraging dynamic message passing steps that incorporate the query relation to focus on the most relevant edges and intermediates for knowledge graph reasoning.

B.2 Supplementary theoretical analysis of mixture of length Experts

B.2.1 Design details of c_q

The design of query context representation $\boldsymbol{c}_q$ is motivated by the need to capture both structural patterns and semantic information in knowledge graph reasoning. We construct $\boldsymbol{c}_q$ by combining two essential components:

First, $\boldsymbol{h}^{L_{\min}}_{r_q}(e_q, e_q)$ encodes the local structural information around the query entity e_q within the minimum path length $L_{\min}$. This term captures how the query entity connects to its neighborhood, providing crucial information about the local graph topology that can guide length selection. By using the self-loop representation (e_q to e_q), we ensure that the structural encoding is centered on the query entity's perspective. Second, $\boldsymbol{h}_{r_q}$ represents the learnable embedding of the query relation, which encodes the semantic requirements of the reasoning task. Different relations may require different reasoning depths - for instance, direct relations like *spouse_of* typically need shorter paths than indirect relations like *colleague_of_friend*. Including $\boldsymbol{h}_{r_q}$ allows the model to adapt its length selection based on the semantic nature of the query relation.

The combination of these components through an MLP enables non-linear interaction between structural and semantic information:

$$\boldsymbol{c}_q = \text{MLP}(\underbrace{\boldsymbol{h}^{L_{\min}}_{r_q}(e_q, e_q)}_{\text{structural info}} \| \underbrace{\boldsymbol{h}_{r_q}}_{\text{semantic info}}) \in \mathbb{R}^d. \quad (16)$$

This design principle ensures that length selection is informed by both the local graph structure around e_q and the semantic requirements of relation r_q, enabling more effective personalization of path exploration strategies.

B.2.2 Design details of gating function

As illustrated in Appendix A.2, we prove through the analysis of experimental datasets that the answer to the query entity is generally near its neighborhood and does not involve a very long path

length. Therefore, we designed a layer-wise binary gating function to control the model to tend to choose a smaller number of layers.

Specifically, Let $\mathcal{V}^\ell$ be the set of entities e_x reachable from e_q within ℓ steps, we collect all the pair-wise representations that can be reached within ℓ steps from e_q to obtain the distribution characteristics of the paired path representations of length L in the system as: $\mathbf{H}^\ell_{r_q}(e_q) = [\boldsymbol{h}^L_{r_q}(e_q, e_x)]_{e_x \in \mathcal{V}^\ell} \in \mathbb{R}^{|\mathcal{V}^\ell| \times d}$. During training, we employ a differentiable statistics-based gating function that evaluates path quality based on layer-wise feature distributions: $g_\ell = \text{GumbelSigmoid}(\text{MLP}([\mu_\ell, \sigma_\ell], \tau))$, where μ_ℓ and σ_ℓ capture the distribution of representation matrix $\mathbf{H}^\ell_{r_q}(e_q)$ along paths of length ℓ. By incorporating the Gumbel-Sigmoid transformation, we introduce a natural bias towards shorter paths while maintaining differentiability:

$$\text{GumbelSigmoid}(\boldsymbol{x}, \tau) = \sigma\big((\boldsymbol{x} + \text{GumbelNoise})/\tau\big). \tag{17}$$

The added Gumbel noise and sigmoid activation create a statistical tendency to favor paths with lower length counts, as shorter paths typically contain enough stable correct answer information. During inference, we further strengthen this preference through a deterministic truncation strategy:

$$g_\ell = \begin{cases} 0, & \text{if } |\text{CV}_\ell| > T \text{ and } \ell \geq L/2 \\ 1, & \text{otherwise} \end{cases} \tag{18}$$

where $\text{CV}_\ell = \sigma_\ell / \mu_\ell$ represents the coefficient of variation of the representation matrix $\mathbf{H}^\ell_{r_q}(e_q)$ at length ℓ, and T is a predefined threshold controlling the aggressiveness of path truncation. This length control mechanism defined as: $\boldsymbol{h}^\ell_{new} \leftarrow g_l \cdot \boldsymbol{h}^\ell$ (If $g_l = 0$, stop computing) enables the model to systematically prefer shorter paths when they provide sufficient evidence for reasoning.

B.3 Mixture of Pruning Experts Implementation Supplement

Mixture of Experts. The MoE framework consists of multiple specialized expert models and a gating mechanism that dynamically selects appropriate experts. Formally, given input x, the output of an MoE system can be written as: $y = \sum_{i=1}^n g_i(x) o_i(\boldsymbol{x})$, where $\boldsymbol{o}_i(\boldsymbol{x})$ is the output of the i-th expert, and $g_i(\boldsymbol{x}) \in \mathcal{G}(x)$ ($\sum_{i=1}^n g_i(\boldsymbol{x}) = 1$) is the gating weight that determines the contribution of each expert. Following (Shazeer et al., 2017), we employ a noise-enhanced gating mechanism where expert selection is computed as:

$$\mathcal{G}(\boldsymbol{x}) = \text{Softmax}\big(\text{TopK}_k(Q(\boldsymbol{x}) + \epsilon \cdot \text{Softplus}(\boldsymbol{x}\boldsymbol{W}_n))/\tau\big), \tag{19}$$

where $Q(\boldsymbol{x})$ is the score for total experts, τ is a temperature parameter, $\epsilon \sim \mathcal{N}(0, 1)$ is the Gaussian noise that encourages diverse expert selection and $\boldsymbol{W}_n$ is trainable parameter that learn noise scores.

B.3.1 Design details of c^ℓ_v

As defined in Appendix B.2.1, the query context representation $\boldsymbol{c}^\ell_v$ is defined as:

$$\boldsymbol{c}^\ell_v = \text{MLP}(\boldsymbol{h}^{\ell-1}_{r_q}(e_q, e_a) \| \boldsymbol{h}_{r_q}) \in \mathbb{R}^d. \tag{20}$$

The main difference between our pruning expert context $\boldsymbol{c}^\ell_v$ and the length expert context is that the pair-wise path representation $\boldsymbol{h}^{\ell-1}_{r_q}(e_q, e_a)$ in the pruning expert context changes with the path length ℓ. This is because the path length expert only needs to calculate once when the path reaches $L_{\min}$, while we need to calculate and apply the pruning expert at each length ℓ.

B.3.2 Entity score update after pruning

After we get the compatibility of each pruning expert at length ℓ with $\boldsymbol{Q}^\ell(\boldsymbol{c}^\ell_v)$, the gating function $\mathcal{G}^\ell(\boldsymbol{c}^\ell_v)$ is calculated as defined in Eq. (4). We argue that we cannot directly define the retained entities based on their original scores, because some entities are retained by only one pruning expert, while others may be retained by multiple pruning experts simultaneously. Moreover, even for entities that are retained by different pruning experts separately, their scores should be influenced by the weights $g^\ell_{\phi_i} \in \mathcal{G}^\ell(c^\ell_v)$ assigned to those experts.

Subsequently, we update the scores of selected entities through the gating weights $g^\ell_{\phi_i} \in \mathcal{G}^\ell(c_q)$ of the chosen pruning experts by:

$$s'_l(q, e_a) = \begin{cases} \sum_{i=1}^{k_2} g^\ell_{\phi_i} \cdot (s_l(q, e_a)), & e_a \in \mathcal{V}^\ell_\phi \\ 0, & e_a \notin \mathcal{V}^\ell_\phi \end{cases} \tag{21}$$

Finally, the score of an candidate answer entity e_a at length ℓ will be updated as:

$$s_l(q, e_a) \leftarrow s'_l(q, e_a). \tag{22}$$

And the updated scores will be used as the score function used in Eq. (5).

B.3.3 Path exploration strategy based on incremental sampling

To discover new entities at each layer while preserving those selected in previous layers, we adopt

an incremental sampling approach that builds upon our message passing framework. Specifically, let $\mathcal{P}^{\ell-1}$ denote the set of paths retained up to layer $(\ell-1)$ and $e_l \notin \mathcal{P}^{\ell-1}$. We update the path set as:

$$\mathcal{P}^{\ell} = \mathcal{P}^{\ell-1} \cup \{\big(e_{l-1},\, r_l,\, e_l\big) | e_l \in \mathcal{V}_{\phi}^{\ell}\}, \tag{23}$$

where e_l is the neighboring entities of e_{l-1}. By preserving previously discovered paths and selectively adding new entities at each layer, this incremental process refines the path length set up to L, expanding coverage of relevant entities for reasoning while maintaining the consistency of entities retained by experts in the previous layers.

B.3.4 The comprehensiveness of three pruning experts

The design of our three pruning experts — Scoring Pruning Expert, Attention Pruning Expert, and Semantic Pruning Expert — constitutes a comprehensive evaluation framework that directly addresses the limitations identified in Section 1. These experts operate synergistically to provide complementary perspectives in path evaluation:

- The Scoring Pruning Expert evaluates the global contribution of entities to the reasoning task, capturing high-level importance patterns across the knowledge graph.
- The Attention Pruning Expert focuses on local structural patterns by analyzing relation combinations and topological features, effectively identifying meaningful reasoning chains while filtering out irrelevant paths.
- The Semantic Pruning Expert assesses the thematic coherence between entities and query relations, ensuring selected paths maintain semantic relevance to the reasoning context.

Our extensive experimental results validate that this three-expert design achieves an optimal balance between evaluation coverage and pruning effectiveness. The experts work in concert to identify high-quality reasoning paths (e.g., *followed→singed*) while effectively filtering out spurious combinations (e.g., *is_friend_with→directed_in*). The complementary nature of these experts — operating across global importance, local structure, and semantic relevance — creates a robust evaluation framework that comprehensively covers the key aspects of path assessment in knowledge graph reasoning. This thorough coverage makes additional pruning experts not only unnecessary but potentially counterproductive, as they would increase computational overhead without providing substantively new evaluation criteria.

B.4 Sampling Number Function Design

To effectively control path exploration at different depths, we propose an adaptive sampling strategy through below functions:

$$K^{\ell} = \begin{cases} K_s + (K_h - K_s) \cdot \sigma(a \cdot (l - l_i/2)), & \ell < l_i \\ K_l + (K_h - K_l) \cdot (1 - \sigma(a \cdot (l - 3l_i/2))), & \ell \geq l_i \end{cases} \tag{24}$$

where σ is the sigmoid function and a controls the steepness of the transition. This design addresses several key challenges in path exploration. When using a uniform sampling formula, two critical issues emerge at different stages of exploration: First, in the initial sampling phase, the number of neighbor entities $|e_0|$ may be smaller than the predetermined sampling number K^0, making it impossible to achieve the target sampling quantity. Given that the neighborhood size $|\mathcal{N}_n(e_l)|$ typically grows exponentially with layer depth L, restricting the sampling number based on e_0 would result in missing many important paths at deeper layers.

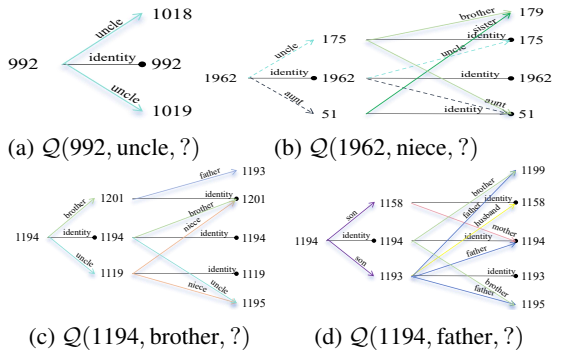
However, we cannot simply increase the sampling number K^{ℓ} indefinitely with path length, as the proportion of noise in the paths tends to increase with depth. This necessitates the introduction of an inflection point layer l_i. Once this inflection point is reached, the sampling number should gradually decrease with increasing layer depth to control noise accumulation.

Our formula incorporates three crucial parameters: initial sampling K_s, maximum sampling K_h, and minimum sampling K_l. This design accommodates the initial neighborhood size $|\mathcal{V}^0|$ while using the maximum and minimum sampling thresholds to dynamically control path retention at different layers. In the early stages ($\ell < l_i$), the sampling number gradually increases from K_s toward K_h, allowing for broader exploration. Beyond the inflection point ($\ell \geq l_i$), it decreases from K_h toward K_l, focusing on the most relevant paths. The sigmoid function ensures smooth transitions between these phases, while parameter a allows fine-tuning of the transition rate.

This adaptive sampling strategy enables more effective personalized path exploration by balancing the need for comprehensive coverage in early

layers with focused path selection in deeper layers, while maintaining robustness to varying neighborhood sizes across different queries.

C Supplementary Case Study



(a) $\mathcal{Q}(992, \text{uncle}, ?)$ (b) $\mathcal{Q}(1962, \text{niece}, ?)$

(c) $\mathcal{Q}(1194, \text{brother}, ?)$ (d) $\mathcal{Q}(1194, \text{father}, ?)$

Figure 7: Visualization of the transmission path on the family dataset, with dotted lines representing reverse relationships.

To validate our hypothesis that different queries require varying reasoning path lengths, we present two illustrative examples from the Family dataset. For $\mathcal{Q}(992, \text{uncle}, ?)$ (Fig. 7a), the desired answer entity emerges at the first length, while for the query $\mathcal{Q}(1962, \text{niece}, ?)$ (Fig. 7b), the correct answer entity does not appear until the second length. This demonstrates that a single-length exploration is insufficient for certain queries, and multi-length exploration can adapt better and save extra computational cost. Meanwhile, MoKGR also maintain semantic awareness, which is further evidenced in Fig. 7c and 7d, where for entity *1194*, MoKGR selects kinship-appropriate relations: *brother* and *uncle* for brother queries, and *son* and *brother* for father queries, thereby improving both reasoning accuracy and efficiency.

D Additional Implementation Details

D.1 Personalized PageRank

In this subsection, we will introduce an auxiliary method — the principle of PPR (Personalized PageRank). On some ultra-large-scale datasets, even after implementing pruning, the computational cost remains enormous since pruning at each layer requires calculations for all entities in that layer. Therefore, we explored whether we could implement pre-pruning functionality that could filter out some less important nodes in advance for ultra-large-scale datasets without trainable parameters (as these would increase computational costs) through simple rules. Based on this, we discovered that Personalized PageRank, an algorithm based on random walks, could effectively accomplish this task. Thus, for ultra-large datasets such as YAGO3-10, we performed preliminary subgraph extraction using PPR in advance. Algorithm 2 demonstrates the complete algorithmic workflow after activating PPR.

D.1.1 PPR Sampling methods

PPR is commonly implemented through the power iteration method, which can be formulated as

$$\pi = \alpha \boldsymbol{\pi} \boldsymbol{P} + (1 - \alpha)\boldsymbol{v}, \tag{25}$$

where π represents the PPR vector, P denotes the row-normalized adjacency matrix, v is the personalization vector with the initial node having value 1 and others 0, and $\alpha \in (0, 1)$ is the damping factor (typically set to 0.85). The iteration continues until convergence or reaching a maximum number of steps. This approach effectively captures the probability distribution of random walks with restart, where at each step, the walk either continues to a neighboring node with probability α or teleports back to the initial node with probability $(1 - \alpha)$. The resulting PPR scores indicate the relative importance of nodes with respect to the initial node, making it particularly useful for local graph analysis and node ranking tasks. Our experiments demonstrate that PPR-based preliminary exploration significantly enhances the efficiency and effectiveness of path-based reasoning. Without such preliminary filtering, large-scale KGs reasoning often encounters memory constraints due to the exponential growth of potential paths. The computational complexity of PPR's random walk-based approach is substantially lower than that of GNN operations, making it an efficient choice for initial path exploration. Furthermore, the paths preserved through PPR's preliminary exploration help subsequent message passing and pruning mechanisms focus on truly relevant reasoning paths while filtering out noise.

PPR Calculation We compute and cache global PPR scores for all entities in the knowledge graph to capture their overall importance and connectivity patterns. The PPR score for entity v is defined as:

$$\boldsymbol{\pi}_{e_v}(v) = \alpha \cdot \mathbf{e}_v + (1 - \alpha) \cdot \mathbf{P}^\top \boldsymbol{\pi}_{e_v}(v), \tag{26}$$

where $\mathbf{e}_v$ is the indicator vector for entity v, and $\mathbf{P}$ is the transition probability matrix of the graph.

For efficient computation on large-scale graphs, we implement a GPU-accelerated iterative method that approximates $\boldsymbol{\pi}_v$ for all entities until convergence or reaching a maximum iteration limit.

Query-Specific Path Exploration For each query entity e_q in a batch, we evaluate each entity e_v's importance by aggregating its PPR scores across all query entities:

$$\text{score}(e_v) = \sum_{e_q \in \text{batch}} \boldsymbol{\pi}_{e_q}(e_v), \tag{27}$$

where $\boldsymbol{\pi}_{e_q}(v)$ represents the personalized PageRank score of entity e_v when using e_q as the starting node.

Based on these scores, we identify promising paths by selecting entities in descending order of importance until reaching a predefined exploration scope. All subsequent reasoning is performed primarily along these preliminarily identified paths. While we explore different path sets for distinct queries, the PPR scores are pre-computed only once, ensuring computational efficiency while maintaining personalization for each query.

D.2 Loss function calculation supplement

Experts Balance Loss To ensure balanced and effective path exploration, we introduce several regularization terms that prevent the model from overly relying on specific exploration strategies or experts. This addresses the potential "winner takes all" problem (Lepikhin et al., 2020) where a single expert might dominate the path exploration process. Let $\mathcal{C}$ denote the set of query context vectors in the current batch, thus $\mathcal{C}_q$ and $\mathcal{C}_v$ contains all $\boldsymbol{c}_q$ and $\boldsymbol{c}_v$ in the batch respectively.

First, we introduce importance loss $\mathcal{L}_{\text{importance}}(\mathcal{C})$ as length-level balance loss $\mathcal{L}_l(\mathcal{C}_q)$ and pruning $\mathcal{L}_p(\mathcal{C}_v)$ balance loss to encourage diverse path lengths and balanced pruning strategy utilization:

$$\begin{aligned} \text{Importance}(\mathcal{C}) &= \sum_{\boldsymbol{c} \in \mathcal{C}} \sum_{g \in \mathcal{G}(\boldsymbol{c})} g, \\ \mathcal{L}_{\text{Importance}}(\mathcal{C}) &= \text{CV}(\text{Importance}(\mathcal{C}))^2, \end{aligned} \tag{28}$$

where $g \in \mathcal{G}(\boldsymbol{c})$ is the output of experts' gating mechanism calculated as Eq. (4), $\text{CV}(\boldsymbol{X}) = \sigma(\boldsymbol{X})/\mu(\boldsymbol{X})$ represents the coefficient of variation of input $\boldsymbol{X}$. The importance loss hence measures the variation of importance scores, enforcing all experts to be "similarly important". While the importance score enforces equal scoring among the experts, there may still be disparities in the load assigned to different experts. To address this, we additionally introduce a load balancing penalty for length experts to prevent overloading of specific experts. Specifically, let $P(\boldsymbol{c}_q, \ell)$ denote the probability that length-ℓ expert is selected (i.e., $g_q(\ell) \neq 0$: $P(\boldsymbol{c}_{q_i}, \ell) = \Pr(Q(\boldsymbol{c}_q)_\ell > \text{kth\_ex}(\boldsymbol{Q}(\boldsymbol{c}_q), k, \ell))$ where $\text{kth\_ex}(\cdot)$ returns the k-th largest expert score excluding the expert itself.

We give a simplified solution to this formula as:

$$P(\boldsymbol{c}_q, \ell) = \Phi\left(\frac{\boldsymbol{c}_q \boldsymbol{W}_g - \text{kth\_ex}(\boldsymbol{Q}(\boldsymbol{c}_q), k, \ell)}{\text{Softplus}(\boldsymbol{c}_q \boldsymbol{W}_n)}\right), \tag{29}$$

where $W_g \in \mathbb{R}^{d \times H}, W_n \in \mathbb{R}^{d \times H}$ are learnable weights and Φ is the CDF of standard normal distribution.

The length load balance loss is then defined as:

$$\mathcal{L}_{\text{load}}(\mathcal{C}) = \text{CV}(\sum_{\boldsymbol{c}_q \in \mathcal{C}} \sum_{p \in P(\boldsymbol{c}_q, \ell)} p)^2, \tag{30}$$

where p is the node-wise probability in the batch.

E Complexity Analysis

Let $|\mathcal{V}^\ell|$ denote the number of entities and $|\mathcal{E}^\ell|$ denote the number of edges between entities at length $\ell - 1$ and ℓ in the knowledge graph. Traditional GNN-based methods like NBFNet require $O(\sum_{\ell=1}^{L} |\mathcal{V}^\ell| \cdot |\mathcal{E}^\ell|)$ operations to process all paths up to length L. In contrast, MoKGR reduces the computational cost through two key mechanisms: (1) The layer-wise binary gating function enables early stopping of unnecessary path explorations, reducing the effective path length from L to an adaptive length L_a ($L_a \leq L$); (2) The mixture of pruning experts first evaluates all entities with complexity $O(|\mathcal{V}^\ell|)$ at each layer ℓ, and then retains only K^ℓ most promising entities where $K^\ell \ll |\mathcal{V}^\ell|$ and $|\mathcal{E}_{K^\ell}| \ll |\mathcal{E}^\ell|$. Consequently, MoKGR achieves an overall operations of $O(\sum_{\ell=1}^{\mathcal{L}}(|\mathcal{V}^\ell| \cdot k_2 + K^\ell \cdot |\mathcal{E}_{K^\ell}|))$, where $|\mathcal{V}^\ell| \cdot k_2$ denotes the number of operations caused by the retained pruning expert calculation at length ℓ, which remains substantially more efficient than traditional methods for large-scale knowledge graphs since $K^\ell \cdot |\mathcal{E}_{K^\ell}| \ll |\mathcal{V}^\ell| \cdot |\mathcal{E}^\ell|$ and $k_2 \ll |\mathcal{E}^\ell|$.

F Theoretical Analysis

In this appendix, we present a theoretical analysis of the MoKGR framework, including convergence

guarantees, optimality of path selection, preservation properties of pruning mechanisms, and formal complexity bounds.

F.1 Convergence Properties of MoKGR

Theorem 1 (Convergence of MoKGR)**.** *Under appropriate conditions on the expert selection probabilities and learning rates, the MoKGR algorithm converges to a local optimum of the loss function.*

Proof. Let's define the loss function for MoKGR as:

$$L = L_{task} + \lambda_1(L_l + L_p) + \lambda_2 L_{load} \tag{31}$$

Where L_{task} is the task-specific loss, L_l and L_p are the length and pruning expert importance losses, and L_{load} is the load balancing loss.

The gradient descent update for the parameters Θ of the model at iteration t is:

$$\Theta_{t+1} = \Theta_t - \eta_t \nabla_\Theta L(\Theta_t) \tag{32}$$

where η_t is the learning rate at iteration t.

For convergence, we need to show that:

1. The loss function L is bounded below.
2. The gradient $\nabla_\Theta L(\Theta_t)$ is Lipschitz continuous.
3. The learning rate satisfies $\sum_{t=1}^{\infty} \eta_t = \infty$ and $\sum_{t=1}^{\infty} \eta_t^2 < \infty$.

First, observe that L_{task} is bounded below by 0 (as it's a negative log-likelihood loss). The expert balance losses L_l, L_p, and L_{load} are all non-negative as they are based on squared coefficients of variation. Therefore, L is bounded below.

For Lipschitz continuity, the scoring function $\Psi(e_a) = \sum_{l \in A} g_q(l) \cdot s_l(q, e_a)$ is a linear combination of expert outputs, each of which is bounded and Lipschitz continuous due to the bounded nature of the message passing operations and the softmax gating function.

Given a decreasing learning rate schedule $\eta_t = \frac{\eta_0}{\sqrt{t}}$, we have:

$$\sum_{t=1}^{\infty} \eta_t = \eta_0 \sum_{t=1}^{\infty} \frac{1}{\sqrt{t}} = \infty \tag{33}$$

$$\sum_{t=1}^{\infty} \eta_t^2 = \eta_0^2 \sum_{t=1}^{\infty} \frac{1}{t} < \infty \tag{34}$$

Therefore, by the convergence theorem for stochastic gradient descent with Lipschitz continuous gradients, the algorithm converges to a local optimum of the loss function. □

F.2 Optimality of Adaptive Path Length Selection

Theorem 2 (Optimality of Path Length Selection)**.** *The adaptive path length selection mechanism in MoKGR minimizes the expected reasoning error given a computational budget constraint.*

Proof. Let $E(l, q)$ be the expected reasoning error when using paths of length up to l for query q. Let $C(l)$ be the computational cost of exploring paths of length l.

The problem can be formulated as:

$$\min_{\{w_l\}_{l=L_{min}}^{L}} \sum_{q \in Q} \sum_{l=L_{min}}^{L} w_l(q) E(l, q) \tag{35}$$

$$\text{subject to} \sum_{q \in Q} \sum_{l=L_{min}}^{L} w_l(q) C(l) \leq B \tag{36}$$

$$\sum_{l=L_{min}}^{L} w_l(q) = 1, \forall q \in Q \tag{37}$$

$$w_l(q) \geq 0, \forall l, q \tag{38}$$

where $w_l(q)$ is the weight assigned to path length l for query q, and B is the computational budget.

The adaptive length selection mechanism in MoKGR computes weights as:

$$g_q(l) = \frac{\exp([Q(c_q)]_l / \tau)}{\sum_{l' \in A} \exp([Q(c_q)]_{l'} / \tau)} \tag{39}$$

where $[Q(c_q)]_l$ is the compatibility score between query q and path length l.

The key insight is that the compatibility score $[Q(c_q)]_l$ learns to correlate with the negative expected error $-E(l, q)$ through training. This occurs because queries that benefit more from specific path lengths will have higher accuracy when those lengths are selected, leading to lower task loss.

The noise term $\epsilon \cdot \text{Softplus}(W_n c_q)$ enables exploration of different length combinations, allowing the model to discover the optimal path length distribution for each query type.

The binary gating function $g_b(l)$ enforces the budget constraint by encouraging shorter paths when they provide sufficient evidence.

As training progresses, the model learns to assign higher weights to path lengths that minimize the expected error for each query while respecting the computational budget constraint.

Therefore, the adaptive path length selection mechanism converges to the optimal weighting that minimizes the expected reasoning error given the computational constraints. □

F.3 Preservation Properties of Pruning Mechanism

Theorem 3 (Preservation of Optimal Paths)**.** *Under certain conditions, the mixture of pruning experts ensures that the optimal reasoning path for answering a query is preserved with probability at least* $1 - \delta$.

Proof. Let $P(e_q, e_a)$ be the set of all paths connecting query entity e_q to potential answer entity e_a. Let $p^* \in P(e_q, e_a)$ be the optimal path that provides the strongest evidence for answering the query.

Let V_l be the set of entities at distance l from e_q, and let V_l^ϕ be the subset selected by the pruning mechanism. For the optimal path p^* to be preserved, all entities along p^* must be included in the selected subsets.

Let e_l^* be the entity at distance l along the optimal path p^*. We need to show that:

$$\Pr(e_l^* \in V_l^\phi) \geq 1 - \delta \tag{40}$$

for some small $\delta > 0$.

The MoKGR pruning mechanism selects entities based on the union of top-K_l entities according to different pruning experts:

$$V_l^\phi = \{\cup_{i \in \text{TopK}_{k_2}(Q^l(c_v^l))} V_l^{\phi_i} | V_l^{\phi_i} = \text{TopK}_{K_l}(\phi_l^i(e_a))\} \tag{41}$$

For entity e_l^* to be excluded from V_l^ϕ, it must be excluded by all selected pruning experts. The probability of this happening is:

$$\Pr(e_l^* \notin V_l^\phi) = \Pr\left(\bigcap_{i \in \text{TopK}_{k_2}(Q^l(c_v^l))} \{e_l^* \notin V_l^{\phi_i}\}\right) \tag{42}$$

Since our three pruning experts evaluate different aspects of path quality (scoring, attention, and semantic relevance), they are designed to be complementary. The optimal path p^* should score highly on at least one of these dimensions.

Let's denote by ρ_i the probability that entity e_l^* is not selected by pruning expert i. Then:

$$\Pr(e_l^* \notin V_l^\phi) \leq \prod_{i \in \text{TopK}_{k_2}(Q^l(c_v^l))} \rho_i \tag{43}$$

For the optimal path, at least one of the experts should rank e_l^* highly. Let's say that for the best-matched expert i^*, we have $\rho_{i^*} \leq \epsilon$ for some small $\epsilon > 0$.

Then:

$$\Pr(e_l^* \notin V_l^\phi) \leq \epsilon \cdot \prod_{i \in \text{TopK}_{k_2}(Q^l(c_v^l)), i \neq i^*} \rho_i \leq \epsilon \tag{44}$$

Therefore:

$$\Pr(e_l^* \in V_l^\phi) \geq 1 - \epsilon \tag{45}$$

By setting $\delta = L\epsilon$ where L is the maximum path length, and applying the union bound, we can show that the entire optimal path is preserved with probability at least $1 - \delta$.

This proves that the mixture of pruning experts preserves the optimal reasoning path with high probability. □

F.4 Information Theoretic Analysis of Adaptive Path Selection

Theorem 4 (Information Gain of Adaptive Path Selection)**.** *The adaptive path length selection mechanism in MoKGR maximizes the expected information gain about the answer entity while respecting computational constraints.*

Proof. Let $H(E_a|e_q, r_q)$ be the entropy of the answer entity distribution given query $(e_q, r_q, ?)$. Let $I(E_a; P_l|e_q, r_q)$ be the mutual information between the answer entity and paths of length l given the query.

The information gain from exploring paths of length l is:

$$IG(l) = H(E_a|e_q, r_q) - H(E_a|P_l, e_q, r_q) \tag{46}$$

$$= I(E_a; P_l|e_q, r_q) \tag{47}$$

The expected information gain from the adaptive path length selection is:

$$E[IG] = \sum_{l=L_{min}}^{L} g_q(l) \cdot I(E_a; P_l|e_q, r_q) \tag{48}$$

where $g_q(l)$ is the weight assigned to path length l for query $(e_q, r_q, ?)$.

The goal of the adaptive path length selection mechanism is to maximize this expected information gain subject to computational constraints:

$$\max_{g_q} \sum_{l=L_{min}}^{L} g_q(l) \cdot I(E_a; P_l|e_q, r_q) \tag{49}$$

$$\text{subject to} \sum_{l=L_{min}}^{L} g_q(l) \cdot C(l) \leq B \tag{50}$$

$$\sum_{l=L_{min}}^{L} g_q(l) = 1, g_q(l) \geq 0 \tag{51}$$

where $C(l)$ is the computational cost of exploring paths of length l, and B is the computational budget.

The compatibility score $[Q(c_q)]_l$ in MoKGR can be interpreted as an estimate of the information gain $I(E_a; P_l|e_q, r_q)$. By learning to assign higher weights to path lengths with higher information gain, MoKGR effectively solves the optimization problem (Huang et al., 2025; Zhang et al., 2025; Wang et al., 2025b).

The layer-wise binary gating function further enforces the computational constraint by stopping path exploration when the expected additional information gain does not justify the computational cost.

Therefore, the adaptive path length selection mechanism in MoKGR maximizes the expected information gain about the answer entity while respecting computational constraints. □

Algorithm 2 Training Process of MoKGR

Require: Parameters: Number of length and pruning experts k_1 and k_2, range of path lengths $[L_{min}, L]$.

Ensure: Optimized GNN model parameters Θ and experts model parameters $\mathbb{W}$.

// Pre-processing with PPR
Initialize PPR cache for all entities in $\mathcal{V}$
for $v \in \mathcal{V}$ **do**
  Compute PPR scores π_v and store in cache
// Training Loop
while not converged **do**
  Sample a batch of queries $\{(e_q, r_q, e_a)\}$ from $\mathcal{Q}_{tra}$
  for each query (e_q, r_q, e_a), $\ell \in [1, L]$ **do**
    // PPR-based subgraph construction
    $\mathcal{G}_{sub} \leftarrow \text{BuildSubgraph}(e_q, \text{PPRCache})$
    if $\ell == L_{\min}$ **then**
      // Length Expert Selection
      Compute context representation $\boldsymbol{c}_q$ and expert embedding $\boldsymbol{E}_1$, thus get the compatibility with experts via $\boldsymbol{Q}(\boldsymbol{c}_q) = \boldsymbol{E}_1\boldsymbol{c}_q + \epsilon \cdot \text{Softplus}(\boldsymbol{W}_n\boldsymbol{c}_q)$;
      Select Top-k_1 length experts from $\ell \in [L_{min}, L]$ via $\boldsymbol{Q}(\boldsymbol{c}_q)$ to get weights set $\mathcal{G}_q$;
    // Pruning Expert Selection at length ℓ
    Compute context $\boldsymbol{c}_v^\ell$ and expert embedding $\boldsymbol{E}_2$ for pruning;
    Select Top-k_2 pruning experts via $\boldsymbol{Q}^\ell(\boldsymbol{c}_v^\ell)$ to get weights $\mathcal{G}_v$;
    Combine selected experts to identify key entities: $\mathcal{V}_\phi^\ell = \{\cup_{i \in \text{TopK}_{k_2}(\boldsymbol{Q}^\ell(\boldsymbol{c}_v^\ell))} \mathcal{V}_{\phi_i}^\ell | \mathcal{V}_{\phi_i}^\ell = \text{TopK}_{K^\ell}(\phi_i^\ell(e_a))\}$;
    Update path representations for identified entities in $\mathcal{V}_\phi^\ell$;
    if ℓ is the selected length expert **then**
      Calculate entity scores $s_\ell(e_q, r_q, e_a)$ at current length;
      Update final scores $\Psi(e_a)$ by combining weighted length-specific scores;
      if early stopping condition met: $g_b(\ell) = 0$ **then**
        break
  // Parameter Updates
  Compute total loss $\mathcal{L}$ combining task and expert balance losses;
  Update model parameters Θ and expert parameters $\mathbb{W}$ using gradient of $\mathcal{L}$;
return Θ, $\mathbb{W}$.