HyperGraphPro: Progress-Aware Reinforcement Learning for Structure-Guided Hypergraph RAG

Anonymous ACL submission

Abstract

Multi-hop question answering requires retrieval agents to construct a connected evidence path across reasoning turns rather than retrieve isolated relevant evidence for a single query. However, in existing GraphRAG methods, retrieval and reinforcement learning are often optimized with mismatched signals: retrieval is driven by local semantic relevance, while outcome-level reinforcement learning assigns the reward by assessing the final answer without identifying which retrieval actions progress the reasoning path. We introduce HyperGraphPro, a progress-aware hypergraph RAG framework that focuses on evidence-path progress as the shared objective for retrieval and policy optimization. For retrieval, HyperGraphPro reranks candidate hyperedges using query-conditioned structural distinctiveness, combining entity-query relevance with a local-vs-global hypergraph incidence signal to favor entities that are informative in the current graph neighborhood. For policy optimization, HyperGraphPro replaces uniform trajectory-level reward assignment with stepwise progress-aware policy optimization, which assigns step-level rewards combining final answer reachability and inter-turn entity connectivity. Experiments on multi-hop question answering benchmarks demonstrate that HyperGraphPro consistently improves reasoning accuracy over existing GraphRAG methods.

1 Introduction

Retrieval-augmented generation (RAG) (Lewis et al., 2020; Jin et al., 2025; Song et al., 2025) has been widely used to improve the factuality of large language models (LLMs) by providing output generation with external evidence. However, conventional RAG approaches retrieve evidence as independent text chunks based on semantic similarity to the question. It is often insufficient for multi-hop question answering, where the answer is not dependent on a single locally relevant text chunk but on a sequence of connected evidence paths. For example, solving a compositional question may require iteratively identifying an intermediate entity, then following a relation to another entity to reach the final answer. So, the challenge is not only retrieving relevant facts, but also constructing a coherent reasoning path with the connected evidence.

Graph retrieval-augmented generation (GraphRAG) partially addresses this limitation by representing knowledge as entities and their higher-order relations rather than isolated knowledge chunks. Recent agentic GraphRAG methods (Luo et al., 2025a) further allow language models to iteratively generate retrieval queries, observe graph-structured evidence, and update their reasoning state.

However, current agentic GraphRAG approaches still face a mismatch between local relevance and reasoning progress. In retrieval, they largely rely on local contextual similarity, retrieving evidence that appears semantically relevant to the question, but may fail to capture the relational structure in the nature of graph structures. Furthermore, the retrieved evidence across turns may remain disconnected. This may cause the reasoning drift, where the agent provides plausible but unproductive evidence, accumulates redundant information, or misses bridging entities, that connect the question to the final answer. In reinforcement learning, the objectives in GraphRAG rely only on sparse final-answer rewards without explaining which intermediate retrieval step advanced the reasoning path and caused failure. As a result, useful bridge-retrieval actions may receive little credit, while spurious actions in otherwise successful trajectories may be reinforced.

To address these limitations, we propose **HyperGraphPro**, which aligns both retrieval and policy optimization with graph-structured progress. Our HyperGraphPro consists of two key components: structure-guided retrieval and progress-based step-

wise policy optimization. First, we introduce a **structure-guided hypergraph retrieval** mechanism, which jointly leverages semantic relevance and structural proximity in the knowledge graph to retrieve subgraphs that are not only contextually relevant but also structurally coherent. Second, we present a **progress-based stepwise policy optimization** strategy, which provides dense supervision over intermediate reasoning steps by assessing the action of each step according to its contribution to reasoning progress, rather than relying solely on the correctness of the final answer. Together, these components align both retrieval and policy learning with the compositional nature of graph-based reasoning.

We evaluate HyperGraphPro on 2WikiMultihopQA, HotpotQA, MuSiQue, and NQ using Qwen2.5-3B and Qwen2.5-7B backbones. HyperGraphPro consistently improves over the strongest evaluated baseline, Graph-R1, raising average F1 from 49.89 to 54.56 with Qwen2.5-3B and from 55.94 to 59.38 with Qwen2.5-7B. The improvements are largest on multi-hop benchmarks, where the agent must connect evidence across retrieval steps. Our comprehensive ablation studies further show that structure-guided retrieval and progress-aware policy optimization each improve performance and are complementary when combined.

Our contributions are threefold:

- We propose a **query-conditioned structural distinctiveness score** for knowledge reranking, combining entity-query relevance with local-vs-global hypergraph incidence to prioritize informative entities in the current graph neighborhood.
- We introduce a **progress-aware stepwise optimization**, a dense supervision strategy that assesses each retrieval action based on the contribution to reaching the final answer and inter-turn connectivity.
- We demonstrate that **HyperGraphPro** achieves the best performance among the baselines including GraphRAG and RL-based agentic GraphRAG on multi-hop question answering benchmarks.

2 Related Work

2.1 Graph Retrieval–Augmented Generation

Retrieval-Augmented Generation (RAG) (Lewis et al., 2020; Jiang et al., 2023; Trivedi et al., 2023) has been introduced to address hallucinated or factually incorrect responses generated by large language models. GraphRAG (Edge et al., 2024) has extended the RAG paradigm by leveraging knowledge graphs to provide richer context and improve complex reasoning. Building on GraphRAG, Hyper-RAG (Feng et al., 2026) and HypergraphRAG (Luo et al., 2025b) harness the power of hypergraphs, enabling the capture of higher-order relationships beyond simple pairwise links. In parallel, another line of work (Chen et al., 2026; Guo et al., 2025b; Gutiérrez et al., 2025; Wang et al., 2025; Xu et al., 2025) explores retrieval mechanisms that emphasize computational efficiency without sacrificing reasoning accuracy. More recently, Graph-R1 (Luo et al., 2025a) further extends GraphRAG by leveraging reinforcement learning to enhance adaptability and reasoning performance. Our HyperGraphPro presents structure-guided retriever that re-ranks candidate knowledge using query-conditioned structural distinctiveness.

2.2 Reinforcement Fine-Tuning for LLM

Reinforcement learning (RL), such as RLHF (Ouyang et al., 2022), has emerged as a key technique to boost LLM reasoning abilities, either by optimizing for outcome-based rewards or by using preference feedback. For example, DeepSeek-R1 (Guo et al., 2025a) is trained with Group Relative Policy Optimization (GRPO), which evaluates multiple outputs from the model to estimate relative advantages, thereby eliminating the need for a separate value model for Generalized Advantage Estimation (Schulman et al., 2015). In addition, several works have explored agentic reinforcement learning frameworks (Zheng et al., 2026; Feng et al., 2025; Zeng et al., 2025), enabling LLMs to autonomously decide when and how to invoke agents, thereby greatly improving their multi-turn interaction and tool-use capabilities. Motivated by the success of reinforcement fine-tuning approaches, Search-R1 (Jin et al., 2025) leverages multi-turn interleaved reasoning and search by dynamically retrieving external information, while R1-Searcher (Song et al., 2025) employs a two-stage, outcome-based reinforcement learning scheme that enables LLMs to invoke search engines during reasoning.

3 HyperGraphPro

We present HyperGraphPro, a progress-aware reinforcement learning framework that improves multi-step reasoning and retrieval over knowledge graphs. Our framework is motivated by a simple principle: in multi-hop QA, retrieval and policy learning should be optimized according to whether each turn advances a connected evidence path. We name this quantity *evidence-path progress*. A turn makes progress when it retrieves information that is both informative for reaching the answer and connected to the entities or evidence accumulated in previous turns. Concretely, we begin by describing the background of an RL-based GraphRAG framework, including group-relative policy optimization formulation (Guo et al., 2025a). Next, we introduce our structure-guided hypergraph retrieval mechanism, which exploits hypergraph structure to estimate entity distinctiveness and retrieve more informative evidence. Finally, we present our inter-turn entity connectivity-based step progress-aware policy optimization, which enables fine-grained credit assignment across reasoning steps and guides the LLM agent to iteratively refine its retrieval and reasoning trajectory toward more effective knowledge-grounded generation.

3.1 Outcome-Level Agentic GraphRAG

Reinforcement Learning (RL)-based GraphRAG frameworks, such as Graph-R1 (Luo et al., 2025a), aim to learn an output reasoning trajectory $\boldsymbol{y}$ that leads to a final answer for an input question $\boldsymbol{x}$ by interleaving language reasoning with retrieval over the structured external knowledge graph $\mathcal{G}_K$.

Context similarity-based graph retrieval. GraphRAG leverages knowledge graphs $\mathcal{G}_K$ to model the relational information inherent in the external knowledge. In this work, we use a hypergraph as external knowledge representation since it expresses n-ary relations among more than two entities while conventional graphs model a connection between two entities. Formally, a hypergraph is defined as $\mathcal{G}_K = (V, E)$, where $v \in V$ denotes entities and E denotes a set of edges. Generally, given the knowledge corpus $K = \{\boldsymbol{d}_1, \boldsymbol{d}_2 \ldots, \boldsymbol{d}_N\}$, a (hyper)graph is extracted from knowledge documents $\boldsymbol{d}$ through an LLM:

$$\mathcal{G}_K = (V, E)\,, \text{where } \{(e_i, \mathcal{V}_{e_i})\}_{i=1}^{m} \sim \pi_{\text{ext}}(\boldsymbol{d}) \tag{1}$$

where V, E are sets of nodes and hyperedges, respectively. π_{ext} is an LLM that is prompted to transform the knowledge segment $\boldsymbol{d}$ into a set of relation-entity pairs $\{(e_i, \mathcal{V}_{e_i})\}$, where e_i and $\mathcal{V}_{e_i} = \{v_1, \ldots, v_n\}$ denote the hyperedge and its participating entities, respectively. On the constructed knowledge graph, conventional methods (Jin et al., 2025) perform the retrieval based on their semantic similarity to the query representation. It can be formulated as:

$$\begin{aligned} \mathcal{R}_H(\boldsymbol{q}_t) &= \underset{e_i \in E}{\overset{k}{\operatorname{argmax}}} \Big(\operatorname{sim}(\phi(\boldsymbol{q}_t), \phi(e_i))\Big), \\ \mathcal{F}_t^{\text{ret}} &= \bigcup_{e_j \in \mathcal{R}_H} \{(e_j, V_{e_j}) \mid e_j \in E\}, \end{aligned} \tag{2}$$

where $\phi(\boldsymbol{q}_t)$ denotes the query embeddings, and $\phi(e_i)$ represents the hyperedge embedding. The top-k retrieved set $\mathcal{F}_t^{\text{ret}}$ forms the final knowledge set. Graph-R1 applies a hybrid hyperedge retrieval strategy where both entity and hyperedge similarity measurements are leveraged to retrieve hyperedges.

Group-Relative Policy Optimization. Group-Relative Policy Optimization (GRPO) (Shao et al., 2024; DeepSeek-AI, 2025) is one of the representative RL approaches with its strong performance and efficiency. We apply GRPO to train the agent, which is formulated as:

$$\begin{aligned} &\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{\left[\boldsymbol{x} \sim \mathcal{D}_X, \{\boldsymbol{y}^{(i)}\}_{i=1}^{N} \sim \pi_{\theta_{\text{old}}}(\cdot \mid \boldsymbol{x}; \mathcal{G}_K)\right]} \\ &\left[\frac{1}{|\boldsymbol{y}^{(i)}|} \sum_{t=1}^{|\boldsymbol{y}^{(i)}|} \min\left(\hat{r}_t^{(i)} \hat{A}^{(i)}, \operatorname{clip}\left(\hat{r}_t^{(i)}, 1-\epsilon, 1+\epsilon\right) \hat{A}^{(i)}\right)\right] \\ &- \beta \mathbb{D}_{\text{KL}}\left(\pi_\theta \,||\, \pi_{\text{ref}}\right), \end{aligned} \tag{3}$$

where $\hat{r}_t^{(i)} = \frac{\pi_\theta\left(y_t^{(i)} \mid \boldsymbol{x}, \boldsymbol{y}_{<t}^{(i)}; \mathcal{G}_K\right)}{\pi_{\text{old}}\left(y_t^{(i)} \mid \boldsymbol{x}, \boldsymbol{y}_{<t}^{(i)}; \mathcal{G}_K\right)}$ denotes the likelihood ratio between current π_θ and the old policy model π_{old}. The advantage is calculated as $\hat{A}^{(i)} = \frac{R\left(\boldsymbol{x}, \boldsymbol{y}^{(i)}\right) - \text{mean}\left(\left\{R\left(\boldsymbol{x}, \boldsymbol{y}^{(j)}\right)\right\}_{j=1}^{N}\right)}{F\left(\left\{R\left(\boldsymbol{x}, \boldsymbol{y}^{(j)}\right)\right\}_{j=1}^{N}\right)}$, where $F(\cdot)$ is the normalizer within the group $\left\{R\left(\boldsymbol{x}, \boldsymbol{y}^{(i)}\right)\right\}_{i=1}^{N}$. Most existing approaches apply outcome-supervised settings (Guo et al., 2025a), which assign the same reward at every token in each output $\boldsymbol{y}$ based on the sequence-level reward of the output $\boldsymbol{y}$.

3.2 Structure-Guided Retrieval for Evidence-Path Extension

We introduce a structure-guided hypergraph retrieval module that ranks hyperedges using both

semantic relevance and hypergraph structure. The goal of retrieval in HyperGraphPro is not only to find hyperedges that match the current query, but to select hyperedges that extend the current evidence path. A hyperedge is helpful when its entities are semantically compatible with the query and structurally informative in the local graph region induced by the query. Conventional retrieval methods score each evidence unit independently, often relying on contextual similarity between the query and candidate passages. Differently, our retriever asks whether a hyperedge contains entities that are not only semantically related to the query, but also structurally distinctive within the query-induced region of the hypergraph

At reasoning step t, the agent generates a query $\mathbf{q}_t$ for the retrieval. We first extract the set of query entities, denoted by $\mathcal{V}(\mathbf{q}_t)$, and retrieve candidate hyperedges based on both semantic relevance and structural distinctiveness.

Entity-query semantic relevance. We measure how well each entity $v \in V$ semantically aligns with the current query $\boldsymbol{q}_t$, which is computed as:

$$s_t(v) = \mathrm{sim}(\phi(v), \phi(\mathcal{V}(\mathbf{q}_t))), \qquad \forall v \in V, \tag{4}$$

where $\phi(v)$ denotes the embedding of entity v, and

$$\phi(\mathcal{V}(\mathbf{q}_t)) = \frac{1}{|\mathcal{V}(\mathbf{q}_t)|} \sum_{u \in \mathcal{V}(\mathbf{q}_t)} \phi(u) \tag{5}$$

is the aggregated embedding of the entities mentioned in the query. The score $s_t(v)$ captures the semantic compatibility between entity v and the current required information.

Structure-guided entity distinctiveness. Semantic similarity alone is often insufficient in hypergraphs, because highly frequent entities may receive large similarity scores even when they provide limited discriminative value for multi-hop retrieval. To address this issue, we introduce an entity distinctiveness term that reflects how distinctive an entity is within the query-induced hypergraph neighborhood by comparing its local participation frequency against its global participation frequency.

Assume that

$$E(\mathbf{q}_t) = \{e \in E \mid V_e \cap V(\mathbf{q}_t) \neq \emptyset\} \tag{6}$$

denotes query-induced neighborhood, which is the set of hyperedges incident to at least one query entity, where V_e denotes the entity set of hyperedge e. For each entity $v \in V$, we define its query-conditioned distinctiveness of v as

$$D_t(v) = \log\left(1 + \frac{|\{e \in E(\mathbf{q}_t) \mid v \in V_e\}|}{|\{e \in E \mid v \in V_e\}|}\right). \tag{7}$$

This score becomes larger when an entity v appears frequently in the query-induced neighborhood, but not ubiquitously throughout the entire hypergraph. In this sense, $D_t(v)$ acts as a query-conditioned structural prior that downweights overly common entities and amplifies entities that are more informative for the current retrieval context.

Hyperedge re-scoring. Given a candidate hyperedge $e \in E$, we first normalize the semantic relevance of each entity within the hyperedge:

$$\tilde{s}_t(v, e) = \frac{s_t(v)}{\sum_{u \in V_e} s_t(u)}, \qquad v \in V_e. \tag{8}$$

This normalization reflects the relative semantic contribution of entity v within hyperedge e, rather than using its global relevance score in isolation.

We then combine semantic relevance and structural distinctiveness to obtain the entity contribution score

$$r_t(v, e) = \tilde{s}_t(v, e) \cdot D_t(v), \tag{9}$$

and define the overall score of hyperedge e as

$$R_t(e) = \sum_{v \in V_e} r_t(v, e). \tag{10}$$

Therefore, $R_t(e)$ can be interpreted as the expected structural distinctiveness of the entities in hyperedge e, weighted by their semantic relevance to the current query. A hyperedge receives a high score when its query-relevant entities are also distinctive within the query-induced hypergraph neighborhood.

Finally, we rank all candidate hyperedges by $R_t(e)$ and select the top-k hyperedges:

$$\mathcal{E}_t^{\mathrm{ret}} = \mathrm{TopK}_{e \in E}\, R_t(e). \tag{11}$$

The retrieved fact set is then given by

$$\mathcal{F}_t^{\mathrm{ret}} = \bigcup_{e \in \mathcal{E}_t^{\mathrm{ret}}} \{(e, V_e)\}. \tag{12}$$

3.3 Step Progress-Aware Policy Optimization for Evidence-Path Credit Assignment

RL-based agentic GraphRAG methods (Jin et al., 2025; Luo et al., 2025a) typically optimize trajectories with sparse outcome rewards, assigning the same final-answer signal to all reasoning turns. This may be problematic in multi-hop retrieval, where different steps play different roles: some extend the current evidence path, and others may retrieve redundant or structurally disconnected evidence. We therefore introduce a progress-aware stepwise policy optimization objective in which the agent iteratively generates sub-queries $\boldsymbol{q}_t$ and interacts with a knowledge hypergraph $\mathcal{G}_K$. Our HyperGraphPro can be formulated as a stepwise reasoning process where the agent iteratively generates sub-queries $\boldsymbol{q}_t$ for each turn t and interacts with a knowledge hypergraph.

Informational progressive dense rewarding. We apply the dense reward scoring mechanism to reflect the step-level progress. We first design the informational progress score r_t^{ip} at step t, which captures the certainty of reaching ground-truth outputs y^* after generating the current intermediate step. Specifically, we estimate it as the expected final outcome reward of sampled generations from the current state. We define the rollout-estimated answer-reachability value as $\widehat{V}_h = \frac{1}{M}\sum_{m=1}^{M}\mathcal{R}\left(x, \tilde{\boldsymbol{y}}_t^{(m)}\right)$, where $\tilde{\boldsymbol{y}}_t^{(m)} \sim \pi_{\theta_{\text{old}}}\left(\cdot \mid \boldsymbol{x}, \boldsymbol{y}_{<t}; \mathcal{G}_K\right)$ is the generation from the current state t and M is the number of sampling. The answer-progress reward is calculated as the increase in this estimated value caused by the current reasoning turn:

$$r_t^{\text{ip}} = \widehat{V}_t - \widehat{V}_{t-1}. \tag{13}$$

Here, $\widehat{V}_0$ is computed from the initial question before any retrieved hypergraph context is observed.

Structure-consistent progressive dense rewarding. While r_t^{ip} quantifies the informativeness of the intermediate thoughts at current step t, it does not enforce *graph-consistent* progress in multi-hop reasoning. We therefore introduce structural shaping terms that encourage progress along coherent chains in the knowledge hypergraph.

Let $\mathcal{V}(\cdot)$ denote the set of entities (nodes) contained within a given text or hyperedge. We define two connectivity scores for the retrieved hyperedge $\mathcal{G}_t$. We design a structure-progress reward as:

$$r_t^{sp} = r_t^{con} + r_t^{ans}, \tag{14}$$

where the reward consists of the connectivity score and answer reachability score. To ensure consistent multi-hop reasoning, we reward retrievals that share entities with the previously generated state $s_{<t}$ or previously retrieved contexts. This encourages the agent to extend existing reasoning chains rather than retrieving isolated facts:

$$r_t^{con} = \frac{|\mathcal{V}(\mathcal{G}_t) \cap \mathcal{V}(s_{<t})|}{|\mathcal{V}(\mathcal{G}_t)|} \tag{15}$$

To guide the agent toward the solution, we provide a sparse reward when the retrieved hyperedge contains entities present in the ground truth answer y^* in the training phase:

$$r_t^{ans} = \frac{|\mathcal{V}(\mathcal{G}_t) \cap \mathcal{V}(y^*)|}{|\mathcal{V}(\mathcal{G}_t)|} \tag{16}$$

Final Reward. We first define our local progressive reward ρ_t as:

$$\rho_t = \lambda_{\text{ip}} r_t^{\text{ip}} + \lambda_{\text{sp}} r_t^{\text{sp}}, \tag{17}$$

This yields step-level supervision that reflects both (i) *informational progress* (uncertainty reduction) and (ii) *structural progress* (coherent traversal of the knowledge hypergraph).

Unlike standard outcome-based GRPO, we do not assign the scalar reward to every generated trajectory. Instead, for each turn t, we compute a future shaped return as our final reward:

$$G_t^{(i)} = \sum_{u=t}^{T_i} \rho_u^{(i)} + \gamma^{T_i+1-t} r_{\text{outcome}}^{(i)}, \tag{18}$$

where T_i is the maximum number of turn of the i-th trajectory, r_{outcome} is the reward value calculated by outcome reward functions (*e.g.*, format reward, accuracy reward) and $\gamma \in [0, 1]$ is a discount factor. In our experiments, we set $\gamma = 1$.

Stepwise group-relative policy optimization. Different from standard GRPO using sequence-level advantage $\hat{A}$, our HyperGraphPro is trained with stepwise group-relative policy optimization using step-level modulated advantage $\hat{\tilde{A}}$ as follows:

$$\begin{aligned}\mathcal{J}_{\text{StepGRPO}}(\theta) &= \mathbb{E}_{\left[\boldsymbol{x}\sim\mathcal{D}_X, \{\boldsymbol{y}^{(i)}\}_{i=1}^{N}\sim\pi_{\theta_{\text{old}}}(\cdot|\boldsymbol{x};\mathcal{G}_K)\right]} \\ &\left[\frac{1}{|\boldsymbol{y}^{(i)}|}\sum_{t=1}^{|\boldsymbol{y}^{(i)}|} \min\left(\hat{r}_t^{(i)}\hat{\tilde{A}}_t^{(i)}, \text{clip}\left(\hat{r}_t^{(i)}, 1-\epsilon, 1+\epsilon\right)\hat{\tilde{A}}_t^{(i)}\right)\right] \\ &- \beta\mathbb{D}_{\text{KL}}\left(\pi_\theta \,||\, \pi_{\text{ref}}\right),\end{aligned} \tag{19}$$

Method	2Wiki.		HotpotQA		MuSiQue		Multihop Avg.		NQ		All Avg.	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Qwen2.5-3B-Instruct												
NaiveGeneration	2.34	7.59	6.25	11.16	0.78	3.67	3.12	7.47	2.34	8.90	2.93	7.83
StandardRAG	3.91	12.52	7.03	15.41	0.00	2.92	3.65	10.28	0.00	10.69	2.74	10.39
SFT	7.03	12.40	10.94	16.48	1.56	5.04	6.51	11.31	3.12	11.23	5.66	11.29
R1	20.31	28.45	20.31	25.33	3.12	8.07	14.58	20.62	11.72	21.51	13.87	20.84
Search-R1	31.25	38.04	38.28	43.84	3.91	7.65	24.48	29.84	24.22	37.96	24.42	31.87
R1-Searcher	13.28	23.50	35.94	42.44	7.81	12.81	19.01	26.25	24.22	36.53	20.31	28.82
Graph-R1	50.00	57.56	50.78	56.75	32.81	40.51	44.53	51.61	30.47	44.75	41.02	49.89
HyperGraphPro (Ours)	**55.47**	**61.72**	**55.47**	**61.90**	**37.50**	**47.27**	**49.48**	**56.96**	**34.38**	**47.34**	**45.71**	**54.56**
Qwen2.5-7B-Instruct												
NaiveGeneration	3.12	12.25	6.25	16.58	0.00	4.06	3.12	10.96	1.56	13.00	2.73	11.47
StandardRAG	7.81	12.75	10.16	21.10	0.78	4.53	6.25	12.79	1.56	15.97	5.08	13.59
SFT	11.72	20.28	19.53	27.59	5.47	10.02	12.24	19.30	5.12	19.02	10.46	19.23
R1	25.00	30.99	31.25	37.05	7.03	14.53	21.09	27.52	16.41	28.45	19.92	27.76
Search-R1	36.72	41.29	44.53	50.85	14.84	22.35	32.03	38.16	32.03	45.88	32.03	40.09
R1-Searcher	27.34	33.96	39.84	46.36	10.16	16.63	25.78	32.32	32.03	44.93	27.34	35.47
Graph-R1	55.47	65.04	57.03	62.69	36.72	46.17	49.74	57.97	33.59	49.87	45.70	55.94
HyperGraphPro (Ours)	**59.38**	**69.75**	**60.94**	**67.57**	**39.84**	**49.47**	**53.39**	**62.26**	**35.94**	**50.71**	**49.03**	**59.38**

Table 1: Experimental results on multihop question-answering benchmark datasets such as 2WikiMultiHopQa, HotpotQA, and MuSiQue, and a singlehop question-answering dataset such as NQ. Multihop Avg. indicates the mean performance over multihop datasets. All Avg. indicates the mean performance over all datasets.

where $\hat{\tilde{A}}_t^{(i)} = \frac{G_t^{(i)} - \mu_{G_t}}{\sigma_{G_t}}$ denotes the stepwise modulated advantage at reasoning turn t for trajectory i, enabling the policy to focus on local rewards associated with specific reasoning decisions. This allows for more fine-grained credit assignment during training, especially beneficial in multi-hop or structured reasoning settings.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate our proposed methods on three multi-hop question answering benchmark datasets: 2WikiMultihopQA (**2Wiki.**) (Ho et al., 2020), **HotPotQA** (Yang et al., 2018), **MuSiQue** (Trivedi et al., 2022). We also adopt a large-scale single-hop QA dataset, Natural Questions (**NQ**) (Kwiatkowski et al., 2019) to explore the performance under the single-hop questions.

Models. We conduct experiments using `Qwen-2.5-Instruct` (3B, and 7B) (Qwen et al., 2025) as the base LLMs following existing works (Luo et al., 2025a). For the retrieval, we employ `bge-large-en-v1.5` (Xiao et al., 2024) following Graph-R1 (Luo et al., 2025a). Additional implementation details are in the supplement.

Baselines. To show the effectiveness of our HyperGraphPro over previous methods, we compare it with the baselines: **NaiveGeneration**, **StandardRAG**, **SFT**, **R1** (Shao et al., 2024), **Search-R1** (Jin et al., 2025), **R1-Searcher** (Song et al., 2025), and **Graph-R1** (Luo et al., 2025a).

4.2 Experimental Results

We evaluate our proposed methods using two variants of `Qwen2.5-Instruct` (3B and 7B) in Table 1. HyperGraphPro consistently achieves the best average performance, showing remarkable improvements over both naive generation and retrieval-augmented baselines. With `Qwen2.5-3B-Instruct`, our method reaches an average F1 of 54.56, compared to 49.89 from the strongest baseline (Graph-R1). Similarly, under the `Qwen2.5-7B-Instruct`, our method achieves the highest overall performance with an average F1 of 59.38, improving upon the best baseline (Graph-R1, 55.94) by more than 3.4 points. The improvement is more pronounced on multi-hop question-answering datasets than on NQ. For `Qwen2.5-3B-Instruct`, the multihop average F1 gain over Graph-R1 is +5.35, compared with +2.59 on NQ. For `Qwen2.5-7B-Instruct`, the multihop average F1 gain is +4.29, compared with +0.84 on NQ. This result supports that progress-aware hypergraph retrieval and stepwise policy optimization are more beneficial when more connecting evidence is required across multiple reasoning turns.

Method	2Wiki.		HotpotQA		MuSiQue		2Wiki.		HotpotQA		MuSiQue	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
	Qwen3-0.6B						***Qwen3-1.7B***					
Search-R1	22.66	29.28	31.25	40.99	7.81	12.82	31.25	36.79	39.06	44.58	10.94	17.23
Graph-R1	24.22	36.03	32.81	42.21	19.53	27.38	42.97	46.55	46.88	50.25	29.69	37.72
HyperGraphPro (Ours)	**28.91**	**40.72**	**36.72**	**47.39**	**23.44**	**30.42**	**47.66**	**50.91**	**50.78**	**55.02**	**32.81**	**40.73**

Table 2: Experimental results using Qwen3-0.6B and Qwen3-1.7B.

Str. guided ret.	Progress-aware PO	2Wiki. EM	2Wiki. F1	MuSiQue EM	MuSiQue F1
		50.00	57.56	32.81	40.51
✓		51.56	58.45	33.59	42.14
	✓	53.91	60.47	34.38	45.16
✓	✓	**55.47**	**61.72**	**37.50**	**47.27**

Table 3: Ablation studies of our HyperGraphPro on multi-hop question answering datasets with Qwen-2.5-3B. (Str. guided ret.: Structure-guided hypergraph retrieval, Progress-aware PO: Progress-aware Step-level policy optimization).

Info. Progress	Structure Progress	2Wiki. EM	2Wiki. F1	MuSiQue EM	MuSiQue F1
		51.56	58.45	33.59	42.14
✓		53.13	59.61	35.16	44.81
	✓	54.69	60.55	34.38	44.29
✓	✓	**55.47**	**61.72**	**37.50**	**47.27**

Table 4: Performance comparison according to the policy optimization using Qwen-2.5-3B. Step progress means an informational progress score. Structure progress means structure-consistent progressive dense rewarding.

Retrieval method	2Wiki. EM	2Wiki. F1	MuSiQue EM	MuSiQue F1
No Retrieval	20.31	28.45	3.12	8.07
Knowledge corpus (Ctxt.)	31.25	43.84	3.91	7.65
Hyperedge (Ctxt.)	50.78	58.13	32.03	40.31
Graph-R1 (Ctxt.)	53.91	61.21	34.38	43.41
HyperGraphPro (Ours) (Ctxt.+Struct.)	**55.47**	**61.72**	**37.50**	**47.27**

Table 5: Performance comparison according to the retrieval methods on multi-hop question answering datasets using Qwen-2.5-3B. Ctxt. indicates the natural language context-based relevance. Struct. indicates the graph structure-based relevance.

Method	2Wiki. # Turns	2Wiki. F1	MuSiQue # Turns	MuSiQue F1
Qwen2.5-3B-Instruct				
Graph-R1	2.63	57.56	3.79	40.51
HyperGraphPro (Ours)	**2.45**	**61.72**	**3.16**	**47.27**
Qwen2.5-7B-Instruct				
Graph-R1	2.72	65.04	3.92	46.17
HyperGraphPro (Ours)	**2.56**	**69.75**	**3.54**	**49.47**

Table 6: Comparison on the number of turns between Graph-R1 and HyperGraphPro (Ours).

4.3 Additional Experimental Results

Experimental results on Qwen3. To assess the effectiveness of our HyperGraphPro on other variants of LLMs, we conduct the experiments using `Qwen3-0.6B` and `Qwen3-1.7B` (Table 2). From the table, our HyperGraph outperforms other baseline methods using Qwen3 models. This experimental result demonstrates that our proposed method is robust to small size variants of large language models.

Ablation studies. To evaluate the effectiveness of the structure-guided hypergraph retrieval (Sec. 3.2) and progress-aware policy optimization (Sec. 3.3), we conduct an ablation study on two multi-hop QA benchmarks in Table 3. The results show that both components contribute to performance improvements across all datasets. Specifically, applying progress-aware stepwise policy optimization leads to better learning and enhanced reasoning ability. Similarly, incorporating our structure-guided hypergraph retrieval framework alone yields consistent gains over the baseline, demonstrating the benefit of integrating hypergraph structural information during retrieval. Then, combining both components, our full model achieves the best performance, with notable improvements in F1 scores, such as +4.16 on 2Wiki. and +6.76 on MuSiQue over the baseline. These results confirm that both finer-grained policy updates and structure-aware retrieval are helpful for improving multi-hop question answering with large language models.

Policy optimization analysis. Additionally, we conduct an ablation study to examine the effect of different progress-aware dense rewarding functions used for policy optimization. Table 4 reports the experimental results on 2Wiki and MuSiQue. The best performance is obtained when both informational progress score r_t^{ip} and structure-consistent progressive r_t^{sp} dense rewarding are jointly applied. These results suggest that the two dense reward signals capture complementary aspects of reasoning progress, and that their combination more effec-

Question & Answer	Graph-R1	HyperGraphPro (Ours)
Question: When is election day for senate in the state where Taylor Hicks is from? **Answer:** December 12, 2017	⟨think⟩ To answer this question, I need to determine the state Taylor Hicks is from and then find out when the election day for the senate is in that state. ⟨/think⟩ ⟨query⟩ where is Taylor Hicks from ⟨/query⟩ ⟨knowledge⟩ According to season five winner Taylor Hicks, who is from the state of Alabama, ... ⟨/knowledge⟩ ⟨think⟩ The knowledge base states that Taylor Hicks, the season five winner, is from Alabama. Now I need to find out when the election day for the senate is in Alabama. ⟨/think⟩ ⟨query⟩ election day for senate in Alabama ⟨/query⟩ ⟨knowledge⟩ A special election for the United States Senate in Alabama took place on December 12, 2017,... ⟨/knowledge⟩ ⟨think⟩ The knowledge base provides information about a special election for the United States Senate in Alabama... Typically, senate elections occur every two years on the same day as the general midterm elections, which is usually the first Tuesday after the first Monday in November. ⟨/think⟩ ⟨query⟩ when is the election day for senate in the united states ⟨/query⟩ ⟨knowledge⟩ A special election for the United States Senate in Alabama took place on December 12, 2017,... The 2018 United States House of Representatives elections will be held on November 6, 2018. ⟨/knowledge⟩ ⟨think⟩ Based on the information provided, the typical election day for the senate in the United States is the first Tuesday after the first Monday in November every two years. This is the general midterm election day. ⟨/think⟩ ⟨answer⟩ The election day for the senate in the United States is the first Tuesday after the first Monday in November. ⟨/answer⟩	⟨think⟩ To answer this question, I need to find out which state Taylor Hicks is from first. Then, I can find the election day for the senate election in that state. ⟨/think⟩ ⟨query⟩ Taylor Hicks state ⟨/query⟩ ⟨knowledge⟩ According to season five winner Taylor Hicks, who is from the state of Alabama, ... ⟨/knowledge⟩ ⟨think⟩ From the search results, we can see that Taylor is from the state of Alabama. Now, I need to find out the election day for the senate election in Alabama. ⟨/think⟩ ⟨query⟩ election day for senate election in Alabama ⟨/query⟩ ⟨knowledge⟩ A special election for the United States Senate in Alabama took place on December 12, 2017,... ⟨/knowledge⟩ ⟨think⟩ From the search results, the special election for the United States Senate in Alabama took place on December 12, 2017. ⟨/think⟩ ⟨answer⟩ December 12, 2017 ⟨/answer⟩

Table 7: Comparison on outputs generated by Graph-R1 and HyperGraphPro (Ours).

tively guides the policy toward accurate multi-hop reasoning.

Retrieval method analysis. Table 5 compares different retrieval strategies on multi-hop QA datasets using `Qwen2.5-3B-Instruct`. Our HyperGraphPro consistently achieves the best performance across all datasets, highlighting the effectiveness of integrating both contextual and structural relevance in retrieval. Compared to the text-only retrieval baselines (Knowledge corpus and Hyperedge), HyperGraphPro yields substantial gains, demonstrating that leveraging graph structures enables more accurate and semantically coherent knowledge selection. These results confirm that combining semantic and structural information is crucial for improving reasoning consistency in complex question answering.

Turn length analysis. We compare the average number of reasoning turns of HyperGraphPro with Graph-R1 using `Qwen-2.5-3B-Instruct` and `Qwen-2.5-7B-Instruct` in Table 6. The results show that HyperGraphPro achieves higher performance with fewer turns even though the number of turns is not used as a reward signal. This demonstrates that our method improves the reasoning effectiveness with less retrieval.

4.4 Qualitative Analysis

Here, we conduct a qualitative comparison between Graph-R1 and HyperGraphPro to illustrate how our reasoning framework enhances multi-hop reasoning (Table 7). While Graph-R1 correctly retrieves relevant knowledge snippets, it often produces redundant or conflicting reasoning steps—such as repeatedly querying the election day in Alabama and overgeneralizing the final answer to the national level. In contrast, HyperGraphPro successfully generates the reasoning trajectory by progressively refining each step, leveraging retrieved graph context to eliminate unnecessary or misleading hops. As shown in the example, HyperGraphPro accurately identifies the specific "special election" event on December 12, 2017, by integrating contextual knowledge about the Senate vacancy, while Graph-R1 misleads itself toward the general midterm schedule. This highlights that HyperGraphPro effectively mitigates reasoning drift and enables precise, context-aware multi-hop inference.

5 Conclusion

We have introduced HyperGraphPro, a step-aware reinforcement learning framework for graph-based retrieval and reasoning. The core idea is to consider evidence-path progress in both retrieval and policy optimization. While prior GraphRAG frameworks advanced agentic multi-step reasoning, they remain limited by structural unawareness and sparse reward feedback. HyperGraphPro overcomes these issues through structure-aware retrieval and step-wise advantage modulation, enabling more coherent reasoning. Experiments on knowledge-intensive QA benchmarks show that HyperGraphPro consistently improves answer accuracy on knowledge-intensive QA benchmarks.

Limitations

HyperGraphPro method is significantly influenced by the prompt, retriever, the base LLM, and the given input graph. Specifically, based on the structure and quality of the input graph, the downstream task performance can vary substantially.

Ethics Statement

As we use LLM with external knowledge bases, the outputs may contain unintentionally hallucinating, harmful, sensitive, or biased content. Therefore, the LLM and the knowledge base should be carefully selected to evade such situations.

References

Boyu Chen, Zirui Guo, Zidan Yang, Yuluo Chen, Junze Chen, Zhenghao Liu, Chuan Shi, and Cheng Yang. 2026. PathRAG: Pruning graph-based retrieval augmented generation with relational paths. In *AAAI*.

DeepSeek-AI. 2025. Deepseek-v3.2: Pushing the frontier of open large language models.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. 2025. Group-in-group policy optimization for llm agent training. In *NeurIPS*.

Yifan Feng, Hao Hu, Xingliang Hou, Shiquan Liu, Shihui Ying, Shaoyi Du, Han Hu, and Yue Gao. 2026. Hyper-RAG: Combating LLM Hallucinations using Hypergraph-Driven Retrieval-Augmented Generation. *Nature Communications*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 180 others. 2025a. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Nature*, 645:633–638.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2025b. LightRAG: Simple and fast retrieval-augmented generation. In *EMNLP*.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. 2025. From RAG to Memory: Non-parametric continual learning for large language models. In *ICML*.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *COLING*, pages 6609–6625.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. In *EMNLP*.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-R1: Training llms to reason and leverage search engines with reinforcement learning. In *COLM*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, and 1 others. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *NeurIPS*, volume 33, pages 9459–9474.

Haoran Luo, Guanting Chen, Qika Lin, Yikai Guo, Fangzhi Xu, Zemin Kuang, Meina Song, Xiaobao Wu, Yifan Zhu, Luu Anh Tuan, and 1 others. 2025a. Graph-r1: Towards agentic graphrag framework via end-to-end reinforcement learning. *arXiv preprint arXiv:2507.21892*.

Haoran Luo, Guanting Chen, Yandan Zheng, Xiaobao Wu, Yikai Guo, Qika Lin, Yu Feng, Zemin Kuang, Meina Song, Yifan Zhu, and 1 others. 2025b. HyperGraphRAG: Retrieval-augmented generation via hypergraph-structured knowledge representation. In *NeurIPS*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. In *NeurIPS*, volume 35, pages 27730–27744.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, and 1 others. 2019. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2015. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025. Smart-searcher: Incentivizing the dynamic knowledge acquisition of llmsvia reinforcement learning. In *EMNLP Findings*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multi-hop questions via single-hop question composition. *TACL*, 10:539–554.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *ACL*, pages 10014–10037.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. 2020. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl.

Nengbo Wang, Xiaotian Han, Jagdip Singh, Jing Ma, and Vipin Chaudhary. 2025. CausalRAG: Integrating causal graphs into retrieval-augmented generation. In *ACL Findings*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, and 1 others. 2020. Transformers: State-of-the-art natural language processing. In *NeurIPSW*, pages 38–45.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack: Packed resources for general chinese embeddings. In *SIGIR*.

Tianyang Xu, Haojie Zheng, Chengze Li, Haoxiang Chen, Yixin Liu, Ruoxi Chen, and Lichao Sun. 2025. Noderag: Structuring graph-based rag with heterogeneous nodes. *arXiv preprint arXiv:2504.11544*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*.

Siliang Zeng, Quan Wei, William Brown, Oana Frunza, Yuriy Nevmyvaka, and Mingyi Hong. 2025. Reinforcing multi-turn reasoning in llm agents via turn-level credit assignment. *arXiv preprint arXiv:2505.11821*.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*.

Ziwei Zheng, Michael Yang, Jack Hong, Chenxiao Zhao, Guohai Xu, Le Yang, Chao Shen, and Xing Yu. 2026. Deepeyes: Incentivizing" thinking with images" via reinforcement learning. In *ICLR*.

A More implementation details

We use 4 NVIDIA A6000 GPUs for `Qwen-2.5-3B-Instruct`. For `Qwen-2.5-7B-Instruct`, we use 4 NVIDIA H200 GPUs. Our code is implemented based on PyTorch library (Paszke et al., 2019). Also, we adopt Huggingface transformers (Wolf et al., 2020) and trl library (von Werra et al., 2020) to train and inference our HyperGraphPro. Following Graph-R1 (Luo et al., 2025a), we uniformly sample 5,120 samples per dataset for training and 128 instances for the evaluation. We use 128 batch size and set max length as 4096. For each turn, we retrieve top-5 hyperedges and the epoch is set to 3 following other works (Luo et al., 2025a) for a fair comparison. We use $\lambda_{ip} = 0.5, \lambda_{sp} = 0.5, M = 4$ for the hyperparameter. In addition, we use an automated grammar checker for proofreading only.

B Prompt

Table 8 shows the input template for HyperGraphPro following Graph-R1 (Luo et al., 2025a). As shown in Figure 1, we use the same n-ary relation-extraction prompt as HyperGraphRAG (Luo et al., 2025b).

C Full Efficiency Analysis

Table 9 compares the average number of turns between Graph-R1 and HyperGraphPro across four multi-hop QA benchmarks. From the table, HyperGraphPro consistently achieves comparable or fewer reasoning turns while delivering substantially higher F1 scores across all datasets and model scales although it is not explicitly designed to improve the efficiency of the model. On average, HyperGraphPro reduces the number of turns or maintains similar turn counts relative to Graph-R1, indicating improved reasoning efficiency alongside accuracy gains. These results suggest that the proposed progress-aware reinforcement learning implicitly encourages more effective multi-step retrieval and reasoning. In the future work, incorporating explicit turn-aware or efficiency-driven rewards may further reduce reasoning steps and enhance overall inference efficiency.

D Reward functions

Following existing works (Luo et al., 2025a), we define a reward function $R(\boldsymbol{\tau})$ with a *format reward* $R_{\text{format}}(\tau)$ and an *answer reward* $R_{\text{answer}}(\boldsymbol{y}_T^{\text{ans}})$, leading both informative retrieval and accurate answer generation.

Format Reward. The format reward $R_{\text{format}}(\tau)$ encourages the agent to follow the intended reasoning structure. At each step, we check whether the output includes a well-formed block $\left(a_t^{\text{think}}, a_t^{\text{type}}, a_t^{\text{out}}\right)$. Each valid step receives a reward of 0.5, capped at a total of 1.0:

$$R_{\text{format}}(\boldsymbol{x}, \boldsymbol{y}) = \min\left(1.0,\ 0.5 \cdot \sum_{t=1}^{T} \mathbb{I}\left\{(\boldsymbol{y}_t) \text{ is well-formed}\right\}\right), \tag{20}$$

where $\mathbb{I}\{\cdot\}$ is an indicator function that returns 1 if the step output matches the expected format.

Answer Reward. The answer reward $R_{\text{answer}}(\boldsymbol{a}_T^{\text{ans}})$ measures the semantic correctness of the generated answer $\boldsymbol{a}_T^{\text{ans}}$ by comparing it with the ground-truth answer y_q^* using a token-level F1 score:

$$R_{\text{answer}}(\boldsymbol{x}, \boldsymbol{y}) = \frac{2 \cdot \left|\text{tokens}(\boldsymbol{y} \cap \text{tokens}(y_q^*)\right|}{\left|\text{tokens}(\boldsymbol{a}_T^{\text{ans}})\right| + \left|\text{tokens}(y_q^*)\right|}, \tag{21}$$

where $|\cdot|$ denotes multiset cardinality. The function $\text{tokens}(\cdot)$ applies standard preprocessing including lowercasing and whitespace-based tokenization.

Overall Outcome Reward. The total reward for a reasoning trajectory τ is defined as:

$$\begin{aligned} r_{\text{outcome}} &= -1.0 + R_{\text{format}}(\boldsymbol{x}, \boldsymbol{y}) + \\ &\mathbb{I}\{R_{\text{format}}(\boldsymbol{x}, \boldsymbol{y}) = 1.0\} \cdot R_{\text{answer}}(\boldsymbol{x}, \boldsymbol{y}). \end{aligned} \tag{22}$$

This ensures that answer correctness is only rewarded when the format is structurally valid. With the outcome-directed reward $R(\boldsymbol{x}, \boldsymbol{y})$, high answer quality is attainable through structurally coherent and reasoning-complete trajectories τ with multi-turn iteration within knowledge hypergraph $\mathcal{G}_K$.

E Evaluation Metrics

To evaluate each method across benchmarks, we mainly consider three metrics. First, we report Exact matching (**EM**), which calculates the correctness of the prediction by comparing the predicted answer $\boldsymbol{y}$ with ground truth $\boldsymbol{y}^\star$:

$$\text{EM} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\left(\text{norm}\left(\boldsymbol{y}\right) = \text{norm}\left(\boldsymbol{y}^\star\right)\right), \tag{23}$$

You are a helpful assistant. Answer the given question. You can query from knowledge base provided to you to answer the question. You can query knowledge as many times as you want. You must first conduct reasoning inside `<think>`...`</think>`. If you need to query knowledge, you can set a query statement between `<query>`...`</query>` to query from knowledge base after `<think>`...`</think>`. When you have the final answer, you can output the answer inside `<answer>`...`</answer>`. Question: question. Assistant:

Table 8: Input template for HyperGraphPro following Graph-R1 (Luo et al., 2025a). question will be replaced with the specific user query. Note that the knowledge retrieved is placed within `<knowledge>`...`</knowledge>` after `</query>`.

```
-Goal-
Given a text document that is potentially relevant to this activity and a list of entity types, identify all entities of those types from the text and all relationships among the identified entities.
Use {language} as output language.

-Steps-
1. Divide the text into several complete knowledge segments.  For each knowledge segment, extract the following information:
-- knowledge_segment: A sentence that describes the context of the knowledge segment.
Format each knowledge segment as ("hyper-relation"{tuple_delimiter}<knowledge_segment>)

2. Identify all entities in each knowledge segment. For each identified entity, extract the following information:
- entity_name: Name of the entity, use same language as input text. If English, capitalized the name.
- entity_type: Type of the entity.
- entity_description: Comprehensive description of the entity's attributes and activities.
Format each entity as ("entity"{tuple_delimiter}<entity_name>{tuple_delimiter}<entity_type>{tuple_delimiter}<entity_description>)

3. Return output in {language} as a single list of all the entities and relationships identified in steps 1 and 2. Use **{record_delimiter}** as the list delimiter.

4. When finished, output {completion_delimiter}

######################
-Examples-
######################
{examples}

#############################
-Real Data-
######################
Text: {input_text}
######################
```

Figure 1: Prompt for n-ary relation extraction π_{ext}.

	2Wiki.		HotpotQA		MuSiQue		NQ	
Method	# Turns	F1	# Turns	F1	# Turns	F1	# Turns	F1
Qwen2.5-3B-Instruct								
Graph-R1	2.63	57.56	3.65	56.75	3.79	40.51	1.21	44.75
HyperGraphPro (*ours*)	**2.45**	**61.72**	**3.58**	**61.90**	**3.16**	**47.27**	**1.19**	**47.34**
Qwen2.5-7B-Instruct								
Graph-R1	2.72	65.04	**3.38**	62.69	3.92	46.17	**1.13**	49.87
HyperGraphPro (*ours*)	**2.56**	**69.75**	3.41	**67.57**	**3.54**	**49.47**	1.15	**50.71**

Table 9: Full comparison on the number of turns between Graph-R1 and HyperGraphPro (Ours).

In parallel, **F1** score quantifies token-level overlap between the prediction $\boldsymbol{y}$ and the ground truth $\boldsymbol{y}^\star$, defined as:

$$\text{F1} = \frac{1}{N}\sum_{i=1}^{N} \frac{2 \cdot |\text{tokens}(\boldsymbol{y}) \cap \text{tokens}(\boldsymbol{y}^\star)|}{|\text{tokens}(\boldsymbol{y})| + |\text{tokens}(\boldsymbol{y}^\star)|} \quad (24)$$

F Baselines

To show the effectiveness of our HyperGraphPro over previous methods, we compare it with the following baselines:

- **NaiveGeneration**: produces answers directly from LLM using only the input question without any retrieval, which can be seen as the simplest lower-bound reference.
- **StandardRAG** (Lewis et al., 2020): A conventional retrieval-augmented generation framework where a semantic retriever selects chunks, and LLM generates responses conditioned on them.
- **Supervised Fine Tuning (SFT)** (Zheng et al., 2024): LLM fine-tuned in a supervised manner on question–answer pairs, without any multi-turn reasoning or reinforcement learning.
- **R1** (Shao et al., 2024): A policy optimized with GRPO that directly generates answers from prompts, without retrieval, focusing solely on improving outcome.
- **Search-R1** (Jin et al., 2025): A GRPO-trained retrieval model that operates in multiple turns, iteratively refining queries and retrieved chunks under a unified policy.

- **R1-Searcher** (Song et al., 2025): A two-phase GRPO approach with retrieval, which is initially guided by format-level rewards to generate structured reasoning traces, followed by optimization with answer-level rewards.

- **Graph-R1** (Luo et al., 2025a): An agentic GraphRAG system using end-to-end reinforcement learning over a lightweight hypergraph to iteratively think-retrieve-rethink-generate.