BREAK THE CHAIN, GAIN A MARGIN: A THEORY OF MULTI-STEP REASONING IN LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Despite advances in retrieval-augmented reasoning through reinforcement learning (RL), it remains unclear when and why language models struggle with multi-step reasoning. We study this issue using a graph-based mathematical framework, in which a language model (LM) identifies valid reasoning steps by using statements as nodes and correct links as edges. We present a key theoretical result suggesting that such LMs are capable of internally identifying long reasoning chains. However, without multi-step reasoning, their confidence (margin) in distinguishing true from false links decreases as chains grow longer, making long-range dependencies increasingly error-prone. Motivated by this analysis, we develop theoretical principles for multi-step reasoning that break long chains into shorter segments with larger training margins. We instantiate these principles in PHyRR (Plan-Conditioned Hypergraph Retrieval-Reasoning), a RL framework for question answering (QA). PHyRR first decomposes queries into smaller sub-goals during planning, then performs hypergraph retrieval and reasoning adaptively based on the resulting plan. Its RL rewards further encourage reasoning steps that are theoretically predicted to be more effective. Experiments confirm the predicted margin behavior and show that PHyRR outperforms baselines across four QA benchmarks, with the largest improvements on multi-hop reasoning tasks.

1 INTRODUCTION

To tackle complicated reasoning tasks, modern Large Language Models (LLMs) rely on multiple stages of internal reasoning and information retrieval from external knowledge sources. Among various strategies for this purpose, Reinforcement Learning (RL)-based methods have recently received particular attention as they let a model automatically decide when to retrieve as it reasons. The RL approach is particularly advantageous in multi-step question-answering (QA), where the external evidence needed later is only identifiable once earlier reasoning steps are worked out. Nevertheless, the mechanisms in multi-step reasoning remain insufficiently understood, and it is still unclear how the chains of reasoning and retrieval simplify difficult tasks that require long internal reasoning chains.

In this paper, we suggest a theoretical framework to understand and improve chains of reasoning and retrieval. We mathematically describe the basic ability of a language model (LM) to "reason," as the problem of representing an *inference graph* where statements are nodes, and valid support among statements are edges. Within this theoretical analysis, we show some key results. First, the inference graph can be represented in a fixed-size space that depends only on its width; long chains do not by themselves require a larger representation. Hence, LMs are capable of internally identifying long reasoning chains. Second, the confidence in learning any such representation shrinks steadily as chains grow longer. We describe this confidence in terms of a parameter called representation *margin*. Long chains are representable, but the specific links they depend on sit increasingly close to the decision boundary. In this way, we explain why accuracy degrades with reasoning depth.

Motivated by this analysis, we further establish a set of theoretical principles for multi-step reasoning, which revolve around the idea of decomposing a long chain into smaller segments that can be learned with higher confidence. Based on these principles, we develop PHyRR, an RL framework for QA. PHyRR employs a novel initial planning component that decomposes a query into a sequence of atomic sub-goals, after which reasoning and retrieval are performed adaptively based on the

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

plan along a single trajectory. To improve information retrieval, we also introduce a method by constructing a knowledge hypergraph. Finally, supported by our theoretical principles, we propose a new multi-component reward during the RL optimization.

Our contributions are summarized as follows:

1. We present a graph-theoretic analysis of LMs, explaining that long chains of internal reasoning are learned by less confidence, and justifying external chains as a method to cope with this effect.

2. Based on the theoretical analysis, we develop PHyRR framework, including components namely initial planning, an adaptive retrieval-reasoning process including hypergraph retrieval based on plan, and a multi-component RL reward function.

3. We empirically validate both the theory and the framework. In benchmark experiments, PHyRR consistently shows better accuracy, and ablation studies verify the importance of the theoretically grounded components.

2 Theoretical Basis for Reasoning

In this section, we demonstrate provable benefits of the chain of reasoning by focusing on the basic ability of language models (LMs) in predicting the relation between two given statements, a property that indicates their capabilities in other downstream tasks, such as text generation. In the following, we give a formal definition of this feature and discuss its relation to the chain of reasoning. Due to the complexity of the LLMs, we carry out our analysis in a simplified and idealized scenario.

2.1 Language Models as Graph Embedding

Consider a directed *inference graph* $\mathcal{G} = (V, E)$ with nodes V and directed edges (arrows) E. Each node $v \in V$ corresponds to a possible statement (i.e., a valid string of tokens) in a language, and each edge $(u, v) \in E$, which we simply call a *reason*, shows that the source statement u semantically supports the target v, a relation that we also denote by $u \to v$. We assume that this relation forms a **partially ordered set (poset)**, i.e., it is reflexive, antisymmetric and transitive.

In practice, $\mathcal{G}$ is large and only fractionally observed. A language model (LM) provides a convenient means to predict whether a given unseen pair (u, v) belongs to $\mathcal{G}$, i.e., the relation $u \to v$ holds true, or not. Here, we consider an ideal scenario, where this goal is achieved by embedding the nodes in a Euclidean space, as defined below:

Definition 1. *For an embedding dimension m, consider the pair of maps $\mathrm{k} : V \to \mathbb{R}^m$ and $\mathrm{q} : V \to \mathbb{R}^m$, respectively called key and query, such that for every edge $e = (u, v) \in E$ it holds that $\langle \mathrm{q}(u), \mathrm{k}(v) \rangle > 0$ and for every non-edge $(u, v) \notin E$ it holds that $\langle \mathrm{q}(u), \mathrm{k}(v) \rangle \leq 0$. Then, we say that (k, q) is an embedding of $\mathcal{G}$.*

However, this definition is ambiguous in multiple ways. For example, one can arbitrarily scale the keys and queries, or introduce proper rotations to them, without affecting their inner product. To avoid such ambiguities, we introduce the following:

Definition 2. *For a finite graph with $n = |V|$ vertices and $N = |E|$ edges, we say that an embedding (k, q) is* **canonical** *if*

$$\frac{1}{n} \sum_{v \in V} \mathrm{k}(v)\mathrm{k}^T(v) = I, \quad \frac{1}{n} \sum_{v \in V} \mathrm{q}(v)\mathrm{q}^T(v) = D, \tag{1}$$

where I is the identity matrix and D is a diagonal matrix with non-negative diagonals and $\mathrm{Tr}(D) = N$. Moreover, we refer to the smallest value ϵ of $\langle \mathrm{q}(u), \mathrm{k}(v) \rangle$ for an edge $e = (u, v) \in E$ as the **margin** *and the corresponding edges as* **critical edges**.

Canonical embeddings are deeply connected to singular value decomposition (SVD) of weighted adjacency matrices. Their margin ϵ also reflects the difficulty of embedding a graph by low-rank decomposition. In Appendix B, we clarify this fact by studying a scenario where edges and non-edges of $\mathcal{G}$ are independently sampled, and an observed adjacency matrix $\hat{A}$ is formed by setting observed edges and non-edges to two distinct scalar values, respectively. The following informal theorem summarizes the findings in appendix B.

Theorem 1 (informal)**.** *Denote by $\rho := \frac{N}{n^2}$ the edge density of $\mathcal{G}$. Suppose that for a fixed number m, truncating the adjacency matrix A to the m largest singular values $\{\sigma_i\}$ provides a canonical embedding of $\mathcal{G}$ with margin ϵ and we have $\sigma_i \sim \bar{\sigma}_i\sqrt{N}$ for distinct constants $\bar{\sigma}_i$. Moreover, the average number b of sampled edges satisfies $b \sim n$ and refer to $\sigma := \sqrt{n/b}$ as the effective sampling noise. Consider the embedding provided by the singular vectors of the m largest singular values of $\hat{A}$. For this embedding, the asymptotic probability of mistaking each edge purely depends on the true embedding of this edge as well as σ, ρ. In particular, for small values of σ, the probability of missing critical edges is an exclusive function of ϵ, σ, ρ.*

Theorem 1 identifies three key factors ϵ, ρ and σ in the edge prediction error of the LMs trained by SVD. Concrete expressions are given in corollary 1 in the appendix. While real-world LMs are different in training, one may expect that these factors play a similar prominent role in them, too. In a numerical experiment (Figure 4 in Appendix B), we evaluate this relationship that confirms the dependency of the probability of missing edges on three key factors.

2.1.1 When Does an Embedding Exist?

In this part, we shows that the transitivity property of the inference graphs encourages the existence of their low-dimensional embedding. This observation indicates that LMs are capable of internally encoding long reasoning chains. However, we also observe that long chains force the margin ϵ of the embedding to shrink, which, according to Theorem 1, makes the embedding exceedingly more difficult to be correctly learned. This observation constitutes the core of our justification for the chain of reasoning as a way to combat the shrinkage of the margin and its resulting growth of training error.

To express our result, we remind that the height of a poset is the length of its longest chain (i.e., a total ordering included in it), and its width is the minimum number of disjoint chains entirely covering (partitioning) it. For an inference graph, each covering chain can be interpreted as a *mode of inference*, and hence the width reflects the number of such modes. The height is the longest viable reasoning chain. Then, we provide the following result (see appendix C for the proof):

Theorem 2. *Given an inference graph $\mathcal{G}$ with width w and height h:*

1. *there exists a $2w-$dimensional canonical embedding of $\mathcal{G}$, and*
2. *the margin ϵ of every $m-$dimensional canonical embedding of $\mathcal{G}$ satisfies $\epsilon \leq c\sqrt{\frac{m}{h}}$, where c is a universal constant*[1].

Theorem 2 shows that the minimum required dimension of embedding purely depends on the width, and embedding of long chains of arbitrary length h is possible. However, an LM targeting such long chains invariably encodes some critical edges with a margin of $O(1/\sqrt{h})$, which are easy to miss by Theorem 1. (We verify both parts of this theorem numerically shown in Figures 5–6 in Appendix C.)

2.2 How does the chain of reasoning help?

We frame the chain of reasoning as a solution to the challenge presented in section 2.1.1, namely the difficulty of learning long chains by Euclidean embedding. The idea behind this approach is to restrict the embedding to a sub-poset of $\mathcal{G}$ with a substantially smaller height and attempt to predict a reason $u \to v$ by finding a path between them within the sub-poset. This is formalized in the following:

Definition 3. *For an inference graph $\mathcal{G} = (V, E)$, we say that a pair (E_s, E_*) of subsets of E is an $s-$decomposition if E_s is a poset of height less than s and the transitive closure of $E_\mathrm{s} \cup E_*$ is E. In other words, each pair $(u, v) \in E$ is connected by a path in $E_\mathrm{s} \cup E_*$.*

depends purely Based on the above, the role of the LM in a chain of reasoning is to embed only the edges in E_s. The reasons in E_* are manually imposed. This may include various operations. For example, $(u, v) \in E_*$ may mean that v is obtained by appending to u particular prompts or external information retrieved by querying u. Note that the existence of E_* is essential, as it breaks the long chains into smaller pieces covered by E_s. This observation suggests that **prompt engineering, information retrieval, and other hand-crafted actions play a critical role in the success of the chain of reasoning**.

[1] For example, $c = 4$ works.

3 PHYRR FRAMEWORK

Our theory in Section 2 suggests two sets of basic principles for a suitable chain of reasoning. These principles are the basis of the PHyRR framework and are detailed below:

Ease of Training: The main goal of the decomposition (E_{s}, E_*) is to make sure that training an embedding for E_{s} (fine-tuning by RL) is easier than the entire graph. Theorem 1 suggests controlling three factors to achieve this:

1. The effective noise σ should be reduced.
2. The edge density ρ should be reduced by selecting fewer edges in E_{s}.
3. The margin ϵ is increased.

Ease of implementation: The strategy of decomposing the graph into (E_{s}, E_*) poses a natural difficulty during inference, one needs to identify a connecting chain in $E_{\mathrm{s}} \cup E_*$ between two given nodes. This problem is resolved by making sure that such a chain is unique. In other words, one ideally requires that every node in $E_{\mathrm{s}} \cup E_*$ has at most one direct predecessor and hence $E_{\mathrm{s}} \cup E_*$ consists of parallel paths that we refer to as *trajectories*. In this case, an LLM can implement not only the edges in E_s but an entire trajectory by invoking edges in E_*, as external actions through generating special tokens.

PHyRR realizes these principles through three key components. The *planning component* (section 3.1) addresses ease of training by defining short, atomic sub-goals before reasoning begins. In this component, a plan is generated once, at the start of the trajectory, and then held fixed as an anchor for the retrieval and reasoning that follow. In this way, it keeps E_s small at each step and also increases the margin ϵ by inducing shorter segments. The planning component also addresses ease of implementation by shifting the entire reasoning chain to a region of the underlying inference graph with reduced uncertainty, hence unique chains.

After planning, PHyRR applies an *iterative and adaptive retrieval-reasoning process* (Section 3.2), solely conducted by an LLM. The underlying retrieval mechanism (Section 3.2.1) provides a key part of the externally imposed edges E_*: each retrieved fact becomes a connecting edge in it. This process addresses ease of implementation too: at each step, the RL-trained policy (i.e., the LLM) automatically decides either to generate a query and actuate external retrieval (E_*), or to undergo internal reasoning (E_s). This mixed process of retrieval and reasoning is what RL optimizes (Section 3.3). The third component of PHyRR is accordingly a *multi-component reward* (Section 3.3.1) toward low noise, low density, and high margin as described above. We describe these component in the following sections.

3.1 PLANNING

The planning component in PHyRR breaks each query into a small number of short sub-goals. Then, the model is not required to discover such a decomposition itself during the reasoning process, which reduces the probability of a wrong chain of reasoning and directly enhances the model's confidence at each step. This component also addresses a practical problem in multi-hop QA, known as the cold-start effect, where a model starts a limited exploration without a full overview of the question. As such, planning decreases initial uncertainty by avoiding guessing a possible reasoning trajectory. We describe the planning component as a randomized map $\mathrm{plan}_{\mathrm{I}}(q) = \{p_1, p_2, \ldots, p_M\} \sim P^{(0)}(\cdot \mid q)$ of the query q. This map is implemented by an LLM and a prompt $\mathcal{P}_{\mathrm{plan}}$ (Appendix E), instructing it to analyze the question and break it down into small parts, together with a high-level roadmap. All sub-goals are constructed to be semantically atomic, i.e., focusing on only one reasoning intention, non-redundant, and sufficiently abstract. This plan acts as a constant anchor for the entire trajectory, including every reasoning and retrieval step, assisting them in deciding the next step better.

3.2 ITERATIVE RETRIEVAL–REASONING THROUGH PLAN

Now, we focus on the chain of reasoning and retrieval, which we define as a sequence $\tau := \{y_t\}$ of *links* y_t. Each link $y_t = (\lambda_t, s_t)$ at an iteration $t = 0, 1, \ldots$ consists of a generated string s_t and an index $\lambda_t \in \{\texttt{plan}, \texttt{think}, \texttt{retrieve}, \texttt{answer}\}$ describing how it is generated. A trajectory is a realization of τ. Further, we denote by τ_t the trajectory generated up to (not including)

time t. In particular, $\tau_0 = \{\}$. As explained in Section 3.1, PHyRR initializes a trajectory by generating a plan plan_I. The trajectory is terminated after the index `answer` is invoked. Accordingly, at the termination time T, we set $\lambda_T = \texttt{answer}$ and use s_T as the final answer. In the middle of the trajectory, i.e., $t = 1, 2, \ldots, T-1$, the indices `think, retrieve` are utilized. Given a query q, the generated plan plan_I, and an external knowledge source $\mathcal{GH}$, the trajectory τ is sequentially generated by the LLM with trainable weights θ, under the prompt $\mathcal{P}_{\mathrm{AgenticQA}}$ (Figure 9, Appendix E). We denote this procedure by $P_\theta(\tau \mid q, \mathrm{plan}_\mathrm{I};\ \mathcal{GH})$, emphasizing its dependence on the plan, the LLM weights θ, and the knowledge hypergraph $\mathcal{GH}$ for retrieval. Here, $\mathcal{GH}$ is a structured knowledge source built over the external corpus, whose construction and use in retrieval we describe in Section 3.2.1. The sequential nature of the trajectory is written as:

$$P_\theta(\tau \mid q; \mathrm{plan}_I, \mathcal{GH}) = \prod_{t=1}^{T} \pi_\theta(y_t \mid \tau_t; \mathcal{GH}), \tag{2}$$

where $\pi_\theta(y_t \mid \tau_t; \mathcal{GH})$ describes how an individual link y_t is generated from the previous links in τ_t. A key aspect of the relation in equation 2 is that it allows the reasoning steps with $\lambda_t = \texttt{think}$ to actively invoke and control retrieval steps with $\lambda_t = \texttt{retrieve}$, while the retrieved information reshapes subsequent reasoning steps with $\lambda_t = \texttt{think}$ or $\lambda_t = \texttt{answer}$. This reflects the core aspect of the PHyRR, namely the progressive exploration of the structured knowledge in a dynamic and plan-aware that supports multi-hop reasoning. Depending on the index λ_t of each link y_t, the description of the policy π_θ is different and corresponds to one of the two above mechanisms. During the reasoning steps with $\lambda_t = \texttt{think}$ or $\lambda_t = \texttt{answer}$, the LLM further generates s_t as a string following (τ_t, λ_t). However, for the retrieval steps with $\lambda_t = \texttt{retrieve}$, the LLM is interrupted and $s_t = c_t$ is generated by equation 3 where $r_{<t}$ is the collection of all reasoning links before t, i.e., all $y_{t'}$ at $t' < t$ with $\lambda_{t'} = \texttt{think}$. We describe this stage in Section 3.2.1.

$$c_t \sim P_{\mathrm{ret}}(c_t \mid r_{<t},\ q;\ \mathrm{plan}_\mathrm{I}, \mathcal{GH}). \tag{3}$$

3.2.1 Hypergraph Retrieval based on Plan

As previously mentioned, the retrieval mechanism targets the key component of breaking long reasoning chains with externally imposed edges (E^*). In this way, it reduces the internal chain of reasoning required by LLMs and increases their margin. This is what makes retrieval essential to PHyRR framework. In PHyRR, we retrieve information from a knowledge hypergraph rather than text or a graph, because a hypergraph can support higher-order relationships. This allows a single retrieval step to replace multiple steps based on distinct queries and reduce the chain even further. We build a knowledge hypergraph $\mathcal{GH} = (V, E)$, where $V = \{v_1,\ v_2,\ \ldots, v_n\}$ is a set of nodes representing entities or concepts, and $E = \{e_1, e_2\ \ \ldots, e_m\}$ is the set of hyperedges representing higher-order relations. In particular, each hyperedge $e_j \in E$ is described by $e_j = (s_j, \ell_j)$, where $s_j = \{v_{j,1},\ v_{j,2},\ \ldots, v_{j,l_j}\} \subseteq V$ is a non-empty subset of participating nodes, ℓ_j is the natural language description of the relation e_j represents. With an abuse of notation, we refer to the first component (node subset) of a hyperedge e by s_e. For a given corpus $\mathcal{K}$, we construct $\mathcal{GH}$ using an LLM(Appendix D). To reduce ambiguity from synonymous or overlapping facts, we merge semantically equivalent entities and redundant relations (details in Appendix D).

Then, to improve multi-step reasoning, we retrieve from the knowledge hypergraph $\mathcal{GH}$ within the reasoning chain, conditioning on both the reasoning so far and the plan. We explain this procedure below. In this part, we frequently refer to a given, but arbitrary similarity function $\mathrm{sim}(.,.)$ between real-valued vectors. In retrieval states with $\lambda_t = \texttt{retrieve}$, the system requests retrieval by generating a new query q_t based on the reasoning states $r_{<t}$, the plan and the input query q:

$$q_t \sim P_{\text{query}}(q_t \mid r_{<t},\ q; \mathrm{plan}_\mathrm{I}). \tag{4}$$

Then, we use the query q_t to retrieve the response c_t from the hypergraph $\mathcal{GH}$:

$$c_t \sim P_{\mathrm{extract}}(c \mid q_t; \mathrm{plan}_\mathrm{I}, \mathcal{GH}). \tag{5}$$

Note that the distribution P_{ret} in equation 3 coincides with the marginalization of the product $P_{\mathrm{extract}} P_{\mathrm{query}}$, given in equation 4 and equation 5, on the locally generated query q_t. The retrieved knowledge c_t provides relevant entities and hyperedges to the current state of reasoning, and is incorporated in generating the next reasoning states. While existing methods rely on entity expansion and hyperedge matching (Luo et al., 2025; 2026), we extend this process by incorporating planning to guide retrieval. Specifically, we adapt both components to the plan plan_I in addition to the query q_t.

This transforms retrieval from a static similarity search into a dynamic process based on a plan. So, we first select the most relevant sub-plan p^* to q_t based on $p^* = \arg\max_{i \in \{p_1,\ldots,p_M\}} \ \mathrm{sim}\big(\phi(q_t), \phi(p_i)\big)$. Next, the query undergoes two parallel process paths, which are then fused:

(I) Plan-Guided Entity-Based Hyperedge Retrieval: We first identify a set of top-ranked entities $\mathrm{Ret}^{\mathrm{V}}(q_t)$ from V based on their similarity to V_{q_t}, weighted with the selected sub-plan p^*:

$$\mathrm{Ret}^{\mathrm{V}}(q_t) = \arg\overset{k}{\max}_{v \in V}\Big(\mathrm{sim}\big(\phi(V_{q_t}), \phi(v)\big) \cdot \psi_V(v, p^*)\Big), \tag{6}$$

where $\phi(V_{q_t})$is the aggregated embedding of the entities extracted from q_t, and $\psi_V(v, p^*)$ is a softmax normalized similarity between v and p^* that promotes selecting entities well aligned with the plan. We use $\mathrm{Ret}^{\mathrm{V}}(q_t)$ to collect a sorted list $\mathrm{Fact}^{\mathrm{V}}{}_{q_t}$ of related hyperedges to q_t:

$$\mathrm{Fact}^{V}_{q_t} = \bigcup_{v_i \in \mathrm{Ret}^{\mathrm{V}}(q_t)} \{e \in E \mid v_i \in s_e, \ \}, \tag{7}$$

(II) Plan-Guided Direct Hyperedge Retrieval: We directly retrieves a sorted list $\mathrm{Fact}^{\mathrm{E}}(q_t)$ of hyperedges based on their similarity to the query q_t and the current sub-plan p^* in equation 8, where $\psi_H(e, p^*)$ is another softmax normalized weight between the hyperedge and the current sub-plan.

$$\mathrm{Fact}^{\mathrm{E}}(q_t) = \arg\overset{k}{\max}_{e \in E}\Big(\mathrm{sim}\big(\phi(q_t), \phi(e)\big) \cdot \psi_H(e, p^*)\Big), \tag{8}$$

To produce the retrieved knowledge c_t, that are most relevant to the reasoning and the plan, we use a reciprocal rank $\mathrm{Score}(f \mid p^*) = \sum_{j \in \{\mathrm{V},\mathrm{E}\}} \frac{1}{\mathrm{rank}_j(f)}$, to combine $\mathrm{rank}_{\mathrm{V}}(f)$ and $\mathrm{rank}_{\mathrm{E}}(f)$ which denote the rank of the fact f in the plan-based retrieval lists $\mathrm{Fact}^{V}_{q_t}$ and $\mathrm{Fact}^{E}_{q_t}$, respectively.

3.3 Policy Optimization via GRPO

Finally, we explain training PHyRR, which is done by tuning the weights θ of the LLM. For this reason, we define a trajectory-level reward function $R(\tau)$, reflecting the overall quality of a trajectory including the correctness of the final answer s_T, the coherence and usefulness of intermediate reasoning steps, and the relevance and effectiveness of the retrieved information. We describe this reward function in Section 3.3.1. Training entails maximizing the expected reward of the trajectory:

$$\max_{\theta} \ \mathbb{E}_{\tau \sim P_\theta(\tau \mid q; \mathrm{plan}_{\mathrm{I}}, \mathcal{GH})}\big[R(\tau)\big] - \beta\, \mathcal{D}_{\mathrm{KL}}[\pi_\theta \,\|\, \pi_{\mathrm{ref}}], \tag{9}$$

where the expectation is also taken over $q \sim \mathcal{D}, \ \mathrm{plan}_{\mathrm{I}} \sim P^{(0)}(\mathrm{plan}_{\mathrm{I}} \mid q)$ with $\mathcal{D}$ denoting the query distribution. The term $-\beta\, \mathcal{D}_{\mathrm{KL}}[\pi_\theta \,\|\, \pi_{\mathrm{ref}}]$ is a common regularization based on the Kullback–Leibler divergence $\mathcal{D}_{\mathrm{KL}}$ of the distributions generated by the trained LLM and a reference LLM. To solve the problem in equation 9, we use Group Relative Policy Optimization (GRPO) Shao et al. (2024) by framing equation 9 as an RL problem with π_θ in equation 2 as the policy and τ_t as the states.

3.3.1 Multi-Component Reward Design

We introduce a multi-component reward function $R(\tau)$ to evaluate retrieval and reasoning throughout a trajectory τ in equation 10, where $(\omega_i)_i$ denotes the weights of the sub-rewards.

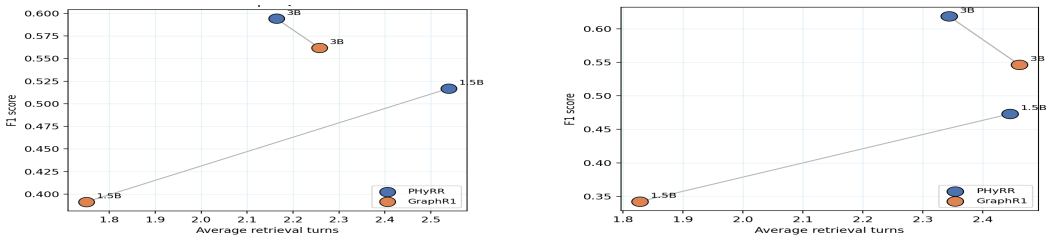
$$R(\tau) = \omega_1 R_{\text{Answer}}(\tau) + \omega_2 R_{\text{Format}}(\tau) + \omega_3 R_{\text{Reasoning}}(\tau) + \omega_4 R_{\text{Retrieval}}(\tau) \tag{10}$$

Answer Reward: We define the answer reward R_{Answer} as the F1 score between the predicted answer a generated in τ and the gold answer a_*, which is widely used for evaluating the predicted answer.

Format Reward: We introduce the format reward R_{Format} to enforce a structural output format as:

$$R_{\text{Format}}(\tau) = \frac{1}{3}\left(\frac{1}{T}\sum_{t=1}^{T} I_t^{\mathrm{T}} + \frac{1}{\max(1, T_{\mathrm{R}})}\sum_{t=1}^{T} I_t^{\mathrm{R}} + I_t^{\mathrm{A}}\right), \tag{11}$$

where T_{R} is the number of retrieval steps. Here, I_t^{T}, I_t^{R}, and I_t^{A} are indicator functions that respectively check whether step t has a well-formed `<think>...</think>` tags, a valid `<query>..</query>` and `<Information>...</Information>` pairs, and exactly one well-formed `<answer>...</answer>`, returning 1 if true and 0 otherwise.



(a) HotpotQA Dataset (b) 2Wiki Dataset

Figure 1: PHyRR vs GraphR1 on Qwen2.5-3B and 1.5B in term of retrieval turns.

Reasoning and Retrieval Rewards: We introduce the reasoning and retrieval rewards in Eqs.12 and 13, respectively. If $R_{Answer} = 1$, we set the these rewards to 1; otherwise, we use their computed values. Let reasoning trajectory be $\tau_{\text{think}} = \{r_1, \ldots, r_T\}$ consisting of all links y_t with $\lambda_t = \texttt{think}$, K_t denotes the retrieved knowledge at step t, $K_R = \bigcup_t K_t$ the total retrieved knowledge, Q_t the query at step t, P the reasoning plan and $V(\cdot)$ the set of entities extracted from a text block.

$$R_{\text{Reasoning}} = \gamma.\left(I_{sgn}[\,|\tau_{\text{think}}| \in [S_{\min}, S_{\max}]\,] + I[\mathcal{L} \cap \tau_{\text{think}} \neq \emptyset] - I\left[\frac{|\mathcal{V}(\tau_{\text{think}}) \setminus \mathcal{V}(K_R)|}{|\mathcal{V}(\tau_{\text{think}})|} > 0.5\right]\right) \tag{12}$$

$$R_{\text{Retrieval}} = \gamma.\left(I_{sgn}[a_* \cap K_R \neq \emptyset] + I\left[\left(\frac{1}{T_R}\sum_{t=1}^{T_R}\frac{|Q_t \cap (P \cup K_{t-1})|}{|Q_t|)}\right) > 0.5\right]\right) \tag{13}$$

where $I_{sgn}[.]$ is a bipolar indicator that gives +1 if true and -1 otherwise, $I[.]$ is the standard indicator returning 1 if true and 0 otherwise, and γ is a scaling hyperparameter. In reasoning reward, $I[\mathcal{L} \cap \tau_{\text{think}} \neq \emptyset]$ shows the reasoning trajectory contains at least one keyword from the predefined logical/verification vocabulary set $\mathcal{L}$. The last term is a hallucination penalty that prevents the generation of unsupported facts that do not exist in the retrieved knowledge. In the retrieval reward, the first term checks whether the gold answer overlaps with the retrieved knowledge or not.The second term checks if each query follows the plan and previously retrieved information by measuring how well it connects to them.

4 Experimental Evaluation

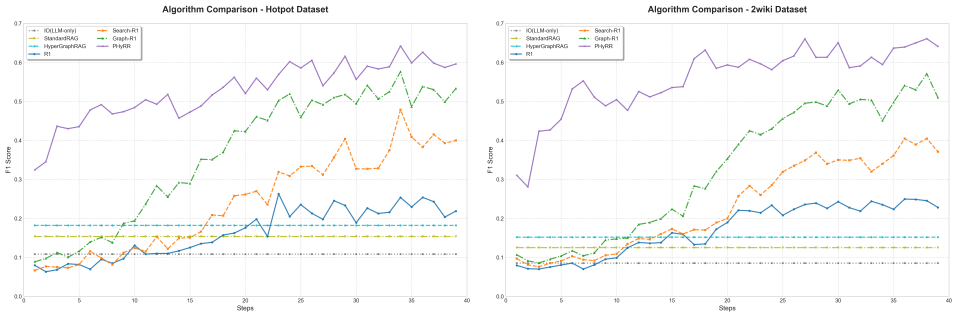
This section presents evaluation of PHyRR through three research questions: (RQ1) Does it outperform QA baselines? (RQ2) How do its components contribute to performance? (RQ3) Does plan improve early-stage performance and reduce cold-start issues?

4.1 Performance Results (RQ1)

We compared the performance of PHyRR with baselines shown in Table 1. Based on these results PHyRR achieves the best performance. We can highlight the following findings: First, prompt-engineered models struggle most with multi-hop reasoning, while RL-based methods improve markedly. Second, results show that methods using structured knowledge for retrieval generally outperform text-based retrieval, confirming the value of higher-order relationships; PHyRR improves further still, showing structure alone is not enough. Third, RL-based methods without an effective approach fail to achieve high results, while PHyRR, which combines RL with structured interaction among planning, retrieval, and reasoning and uses a multi-component reward, is consistently better. We also compare PHyRR with the best baseline, GraphR1, in Figure 1 in terms of retrieval turns to see whether its accuracy gains come from retrieving more efficiently or not. On the Qwen2.5-3B, PHyRR achieves a higher F1 with fewer retrieval turns on both datasets, indicating more efficient retrieval. On the Qwen2.5-1.5B, PHyRR uses more retrieval turns, suggesting that the smaller model relies on more additional retrieval to compensate for its weaknesses. We also examine how the models improve step-by-step in Figure 2 . We find that PHyRR gains performance much faster in the early stages, confirming that it effectively avoids the cold-start problem (Section 4.3).

Table 1: The results of different methods on multi-hop QA benchmarks. "–" mean no knowledge interaction while *Text*, and *Graph* represent text-based and graph-based knowledge, respectively.

Method	Category		Multi-Hop QA				Single-Hop QA				AVG	
	Prompt/RL	Knowledge	HotpotQA		2Wiki		PopQA		TriviaQA		Overall	
			EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Qwen2.5-3B-Instruct												
IO (LLM-only)	Prompt	-	4.68	10.82	0.78	8.52	2.34	10.47	3.12	13.08	2.73	10.72
StandardRAG	Prompt	Text	6.37	15.33	1.82	12.49	4.68	14.69	7.03	24.04	4.98	16.64
HyperGraphRAG	Prompt	Graph	7.03	18.19	3.51	15.18	7.81	16.73	9.63	26.82	7.00	19.23
R1	RL	-	17.10	26.27	19.81	24.94	14.32	27.62	39.34	47.50	22.64	31.58
Search-R1	RL	Text	30.10	45.86	28.54	40.46	29.10	36.83	42.50	49.72	32.56	43.22
Graph-R1	RL	Graph	49.06	56.17	48.96	54.63	37.50	45.14	53.91	62.06	47.36	54.50
PHyRR	RL	Graph	**52.34**	**59.41**	**57.03**	**61.86**	**42.97**	**47.94**	**56.59**	**65.85**	**52.23**	**58.77**
Qwen2.5-1.5B-Instruct												
IO (LLM-only)	Prompt	-	3.12	6.49	0.56	7.42	1.56	7.27	3.12	6.82	2.09	7.00
StandardRAG	Prompt	Text	3.90	8.54	1.56	11.32	4.03	12.73	7.81	16.8	4.33	12.35
HyperGraphRAG	Prompt	Graph	4.17	10.09	7.81	13.78	3.61	11.67	7.04	15.41	5.66	12.74
R1	RL	-	13.49	20.11	17.15	26.73	18.86	20.67	21.45	35.04	17.74	25.64
Search-R1	RL	Text	31.25	38.49	18.03	29.4	31.09	39.19	33.75	45.27	28.53	38.09
Graph-R1	RL	Graph	35.16	39.14	28.90	34.23	35.93	42.67	46.09	55.65	36.52	42.92
PHyRR	RL	Graph	**45.31**	**51.69**	**41.41**	**47.30**	**40.62**	**45.94**	**53.91**	**62.88**	**45.31**	**51.95**



(a) HotpotQA Dataset (b) 2Wiki Dataset

Figure 2: Step-wise F1 score on multi-hop datasets based on Qwen2.5-3B.

4.2 Ablation Studies (RQ2)

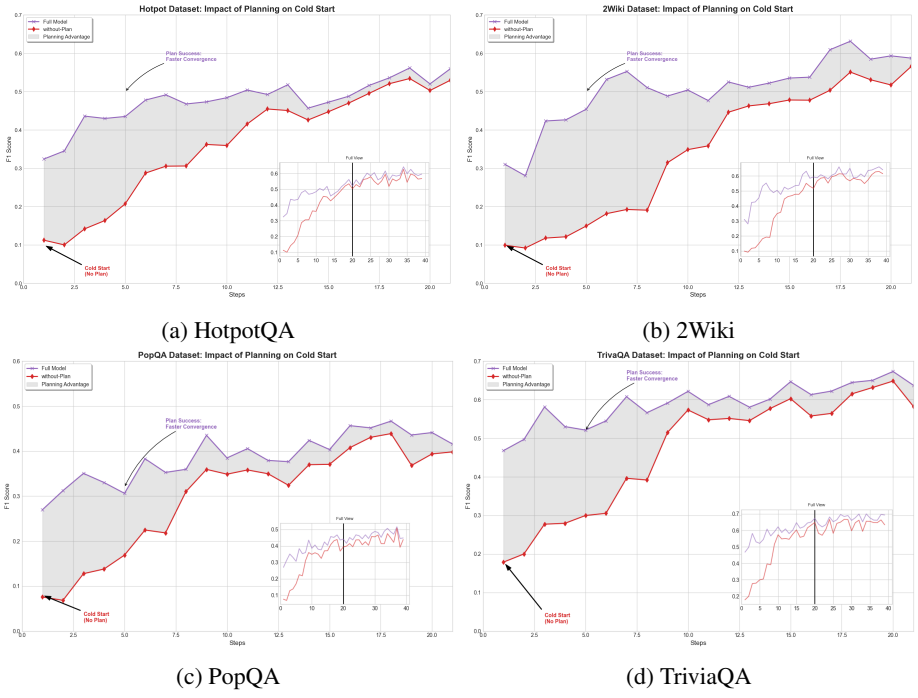
Planning and the multi-component reward are motivated by ease of training (Section 3). Here, we test how critical these two components are in the PHyRR design. We start from the full model and incrementally disable planning and the components of the reward, as shown in Table 2. Removing the plan decreases the performance, particularly on multi-hop tasks, exactly where longer chains make the margin problem worse. Replacing the multi-component reward with simple outcome and format rewards shows that performance drops as well. This result confirms that planning and the multi-reward are necessary, not just theoretically motivated. Although the impact of these components is visible across all models, it is especially strong for the Qwen2.5-1.5B, suggesting that models with less capacity depend more on both components. For example, on 2Wiki, the Qwen2.5-1.5B model's EM drops from 41.41 to 28.9, much sharper than the drop seen on the Qwen2.5-3B.

4.3 Planning and Cold Start Analysis (RQ3)

To address RQ3, we compare PHyRR framework against its version without the planning component during learning. As shown in Figure 3, the version without the plan shows a clear cold-start problem: F1 is low for early steps, while PHyRR starts much higher. e.g., on PopQA (Figure 3c), the PHyRR reaches 0.30–0.35 F1 at steps 3–4, while its version without planning needs until steps 8–9. Planning helps small backbones more; e.g., averaged F1 across datasets shows that removing the plan drops performance by 3.12% for Qwen2.5-3B and 6.09% for Qwen2.5-1.5B. So while Qwen2.5-3B has higher overall performance, Qwen2.5-1.5B performance relies more on planning. We also checked

Table 2: Ablation study of PHyRR. PL = Plan, MR = Multi-reward, HG = Hypergraph. Parenthetical values (Δ) show the gain from adding each component on top of the row above.

Method	Components	Multi-Hop QA				Single-Hop QA				Avg	
		HotpotQA		2Wiki		PopQA		TriviaQA		Overall	
		EM	F1	EM	F1	EM	F1	EM	F1	EM/Δ	F1/Δ
Qwen2.5-3B-Instruct											
w/o PL, w/o MR	(HG)	49.59	56.31	49.09	56.72	37.50	44.85	53.47	64.22	47.41(–)	55.53(–)
w/o PL	(MR + HG)	51.62	58.84	53.90	60.72	42.18	47.61	54.69	64.21	50.60(+3.19)	57.85(+2.32)
PHyRR (Full)	(PL+MR+HG)	**52.34**	**59.41**	**57.03**	**61.86**	**42.97**	**47.94**	**56.59**	**65.85**	52.23(+1.64)	58.77(+0.92)
Qwen2.5-1.5B-Instruct											
w/o PL, w/o MR	(HG)	41.41	47.99	28.90	36.44	36.72	44.17	50.01	56.67	39.26(–)	46.32(–)
w/o PL	(MR + HG)	42.18	49.54	38.27	46.83	38.28	44.51	51.47	60.47	42.55(+3.29)	50.34(+4.02)
PHyRR (Full)	(PL+MR+HG)	**45.31**	**51.69**	**41.41**	**47.30**	**40.62**	**45.94**	**53.91**	**62.88**	45.31(+2.76)	51.95(+1.62)



(a) HotpotQA (b) 2Wiki

(c) PopQA (d) TriviaQA

Figure 3: Impact of planning in terms of F1 on Qwen2.5-3B.

whether generated queries during learning in PHyRR actually follow the plan or just ignore it, so we measure word overlap between each generated query and its best-matching plan. The results show that average alignment is 75%, and 70% for Qwen2.5-3B and Qwen2.5-1.5B, respectively. This confirms the plan actually acts as an anchor rather than being ignored after the first step.

5 Conclusion

In this work, we studied why multi-hop reasoning is hard for LMs. We framed this question as a graph representation problem, where we assumed statements as nodes and valid reasoning steps as edges. We showed that while long chains can be represented, the confidence of the model that separates correct links from wrong ones shrinks as chains increase, making them easier to learn wrong. This motivated PHyRR, which includes planning followed by an adaptive retrieval-reasoning process. PHyRR is optimized with RL, based on a theoretically grounded multi-component reward. The planning component creates a plan that breaks a query into short sub-goals and then performs hypergraph retrieval and reasoning adaptively based on the resulting plan. Experimental results showed the superiority of PHyRR over baselines across four QA benchmarks.

AI USE STATEMENT

In this work, we used generative AI tools for language polishing and improving manuscript readability. We also used generative AI tools for some code generation. We have reviewed all AI-assisted work.

REFERENCES

Florent Benaych-Georges and Raj Rao Nadakuditi. The singular values and vectors of low rank perturbations of large rectangular random matrices. *Journal of Multivariate Analysis*, 111:120–135, 2012.

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 2318–2335. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.137. URL https://aclanthology.org/2024.findings-acl.137/.

Mingyang Chen, Linzhuang Sun, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. Research: Learning to reason with search for llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2503.19470.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, et al. DeepSeek-R1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025a. doi: 10.1038/s41586-025-09422-z. URL https://www.nature.com/articles/s41586-025-09422-z.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. LightRAG: Simple and fast retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 10746–10761. Association for Computational Linguistics, 2025b. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.568. URL https://aclanthology.org/2025.findings-emnlp.568/.

Haoyu Han, Li Ma, Yu Wang, Harry Shomer, Kai Guo, Yongjia Lei, Zhisheng Qi, Zhigang Hua, Bo Long, Hui Liu, Charu Aggarwal, and Jiliang Tang. Rag vs. graphrag: A systematic evaluation and key insights. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '26, pp. 8966–8977. Association for Computing Machinery, 2026. ISBN 9798400722592. doi: 10.1145/3770855.3817575. URL https://doi.org/10.1145/3770855.3817575.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625. International Committee on Computational Linguistics, 2020. doi: 10.18653/v1/2020.coling-main.580. URL https://aclanthology.org/2020.coling-main.580/.

Yu Huang, Zixin Wen, Aarti Singh, Yuejie Chi, and Yuxin Chen. Transformers provably learn chain-of-thought reasoning with length generalization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=aE0bCvXXBt.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, et al. OpenAI o1 System Card, 2026. URL https://arxiv.org/abs/2412.16720.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Rwhi91ideu.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual*

Meeting of the Association for Computational Linguistics, pp. 1601–1611. Association for Computational Linguistics, 2017. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147/.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474. Curran Associates Inc., 2020.

Qiwei Li, Zuchao Li, Ping Wang, Haojun Ai, and Hai Zhao. Hypergraph based understanding for document semantic entity recognition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 2950–2960. Association for Computational Linguistics, 2024a. doi: 10.18653/v1/2024.acl-long.162. URL https://aclanthology.org/2024.acl-long.162/.

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. Chain of thought empowers transformers to solve inherently serial problems. In *International Conference on Learning Representations*, volume 2024, pp. 11911–11943, 2024b. URL https://icml.cc/virtual/2023/27237.

Haoran Luo, Haihong E, Guanting Chen, Yandan Zheng, Xiaobao Wu, Yikai Guo, Qika Lin, Yu Feng, Zemin Kuang, Meina Song, Yifan Zhu, and Anh Tuan Luu. HyperGraphRAG: Retrieval-augmented generation via hypergraph-structured knowledge representation. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 152206–152234. Curran Associates, Inc., 2025. doi: 10.52202/085713-5089. URL https://www.proceedings.com/085713-5089.html.

Haoran Luo, Haihong E, Guanting Chen, Qika Lin, Yikai Guo, Fangzhi Xu, Zemin Kuang, Meina Song, Xiaobao Wu, Yifan Zhu, and Anh Tuan Luu. Graph-R1: Towards agentic graphrag framework via end-to-end reinforcement learning. In *Proceedings of the 43rd International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2026. URL https://icml.cc/virtual/2026/poster/63269.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pp. 9802–9822. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-long.546. URL https://aclanthology.org/2023.acl-long.546/.

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

Aditi Singh, Abul Ehtesham, Saket Kumar, Tala Talaei Khoei, and Athanasios V. Vasilakos. Agentic retrieval-augmented generation: A survey on agentic rag, 2026. URL https://arxiv.org/abs/2501.09136.

Qwen Team. Qwen2.5: A party of foundation models, 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22. Curran Associates Inc., 2022. ISBN 9781713871088.

Yuyang Wu, Yifei Wang, Ziyu Ye, Tianqi Du, Stefanie Jegelka, and Yisen Wang. When more is less: Understanding chain-of-thought length in LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=6QDFsYxtI1.

Zhishang Xiang, Chuanjie Wu, Qinggang Zhang, Shengyuan Chen, Zijin Hong, Xiao Huang, and Jinsong Su. When to use graphs in rag: A comprehensive analysis for graph retrieval-augmented generation. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust

(eds.), *International Conference on Learning Representations*, volume 2026, pp. 66145–66178, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/6c9e01d6cefbbf4cdd265032550e767f-Paper-Conference.pdf.

Tong Yang, Yu Huang, Yingbin Liang, and Yuejie Chi. Multi-head transformers provably learn symbolic multi-step reasoning via gradient descent. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=qFC728XyeM.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380. Association for Computational Linguistics, 2018. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

Chuanyue Yu, Kuo Zhao, Yuhan Li, Heng Chang, Mingjian Feng, Xiangzhe Jiang, Yufei Sun, Jia Li, Yuzhi Zhang, Qingyun Sun, Jianxin Li, and Ziwei Zhang. Graphrag-r1: Graph retrieval-augmented generation with process-constrained reinforcement learning. In *Proceedings of the ACM Web Conference 2026*, WWW '26, pp. 1398–1409. Association for Computing Machinery, 2026. ISBN 9798400723070. doi: 10.1145/3774904.3792589. URL https://doi.org/10.1145/3774904.3792589.

A Related Work

Retrieval-Augmented Generation (RAG): RAG is one of the most used approaches that allows using external information sources to increase the capacity of LLMs Lewis et al. (2020). Several improvements were proposed over time, ranging from one-shot retrieval to iterative retrieval approaches.Recently, agent-based methods let models decide how to use retrieval during generation Jin et al. (2025) Singh et al. (2026). Among recent approaches, using structured retrieval has gained attention Han et al. (2026). Some structured retrieval methods known as Graph Retrieval-Augmented Generation (GraphRAG) retrieve form knowledge graphs Xiang et al. (2026) Guo et al. (2025b) There are many variations of GraphRAG, including different architectures, richer information types Luo et al. (2025) Luo et al. (2026).

Reasoning in Large Language Models: Reasoning is an essential ability in LLMs, and the current research has led to significant improvements. Some early research used chain-of-thought (CoT) prompting Wei et al. (2022), where models learn to solve a problem by reasoning through generating the thinking sequence before the final answer. In addition, some recent work moved towards training-based approaches, including OpenAI Jaech et al. (2026), DeepSeek-R1Guo et al. (2025a), which illustrate how RL can significantly improve the step-by-step reasoningChen et al. (2025).

RL for Retrieval-Augmented Reasoning: Recently, researchers have worked to improve the performance of LLMs using RL.Recent studies have shown that RL algorithms such as Proximal Policy Optimization (PPO) or Group Relative Policy Optimization (GRPO) Shao et al. (2024) can improve the retrieval and reasoning capabilities in LLMs Chen et al. (2025)Guo et al. (2025a). Recently, some RL agents have been developed to model interactions among retrieval and reasoning Guo et al. (2025a)Jin et al. (2025).Some frameworks further extend this idea to structured or graph-based settings, such as GraphR1 Luo et al. (2026) and GraphRAG-R1Yu et al. (2026), highlighting the potential of RL for GraphRAG-style systems.

Theoretical Understanding of Reasoning in LLMs: Although LLMs have achieved strong empirical performance, important questions about the mechanisms multi-step reasoning remain unanswered. The theoretical works have examined this from different angles. In Wu et al. (2026), error accumulation is analyzed, showing how errors propagate and how accuracy follows chain length. The language of circuit complexity is used to examine the capability of transformers in reasoningLi et al. (2024b) .Other works analyze gradient descent and show that transformers learn multi-step reasoning such as path-finding on trees or state-tracking Yang et al. (2025)Huang et al. (2025).

B Canonical embedding and Low-Rank Decomposition

In this part, we elaborate on the discussion in section 2.1 by showing that canonical embeddings are tightly connected to the SVD of suitably weighted adjacency matrices. To begin with, consider an arbitrary embedding $(\mathrm{k},\ \mathrm{q})$ of a graph $\mathcal{G}$ with dimension m and define its *associated weight matrix* $A = (\langle \mathrm{q}(u), \mathrm{k}(v)\rangle)_{u,v}$. Scale A such that $\|A\|_{\mathrm{F}}^2 = N$. This makes A comparable to the unweighted adjacency matrix of $\mathcal{G}$. Note that by construction, A has a rank of at most m. Then, take $A = U\Sigma V^T$ as its truncated SVD with only m largest singular values kept (the rest are zero). This means that U, V are $n \times m$. Now, we may take the rows of $\sqrt{n}U$ and $\sqrt{n}V\Sigma$ as another set of keys and queries associated with the scaled weights A and verify that they satisfy the relations in equation 1, because $U^TU = V^TV = I$ and $Tr(\Sigma^2) = \|A\|_{\mathrm{F}}^2 = N$. This construction shows that requiring a canonical embedding is not more restrictive than requiring an arbitrary embedding, as they are associated with the same scaled weights A. Specifically, every $m-$dimensional canonical embedding with a given margin ϵ is obtained by the truncated SVD of a suitable map A given by the following:

Definition 4. *We say that a map $A = V \times V \to \mathbb{R}$ is $m-$***compatible** *with the graph $\mathcal{G}$ if A has rank at most m (as a square matrix), $\|A\|_{\mathrm{F}}^2 = N$, and $A(u, v) > 0$ for every edge and $A(u, v) \leq 0$ for nonedges. We refer to the smallest value of $A(u, v)$ for the edges as the* **margin** *of A.*

In the next definition, we also consider a more general case with a non-trivial truncation step, which is more likely to appear in practice:

Definition 5. *For a map $A = V \times V \to \mathbb{R}$ with SVD $A = U\Sigma V^T$ (as a square matrix), we refer to $A'_m = U\Sigma'_m V^T$ as the $m-$***spectral truncation** *of A, where Σ'_m retains the m largest diagonals of Σ and sets the rest to zero. Then, we say that A is $m-$***adaptable** *to the graph $\mathcal{G}$ if $\|A\|_{\mathrm{F}}^2 = N$*

and $A'_m(u,v) > 0$ for every edge and $A'_m(u,v) \leq 0$ for nonedges. We refer to the smallest value of $A'_m(u,v)$ for the edges as the **margin** *of A.*

Note that in this extended definition, the $m-$spectral truncation A'_m becomes an $m-$compatible map only after a suitable scaling.

B.1 Statistical Hardness of Learning Embeddings

Language models are trained by noisy observations of the inference graph $\mathcal{G}$. In this section, we establish that despite the large size of $\mathcal{G}$, the existence of a low-rank embedding as predicted by Theorem 2 allows us to learn a fairly accurate estimate of it. However, we show that the noise imposes a shrinkage and perturbation effect on the embedding (k, q) which can cause edges corresponding to a small margin flip. This precisely means that we might get $\langle \mathrm{k}(u), \mathrm{q}(v) \rangle > 0$ for a non edge (hallucination) or $\langle \mathrm{k}(u), \mathrm{q}(v) \rangle \leq 0$ for an edge (missing). We model the observation process as a random map $\hat{A} : V \times V \to \mathbb{R}$ assigning to every pair of nodes a real number, which indicates not only the observed edges or non-edges, but also the unobserved pairs. Our analysis is based on the following assumption:

Assumption 1. *Each observation $\hat{A}(u,v)$ is an independent random variable with the third absolute moment bounded by a universal constant c. Moreover, with $\mathbb{E}[\hat{A}(u,v)] = \frac{1}{\sqrt{N}} A(u,v)$ and Var$[\hat{A}(u,v)] = \frac{\sigma^2}{n}$, where σ is a positive constants and $A(u,v)$ is an $m-$adaptable weight to $\mathcal{G}$ with N edges and $n = o(N)$ nodes. Denoting the ith largest singular value of A by σ_i, we have $\frac{\sigma_i}{\sqrt{N}} \to \bar{\sigma}_i$ for $i \geq m$ with distinct constants $\bar{\sigma}_i$.*

For an illustration of this assumption, see Appendix B.2. While LMs are trained by complex algorithms, we consider a simpler learning scheme by truncating the SVD of $\hat{A}$ (regarded as a matrix). The following result characterizes this method, which can be obtained by slight simplification of the results in Benaych-Georges & Nadakuditi (2012). Hence, the proof is omitted.

Theorem 3. *Suppose that Assumption 1 is satisfied. Take $\Omega = \left(U \in \mathbb{R}^{n\times m}, \{\sigma_i\}_{i\in[m]}, V \in \mathbb{R}^{n\times m}\right)$ as the SVD of $\hat{A}$, truncated to the m largest singular values. Supposed that $\sigma < \bar{\sigma}_i$. Define $\hat{\Omega} = \left(\hat{U} \in \mathbb{R}^{n\times m}, \{\hat{\sigma}_i\}_{i\in[m]}, \hat{V} \in \mathbb{R}^{n\times m}\right)$ as follows:*

$$\hat{U}_i = \kappa_i U_i + \frac{\sigma}{\bar{\sigma}_i \sqrt{n}} G_i, \quad \hat{V}_i = \kappa_i V_i + \frac{\sigma}{\bar{\sigma}_i \sqrt{n}} H_i,$$
$$\hat{\sigma}_i = \bar{\sigma}_i + \frac{\sigma^2}{\bar{\sigma}_i}, \quad \kappa_i^2 = 1 - \frac{\sigma^2}{\bar{\sigma}_i^2} \tag{14}$$

where the subscript i denotes the ith column of a matrix and G_i, H_i are independent standard Gaussian random vectors. Then, for every 1-Lipschitz real function $\Phi(\Omega)$ (under Frobenius norm of matrices and supremum norm of arguments) we have $\left|\mathbb{E}[\Phi(\hat{\Omega})] - \mathbb{E}[\Phi(\hat{\Omega})]\right| \leq \frac{c'}{\sqrt{n}}$ for a universal constant c'.

Theorem 3 characterizes the perturbation of the finitely truncated SVDs, caused by random observations. It shows that, for a sufficiently large number of observations, the singular vectors are corrupted by shrinkage factors κ_i and noise, and the singular values shift due to the observation noise.

B.2 Example of Graph Embedding by SVD

In this part, we elaborate on the example introduced in section 2.1. In this example, we assume that $n = o(N)$ and observe each pair (u,v) with an equal probability $p = \frac{n}{4N\sigma^2}$, and respectively set $\hat{A}(u,v) = p+1, p-1$ whenever an edge or a non-edge (negative sampling) is observed and set $\hat{A}(u,v) = p$ when the pair remains unobserved. Note that for simplicity, there is no noise in observation. It is simple to check that we may write $\hat{A}(u,v) = \frac{n}{2\sigma^2\sqrt{N}} \left(\frac{1}{\sqrt{N}} A + \frac{\sigma}{\sqrt{n}} G\right)$, where A is the adjacency matrix, G is a matrix of independent, centered bounded binary random variables with unit variance. Now, we can express the exact result mentioned in Theorem 1. Suppose that we take the SVD of $\hat{A}$ truncated to the largest m singular values. Then, we may invoke Theorem 3 according

to which the singular vectors and values of $\hat{A}$ are given by $\hat{U}, \hat{V}$ and $\frac{n}{2\sigma^2\sqrt{N}}\hat{\sigma}_i$ in equation 14, respectively. Then, the predicted score $\langle \mathrm{k}(u), \mathrm{q}(v) \rangle$ for an edge is given by the random variable

$$T(u,v) = \sum_{i=1}^{m} \left(\kappa_i U_i(u) + \frac{\sigma}{\bar{\sigma}_i \sqrt{n}} G_i(u) \right) \left(\kappa_i V_i(v) + \frac{\sigma}{\bar{\sigma}_i \sqrt{n}} H_i(v) \right) \left(\bar{\sigma}_i + \frac{\sigma^2}{\bar{\sigma}_i} \right). \tag{15}$$

This leads to the following corollary, which is the exact statement of Theorem 1 in the main body of the paper:

Corollary 1. *The probability of missing an edge (u,v) is given by $\Pr(T(u,v) \leq 0)$ and the probability of hallucinating a nonedge (u,v) is given by $\Pr(T(u,v) > 0)$. These parameters solely depend on the true embedding $(U_i(u), V_i(v))$ and singular values $(\bar{\sigma}_i)$ as well as ρ, σ. In addition, suppose that $\max_{i,u} |U_i(u)| = O(\frac{1}{\sqrt{n}})$ and $\max_{i,u} |V_i(u)| = O(\frac{1}{\sqrt{n}})$. Then, the first order expansion of $T(u,v)$ is give by*

$$T(u,v) = \sum_{i=1}^{m} U_i(u) V_i(v) \bar{\sigma}_i + \frac{\sigma}{n} \Gamma(u,v) + O(\sigma^2), \tag{16}$$

where $\Gamma(u,v) := \sqrt{n} \left(\sum_{i=1}^{m} \frac{U_i(u) H_i(v)}{\bar{\sigma}_i} + \sum_{i=1}^{m} \frac{U_i(u) H_i(v)}{\bar{\sigma}_i} \right)$ is a centered Gaussian random variable with finite variance. In particular, for a critical edge, we have $\sum_{i=1}^{m} U_i(u) V_i(v) \bar{\sigma}_i = \frac{\epsilon}{\sqrt{N}}$ and hence for small σ, the probability of missing a critical edge is given by

$$\Pr\left(\frac{\epsilon}{\sqrt{N}} + \frac{\sigma}{n} \Gamma(u,v) \leq 0 \right) = \Pr\left(\epsilon + \sigma \sqrt{\rho} \Gamma(u,v) \leq 0 \right),$$

depending on ϵ, σ, ρ.

C Proof of Theorem 2

Let us first prove the result for a chain (total ordering), where $w = 1$, $h = n$, and $N \asymp n^2$. First, we index the elements by $i = 1, 2, \ldots, n$ in an ascending order (i.e. $u_i \to u_j$ for $i \leq j$). Take $A = (a_{ij})$ where $a_{ij} = C(i - j + 1)$ with a suitable normalization factor C to make $\|A\|_{\mathrm{F}}^2 = N$. Then, we observe that A is a 2-compatible map, which proves part 1. For part b define $Q = (q_{ij})$, where

$$q_{ij} = \begin{cases} -\sqrt{\frac{n-j+1}{n(j-1)}} & i < j \\ \sqrt{\frac{j-1}{n(n-j+1)}} & i \geq j > 1 \\ \frac{1}{\sqrt{n}} & j = 1 \end{cases} \tag{17}$$

One can verify by direct calculations that Q is an orthonormal matrix. Assume an $m-$compatible weight matrix A with margin ϵ. Since the sign pattern of A and Q are identical, we have

$$\mathrm{Tr}(Q^T A) = \sum_{i,j} q_{ij} a_{ij} \geq \epsilon \sum_{i \geq j} q_{ij} = \epsilon \sqrt{n} + \epsilon \sum_{j>1} (n - j + 1) \sqrt{\frac{j-1}{n(n-j+1)}} = \\ \epsilon \sqrt{n} + n^{\frac{3}{2}} \epsilon \sum_{j} \frac{1}{n} \sqrt{\left(\frac{j-1}{n}\right)\left(\frac{n-j+1}{n}\right)}, \tag{18}$$

where the last term is a Riemannian sum and converges to $\int_0^1 \sqrt{x(1-x)}\, dx = \frac{\pi}{8}$. We conclude that there exists a universal constant c such that $\mathrm{Tr}(Q^T A) \geq c n^{\frac{3}{2}} \epsilon$ (in fact, $c = \frac{\pi - 1}{8}$ works). Now, we apply the arithmetic mean -quadratic mean (AM-QM) inequality to the singular values of A and note that there are only m nonzero singular values, which leads to $\sqrt{m} \|A\|_{\mathrm{F}} \geq \|A\|_*$, where $\|A\|_*$ denotes the trace norm (Schatten 1-norm) of A. Moreover, we have

$$\|A\|_* = \max_{P \in \mathbf{O}(n)} \mathrm{Tr}(P^T A) \geq \mathrm{Tr}(Q^T A) \geq c n^{\frac{3}{2}} \epsilon, \tag{19}$$

where $\mathbf{O}(n)$ denotes the $n-$dimension orthogonal group. Noting that $\|A\|_{\mathrm{F}} = n$, we conclude that $n\sqrt{m} \geq c n^{\frac{3}{2}} \epsilon$, which gives the result.

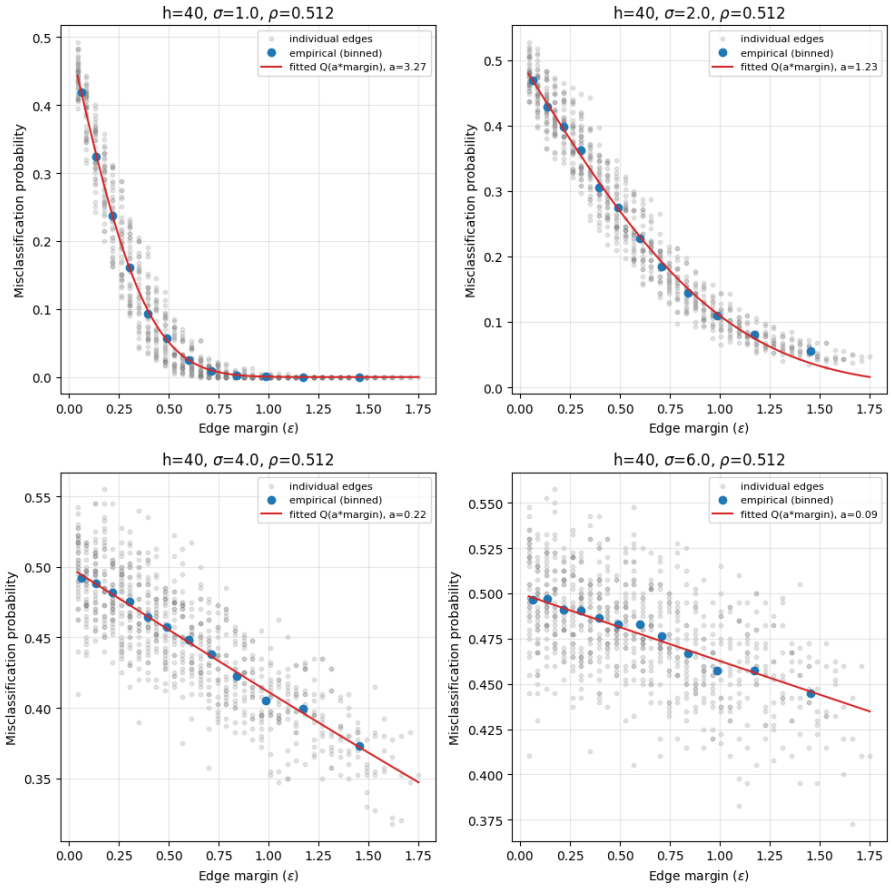


Figure 4: Misclassification probability vs margin across different noise $\sigma = 1, 2, 4, 6$.

Now, we consider a general poset. Take $C_1, C_2, \ldots, C_w$ as disjoint chains covering the poset. Take $e_1, e_2, \ldots, e_w$ as an orthonormal basis of $\mathbb{R}^w$. We already proved above that there exists a 2-dimensional embedding $\mathrm{k}^{(k)}(u), \mathrm{q}^{(k)}(u)$ on C_k. We set $\mathrm{k}^{(k)}(u) = \mathrm{q}^{(k)}(u) = 0$ if u does not belong to C_k and define

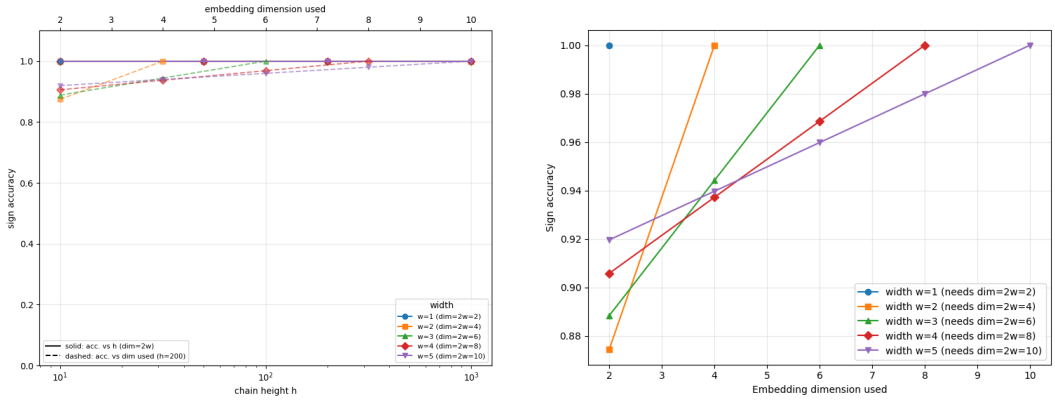
$$\mathrm{k}(u) := \sum_k \mathrm{k}^{(k)}(u) \otimes e_k, \quad \mathrm{q}(u) := \sum_k \mathrm{q}^{(k)}(u) \otimes e_k, \tag{20}$$

where $\otimes$ denotes the outer product. Note that k, q ar $2w-$dimensional and

$$\langle \mathrm{q}(u), \mathrm{k}(v) \rangle = \sum_k \langle \mathrm{q}^{(k)}(u), \mathrm{k}^{(k)}(v) \rangle \tag{21}$$

Since the chains partition the poset, only one of the summands can be nonzero when u, v belong to the same chain. As a result, $\langle \mathrm{q}(u), \mathrm{k}(v) \rangle > 0$ if and only if $u \to v$. This proves part 1. For part 2, take a canonical embedding of the poset with margin ϵ and the corresponding compatible weights A. Take A' as the restriction of A to a block corresponding to the longest chain k and note that $A' \times \frac{N}{N_k}$ is a compatible weight with margin $\epsilon \frac{N}{N_k} \geq \epsilon$, where N_k is the number of edges in the longest chain. Applying the bound derived for the single chain completes the proof[2].

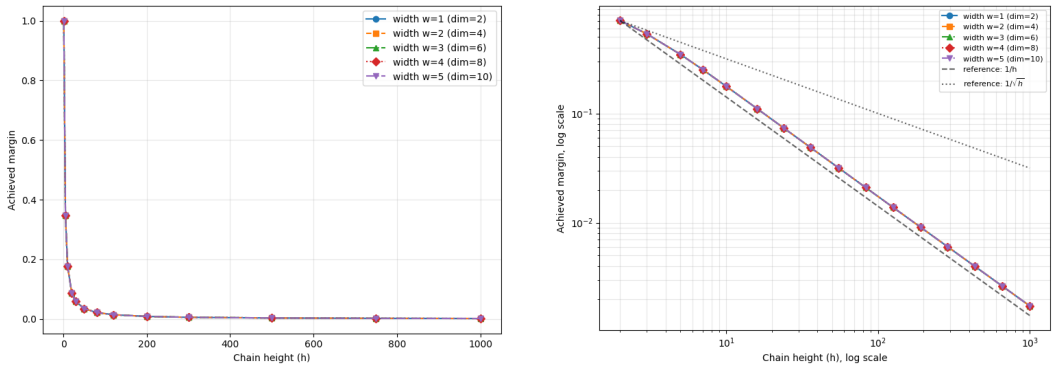
[2]One can refine this bound by repeating the argument for a single chain with blocks of Q matrices for each chain and conclude that $\epsilon \leq c\sqrt{m \sum_k h_k^2} / \left(\sum_k h_k^{\frac{3}{2}} \right)$, where h_k is the length of each chain. This, for example, shows that the margin is even smaller if there are many long chains. We omit this result as it is beyond our goal.



(a) Accuracy vs. chain and embedding dimension

(b) zoomed of accuracy vs. embedding dimension

Figure 5: Embedding dimension depends on width w, not on height h. **(a)** The solid curves (bottom axis) show accuracy is 1.0 for any chain height h. The dashed curves (top axis) show accuracy as the number of used embedding dimensions that is below $2w$.**(b)** A zoomed view of the dashed curves.



(a) Linear scale

(b) Log-log scale, with reference slopes

Figure 6: Margin shrinks as chain height h grows, independent of width w.

D KNOWLEDGE HYPERGRAPH REPRESENTATION

We detail here how the knowledge hypergraph $\mathcal{GH}$ is constructed to use during the retrieval process. For a given corpus $\mathcal{K}$, we construct $\mathcal{GH}$ in two steps:

(A) Initial hypergraph construction: First, we use an LLM with a prompt $\mathcal{P}_{\text{Hyper}}$ (Shown in Figure 7) to extract n-ary relational facts through an automated pipeline Luo et al. (2025). For each document $d \in \mathcal{K}$, extract a sequence of hyperedges e_i for $i = 1, \ldots, n(d)$, with the corresponding set of participating entities s_e, where $n(d)$ denotes the number of facts extracted from the document d. Then, we construct $\mathcal{GH}^0$ by aggregating all extracted hyperedges and their entities across the corpus, i.e., $E^0 = \bigcup_{d\in\mathcal{K}} \bigcup_{i=1}^{n(d)} \{e_i\}$ and $V^0 = \bigcup_{d\in K} \bigcup_{i=1}^{n(d)} \{s_{e_i}\}$.

(B) Synonym and overlapping detection: Despite the capability of hypergraphs , they are prone to the ambiguity imposed by synonyms or overlaps among hyperedges Li et al. (2024a). To alleviate this problem, we refine $\mathcal{GH}^0$ with synonym and overlapping detection. For synonyms, we calculate a contextual embedding v_i^{ctx} for each entity $v_i \in V^0$, as follows:

$$v_i^{\text{ctx}} = \phi(v_i) + \frac{1}{|\text{ Hyper}_i\ |} \sum_{e\in\text{Hyper}_i} \phi(e), \tag{22}$$

where the set $\text{Hyper}_i := \{e \mid e \in E, v_i \in s_e\}$, including all hyperedges e with v_i in its set of entities s_e. $\phi(\cdot)$ is a text encoder that maps an input (an entity or a relation) into a dense embedding.

Prompt for initial hypergraph construction

System prompt:
You are a Hypergraph constructor. Given a text document, identify all entities and the complex n-ary relationships (hyperedges) connecting them. A hyperedge can connect two or more entities simultaneously to represent a complete relational fact.

Instruction:
1. **Divide and Analyze:** Break the text into several complete knowledge segments. Each segment should represent a coherent piece of information or a specific relation.
2. **Extract Relations**: For each knowledge segment, extract:
   knowledge_segment: A natural language describing the context of the relation.
   Format: ("hyperedge"|<knowledge_segment>)
3. **Extract Entities**: Identify all entities in each knowledge segment. For every entity, extract:
   entity_name: Name of the entity (Capitalized if English).
   entity_type: Type of the entity (e.g., Organization, Person, Concept).
   entity_description: Brief description of the entity.
   Format: ("entity"|<entity_name>|<entity_type>|<entity_description>)

Output Requirements:
Return a single, continuous list of all identified entities and relationships. Crucial: Separate every single entity and relationship entry using ||. Do not include introductory text or conversational filler. End the entire output with <END>.

Input:
Text: {input_text}

Figure 7: Prompt $\mathcal{P}_{\mathrm{Hyper}}$ for initial hypergraph construction (Step A).

The Planning Prompt

System prompt:
You are a Strategic Planner for a question-answering agent. Your task is to generate a concise reasoning roadmap that helps an agent answer the given question.

Instruction:
1. Identify the minimum set of sub-goals required to answer the question.
2. Decompose multi-hop questions into dependent sub-goals.
3. Keep simple questions as a small sub-goal.
4. Use conditional dependencies when useful
5. Avoid unnecessary and redundant steps.

Output Requirements:
- Return ONLY the roadmap.
- Wrap the roadmap inside <plan>...</plan>.
- Number each step.
- Separate steps using ||.
- End with <END>.

Input:
Query: {input_query}

Figure 8: Prompt $\mathcal{P}_{\mathrm{Plan}}$ for creating plan.

The Prompt

System prompt:
Answer the given question using the provided Reasoning Plan as a flexible roadmap to guide reasoning and knowledge retrieval. You may query the knowledge base whenever additional information is required. Multiple queries are allowed if necessary.

Instruction:
1. You must reason inside <think>...</think> before every query and before the final answer.
2. If retrieval is needed, use <query> {"query": "your search statement"} </query>
3. The final answer inside <answer>...</answer> must be the shortest possible string:

Input:
Reasoning Plan: {reasoning_plan}
Question: {input_query}

Figure 9: Prompt $\mathcal{P}_{\mathrm{AgenticQA}}$

Table 3: Baseline methods.

Category	Method	Retrieval	Algorithm
Prompt-based	IO (LLM-only)	–	–
	StandardRAG	Text	–
	HyperGraphRAG	Graph	–
RL-based	R1	–	GRPO
	Search-R1	Text	GRPO
	Graph-R1	Graph	GRPO

Subsequently, we look for entities with the same concept, based on the similarity score:

$$\mathrm{sim}_{\mathrm{multi}}(v_i, v_j) = \alpha \cdot \cos(\phi(v_i), \phi(v_j)) + (1-\alpha) \cdot \cos(v_i^{\mathrm{ctx}}, v_j^{\mathrm{ctx}}), \tag{23}$$

where $\alpha \in [0,1]$ is a weighting parameter. An entity pair (v_i, v_j) is considered a candidate for synonym merging if $\mathrm{sim}_{\mathrm{multi}}(v_i, v_j)$ bigger than a threshold. Each candidate pair is then checked by LLM, which decides whether the two entities refer to the same real-world concept. We keep only the confirmed pairs and build a similarity graph over them where entities connected directly or through a chain of confirmed pairs form a new synonym hyperedge e_{syn} added to E_{syn}.

For overlaps, we identify hyperedge pairs $e_i = (s_i, \ell_i) \in E^0$ and $e_j = (s_j, \ell_j) \in E^0$ that share entities $s_i \cap s_j \neq \emptyset$ and their relation descriptions are semantically similar based on similarity $\cos(\ell_i, \ell_j)$.

Then we use an LLM to confirm that these hyperedges are truly redundant. Confirmed pairs are merged into $e_{\mathrm{overlap}} = (s_i \cup s_j, \ell_{\mathrm{overlap}})$ added to E_{overlap}, where ℓ_{overlap} concatenates the two original relation descriptions. Finally, we update the initial hypergraph as $\mathcal{GH} = (V^0, E^+)$ where $E^+ = E^0 \cup E_{\mathrm{overlap}} \cup E_{\mathrm{syn}}$.

E Prompt Used in PHyRR

We use prompts for some parts, including knowledge hypergraph construction (Section D), planning, and the main agentic QA. The planning component uses $\mathcal{P}_{\mathrm{Planner}}$ prompt, shown in Figure 8, to generate an initial plan, and the agentic QA uses $\mathcal{P}_{\mathrm{AgenticQA}}$ prompt as shown in Figure 9.

F Experimental Settings

Datasets: We evaluate PHyRR framework on four QA benchmarks, including multi-hop benchmarks HotpotQA Yang et al. (2018) and 2WikiMultiHopQA Ho et al. (2020), and single-hop PopQA Mallen et al. (2023) and TriviaQA Joshi et al. (2017), based on the FlashRAG toolkit.These datasets cover diverse reasoning complexity, evidence distribution, and single-hop and multi-hop reasoning tasks.

Evaluation Metrics: We used Exact Match (EM) and F1 as measures for evaluation of PHyRR, which are widely used in this domain.EM evaluates whether the predicted answers match exactly with the gold answers or not, while F1 evaluates the overlap between them.

Baselines: We compare PHyRR with six baselines, as shown in Table 3, covering both prompt-based and RL-based methods with different retrieval strategies, including no/text/graph retrieval. The baselines are: 1. IO (LLM-only): LLM directly generates answers using only its internal knowledge, without usingany extra reasoning steps or external knowledge. 2. StandardRAGLewis et al. (2020): LLM first uses external knowledge to find relevant information and then generates answers based on it. 3. HyperGraphRAGLuo et al. (2025): LLM retrieves information from external knowledge represented as a hypergraph before generating answers. 4. R1Guo et al. (2025a): An RL approach that improves LLM's reasoning to generate answers without any retrieval. 5. Search-R1Jin et al. (2025): An RL approach that trains LLMs to automatically retrieve relevant information from external knowledge to improve reasoning, and finally 6. Graph-R1Luo et al. (2026): An RL approach to improve interaction between external knowledge based on graph and reasoning.

Implementation Details: We use Qwen-2.5-3B-Instruct and Qwen-2.5-1.5B-Instruct Team (2024) as the backbone LLM. We use the FlagEmbedding framework with top-5 relevance for retrieval and the embedding model bge-large-en-v1.5 for embeddingChen et al. (2024).