ThoughtForest-KGQA: A Multi-Chain Tree Search for Knowledge Graph Reasoning

Anonymous Author(s)

Abstract

Most multi-hop Knowledge Graph Question Answering (KGQA) methods utilize fixed pruning strategies that, while efficient, critically impair the diversity of answer paths and fail to discover complex or less common correct answers.

To address these limitations, this paper introduces ThoughtForest-KGQA, a novel multi-chain tree search algorithm. The method employs a dual-level reinforcement learning framework where a local-level agent optimizes individual reasoning chains by capturing fine-grained semantic details in the knowledge graph. Concurrently, a global-level agent strategically coordinates the simultaneous exploration of multiple chains. Comprehensive evaluations conducted across two distinct KGQA benchmarks reveal that this approach identifies a broader spectrum of correct answers, setting a new state-of-the-art in the field.

CCS Concepts

• **Computing methodologies** → **Knowledge representation and reasoning**; *Neural networks*; *Reinforcement learning*.

Keywords

Knowledge Graph, Multi-hop Reasoning, Reinforcement Learning,

1 Introduction

Knowledge Graphs (KGs) have emerged as powerful and essential structures for organizing and representing vast amounts of real-world entities and their intricate relationships. As these repositories of structured information continue to expand in both scale and complexity, the imperative to effectively query and reason over them for non-trivial information becomes increasingly critical. The field of Knowledge Graph Question Answering (KGQA) directly addresses this challenge, seeking to provide users with precise, factual answers by navigating and inferring from the rich tapestry of connections inherent within KGs. This ability to move beyond simple fact retrieval towards more complex inferential reasoning is essential for unlocking the full potential of these large-scale knowledge bases.

A particularly challenging task within KGQA is multi-hop reasoning, where deriving an answer necessitates traversing multiple relational edges and synthesizing information across disparate parts of the graph[5]. While prevailing paradigms for multi-hop KGQA have demonstrated considerable success, they often rely on fixed pruning strategies[7, 11]to improve efficiency. These strategies, however, critically impair answer-path diversity and can fail to uncover complex or less common correct answers. Furthermore, many existing methods focus on reasoning along a single[4], ostensibly optimal path or rely on static entity scoring[1, 2], and the inferencing models employed demonstrate limitations in effectively processing multiple concurrent reasoning paths. This creates a significant bottleneck, preventing the full exploration of the rich information contained within the knowledge graph.

To overcome these challenges, this paper introduces ThoughtForest-KGQA, a novel multi-chain tree search algorithm designed to fully leverage the richness within multiple reasoning paths. The core of our novel perspective is a hierarchical reinforcement learning framework that diverges from traditional single-path methods by dynamically guiding subgraph exploration using beam information. Instead of relying on a single chain of reasoning as in Fig 1, our approach strategically coordinates the simultaneous exploration of multiple chains, allowing for a more comprehensive and robust search of the solution space. This method ensures that both broad, relational context and entity-specific details are considered when constructing answer paths.

We present the following contributions. First, We reformulate multi-hop KGQA problem as a multi-path subgraph exploration problem. Second, We propose a dual-level reinforcement learning framework that integrates a high-level graph policy for global relational selection and a low-level path policy for fine-grained entity navigation. Third, we demonstrate that our model outperforms baselines on two real-world datasets.

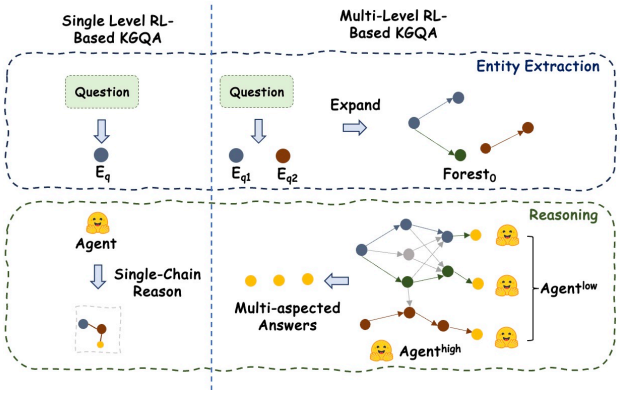


Figure 1: Comparative Frameworks for KGQA: Single-Level RL vs.ThoughtForest-KGQA

2 Related Work

This section briefly reviews prior work in Knowledge Graph Question Answering (KGQA), focusing on reasoning strategies and the role of reinforcement learning.

Reinforcement Learning in KGQA. Reinforcement learning (RL) is applied to KGQA by framing it as a sequential decision-making problem where agents are rewarded for discovering valid answer paths [13]. The pioneering DeepPath system modeled this as a Markov Decision Process (MDP), where an agent traverses the graph by selecting relations from its current entity [11]. However, a key limitation of these early RL methods is their focus on finding

a single reasoning path, which makes them suffer from limited exploration capacity, primarily due to fixed pruning strategies and reliance on static reward signals that fail to reflect intermediate progress[3, 12].

Reasoning Strategies. More recent KGQA methods have sought to overcome the limitations of single-chain RL by incorporating broader context. ReaRev [7] interleaves path exploration with dynamic revision of the search frontier based on partial feedback, allowing limited backtracking. R3-RAG [6] extends Retrieval Augmented Generation by integrating reasoning traces into retrieval, thus refining candidate subgraphs via iterative retrieval-reasoning loops. Although these advances better exploit intermediate evidence, they still maintain a single active thread of reasoning or require expensive subgraph reconstruction at each step.

3 Methodology

Aforementioned KGQA frameworks typically reasoning along a single, ostensibly optimal path, and the inferencing models employed demonstrate limitations in effectively processing multiple concurrent reasoning paths. To fully leverage the richness within multiple reasoning paths, we introduce ThoughtForest-KGQA, which employs dual-level policy networks, jointly trained through a hierarchical reinforcement learning framework to perform diversified traversal across relations and nodes based on the current context and reasoning history. This allows the agent to dynamically explore multiple promising paths simultaneously, rather than being limited to a single trajectory.

3.1 Definitions

KGQA. Knowledge Graph Question Answering (KGQA) aims to find accurate answers to natural language questions by reasoning over a structured knowledge graph (KG). Given a question q and a KG $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, where $\mathcal{E}$ is the set of entities, $\mathcal{R}$ is the set of relations, and $\mathcal{T} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is the set of triples, the task is to identify an answer set $\{e_i\} \subseteq \mathcal{E}$ that correctly answers q. Rather than committing to a single linear path through $\mathcal{G}$, we model the reasoning process as a search over a forest of partial paths, each of which may branch in different directions.

States. To support parallel exploration of multiple candidate chains, we employ a hierarchical state representation with two levels. At each time step t, we represent the high level state as $s_t^{\text{high}} = (\mathcal{F}_t, \Phi(q))$, providing a strategic, overall summarization of the ongoing multi-chain tree search, where $\mathcal{F}_t$ is the forest of current subgraph and $\Phi(q)$ is the fixed question embedding. A low-level state $s_t^{\text{low},(k)} = (v_t^{(k)}, \mathcal{P}_t^{(k)})$ facilitates tactical, fine-grained path construction, corresponding to a focused beam of promising candidate chains selected from the broader forest $\mathcal{F}_t$.

Actions. Corresponding to our hierarchical state representation, the action space in ThoughtForest-KGQA is likewise structured at two levels. At time t, the high-level policy picks relation $\{r_t\}$ for $\mathcal{F}_t$. For the low-level policy chain k, it must select the next entity $v_{t+1}^{(k)}$ among all neighbors reachable via $r_t^{(k)}$.

3.2 ThoughtForest-KGQA overview

In contrast to single-chain reasoning approaches[15] , a crucial aspect of our method is the combination of dual-level multi-chain tree search agent (see Figure 2b). Specifically, the overall score for a potential expansion from chain $\mathcal{P}_t^k$ via relation to entity is determined by the product of the score assigned to the relation by the high-level component and the score assigned to entity by the low-level component. This synergistic scoring ensures that both the relational context and the entity-specific fit are considered. Based on these combined product scores, the system selects the top K tuple expansions across all possibilities arising from the K active chains as show in Figure 2a. These chosen expansions form the new beam of K reasoning processes for the subsequent time step $t + 1$. We employ an LSTM to encode the path-specific history, enabling the model to capture sequential dependencies within each path. In the multi-chain setting, these LSTM modules operate in parallel, each managing a distinct candidate path. The modified thought forest $\mathcal{F}_{t+1}$, directly determines the next high-level state s_{t+1}^{high}. Subsequently, for each of the K newly advanced chains in the beam, its individual low-level state is updated to $s_{t+1}^{low(k)}$. This update reflects its new frontier node $v_{t+1}^{(k)}$ and its extended path history $\mathcal{P}_{t+1}^{(k)}$ within the context of the new global forest $\mathcal{F}_{t+1}$ and query.

3.3 Hierarchical Inference

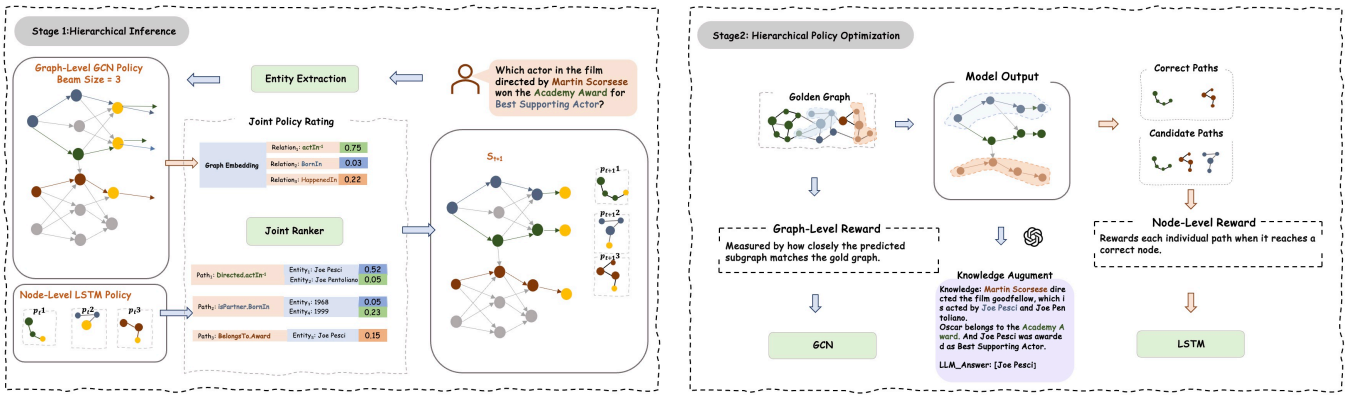
This section details the specific neural network architectures employed for the graph-level and node-level controllers, including their respective policy networks baselines, as well as the graph neural network (GNN) used for node representation.

The ThoughtForest-KGQA reasoning process commences with the interpretation of the input natural language query, q. This initial step involves identifying the primary topic entities mentioned in the query. The initial thought forest, $\mathcal{F}_0$, is then constructed based on these extracted components. In its simplest form, $\mathcal{F}_0$ can be initialized as a set of disconnected single-node graphs.

The graph-level policy network then acts as a strategic director, responsible for guiding the overall direction of subgraph extension by selecting promising relation types to explore. At each decision step t, to process the 'thought forest $\mathcal{F}_t$, we first employ a multi-layer Graph Convolutional Network (GCN) to generate rich, structure-aware embeddings for all nodes within $\mathcal{F}_t$. These node embeddings are then aggregated to produce a condensed vector representation, $h(\mathcal{F}_t)$, which is then passed through an MLP together with the query embedding to form a unified global summary. This contextualized summary z_t then calculate the similarity score between the embedding of candidate relations, and give the score. These scores are then normalized via softmax to yield the high-level policy

$$\pi^{high}(r \mid s_t^{high}) = \frac{exp(z_t^\top \Phi(r))}{\sum_{r' \in \mathcal{R}_{t+1}} exp(z_t^\top \Phi(r'))} \tag{1}$$

Following the strategic relational guidance provided by the graph-level policy, the node-level policy networks are responsible for the fine-grained, tactical selection of specific entities to extend each of the K active reasoning chains. Each chain $k \in \{1, \ldots, K\}$ is managed by its own instance of this node-level agent. At each time



(a) Stage 1: Hierachical Inference Process **(b) Stage 2: Hierarchical Policy Optimization**

Figure 2: RL Framework Overview for ThoughtForest-KGQA

step t, for a given active chain k, its state is primarily characterized by the hidden state $h_t^{(k)}$ of a Long Short-Term Memory (LSTM) network. This $h_t^{(k)}$ encapsulates the history of the k-th path $P_t^{(k)}$ traversed so far, including the sequence of entities and relations. The node-level policy then can be conceptualized as the softmax over these scores:

$$\pi^{low}(e \mid s_t^{low,(k)}) = \frac{exp((h_t^{(k)})^\top)\Phi(e)}{\sum_{e' \in Nbr(v_t^{(k)})} exp((h_t^{(k)})^\top)\Phi(e)} \tag{2}$$

where $\Phi(e)$ is the embedding of the candidate entity. Then we perform a joint beam search over all candidate (r, e) pairs. Across all K active chains and their candidate pairs, beam search retains the top-K combined actions given by the product of the graph-level and node-level policy probabilities. Each selected $(r_t^{(k)}, v_{t+1}^{(k)})$simultaneously advances the high-level forest to $\mathcal{F}_{t+1}$ and updates the low-level path to $P_{t+1}^{(k)}$.

3.4 Hierarchical Policy Optimization

In ThoughtForest-KGQA, we employ separate reward metrics tailored for the graph-level and node-level policies, reflecting their distinct roles within our hierarchical reinforcement learning framework. A key characteristic of our training strategy is the deferral of all Proximal Policy Optimization (PPO) updates to the terminal timestep of an episode. This approach is adopted due to the inherent sparsity of the high-level reward, which is only concretely realized upon episode completions.

The node-level policies receive immediate feedback to guide their tactical entity selections. Specifically, at each step t, if a node-level agent's chosen entity e_{t+1} corresponds to a ground-truth answer node $e_{t+1} \in \mathcal{E}^*$, an immediate binary reward $r_t^{low} = +1$ is issued. This dense reward structure directly incentivizes the LSTM-based node-level policies to navigate towards known correct entities. In contrast, the graph-level policy is guided by a more global, sparse signal related to the overall quality of the exploration. Upon episode termination at step T, we evaluate the final thought forest $\mathcal{F}_T$ against the golden subgraph $\mathcal{G}^*$ by its edge coverage C_T. We then convert this coverage into a terminal reward via $exp^{\alpha(C_T-1)}$ which encourages the graph-level policy to maximize subgraph completeness.

Both dense, stepwise node-level rewards and the sparse, end-of-episode graph-level reward are jointly leveraged within a unified PPO optimization procedure. Concretely, We first collect full episode trajectories under the current high-level and low-level policies, π^{high} and π^{low}, recording all actions, states, and rewards for each timestep. For every timestep t, we compute the cumulative discounted return for both levels:

$$\begin{cases} G_t^{\text{low}} = \sum_{k=t}^{T-1} \gamma_{\text{low}}^{k-t}\, r_k^{\text{low}}, \\ G_t^{\text{high}} = \gamma_{\text{high}}^{T-t}\, R_{\text{high}} \end{cases} \tag{3}$$

reflecting the terminal reward structure. The graph-level value network $V_\pi^{high}(s_t^{high})$ approximates this return, trained by minimizing

$$L_{val^{high}} = \sum_{t=0}^{T-1} (V_\pi{}^{high}(s_t^{high}) - G_t{}^{high}) \tag{4}$$

Given the terminal reward, the TD-error for GAE simplifies to $G_t^{\text{high}} - V_\pi^{\text{high}}(s_t^{high})$. We apply the standard PPO objective independently to both the graph- and node-level policies. Concretely, for each policy we optimize:

$$L_{\text{clip}} = -\sum_{t=0}^{T-1} \min\Big(\rho_t, \hat{A}t, ; \text{clip}\big(\rho_t, 1-\epsilon, 1+\epsilon\big), \hat{A}_t\Big) \tag{5}$$

where ρ_t is the ratio of new to old policy probabilities and $\hat{A}_t$ is the advantage estimate. We then add a value-function loss $cv\, L_{\text{val}}$ and an entropy bonus $c_{\text{ent}}\, L_{\text{ent}}$ to encourage exploration. In practice, this single PPO loss

$$L_{\text{PPO}} = L_{\text{clip}} + c_v\, L_{\text{val}} + c_{\text{ent}}\, L_{\text{ent}} \tag{6}$$

s applied independently to both the high-level and low-level policies.

The joint training procedure begins by iteratively collecting N full episode trajectories using the current policies. Following these rollouts, returns and advantages are calculated independently for

both hierarchical levels, effectively trainint both policy levels in tandem: the high-level policy learns from sparse, coverage-based terminal rewards, while the low-level policy is guided by denser, more immediate rewards. .

4 Experiments

To validate the effectiveness of ThoughtForest-KGQA, we conducted comprehensive experiments on widely-used KGQA benchmarks. Our evaluation was designed to answer two primary research questions:

(1) Does ThoughtFores-KGQA outperform existing state-of-the-art methods on established KGQA tasks?
(2) How does the dual-level reinforcement learning framework contribute to performance, particularly in terms of answer-path diversity and finding complex answers?

4.1 Experimental Settings

Datasets: Our primary experiments were conducted on two widely-used KGQA benchmarks: WebQuestionsSP (WebQSP) [9] and MetaQA (3-hop) [14]. WebQSP utilizes Freebase as its underlying knowledge graph, while MetaQA uses its own distinct KG.

Baselines: We compared ThoughtForest-KGQA against a suite of strong baseline methods, including GNN-based models, embedding-based approaches like EmbedKGQA[8], subgraph retrieval models like GraftNet[10], and other reinforcement learning agents such as DeepPath[11].

Metrics: Performance was evaluated using two standard metrics: **Hits@1**, which measures the accuracy of the top one candidate answers, **Hits@3**, which measures ground-truth answers appearing among the top three candidates and **F1 Score**, which provides a balanced measure of precision and recall.

4.2 Quantitative Results

The main performance comparison is presented in Table **??**. The results show that ThoughtForest-KGQA achieves a new state-of-the-art, demonstrating a significant performance improvement over all baseline methods across both datasets.

4.3 Case Study: Multi-Chain Reasoning

To illustrate how our dual-level framework enhances answer-path diversity, we can examine the hierarchical inferencing process depicted in Figure 2. For a complex question like, "Which actor in the film directed by Martin Scorsese won the Academy Award for Best Supporting Actor?", a single-path reasoning model might fail if it commits to an incorrect early step.

In contrast, ThoughtForest-KGQA initiates a multi-chain search. The graph-level policy strategically identifies promising relations to explore (e.g., `directed_by`, `won_award`), while multiple node-level agents explore these paths in parallel from the extracted entities ("Martin Scorsese," "Academy Award"). As shown in the "Model Output" stage of Figure 2, the framework maintains a "thought forest" of multiple candidate paths simultaneously. This ability to manage and evaluate several reasoning chains at once allows the model to uncover complex and less common answers that would be missed by fixed pruning or single-path strategies, directly addressing our second research question. "'

Table 1: Performance comparison on WebQSP and MetaQA-3hop. Best results for each metric are in bold. (– indicates not reported or not applicable; * typically similar to F1 if answer is single entity, may differ for set answers). Hits@3 values in *italics* are informed estimations.

Dataset	Method	Hits@1	F1	Hits@3
WebQSP	EmbedKGQA [8]	70.1	47.8	*79.5*
	GraftNet [10]	72.5	50.7	*81.0*
	DeepPath [11]	74.3	–	*82.3*
	Minerva [4]	68.5	–	*76.8*
	ReaRev [7]	**80.3**	70.9	76.5
	Ours	79.1	**72.2**	***85.8***
MetaQA-3Hop	EmbedKGQA	70.5	*68.0**	*81.2*
	GraftNet	78.0	*75.5*	*86.3*
	DeepPath	80.5	–	*89.0*
	Minerva	55.2	–	*67.5*
	ReaRev	86.5	84.2	*93.0*
	Ours	**88.1**	**96.2**	**98.1**

5 Conclusion

In this paper, we explored the task of multi-hop Knowledge Graph Question Answering (KGQA), focusing on addressing the limitations of fixed pruning strategies that typically restrict the diversity and depth of reasoning paths. To tackle this, we introduced ThoughtForest-KGQA, a novel hierarchical reinforcement learning framework utilizing dual-level policies for multi-chain tree search. Our graph-level policy strategically guides the exploration of multiple reasoning paths, while the node-level policies finely evaluate candidate entities at each step. Experimental results on two established KGQA benchmarks demonstrated that ThoughtForest-KGQA significantly surpasses baseline methods, achieving superior accuracy and answer-path diversity. A current limitation of our approach is that the node-level policy relies on immediate rewards directly tied to entity correctness, which may limit exploration in scenarios with sparse correct entities.

GenAI Usage Disclosure

We utilized DeepSeek-V3 for assistance with language polishing and LaTeX formatting of this manuscript. All core content, experimental results, and scientific claims were generated by the human authors. The authors critically reviewed and edited all AI-generated suggestions and take full responsibility for the final version of this paper.

References

[1] Medina Andresel, Trung-Kien Tran, Csaba Domokos, Pasquale Minervini, and Daria Stepanova. 2023. Combining inductive and deductive reasoning for query answering over incomplete knowledge graphs. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 15–24.
[2] Yushi Bai, Xin Lv, Juanzi Li, and Lei Hou. 2023. Answering complex logical queries on knowledge graphs via query computation tree optimization. In *International Conference on Machine Learning*. PMLR, 1472–1491.
[3] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*,

Vol. 38. 17682–17690.

[4] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2017. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *arXiv preprint arXiv:1711.05851* (2017).

[5] Ruosen Li, Zimu Wang, Son Tran, Lei Xia, and Xinya Du. 2024. MEQA: A Benchmark for Multi-hop Event-centric Question Answering with Explanations. *Advances in Neural Information Processing Systems* 37 (2024), 126835–126862.

[6] Yuan Li, Qi Luo, Xiaonan Li, Bufan Li, Qinyuan Cheng, Bo Wang, Yining Zheng, Yuxin Wang, Zhangyue Yin, and Xipeng Qiu. 2025. R3-RAG: Learning Step-by-Step Reasoning and Retrieval for LLMs via Reinforcement Learning. *arXiv preprint arXiv:2505.23794* (2025).

[7] Costas Mavromatis and George Karypis. 2022. Rearev: Adaptive reasoning for question answering over knowledge graphs. *arXiv preprint arXiv:2210.13650* (2022).

[8] Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the 58th annual meeting of the association for computational linguistics*. 4498–4507.

[9] Haitian Sun, Tania Bedrax-Weiss, and William W Cohen. 2019. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. *arXiv preprint arXiv:1904.09537* (2019).

[10] Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. *arXiv preprint arXiv:1809.00782* (2018).

[11] Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. *arXiv preprint arXiv:1707.06690* (2017).

[12] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems* 36 (2023), 11809–11822.

[13] Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. 2025. Does Reinforcement Learning Really Incentivize Reasoning Capacity in LLMs Beyond the Base Model? *arXiv preprint arXiv:2504.13837* (2025).

[14] Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[15] Anjie Zhu, Deqiang Ouyang, Shuang Liang, and Jie Shao. 2022. Step by step: A hierarchical framework for multi-hop knowledge graph reasoning with reinforcement learning. *Knowledge-Based Systems* 248 (2022), 108843.