AgentGL: Towards Agentic Graph Learning with LLMs via Reinforcement Learning

Yuanfu Sun[1,2*], **Kang Li**[3*], **Dongzhe Fan**[1,2], **Jiajin Liu**[1,2], **Qiaoyu Tan**[1†]
[1]New York University Shanghai [2]New York University [3]Tsinghua University
{yuanfu.sun, qiaoyu.tan}@nyu.edu, lik24@mails.tsinghua.edu.cn

Abstract

Large Language Models (LLMs) increasingly rely on agentic capabilities—iterative retrieval, tool use, and decision-making—to overcome the limits of static, parametric knowledge. Yet existing agentic frameworks treat external information as unstructured text and fail to leverage the topological dependencies inherent in real-world data. To bridge this gap, we introduce Agentic Graph Learning (AGL), a paradigm that reframes graph learning as an interleaved process of topology-aware navigation and LLM-based inference. Specifically, we propose AgentGL, the first reinforcement learning (RL)–driven framework for AGL. AgentGL equips an LLM agent with graph-native tools for multi-scale exploration, regulates tool usage via search-constrained thinking to balance accuracy and efficiency, and employs a graph-conditioned curriculum RL strategy to stabilize long-horizon policy learning without step-wise supervision. Across diverse Text-Attributed Graph (TAG) benchmarks and multiple LLM backbones, AgentGL substantially outperforms strong GraphLLMs and GraphRAG baselines, achieving absolute improvements of up to 17.5% in node classification and 28.4% in link prediction. These results demonstrate that AGL is a promising frontier for enabling LLMs to autonomously navigate and reason over complex relational environments. The code is publicly available at https://github.com/sunyuanfu/AgentGL.

1 Introduction

Large Language Models (LLMs) have achieved strong performance across NLP tasks through their broad linguistic and reasoning capabilities (Achiam et al., 2023; Yang et al., 2025). Yet their parametric knowledge alone is insufficient for many specialized or fast-evolving domains (Lewis et al., 2020). To bridge this gap, Retrieval-Augmented Generation (RAG) (Gao et al., 2023) and more recent agentic search frameworks (Li et al., 2025; Jin et al., 2025; Chen et al., 2025) allow LLMs to iteratively query external resources and integrate retrieved evidence into a dynamic chain of thought.

Despite the power of agentic paradigms, they mainly operate on unstructured text, overlooking the relational structures that underpin many corpora. In critical domains such as citation networks (Yang et al., 2016), social platforms (Hamilton et al., 2017), and commercial ecosystems (Shchur et al., 2018), information naturally manifests as Text-Attributed Graphs (TAGs), where meaning is derived from the interplay between textual content and graph topology. Consequently, agentic systems that rely solely on lexical similarity cannot harness these structural dependencies. This raises a central question: *Can the agentic learning paradigm be extended to graph-structured environments to enable dynamic, topology-aware reasoning, and how can such a system be built efficiently?*

Existing graph learning efforts only partially address this need. Traditional GNNs (Kipf and Welling, 2016; Velickovic et al., 2017) model structural signals but struggle with rich textual semantics (Yan et al., 2023). Recent LLM-based Graph Models (GraphLLMs) integrate LLMs with graph information via graph-guided prompting or instruction tuning (e.g., GraphGPT (Tang et al., 2024), GraphICL (Sun et al., 2025)), but these models rely on static graph context extracted once at inference time, preventing adaptive exploration. GraphRAG systems (Jimenez Gutierrez et al., 2024; Dong et al., 2025) construct large text-enriched knowledge graphs (KGs) from corpora, yet these reconstructed KGs are costly to build and do not preserve the native topological correlations present in real TAGs. Consequently, neither GraphLLMs nor GraphRAG offers mechanisms for dynamic evidence acquisition over real-world graph structure.

This motivates the emergence of Agentic Graph

[*]Equal contribution
[†]Corresponding author

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 25313–25335

Learning (AGL), a new direction where a LLM agent can autonomously navigate a graph, accumulate structural evidence, and iteratively refine its search trajectory based on on-the-fly reasoning.

However, realizing AGL is non-trivial due to two fundamental challenges. ***(C1) Topology-aware navigation.*** Evidence on a graph is multi-scale: some clues appear in tightly local neighborhoods, whereas others emerge only through broader structural patterns. An agent must decide where to go next in a combinatorial space while avoiding redundant or uninformative regions. ***(C2) Long-horizon policy optimization.*** Effective graph reasoning frequently requires multi-step exploration, but ground-truth search trajectories are rarely available. This makes it difficult to learn policies that balance exploration, exploitation, and reasoning depth, and easy for agents to drift into irrelevant branches or incur unnecessary tool calls. Addressing these challenges demands a principled formulation of graph-native action spaces and stable training mechanisms for long-horizon decision-making.

To address these challenges, we propose `AgentGL`, a framework that formulates graph learning as an agentic decision-making process optimized through reinforcement learning (RL). `AgentGL` equips LLM with a suite of graph-native search tools, including local neighborhood expansion, hop-constrained traversal, and global evidence probing that enable multi-scale structural exploration tailored to the task. To prevent over-searching and encourage deeper reasoning on retrieved evidence, we introduce search-constrained thinking, a mechanism that biases the LLM agent toward reflective inference before invoking additional graph queries. To support stable long-horizon learning without step-by-step trajectory supervision, we further develop a graph-conditioned curriculum RL strategy that progressively increases topology exploration difficulty, integrates multi-faceted rewards, and enforces efficient use of graph tools under limited budgets. Together, `AgentGL` enables LLM agents to learn adaptive, topology-aware search policies that significantly enhance performance on diverse graph reasoning tasks.

- ✦ We study Agentic Graph Learning (AGL), a new paradigm that treats graph learning as an interleaved process of topology-aware exploration and LLM-based reasoning. This formulation unifies graph structure, text semantics, and agentic decision-making under a single framework.
- ✦ We propose `AgentGL`, the first RL-driven AGL framework that synergizes *structural perception*, *strategic reasoning*, and *policy learning*. Specifically, it orchestrates graph-native search tools and search-constrained thinking to navigate complex topologies, employing graph-conditioned curriculum-based RL to optimize the policy without step-wise supervision.
- ✦ We evaluate `AgentGL` across multiple TAG benchmarks and graph tasks, demonstrating strong improvements over leading GraphLLM and GraphRAG baselines. Specifically, it delivers absolute accuracy improvements of up to **17.5%** in node classification and up to **28.4%** in link prediction across diverse LLM backbones.

2 Related Work

Graph Learning with LLMs. Recent work has focused on bridging the gap between graph-structured data and LLMs to facilitate graph reasoning. One line textualizes local structures into natural-language descriptions to support LLM reasoning and contextualized representations (Zhao et al., 2023; Guo et al., 2023; Chen et al., 2024c; Li et al., 2024; Shi et al., 2024; Fang et al., 2024; He et al.). Another line derives graph tokens or structure-aware embeddings and injects them into prompts for graph instruction tuning (Tang et al., 2024; Chen et al., 2024b; Liu et al., 2024b; Sun et al., 2026; Zhang et al., 2025), or performs training-free inference via graph in-context learning (Sun et al., 2025; Huang et al., 2023; Liu et al., 2024a). Despite progress, these pipelines are largely static, limiting adaptation when additional evidence is needed during inference time.

Grounding LLMs with External Knowledge. While standard RAG improves factuality via static retrieval (Lewis et al., 2020; Gao et al., 2023), agentic search advances this by enabling iterative reasoning through reinforcement learning (Jin et al., 2025; Song et al., 2025; Chen et al., 2025) or prompting (Yao et al., 2022; Press et al., 2023; Li et al., 2025). However, these methods predominantly target unstructured text. To incorporate structure, GraphRAG approaches (He et al., 2024; Jimenez Gutierrez et al., 2024; Dong et al., 2025; Han et al., 2024) retrieve evidence from graph-structured data. Yet, they often rely on synthetic graphs reconstructed from flat corpora and their task objectives are fundamentally different from GL (A.2). Even native-graph methods remain lim-

ited: GraphCoT (Jin et al., 2024) focuses on graph QA, while GraphSearch (Liu et al., 2026) targets graph learning, yet both rely on heuristic prompting with limited optimization, often yielding suboptimal solutions.

3 Problem Statement

We study agentic graph learning (AGL) on a TAG $\mathcal{G} = (\mathcal{V}, \mathcal{A}, \mathcal{T})$, where $\mathcal{V}$ is the node set, $\mathcal{A}$ is the adjacency matrix, and $\mathcal{T} = \{\mathbf{t}_v \mid v \in \mathcal{V}\}$ contains node texts. In this paper, we focus on two classical GL tasks: Node Classification and Link Prediction. Specifically, given a query Q, a target instance x (e.g., a node $v \in \mathcal{V}$ or a node pair $(u, v) \in \mathcal{V} \times \mathcal{V}$) and ground-truth label y, the goal is to predict y by grounding the decision in graph-derived evidence.

Formally, we formulate AGL as a sequential decision process on graph $\mathcal{G}$. Given a target x and query Q, the policy π_θ iteratively samples actions $a_t \sim \pi_\theta(\cdot|h_t)$ from $\mathcal{S} \cup \{\textsc{Answer}\}$, where $\mathcal{S}$ denotes a predefined set of graph-native search tools for multi-scale exploration. This interaction yields a trajectory τ containing the accumulated evidence E and the final prediction $\hat{y}$. Our goal is to optimize θ to maximize the expected reward $\mathcal{R}$ over the dataset $\mathcal{D}$: $\mathcal{J}(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[\mathcal{R}]$.

4 Methodology

We present the AgentGL framework (Figure 1), which organizes learning around two complementary components: graph-native policy bootstrapping (Sec. 4.1), where the agent acquires core navigation behaviors, and search-efficiency optimization (Sec. 4.2), which regulates tool use during long-horizon reasoning. Both components are trained under a graph-conditioned curriculum learning regime (Sec. 4.3) designed to improve stability and accelerate convergence.

4.1 Graph-Native Search Policy Bootstrapping

We begin by formulating the RL objective function, designed to empower the LLM agent to autonomously explore the graph structure while preserving its reasoning capabilities, defined as:

$$\mathcal{J}(\theta) = \mathbb{E}_{\substack{(x,Q,y^*)\sim\mathcal{D} \\ \tau\sim\pi_\theta(\cdot|x,Q,\mathcal{G}_\mathcal{S})}} \Big[\mathcal{R}(\hat{y}, y^*) - \beta \cdot \mathbb{D}_{\text{KL}}(\pi_\theta \parallel \pi_{\text{ref}})\Big].$$

where $\mathcal{G}_\mathcal{S}$ signifies the graph environment accessed via the toolset $\mathcal{S} = \{\tau_{\text{1HOP}}, \tau_{\text{2HOP}}, \tau_{\text{SS}}, \tau_{\text{DENSE}}\}$; $\mathcal{R}(\hat{y}, y^*)$ is the outcome-based reward; $\mathbb{D}_{\text{KL}}$ represents the token-level KL divergence between the current policy π_θ and the reference policy π_{ref}; and β is the coefficient controlling the KL penalty strength. To bootstrap the graph-native search (GNS) policy, we next introduce the GNS tools in $\mathcal{S}$ for searching evidence (the text attributes of the collected candidates) directly from the TAG.

Definition 4.1 (***1-hop Neighborhood Search***)**.** Given a query Q and an input x, we simplify the process by treating x as a pair (u, v) (if x is a single node u, we set $v = u$). Let $\mathcal{C} = \mathcal{N}_1(u) \cap \mathcal{N}_1(v)$ and $\mathcal{U}_z = \mathcal{N}_1(z) \setminus \mathcal{C}$ for $z \in \{u, v\}$. The tool τ_{1HOP} constructs the result set E by prioritizing common neighbors and balancing exclusive ones:

$$E = \text{TopK}(\mathcal{C}, K) \cup \bigcup_{z\in\{u,v\}} \text{TopK}(\mathcal{U}_z, k_z),$$

where the quotas k_u, k_v satisfy $k_u + k_v = \max(0, K - |\mathcal{C}|) = R$ and represent the balanced allocation defined by:

$$k_u = \min\left(|\mathcal{U}_u|, \max\left(\lceil R/2 \rceil, R - |\mathcal{U}_v|\right)\right).$$

Given the candidate nodes returned by the tools, the next question is how we select the most informative ones; we address this with a ranking score. The ranking score for a neighbor n is computed via cosine similarity against a fusion embedding:

$$s(n) = \cos\left(\mathbf{h}_n,\ \lambda_r \mathbf{h}_Q + (1 - \lambda_r)\mathbf{h}_x\right),$$

where $\mathbf{h}_{(\cdot)}$ denotes the semantic embedding, $\mathbf{h}_x = \frac{1}{2}(\mathbf{h}_u + \mathbf{h}_v)$ averages the target pair, and $\lambda_r \in [0, 1]$ balances the query relevance.

Definition 4.2 (***2-hop Neighborhood Search***)**.** τ_{2HOP} follows an analogous retrieval logic to Definition 4.1, substituting scope $\mathcal{N}_1(\cdot)$ with $\mathcal{N}_2(\cdot)$.

Definition 4.3 (***Structure Salience Search***)**.** Leveraging precomputed PPR scores (Jeh and Widom, 2003) $s'(v)$, τ_{SS} retrieves the TopK globally salient candidates from the entire graph, ranking by $s'(v)$ for nodes or the mean $\frac{1}{2}(s(i) + s(j))$ for pairs.

Definition 4.4 (***Graph Dense Search***)**.** The tool τ_{DENSE} operates identically to Definition 4.3, except that it substitutes the structural score $s'(\cdot)$ with the semantic relevance measured by the cosine similarity of node or pair embeddings $\phi(\cdot)$.

Remark 4.5 (Design Rationale). This toolkit is designed to ensure comprehensive coverage of the graph information space, spanning two critical dimensions: *Local vs. Global* and *Structure vs. Semantics*. τ_{1HOP} and τ_{2HOP} facilitate precise local

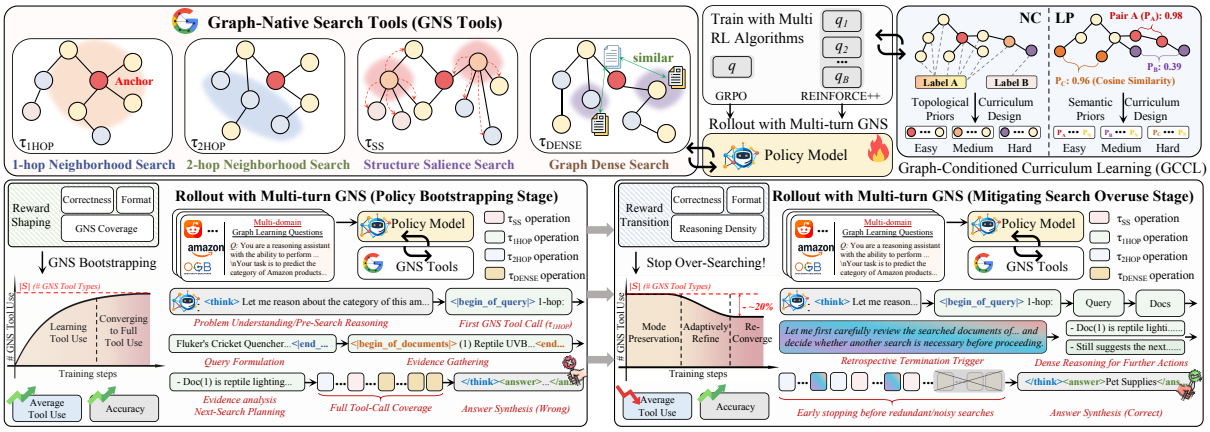


Figure 1: Method Overview. Equipped with graph-native search tools (Top-Left) for structural evidence mining, AgentGL employs a two-stage training strategy building on GCCL (Top-Right). The training progresses from Stage 1: Policy Bootstrapping (Bottom-Left), which uses shaped rewards to instill tool proficiency, to Stage 2: Mitigating Search Overuse (Bottom-Right), which optimizes the trade-off between search efficiency and reasoning accuracy.

grounding by harmonizing explicit topological dependencies with specific query requirements via a weighted fusion strategy. To transcend structural isolation, the global operators offer broader horizons: τ_{SS} acts as a structural prior, identifying topological pivots to guide macro-level reasoning, while τ_{DENSE} adapts the Dense Retrieval paradigm from RAG to graphs, bridging disconnected nodes via latent semantic correlations. Collectively, these primitives empower LLMs to navigate the graph with the same versatility as navigating text.

Optimization with RL Algorithms. To enable LLMs to leverage GNS tools for interleaved reasoning while exploring graph structure, and to avoid the high cost of constructing SFT-style supervision, we directly optimize the policy via RL. Specifically, we instantiate AgentGL with two mainstream critic-free policy optimization algorithms: Group Relative Policy Optimization (GRPO) (Shao et al., 2024) and REINFORCE++ (R++) (Hu et al., 2025) to optimize the AGL process.

Template Design and Trajectories. To support interleaved reasoning and graph-native search in a way that is both learnable and automatically evaluable, we cast AGL as a *reason–act–observe* interaction loop with a strict, machine-parseable interface. Concretely, each prompt specifies (a) a dataset-routed task instruction with a closed label space and target instance text attribute, (b) a toolbox $\mathcal{S}$ of GNS pools with per-pool introduction, and (c) other instructions to steer the model toward the required response format. Within the <think>···</think> block, the model may issue at most one retrieval action per round by emitting a pool-specific query tag <|begin_of_query|> *tool name*:*query* <|end_of_query|>, after which the environment executes the corresponding GNS tool and returns evidence wrapped in <|begin_of_documents|>···<|end_of_documents|>. Formally, let $h_0 = (x, Q)$ denote the initial context. We model the agentic rollout as a recursive state transition process, where the context evolves via the interactive trajectory defined by:

$$h_t = h_{t-1} \oplus (a_t, [\![a_t]\!]_{\mathcal{G}}) \quad \text{s.t.} \quad a_t \sim \pi_\theta(\cdot \mid h_{t-1})$$

where the action $a_t = \langle s_t, q_t \rangle$ specifies a tool selector $s_t \in \mathcal{S}$ and a textual query q_t. The semantic bracket $[\![a_t]\!]_{\mathcal{G}}$ denotes the structural evidence o_t (e.g. text attributes) retrieved from graph $\mathcal{G}$. The operator $\oplus$ recursively appends this interaction turn to the history h_{t-1}. A rollout terminates either when the agent takes the terminal action and decides to output the final answer in <answer>···</answer>, or when the maximum budget B is exhausted.

Reward Shaping. We use a composite reward to provide dense, programmatic supervision for structured tool use while keeping the final objective aligned with task correctness. Concretely, for a trajectory τ with prediction $\hat{y}$, we define

$$R(\tau) = r_{\text{FMT}}(\tau) \; + \; r_{\text{ACC}}(\hat{y}, y) \; + \; r_{\text{COV}}(\tau)$$

Format reward $r_{\text{FMT}}(\tau)$ enforces strict adherence to our tool-use template (tool name + query/args + structured think/answer), making trajectories reliably machine-parsable for stable RL. Accuracy reward $r_{\text{ACC}}(\hat{y}, y) = \lambda_a \mathbb{I}[\hat{y} = y]$ anchors optimization to the end task and prevents reward hacking toward purely "well-formatted" behaviors. GNS coverage reward $r_{\text{COV}}(\tau)$ encourages early exploration of all proposed tools, which is crucial to

prevent early mode collapse to a single default action (one tool or no tool) and to ensure sufficient exploration over the discrete tool-action space.

$$r_{\text{cov}}(\tau) = \eta \sum_{j=1}^{|\mathcal{S}|} \mathbb{I}[\exists\, t:\ a_t = \tau_j]\,, \quad r_{\text{cov}}(\tau) \le |\mathcal{S}|\eta$$

where each tool $\tau_j \in \mathcal{S}$ contributes at most once.

4.2 Less is More: Mitigating Search Overuse

While the bootstrapping stage establishes foundational graph navigation capabilities, it prioritizes *feasibility* over *optimality*, often defaulting to inefficient, exhaustive retrieval. However, given that the effective neighborhood range is highly instance-dependent (Xu et al., 2018), the optimal structural context varies substantially across queries. Indiscriminate tool usage is thus counterproductive: it not only incurs computational overhead but creates structural noise that degrades reasoning fidelity. To address this, we introduce *Search-Constrained Thinking*. This phase *implicitly* optimizes efficiency by compelling the agent to autonomously discern the *minimal sufficient trajectory*—maximizing accuracy by effectively pruning redundant steps. Accordingly, the optimization goal is formulated as:

$$\theta^\star = \operatorname*{argmin}_{\theta} \mathbb{E}_{\tau\sim\pi_\theta}[\,T(\tau)\,] \quad \text{s.t.} \quad \theta \in \operatorname*{argmax}_{\vartheta} \mathcal{J}_{\text{BASE}}(\vartheta)$$

where $\mathcal{J}_{\text{BASE}}$ denotes the bootstrapping objective and $T(\tau)$ denotes the total search cost of trajectory τ. By treating accuracy as a hard constraint, we restrict the efficiency optimization strictly to the optimal solution space, ensuring the agent learns parsimony without compromising performance.

Search-Constrained Thinking. To instantiate the implicit optimization target, we introduce a strategy that enforces a "Think more, Search less: Precision via Parsimony" paradigm. This approach couples retrospective verification with cognitive density constraints to substitute redundant retrieval with deep reasoning, via three components:

Retrospective Termination Trigger. To preclude habitual search continuation, we inject a cognitive interrupt into the context after each tool execution:

```
Let me first carefully review the searched documents of {GNS tool name} and decide whether another search is necessary before proceeding.
```

This trigger acts as a soft constraint, compelling the LLM to explicitly evaluate the sufficiency of the current evidence state $\mathcal{G}_\tau$ during training, transforming the search process from a habitual sequence into a series of deliberate, binary decisions.

Cognitive Density Regularization. To ensure that reduced search frequency stems from efficient information absorption rather than superficial skipping, we impose a penalty on sparse reasoning. Formally, we define segments $\{s_i\}$ as the post-retrieval reasoning blocks dedicated to analyzing the acquired context and define a segment as "deficient" if the token length $\ell(s_i)<$ the threshold δ. We introduce a depth-oriented term r_{depth}:

$$r_{\text{depth}}(z) = \alpha \cdot \mathbb{I}[N_{\text{short}} = 0] - \lambda_d \cdot N_{\text{short}}$$

where N_{short} counts deficient segments. This formulation strictly penalizes fragmented thinking, incentivizing the generation of dense reasoning blocks before further actions.

Adaptive Reward Transition. Reflecting the shift from exploration to exploitation, we discard the coverage incentive r_{COV} while retaining r_{FMT} for format constraints. The main optimization is thus streamlined to the synergistic maximization of accuracy r_{ACC} and reasoning density r_{depth}. This alignment prioritizes deep internal processing over redundant retrieval, naturally converging onto the minimal sufficient trajectory:

$$R(\tau) = r_{\text{FMT}}(\tau) \ + \ r_{\text{ACC}}(\hat{y}, y) \ + \ r_{depth}(z)$$

4.3 Graph-Conditioned Curriculum Learning

To stabilize training and accelerate convergence, we leverage intrinsic graph properties for curriculum design. Unlike reasoning tasks where difficulty estimation relies on expert annotation (Hendrycks et al., 2021) or expensive pilot rollouts (Song et al., 2025), graphs offer a distinct advantage: learnability is directly quantifiable via topological and semantic priors. We formulate an analytical difficulty scoring function $\mathcal{S}(\cdot)$ to proxy hardness, enabling a smooth, cost-free training progression from confident to ambiguous instances for different tasks via graph-conditioned curriculum learning (GCCL).

Node Classification with GCCL. Drawing on prior theoretical insights, node classification difficulty is jointly governed by local homophily and degree magnitude (Tang et al., 2020; Zhu et al., 2020) in many cases. To derive a robust metric $\mathcal{S}_{\text{NC}}(v)$ that approximately estimates difficulty, we rectify homophily estimates using the Wilson Lower Bound, augmented by degree magnitude:

$$\mathcal{S}_{\text{NC}}(v) = \underbrace{\frac{\hat{p}_v + \frac{z^2}{2d_v} - z\sqrt{\frac{\hat{p}_v(1-\hat{p}_v)}{d_v} + \frac{z^2}{4d_v^2}}}{1+\frac{z^2}{d_v}}}_{\text{Wilson Lower Bound}} + \eta \log(1+d_v)$$

where $\hat{p}_v$ is the neighbor label consistency, d_v is the degree, z is the standard normal quantile and η regulates the impact of degree priors. This formulation prioritizes structurally prominent hubs (*Easy*), progresses through intermediate nodes (*Medium*), and defers ambiguous, heterophilous outliers (*Hard*).

Link Prediction with GCCL. Inspired by heuristics in link prediction (Zhang and Chen, 2018; Mao et al., 2023), we posit that "easiness" aligns with the consistency between semantic similarity and label existence. For a link pair $e = (u, v)$ with label $y_e \in \{0, 1\}$, we calculate the score based on cosine similarity of node features $\text{sim}(\mathbf{x}_u, \mathbf{x}_v)$:

$$\mathcal{S}_{\text{LP}}(e) = y_e \cdot \text{sim}(\mathbf{x}_u, \mathbf{x}_v) + (1 - y_e) \cdot \big(1 - \text{sim}(\mathbf{x}_u, \mathbf{x}_v)\big)$$

We prioritize consistent pairs (high-sim positives, low-sim negatives) as *Easy*. The curriculum traverses ambiguous *Medium* instances, deferring *Hard* structural noise-conflicting cases like high-sim negatives to later training iterations.

Training Process. Algorithm 1 outlines the procedure: AgentGL first undergoes graph-native policy bootstrapping (Sec. 4.1), then search-efficiency refinement (Sec. 4.2). Both stages follow an easy-to-hard curriculum in Sec. 4.3. More details are provided in Appendix A.3.

5 Experiments

We conduct extensive experiments to validate the effectiveness of AgentGL. Specifically, we evaluate on 7 TAG datasets spanning 3 domains, compare against 13 baselines across five categories, and test 2 backbone LLMs with different parameter scales.

Datasets. We use the following datasets: (1) ***Citation Networks***: OGB-Arxiv (Hu et al., 2020), PubMed (Sen et al., 2008), and Arxiv-2023 (He et al.); (2) ***Amazon Products***: OGB-Products (Hu et al., 2020), Amazon-Photo, and Amazon-Computers (Shchur et al., 2018); and (3) ***Social Networks***: Reddit (Yan et al., 2025). Additional dataset details and data splits are provided in the Appendix A.1.

Baselines. We compare AgentGL against a diverse set of up-to-date, strong baselines spanning (1) ***GNNs*** (): GraphSAGE (2021), GCN (2016) and RevGAT (2021); (2) ***GraphLLMs*** (): LLaGA (2024a), GraphGPT (2024), GraphPrompter (2024b) and GraphICL(-S1) (2025); (3) ***GraphRAG*** (): LinearRAG (2025), HippoRAG2 (2025) and GraphCoT (A special kind of agent framework) (2024); (4) ***Standard Agentic Search*** (): Search-R1 (2025) and Search-O1 (2025); (5) ***Large Language Models*** (): Qwen2.5-3B/7B-Instruct (2025) (SFT), to comprehensively assess predictive performance.

Setup. For a fair comparison, we use the same LLM backbone as AgentGL for all baselines whose final reasoner is an LLM. For GraphRAG baselines, we construct the graph-based retrieval corpus by collecting the node texts involved in our experiments. For standard agentic search baselines, since they lack native graph-search capability, we replace their original online-search space with the set of graph nodes, ensuring that they can be properly applied to graph reasoning tasks; implementation details are provided in the appendix. Other settings, such as SFT and RL training procedures, follow the original papers. The other implementation details are provided in the Appendix A.3.

5.1 Overall Performance

We first evaluate in-domain and zero-shot transfer performance, with results listed in Table 1. For all methods, we train only on OGB-Arxiv and OGB-Products using different Qwen backbones, and then test on the test splits of all datasets. Based on these results, we summarize the following observations:

Obs 1. AgentGL consistently achieves the best performance across multiple tasks and domains, under diverse graph-reasoning regimes. For node classification (NC), with Qwen7B as the backbone, AgentGL outperforms the baselines by an average of **12.7%** on the in-domain evaluation and **24.4%** on the zero-shot transfer setting. For link prediction (LP), AgentGL achieves an average gain of **26.3%** in-domain and **22.4%** in zero-shot transfer. These improvements are consistent across model scales: with Qwen3B, AgentGL improves over the baselines by **14.5%** on in-domain NC and **26.3%** on in-domain LP, and by **26.6%** and **22.4%** on zero-shot NC and LP, respectively.

Obs 2. AgentGL showcases the promise of interleaved graph reasoning and searching, outperforming static context stuffing. Methods that primarily rely on static "stuffing" (e.g., GraphRAG or GraphLLM) can be competitive in some settings, but are consistently outperformed by AgentGL. Taking Qwen7B LP as an example, AgentGL achieves **47.4%** and **23.2%** higher in-domain performance than GraphRAG and GraphLLM, respectively; these substantial margins are sustained under zero-shot transfer at **35.4%** and **26.9%**. This trend sug-

Tasks	Node Classification							Link Prediction						
Settings	In-Domain		Zero-shot Transfer					In-Domain		Zero-shot Transfer				
Methods	OGB-Arxiv	OGB-Products	PubMed	Photo	Computers	Arxiv-23	Reddit	OGB-Arxiv	OGB-Products	PubMed	Photo	Computers	Arxiv-23	Reddit
GNN-Based Methods														
GCN	60.2 ↑7.7	58.8 ↑10.8	14.1 ↑64.5	8.0 ↑42.8	10.0 ↑50.2	1.9 ↑63.6	7.1 ↑40.0	55.5 ↑38.1	74.7 ↑18.8	51.0 ↑24.8	49.8 ↑19.7	50.5 ↑24.1	52.0 ↑38.1	53.2 ↑34.8
RevGAT	58.9 ↑9.0	56.7 ↑12.9	13.5 ↑65.1	12.1 ↑38.8	5.6 ↑54.6	1.3 ↑64.2	8.2 ↑38.9	59.0 ↑34.6	73.5 ↑20.0	55.8 ↑20.0	59.4 ↑10.1	50.7 ↑23.9	61.8 ↑28.3	61.8 ↑26.2
SAGE	60.1 ↑7.8	56.9 ↑12.7	11.1 ↑67.5	8.2 ↑42.7	7.4 ↑52.8	3.6 ↑61.9	3.1 ↑44.0	61.0 ↑32.6	73.4 ↑20.1	50.9 ↑24.9	51.6 ↑17.9	50.3 ↑24.3	50.5 ↑39.6	57.6 ↑30.4
LLM-based Methods														
Qwen2.5-3B-Instruct														
GraphPrompter	54.1 ↑12.2	61.0 ↑1.3	67.0 ↑7.5	8.5 ↑34.2	29.2 ↑23.1	37.9 ↑25.7	4.7 ↑35.0	79.6 ↑11.8	74.4 ↑15.7	60.9 ↑10.9	50.9 ↑14.9	52.7 ↑15.1	55.8 ↑32.1	46.8 ↑36.4
GraphGPT	12.5 ↑53.8	23.4 ↑38.9	62.3 ↑12.2	3.0 ↑39.7	11.5 ↑40.8	44.6 ↑19.0	9.5 ↑30.2	59.9 ↑31.6	43.3 ↑46.8	44.5 ↑27.4	54.5 ↑11.3	51.1 ↑16.6	63.4 ↑24.5	49.8 ↑33.4
LLaGA	54.6 ↑11.6	57.9 ↑4.4	61.8 ↑12.7	4.2 ↑38.5	12.3 ↑40.0	44.8 ↑18.8	10.1 ↑29.6	79.8 ↑11.7	73.6 ↑16.5	61.5 ↑10.4	51.3 ↑14.5	53.2 ↑14.6	52.1 ↑35.8	43.7 ↑39.5
GraphICL	49.4 ↑16.9	45.8 ↑16.5	53.8 ↑20.7	35.2 ↑7.5	42.4 ↑9.9	52.2 ↑11.4	37.1 ↑2.6	76.3 ↑15.2	69.2 ↑20.9	62.5 ↑9.4	54.8 ↑11.0	57.7 ↑10.1	57.6 ↑30.3	62.0 ↑21.2
LinearRAG	43.8 ↑22.4	52.5 ↑9.8	64.2 ↑10.3	11.8 ↑30.9	20.1 ↑32.2	28.9 ↑34.7	32.7 ↑7.0	47.3 ↑44.1	45.3 ↑44.8	46.0 ↑25.9	47.7 ↑18.1	45.7 ↑22.0	45.9 ↑42.0	50.1 ↑33.1
HippoRAG2	44.2 ↑22.0	53.2 ↑9.1	63.5 ↑11.0	12.2 ↑30.4	23.8 ↑28.5	30.6 ↑32.9	34.7 ↑5.0	48.1 ↑43.4	46.7 ↑43.4	45.9 ↑26.0	48.1 ↑17.7	45.8 ↑21.9	48.5 ↑39.4	49.7 ↑33.5
GraphCoT	44.3 ↑22.0	56.1 ↑6.1	72.8 ↑1.7	38.0 ↑4.7	47.6 ↑4.7	40.1 ↑23.5	37.2 ↑2.5	50.9 ↑40.6	51.0 ↑39.1	52.0 ↑19.9	50.7 ↑15.1	50.2 ↑17.6	51.1 ↑36.8	50.3 ↑32.8
Qwen2.5	49.1 ↑17.2	54.2 ↑8.1	57.7 ↑16.8	10.3 ↑32.4	14.5 ↑37.8	40.8 ↑22.8	9.9 ↑29.8	72.2 ↑19.3	70.3 ↑19.8	54.9 ↑16.9	51.3 ↑14.5	51.5 ↑16.3	48.9 ↑39.0	43.2 ↑40.0
Search-R1	60.2 ↑6.1	58.3 ↑4.0	71.4 ↑3.1	37.8 ↑4.9	42.7 ↑9.6	55.9 ↑7.6	38.7 ↑1.0	82.5 ↑8.9	84.9 ↑5.2	61.8 ↑10.1	54.6 ↑11.2	62.2 ↑5.6	79.6 ↑8.3	69.0 ↑14.2
Search-O1	48.2 ↑18.1	44.7 ↑17.6	53.9 ↑20.6	34.6 ↑8.1	40.1 ↑12.2	51.6 ↑11.9	38.2 ↑1.5	74.4 ↑17.1	65.8 ↑24.3	59.7 ↑12.1	54.9 ↑10.9	56.9 ↑10.9	73.2 ↑14.7	54.5 ↑28.7
AgentGL-R++	65.6	**63.3**	73.6	**42.7**	49.8	60.2	**40.2**	**92.3**	**92.6**	**72.3**	65.2	**68.7**	**90.3**	**86.6**
AgentGL-GRPO	**66.9**	61.2	**75.4**	42.6	**54.8**	**66.9**	39.2	90.6	87.6	71.4	**66.4**	66.8	85.4	79.7
Qwen2.5-7B-Instruct														
GraphPrompter	60.8 ↑8.8	70.2 ↑6.7	80.6 ↑2.1	39.8 ↑19.3	45.2 ↑22.9	62.8 ↑4.7	24.9 ↑29.6	83.2 ↑12.6	80.8 ↑16.1	66.9 ↑12.9	56.1 ↑17.1	61.2 ↑20.3	56.8 ↑35.6	50.9 ↑41.9
GraphGPT	53.8 ↑15.8	59.1 ↑17.8	79.2 ↑3.5	4.4 ↑54.7	12.3 ↑55.8	66.0 ↑1.5	32.2 ↑22.3	50.5 ↑45.3	48.2 ↑48.8	48.7 ↑31.1	49.8 ↑23.4	44.4 ↑37.1	54.7 ↑37.7	33.2 ↑59.6
LLaGA	62.8 ↑6.8	67.4 ↑9.5	78.6 ↑4.1	5.6 ↑53.4	14.6 ↑53.4	67.2 ↑0.3	31.7 ↑22.8	84.1 ↑11.7	79.3 ↑17.7	67.7 ↑12.1	56.8 ↑16.4	60.2 ↑21.3	56.7 ↑35.7	48.5 ↑44.3
GraphICL	66.4 ↑3.2	61.3 ↑15.6	70.4 ↑12.2	42.1 ↑17.0	49.0 ↑19.1	51.0 ↑16.5	48.8 ↑5.6	78.2 ↑17.6	80.9 ↑16.1	72.3 ↑7.5	57.6 ↑15.6	61.8 ↑19.7	72.4 ↑20.0	63.3 ↑29.5
LinearRAG	48.2 ↑21.4	58.3 ↑18.6	72.7 ↑10.0	47.6 ↑11.5	58.3 ↑9.8	53.4 ↑14.1	51.4 ↑3.1	47.6 ↑48.2	48.2 ↑48.8	47.2 ↑32.6	46.3 ↑26.9	46.8 ↑34.7	47.5 ↑44.9	49.8 ↑43.0
HippoRAG2	49.6 ↑20.0	57.2 ↑19.7	75.3 ↑7.4	52.0 ↑7.1	59.0 ↑9.1	51.0 ↑16.5	50.2 ↑4.2	48.3 ↑47.5	47.1 ↑49.9	46.3 ↑33.5	48.2 ↑25.0	44.5 ↑37.0	47.6 ↑44.8	50.5 ↑42.3
GraphCoT	53.1 ↑16.5	62.6 ↑14.3	81.9 ↑0.8	44.2 ↑14.9	54.4 ↑13.7	51.6 ↑15.9	49.8 ↑4.7	50.4 ↑45.3	52.4 ↑44.6	50.5 ↑29.2	51.0 ↑22.2	50.2 ↑31.3	49.9 ↑42.5	52.4 ↑40.4
Qwen2.5	54.7 ↑14.9	55.0 ↑21.9	63.8 ↑18.9	16.4 ↑42.7	29.5 ↑38.6	60.0 ↑7.5	30.9 ↑23.6	74.4 ↑21.3	76.3 ↑20.7	60.2 ↑19.5	53.1 ↑20.1	46.5 ↑35.0	50.9 ↑41.5	47.1 ↑45.7
Search-R1	63.2 ↑6.4	70.4 ↑6.5	81.6 ↑1.1	46.2 ↑12.9	54.6 ↑13.5	66.8 ↑0.7	50.7 ↑3.7	86.6 ↑9.2	89.1 ↑7.8	72.3 ↑7.5	62.1 ↑11.1	70.1 ↑11.4	80.2 ↑12.2	74.1 ↑18.7
Search-O1	60.2 ↑9.4	59.9 ↑17.0	69.5 ↑13.2	41.7 ↑17.4	49.3 ↑18.8	53.8 ↑13.7	47.2 ↑7.2	75.1 ↑20.6	73.9 ↑23.1	65.8 ↑13.9	57.6 ↑15.6	60.6 ↑20.9	74.0 ↑18.4	59.0 ↑33.8
AgentGL-R++	**70.3**	76.8	82.6	58.2	67.5	**67.6**	**54.8**	95.6	**97.4**	**80.4**	**77.0**	**87.3**	90.5	**97.1**
AgentGL-GRPO	68.9	**77.0**	**82.7**	**59.9**	**68.6**	67.4	54.1	**95.9**	96.5	79.1	69.4	75.7	**94.3**	88.5

Table 1: Performance Comparison on Node Classification and Link Prediction benchmarks under In-Domain and Zero-shot Transfer settings. The best results are highlighted in bold. The metric used for two tasks is ACC (%). Red values (↑) indicate the absolute gain over baselines. These denote the average improvement of the two RL variants; for GNN comparisons, they represent the average gain across 3B and 7B backbones.

GNSPB	MSO	Datasets											
		OGB-Arxiv		OGB-Products		Arxiv-2023		PubMed		Amazon-Photo		Amazon-Computers		Reddit	
		ACC(%)	#Search	ACC(%)	#Search	ACC(%)	#Search	ACC(%)	#Search	ACC(%)	#Search	ACC(%)	#Search	ACC(%)	#Search
✓	✗	65.6 ↑3.3	4.00 ↓18.8%	74.1 ↑2.9	4.00 ↓14.0%	66.2 ↑1.2	3.98 ↓10.5%	80.8 ↑1.9	3.99 ↓26.0%	56.9 ↑3.0	3.99 ↓20.5%	67.2 ↑1.4	3.97 ↓19.0%	50.7 ↑3.4	3.97 ↓13.5%
✗	✓	62.3 ↑6.6	0.00 ↑81.2%	71.9 ↑5.1	0.00 ↑86.0%	61.8 ↑5.6	0.11 ↑86.2%	79.1 ↑3.6	0.02 ↑73.2%	54.3 ↑5.6	0.01 ↑79.0%	60.7 ↑7.9	0.01 ↑80.0%	48.7 ↑5.4	0.02 ↑85.2%
✓	✓	68.9	3.25	77.0	3.44	67.4	3.56	82.7	2.95	59.9	3.17	68.6	3.21	54.1	3.43

Table 2: Ablation study on different RL training stages. The red (↑) and green (↓) denote absolute percentage gains and declines, respectively. #Search represents the average search count on the test set of each dataset (Budget=4). The percentage calculation and comparison for the number of searches are conducted based on the budget.

gests that static context injection is more brittle to distribution shifts, while AgentGL's interleaved searching, reasoning loop can adaptively acquire task-relevant evidence and suppress irrelevant context, leading to more robust transfer.

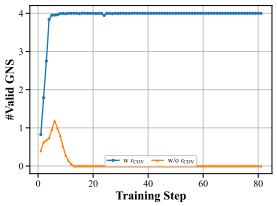
Obs 3. Different RL algorithms yield complementary strengths for AgentGL across graph tasks. Across datasets, AgentGL-R++ and AgentGL-GRPO show a consistent stage/algorithm-dependent profile: GRPO yields higher NC performance by an average of **0.9%** across settings (averaged over Qwen3B/7B), whereas R++ is stronger on LP, improving over GRPO by **3.3%** on average across settings. This indicates a clear tradeoff between algorithms, with task-wise advantages that can be selected based on the target domain.

Obs 4. Scaling up the backbone enhances AgentGL's agentic graph learning capability. Scaling the backbone from 3B to 7B consistently improves AgentGL on both tasks: the average gain is **9.0%** (in-domain) and **11.8%** (zero-shot) for NC, and **5.6%** (in-domain) and **8.7%** (zero-shot) for LP. The improvement is particularly pronounced under zero-shot transfer, indicating that larger backbones better learn and generalize the tool-use policy for adaptive evidence acquisition.

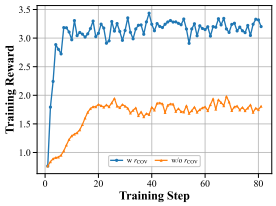
5.2 Impact of Multi-Stage Training

To study the impact of our proposed two-stage RL training: GNS Policy Bootstrapping (GNSPB) and Mitigating Search Overuse (MSO), on both overall performance and search efficiency, we conduct a stage-wise ablation analysis. Specifically, we ablate each training stage and compare the resulting variants in terms of accuracy and tool-call cost, with results reported in Table 2.

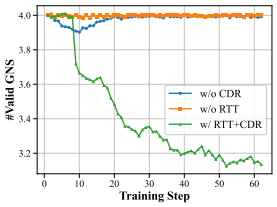
Obs 5. Omitting any RL training stage of AgentGL leads to concurrent drops in both efficiency and performance. When keeping only the GNSPB stage, the presence of $r_{\text{COV}}(\tau)$ encourages the LLM to reliably cover all four tools, which maintains relatively strong performance across datasets; however, it almost always consumes near-full search budgets, increasing the overall cost. In contrast, when keeping only the MSO stage, the policy tends to collapse during training, converging



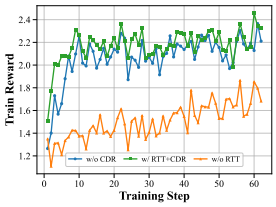
(a) Ablation of $r_{\text{COV}}(\tau)$



(b) Ablation of $r_{\text{COV}}(\tau)$



(c) Ablation of CDR/RTT



(d) Ablation of CDR/RTT

Figure 2: Ablation study of AgentGL(7B)-GRPO on NC: Analysis of valid GNS counts and training rewards.

Model	OGB-Arxiv	Amazon-Photo	Avg.
AGENTGL	**68.9**	**59.9**	**64.4**
w/o r_{COV}	65.2 ↑3.7	55.4 ↑4.5	60.3 ↑4.1
w/o CDR	65.9 ↑3.0	57.1 ↑2.8	61.5 ↑2.9
w/o RTT	65.4 ↑3.5	56.5 ↑3.4	61.0 ↑3.4
w/o GCCL	68.2 ↑0.7	59.3 ↑0.6	63.8 ↑0.6

Table 3: Component-wise ablations of AgentGL. Red numbers denote the absolute improvements.

to the degenerate behavior of making zero searches and carrying this pattern over to inference, which weakens its search capability and degrades performance, resulting in the worst overall results. Only by combining both stages can the model achieve both strong performance and high efficiency; for example, compared to using GNSPB alone, the full method reduces tool calls by about **17.5%** while improving accuracy by an average of **2.4%** on NC.

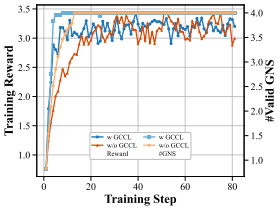
5.3 Component-wise Ablation Analysis

Having established the efficacy of the sequential training, we now isolate the impact of granular components within each stage. Specifically, we ablate individual reward terms and the search-constrained thinking strategy to quantify their distinct contributions to the agent's reasoning capabilities.

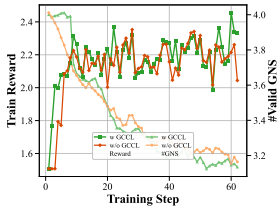
Obs 6. Each component is critical for maintaining the balance between search steps and model performance. For Stage 1, as illustrated in Figure 2a, in the absence of $r_{\text{COV}}(\tau)$, the model fails to acquire effective search habits during training. Consequently, as training steps increase, the agent eventually degenerates to ceasing search operations entirely, maintaining a suboptimal reward level (Figure 2b). Regarding Stage 2, we conducted an ablation study on the Retrospective Termination Trigger (RTT) and Cognitive Density Regularization (CDR). We observe that without CDR, driven by RTT, the model attempts to improve search efficiency in the early phases of Stage 2; however, this improvement is unsustainable, and the model eventually converges to a search step magnitude similar to that of Stage 1 (Figure 2c). Conversely, without RTT, the model persists in the reasoning mode of Stage 1, failing to achieve any efficiency gains. Only the synergistic combination of both components can stably reduce the average search steps, saving approximately **22%** of the search cost when training converges while achieving a **3%** improvement in accuracy (Table 3). Furthermore, we perform an ablation study on λ_r, which governs the embedding-based weighted search for τ_{1HOP} and τ_{2HOP}. As shown in Table 4, the model achieves optimal performance when a balanced weight ($\lambda_r = 0.5$) is applied. This observation underscores the necessity of harmonizing structural topology with semantic similarity, which is essential for comprehensive AGL.

Parameter	Datasets		
	OGB-Arxiv	PubMed	Amazon-Computers
$\lambda_r = 0.0$	67.1	80.6	65.7
$\lambda_r = 0.5$	**68.9**	**82.7**	**68.6**
$\lambda_r = 1.0$	66.9	80.1	66.2

Table 4: Impact of the hyperparameter λ_r.



(a) GCCL in Stage1



(b) GCCL in Stage2

Figure 3: Ablation study of GCCL in different stages for AgentGL(7B)-GRPO on NC.

5.4 Study of Graph-Conditioned Curriculum Learning (GCCL)

Obs 7. GCCL serves to stabilize and expedite convergence across distinct training stages. As illustrated in Figure 3, for Stage 1, GCCL effectively accelerates reward convergence and mitigates oscillations in the later phases of training. A similar trend is observed in Stage 2, where GCCL stabilizes the GNS frequency, maintaining the search steps at a consistently lower magnitude compared to the baseline without GCCL as training progresses. Furthermore, quantitative results in Table 3 corroborate that GCCL not only expedites convergence but also yields an accuracy improvement of approximately

0.65%. In essence, GCCL serves as a stabilizing backbone, effectively guiding the LLM through the complex graph exploration space without succumbing to local optima or early-stage volatility.

6 Conclusion

In this paper, we propose AgentGL, the first RL-driven agentic framework for graph learning, which reformulates Graph Learning as an interleaved process of topology-aware exploration and LLM-based reasoning. AgentGL leverages graph-native search tools for effective navigation and employs a two-stage RL strategy to balance accuracy and efficiency. Across multiple LLM backbones and benchmark settings, AgentGL consistently outperforms strong baselines, including GraphLLMs and GraphRAG methods, achieving the best average performance across all backbones, with absolute gains of up to 17.5% on node classification and 28.4% on link prediction. We hope this work inspires further research into agent-based approaches for complex graph reasoning tasks.

7 Limitations

AgentGL currently operates on text-attributed graphs and does not yet support multimodal-attributed graphs, limiting its applicability in settings where nodes contain richer modal information. Moreover, stable performance in the MSO stage depends critically on a careful trade-off in data allocation between the two stages. It also remains worth investigating whether the MSO stage alters the distribution of tool usage during inference time. The MSO stage is designed to be simple, direct, and effective, and we hope it will encourage future research toward more advanced designs. Finally, extending AgentGL to denser graphs also remains a potential direction for future exploration.

8 Acknowledgments

We sincerely thank the anonymous reviewers, AC and SAC for their valuable feedback. This research is partially supported by National Natural Science Foundation of China (62402320), Shanghai Science and Technology Program (24YF2731000), and the NYU Shanghai Boost Fund.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Mingyang Chen, Linzhuang Sun, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z Pan, Wen Zhang, Huajun Chen, and 1 others. 2025. Learning to reason with search for llms via reinforcement learning. *arXiv preprint arXiv:2503.19470*.

Runjin Chen, Tong Zhao, Ajay Jaiswal, Neil Shah, and Zhangyang Wang. 2024a. Llaga: Large language and graph assistant. *arXiv preprint arXiv:2402.08170*.

Runjin Chen, Tong Zhao, AJAY KUMAR JAISWAL, Neil Shah, and Zhangyang Wang. 2024b. Llaga: Large language and graph assistant. In *Forty-first International Conference on Machine Learning*.

Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, and 1 others. 2024c. Exploring the potential of large language models (llms) in learning on graphs. *ACM SIGKDD Explorations Newsletter*, 25(2):42–61.

Junnan Dong, Siyu An, Yifei Yu, Qian-Wen Zhang, Linhao Luo, Xiao Huang, Yunsheng Wu, Di Yin, and Xing Sun. 2025. Youtu-graphrag: Vertically unified agents for graph retrieval-augmented complex reasoning. *arXiv preprint arXiv:2508.19855*.

Yi Fang, Dongzhe Fan, Daochen Zha, and Qiaoyu Tan. 2024. Gaugllm: Improving graph contrastive learning for text-attributed graphs with large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 747–758.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1).

Jiayan Guo, Lun Du, Hengyu Liu, Mengyu Zhou, Xinyi He, and Shi Han. 2023. Gpt4graph: Can large language models understand graph structured data? an empirical evaluation and benchmarking. *arXiv preprint arXiv:2305.15066*.

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. *arXiv preprint arXiv:2502.14802*.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A Rossi, Subhabrata Mukherjee, Xianfeng Tang, and 1 others. 2024. Retrieval-augmented generation with graphs (graphrag). *arXiv preprint arXiv:2501.00309*.

Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. Harnessing explanations: Llm-to-lm interpreter for enhanced text-attributed graph representation learning. In *The Twelfth International Conference on Learning Representations*.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *Advances in Neural Information Processing Systems*, 37:132876–132907.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. 2025. Reinforce++: Stabilizing critic-free policy optimization with global advantage normalization. *Preprint*, arXiv:2501.03262.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133.

Jin Huang, Xingjian Zhang, Qiaozhu Mei, and Jiaqi Ma. 2023. Can llms effectively leverage graph structural information: when and why. *arXiv preprint arXiv:2309.16595*.

Glen Jeh and Jennifer Widom. 2003. Scaling personalized web search. In *Proceedings of the 12th international conference on World Wide Web*, pages 271–279.

Bernal Jimenez Gutierrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in Neural Information Processing Systems*, 37:59532–59569.

Bowen Jin, Chulin Xie, Jiawei Zhang, Kashob Kumar Roy, Yu Zhang, Zheng Li, Ruirui Li, Xianfeng Tang, Suhang Wang, Yu Meng, and 1 others. 2024. Graph chain-of-thought: Augmenting large language models by reasoning on graphs. *arXiv preprint arXiv:2404.07103*.

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. *arXiv preprint arXiv:2503.09516*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

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. *Advances in neural information processing systems*, 33:9459–9474.

Guohao Li, Matthias Müller, Bernard Ghanem, and Vladlen Koltun. 2021. Training graph neural networks with 1000 layers. In *International conference on machine learning*, pages 6437–6449. PMLR.

Rui Li, Jiwei Li, Jiawei Han, and Guoyin Wang. 2024. Similarity-based neighbor selection for graph llms. *arXiv preprint arXiv:2402.03720*.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025. Search-o1: Agentic search-enhanced large reasoning models. *arXiv preprint arXiv:2501.05366*.

Jiajin Liu, Yuanfu Sun, Dongzhe Fan, and Qiaoyu Tan. 2026. Graphsearch: Agentic search-augmented reasoning for zero-shot graph learning. *arXiv preprint arXiv:2601.08621*.

Yuyan Liu, Sirui Ding, Sheng Zhou, Wenqi Fan, and Qiaoyu Tan. 2024a. Moleculargpt: Open large language model (llm) for few-shot molecular property prediction. *arXiv preprint arXiv:2406.12950*.

Zheyuan Liu, Xiaoxin He, Yijun Tian, and Nitesh V Chawla. 2024b. Can we soft prompt llms for graph learning tasks? In *Companion Proceedings of the ACM on Web Conference 2024*, pages 481–484.

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

Haitao Mao, Juanhui Li, Harry Shomer, Bingheng Li, Wenqi Fan, Yao Ma, Tong Zhao, Neil Shah, and Jiliang Tang. 2023. Revisiting link prediction: A data perspective. *arXiv preprint arXiv:2310.00793*.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711.

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.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine*, 29(3):93–93.

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*.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*.

Yucheng Shi, Qiaoyu Tan, Xuansheng Wu, Shaochen Zhong, Kaixiong Zhou, and Ninghao Liu. 2024. Retrieval-enhanced knowledge editing for multi-hop question answering in language models. *arXiv preprint arXiv:2403.19631*.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592*.

Yuanfu Sun, Kang Li, Pengkang Guo, Jiajin Liu, and Qiaoyu Tan. 2026. Mario: Multimodal graph reasoning with large language models. *arXiv preprint arXiv:2603.05181*.

Yuanfu Sun, Zhengnan Ma, Yi Fang, Jing Ma, and Qiaoyu Tan. 2025. Graphicl: Unlocking graph learning potential in llms through structured prompt design. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 2440–2459.

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. 2024. Graphgpt: Graph instruction tuning for large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 491–500.

Xianfeng Tang, Huaxiu Yao, Yiwei Sun, Yiqi Wang, Jiliang Tang, Charu Aggarwal, Prasenjit Mitra, and Suhang Wang. 2020. Investigating and mitigating degree-related biases in graph convoltuional networks. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1435–1444.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, and 1 others. 2017. Graph attention networks. *stat*, 1050(20):10–48550.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning*, pages 5453–5462. pmlr.

Hao Yan, Chaozhuo Li, Ruosong Long, Chao Yan, Jianan Zhao, Wenwen Zhuang, Jun Yin, Peiyan Zhang, Weihao Han, Hao Sun, and 1 others. 2023. A comprehensive study on text-attributed graphs: Benchmarking and rethinking. *Advances in Neural Information Processing Systems*, 36:17238–17264.

Hao Yan, Chaozhuo Li, Jun Yin, Zhigang Yu, Weihao Han, Mingzheng Li, Zhengxin Zeng, Hao Sun, and Senzhang Wang. 2025. When graph meets multimodal: benchmarking and meditating on multimodal attributed graph learning. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 5842–5853.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31.

Qihai Zhang, Xinyue Sheng, Yuanfu Sun, and Qiaoyu Tan. 2025. Trustglm: Evaluating the robustness of graphllms against prompt, text, and structure attacks. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 5912–5923.

Jianan Zhao, Le Zhuo, Yikang Shen, Meng Qu, Kai Liu, Michael Bronstein, Zhaocheng Zhu, and Jian Tang. 2023. Graphtext: Graph reasoning in text space. *arXiv preprint arXiv:2310.01089*.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in neural information processing systems*, 33:7793–7804.

Luyao Zhuang, Shengyuan Chen, Yilin Xiao, Huachi Zhou, Yujing Zhang, Hao Chen, Qinggang Zhang, and Xiao Huang. 2025. Linearrag: Linear graph retrieval augmented generation on large-scale corpora. *arXiv preprint arXiv:2510.10114*.

A Appendix

A.1 Dataset Details

We evaluate AgentGL on 7 text-attributed graph (TAG) benchmarks spanning three domains: *citation networks*, *e-commerce product graphs*, and *social networks*. Across all TAGs, each node is paired with a piece of natural language text (e.g., title/abstract for papers, product descriptions for items, or post text for online forums), which serves as the semantic grounding for retrieval and reasoning; meanwhile, edges encode native relational structures in the underlying domain, including citation links between papers, co-purchase/co-view relations between products, or interaction/co-posting relations in social platforms. This combination yields a realistic setting for agentic graph learning: the agent must jointly leverage *topology* (where to search) and *semantics* (what evidence says) to solve downstream tasks, while avoiding redundant evidence accumulation.

On each dataset, we consider two classical graph-learning problems: node classification (predicting a node's category label) and link prediction (predicting whether an edge exists between a pair of nodes). Node classification evaluates how effectively the model aggregates multi-hop structural cues together with node text to infer labels, whereas link prediction stresses relational reasoning and neighborhood consistency under sparse supervision. Unless otherwise specified, link prediction is formulated as a binary decision, and node classification uses the original multi-class label space of each dataset. Table 5 summarizes key statistics, including the number of nodes, edges, and classes.

For data splits, except for Reddit and Arxiv-2023, we follow the default split protocol used in GraphICL (Sun et al., 2025) and apply subsampling. For node classification, on the two training datasets (OGB-Arxiv and OGB-Products), we sample 3,000 training nodes each for optimization, and for each dataset we sample 1,000 nodes from the original test split for evaluation. For Arxiv-2023, we keep the original split (He et al.) and likewise perform subsampling on its test split for evaluation. For Reddit, since the original benchmark (Yan et al., 2025) is a multimodal graph, we convert it into a TAG by removing the image attributes of each node and retaining only the original textual fields; we then subsample its test split using the same evaluation protocol.

A.2 More Related Work: GraphRAG vs. AGL

GraphRAG-style methods extend classical RAG by incorporating graph structure into evidence selection and organization, typically for open-ended question answering or long-form generation. In this paradigm, a graph (often a knowledge graph) serves as an index or scaffold that helps retrieve and aggregate textual evidence (documents, passages, entity descriptions, or triples) to ground an LLM's generation (Jimenez Gutierrez et al., 2024; Zhuang et al., 2025). The core objective is therefore *generation quality* (e.g., factuality, faithfulness, relevance), where the graph is an auxiliary structure that improves retrieval and attribution. In contrast, Agentic Graph Learning (AGL) treats the graph as the *primary problem instance* rather than an external knowledge base. The goal is to solve graph learning/reasoning tasks whose correctness depends on structural signals (e.g., neighborhood composition, multi-hop dependencies, or structural ranking), such as node classification, link prediction, and other graph-native queries. Accordingly, the agent interacts with the environment through *graph-native operators* that return nodes' text attributes, and the episode terminates with a discrete task decision, instead of free-form generation. This framing yields trajectories that are inherently *graph-operational*: the policy learns *which structural context to acquire* under a budget and *when to stop*, rather than retrieving text to write an answer. Although several recent works (Luo et al., 2025) have explored agentic GraphRAG, this is not equivalent to agentic graph learning, as the two lines of research focus on fundamentally different objectives. Nevertheless, given their high-level similarities, we select representative (canonical) GraphRAG baselines and adapt them to perform graph reasoning, and we provide a detailed empirical comparison in the experiments section.

A.3 Implementation Details

GraphRAG baselines. For HippoRAG2, LinearRAG and GraphCoT, we follow their original settings whenever applicable. For HippoRAG2, we use gpt-4o-mini for entity extraction and nv-embed-v2 as the embedding model for retrieval. For LinearRAG, we use spaCy for named entity recognition and all-mpnet-base-v2 for embeddings. Both configurations match the default choices reported in their respective papers. To build the retrieval

Domain	Dataset	#Nodes	#Edges	#Classes
Citation Network	OGB-Arxiv	169,343	1,166,245	40
Citation Network	PubMed	19,717	44,338	3
Citation Network	Arxiv-2023	46198	78548	40
Amazon Products	OGB-Products (subset)	54,025	74,420	47
Amazon Products	Amazon-Photo	48,362	500,939	12
Amazon Products	Amazon-Computers	87,229	721,107	10
Social Network	Reddit	15,894	566,160	20

Table 5: Dataset statistics used in this paper across three domains. #Classes correspond to the node classification label space; link prediction is treated as binary.

graph in a way that balances downstream task requirements with the original construction scale of these methods, we subsample 500 nodes from each TAG and construct the GraphRAG index using their *original* node text attributes (before any additional processing), which is then used for retrieval during inference. This subsampled graph is used solely for retrieval/augmentation and is applied consistently across compared methods. For GraphCoT, we follow the original implementation and prompting protocol. In the experiments, we categorize it as a GraphRAG method because it shares key characteristics with other GraphRAG approaches: it is primarily designed for knowledge-intensive QA, and it supports reasoning by retrieving evidence from a graph augmented with external knowledge.

GraphLLM baselines. All GraphLLM baselines are implemented under the same configurations as reported in their original papers, including their default prompting/formatting, graph-to-text serialization strategies, and hyperparameter choices.

GNN baselines. For GNN-based baselines, we adopt the same multi-dataset training and transfer protocol as LLaGA (Chen et al., 2024b).

Standard Agentic Search Baselines. As discussed in the paper, these methods do not natively support search over graph-structured evidence. Moreover, allowing unrestricted online search would make the comparison unfair, since they could directly obtain the answer from the web and bypass any reasoning. Therefore, we keep their original prompting strategies, training protocols, and other settings unchanged, and only replace their online search component with a constrained variant that restricts search to the nodes within the input graph. For Search-R1, we apply GRPO training algorithms.

AgentGL. (Hyper-Parameters) We use OpenRLHF as our primary reinforcement learning training framework. For node classification, each graph-native search tool returns at most 5 retrieved nodes per call, and we append their node text attributes as evidence to the model context. Unless otherwise specified, we sample 16 rollouts per prompt and use a single episode per update, with zero warmup. We set the total training batch size to 128 and the rollout batch size to 32, with a KL regularization coefficient of 0 and a learning rate of 2e-6. We cap the maximum sequence length at 1600 and sample with temperature 1.0. For all text encoding, we use the RoBERTa-Large encoder (all-roberta-large-v1). In GCCL, we set the standard normal quantile z to 1.96 and η to 0.05. All experiments are conducted on a node equipped with 8 NVIDIA H100-80G-SXM5 GPUs and 32 Intel Xeon Platinum 8462Y+ CPU cores (2.8 GHz). We report the average accuracy over 2 rounds.

(GCCL Training) For graph-conditioned curriculum learning, we pre-partition the training nodes into three difficulty strata (easy/medium/hard) and allocate a fixed quota to each stage. For both OGB-Arxiv and OGB-Products, Stage 1 uses 800 easy, 500 medium, and 500 hard samples, while Stage 2 uses 200 easy, 500 medium, and 500 hard samples. Training within each stage is conducted on the allocated data in ascending order of difficulty.

(Reward Details) In Stage 1, we implement a lightweight reward server that scores each rollout by combining (i) task correctness, (ii) format compliance, (iii) tool-usage coverage. Concretely, we first extract the predicted label and compare it with the gold category using a normalized string match. The classification reward is set to 1.5 for an exact match, 0 for a mismatch, and negative when the answer is missing (-1.0) or the sample index

is invalid (-0.5). We additionally apply a format reward to enforce a clean and machine-parsable trajectory structure. A response receives +0.5 if it contains exactly one <think> block and exactly one <answer> block; otherwise it receives -0.5. We further check that query/document delimiters are well-formed (the numbers of begin/end tags match); this adds +0.1 if consistent and -0.3 otherwise. To prevent leakage of tool I/O into the final prediction, we penalize cases where the answer block contains query/document tags (-0.5), where the answer is overly verbose (more than 12 whitespace-separated tokens, -0.2), or where the answer block contains any residual <think> content (-0.3). To encourage the agent to explore different graph-native search tools in the bootstrapping stage, we add a search-coverage reward based on which tool tags appear in the rollout. Each distinct tool used contributes +0.5, capped at 2.0 in total. For Stage 2, we keep the format reward and the classification reward unchanged. The only modification is the cognitive-density reward: if any reasoning segment fails to meet the cognitive density requirement, we apply a penalty of -0.2; otherwise, we add a bonus of +0.5 when all segments satisfy the criterion. The segment-length threshold is set to 100 tokens. For link prediction, the only change is the number of evidence nodes returned by each tool. Specifically, 1-hop Neighborhood Search and 2-hop Neighborhood Search still return up to 5 nodes per call, while Structure Salience Search returns 2 nodes and Graph Dense Search returns 3 nodes.

AI Usage. We utilized AI exclusively for proofreading assistance.

A.4 Additional Experiments

Variance analysis. Table 6 reports the performance variance over three independent runs under our RL training setup. Overall, the observed variances are modest across both node classification and link prediction, suggesting that AgentGL training is reasonably stable in practice. A consistent trend is that smaller backbones exhibit larger variance than their larger counterparts (e.g., 3B vs. 7B), which aligns with the intuition that smaller models are more sensitive to stochasticity in sampling and policy optimization. This gap is especially noticeable on Amazon-Photo, where the 3B variants show higher variance across both tasks.

Sensitivity analysis of K value. We further analyze the sensitivity to the neighborhood size K in the main text. As shown in Table 7, increasing K from 1 to 5 consistently improves accuracy on both OGB-Arxiv and Amazon-Photo, with the best results achieved at K=5 (68.9% and 59.9%, respectively). This suggests that a moderately expanded neighborhood provides more informative structural context for reasoning. In contrast, further increasing K to 7 leads to a slight performance drop on both datasets, indicating that overly large neighborhoods may introduce redundant or noisy information. Overall, these results show that an appropriate neighborhood size is important for balancing contextual richness and noise, and justify our default choice of K=5.

Model	OGB-Products		Amazon-Photo	
	Var(NC)	Var(LP)	Var(NC)	Var(LP)
AgentGL-7B-GRPO	0.3	0.4	0.2	0.3
AgentGL-3B-GRPO	0.6	0.3	0.7	1.1
AgentGL-7B-R++	0.2	0.4	0.5	0.7
AgentGL-3B-R++	0.4	0.6	0.7	0.8

Table 6: Variance of performance over 3 runs for node classification (NC) and link prediction (LP) on two datasets.

Table 7: Sensitivity analysis of K value (AgentGL-7B-GRPO, Node Classification).

Acc (%)	$K=1$	$K=3$	$K=5$	$K=7$
OGB-Arxiv	65.7	68.1	68.9	68.6
Amazon-Photo	56.2	59.1	59.9	59.7

A.5 Case Study

To make AgentGL's decision process more interpretable, we present representative rollouts for both node classification and link prediction. Fig. 4 shows an NC example from the Amazon domain: the model first forms a hypothesis from the anchor text, then verifies it by querying local neighborhoods (1-hop/2-hop) and a global prior (PageRank). We highlight the key reasoning sentences that directly support the final label, illustrating how evidence aggregation over the graph reduces ambiguity and prevents over-reliance on the anchor text alone. Fig. 5 provides an LP example from Reddit, where the model validates a potential edge by searching common 1-hop neighbors; the shared co-post motif offers strong structural evidence that the two endpoints lie in the same tight cluster. Across cases, the agent typically terminates early once the searched evidence becomes self-consistent, avoiding redundant searches under a bounded budget.

A.6 Prompt Template

We use a standardized prompt format to expose graph-native search tools to the policy in a machine-parsable manner. Figs. 6–13 summarize the templates used for node classification and link prediction. Each prompt consists of a `system` instruction specifying the task, the available search pools, and strict output constraints, followed by a `user` message that provides the anchor instance (NC) or a node pair (LP). Placeholders {{...}} are instantiated with dataset-specific task lines, label spaces, relation descriptions, and per-pool Top-K limits, while keeping the action interface invariant across domains. During inference (and RL rollouts), all tool calls must appear inside a single `<think>` block, and the final prediction must be emitted as exactly one label enclosed by `<answer>` tags. This design ensures consistent trajectory logging, robust automatic reward computation, and reproducible evaluation under a fixed search budget.

Algorithm 1: AgentGL: Agentic Graph Learning with RL

Input: Graph $\mathcal{G}$; stage data $\mathcal{D}^{(1)}, \mathcal{D}^{(2)}$; tools $\mathcal{S} = \{\tau_{1HOP}, \tau_{2HOP}, \tau_{SS}, \tau_{DENSE}\}$; ref policy π_{ref}; KL coeff β; budget B; rollouts N; GCCL params: (z, η) (NC), Sim (LP); Stage-1 params: $\lambda_a, \eta_{\text{cov}}$; Stage-2 params: $\lambda_a, \alpha, \lambda_d, \delta$; RL alg ALG $\in \{\text{GRPO}, \text{R++}\}$.

Output: Optimized π_θ.

foreach $s \in \{1, 2\}$ **do** // Two RL Stages
  $\mathcal{D} \leftarrow \mathcal{D}^{(s)}$
  if x *is node* **then** // NC: $S_{NC}(v) = \text{WLB}(\hat{p}_v, d_v; z) + \eta \log(1 + d_v)$
    $Scores \leftarrow \texttt{CalcDiffNC}(\mathcal{D}, \mathcal{G}; z, \eta)$
  else
    $Scores \leftarrow \texttt{CalcDiffLP}(\mathcal{D}; \text{Sim})$
    // $S_{LP}(e) = y_e \text{Sim}(x_u, x_v) + (1 - y_e)(1 - \text{Sim}(x_u, x_v))$
  $\{\mathcal{C}_\text{E}, \mathcal{C}_\text{M}, \mathcal{C}_\text{H}\} \leftarrow \texttt{SplitByDifficulty}(\mathcal{D}, Scores)$
  if $s = 1$ **then** // **Stage 1: Graph-Native Search Policy Bootstrapping**
    // $R(\tau) = r_{\texttt{FMT}}(\tau) + r_{\texttt{ACC}}(\hat{y}, y) + r_{\texttt{COV}}(\tau)$
  else
    // **Stage 2: Mitigating Search Overuse** (inject retrospective trigger after each tool call)
    // $R(\tau) = r_{\texttt{FMT}}(\tau) + r_{\texttt{ACC}}(\hat{y}, y) + r_{\texttt{depth}}(z)$ (discard $r_{\texttt{COV}}$)
  foreach $k \in \{E, M, H\}$ **do**
    while *stage-s budget not exhausted* **do**
      $\mathcal{B} \leftarrow \texttt{SampleBatch}(\mathcal{C}_k)$
      foreach $(x, Q, y) \in \mathcal{B}$ **do**
        for $n \leftarrow 1$ **to** N **do**
          $\tau^{(n)} \leftarrow \texttt{Rollout}(\pi_\theta, \mathcal{G}, \mathcal{S}; x, Q, B, s)$ // $a_t = \langle s_t, q_t \rangle$ or ANSWER
          $r_{\text{FMT}} \leftarrow \texttt{FmtReward}(\tau^{(n)})$;
          $r_{\text{ACC}} \leftarrow \texttt{AccReward}(\hat{y}^{(n)}, y) = \lambda_a \mathbf{1}[\hat{y}^{(n)} = y]$
          if $s = 1$ **then**
            $r_{\text{COV}} \leftarrow \texttt{CoverageReward}(\tau^{(n)}) = \eta_{\text{cov}} \sum_j \mathbf{1}[\exists t : a_t = \tau_j]$
            $R^{(n)} \leftarrow r_{\text{FMT}} + r_{\text{ACC}} + r_{\text{COV}}$
          else
            $r_{\text{depth}} \leftarrow \texttt{DepthReward}(\tau^{(n)}; \alpha, \lambda_d, \delta) = \alpha \mathbf{1}[N_{\text{short}} = 0] - \lambda_d N_{\text{short}}$
            $R^{(n)} \leftarrow r_{\text{FMT}} + r_{\text{ACC}} + r_{\text{depth}}$
      $\theta \leftarrow \texttt{UpdatePolicy}(\theta, \{\tau^{(n)}, R^{(n)}\}, \pi_{\text{ref}}, \beta; \text{ALG})$ // GRPO/R++ w/ KL
return θ

Figure 4: A node classification (NC) case illustrating graph-native tool use. We highlight key reasoning sentences that drive the final decision.

Case Study: Node Classification (NC)

Target instance text attribute:
Tripod is not sturdy enough for heavy use; legs bulge and move, hinge is weak. Not recommended for rifles or kneeling shots.

Ground truth label:
Tripods & Monopods

Model prediction:
Tripods & Monopods (✓ Matched)

Model response:

<think>
The anchor summary is already strongly indicative of a tripod: it talks about leg sections bulging, the legs shifting under load, and a weak hinge, plus mentions kneeling shots / rifles (a common tripod/spotting-scope/shooting-use context). I'll still start with local graph evidence and only expand if anything looks like it could actually be an accessory rather than the tripod itself.

Round 1 — 1-hop: Make the query reflect both the mechanical failure points (hinge/legs) and the use case (kneeling/shooting) so the nearest co-purchases are maximally revealing.

<|begin_of_query|>1-hop:tripod weak hinge legs bulge unstable kneeling shooting<|end_of_query|>
<|begin_of_documents|>
(1) Budget tripod; quick leg locks; okay for light cameras but wobbles when fully extended.
(2) Tripod for spotting scope / range; stable at low height, awkward for kneeling positions.
(3) Lightweight travel tripod with ball head; users mention leg flex and hinge durability issues.
(4) "Heavy duty" tripod claims; complaints about loose joints and bulging leg sections under load.
(5) Compact aluminum tripod; fine for occasional use, not recommended for heavy rifles or heavy camera rigs.
<|end_of_documents|>

These direct neighbors read like the same product family: tripods criticized for stability, load rating, and joint/hinge reliability—plus the same shooting/kneeling angle shows up. That's enough to lean hard toward "Tripods & Monopods," but I'll expand once to check whether the node could be a tripod component (e.g., a head/plate) that's being discussed with tripod-like language.

Round 2 — 2-hop:
Probe for whether the neighborhood shifts toward "plates/heads/clamps" as the main object, or stays centered on the support legs/column system.

<|begin_of_query|>2-hop:tripod load capacity leg locks center column wobble head loosens<|end_of_query|>
<|begin_of_documents|>
(1) Tripod head loosens and tilt lock slips; fine for phones, not for heavier cameras.
(2) Tripod legs slip under load; center column wobble; advertised rating overstated.
(3) Monopod suggested as alternative; compared directly against tripods for portability.
(4) Video tripod: smooth pan but poor tilt stability for heavier rigs.
(5) Travel tripod: stable low, unstable fully extended; needs added weight for stability.
<|end_of_documents|>

Still tripod-first: legs, center column, extension stability, and load capacity dominate the neighborhood, with monopod comparisons appearing naturally. Nothing suggests the anchor is primarily a quick-release plate or a clamp. One global pass can help ensure we're not missing a category boundary in the label set, so I'll do a pagerank lookup.

Round 3 — pagerank:
Use terms that usually surface canonical support gear (tripods/monopods/heads/plates) and see what comes back.

<|begin_of_query|>pagerank:tripod monopod tripod head quick release plate travel tripod<|end_of_query|>
<|begin_of_documents|>
(1) Popular entry-level tripod: quick-release plate, bubble level, weight hook.
(2) Common ball head / tripod head frequently paired with tripods.
(3) Tripod carry bag / case often co-purchased for transport.
(4) Universal quick-release plate accessory used across tripod brands.
(5) Well-known travel tripod line referenced in comparisons (compact fold, lightweight).
<|end_of_documents|>

This is exactly the ecosystem we'd expect when the anchor is a tripod: the graph surfaces other tripods and the usual companion pieces (head/plate/bag), but the core story remains "support legs + stability under load," matching the anchor summary. A fourth "similar" search would likely be redundant, so I'll stop here.
</think>
<answer>Tripods & Monopods</answer>

Figure 5: A link prediction (LP) case from Reddit. The model verifies a strong co-post motif by retrieving dense common 1-hop neighbors.

Case Study: Link Prediction (LP) — Reddit

Target instance text attribute:

Node U: laggar falcon - falco jugger - tal chappar, rajasthan, india

Node V: greater hoopoe-lark : the courtship display of the male consists of rising with fluttering wing-strokes and then diving down with closed wings to a perch. - banni grassland, gujarat, india

Ground truth label:

yes

Model prediction:

yes (✓ Matched)

Model response:

<think>
U and V are both bird-observation style Reddit posts with the same "species – location – India" pattern. The fastest way to verify that is to check for common 1-hop neighbors: if they share multiple direct neighbors that are themselves highly consistent, that's strong evidence of an edge.

Round 1 – 1-hop (prioritize common direct neighbours):

<|begin_of_query|>1-hop:birding india tal chappar rajasthan banni grassland gujarat laggar falcon hoopoe-lark<|end_of_query|>

<|begin_of_documents|>
(1) [common one_hop neighbour] western marsh harrier - little rann of kutch, gujarat, india This neighbour links both Node U and Node V.
(2) [common one_hop neighbour] peregrine falcon - little rann of kutch, gujarat, india This neighbour links both Node U and Node V.
(3) [common one_hop neighbour] common babbler - little rann of kutch, gujarat, india This neighbour links both Node U and Node V.
(4) [common one_hop neighbour] baya weaver - lodai, kutch, gujarat, india This neighbour links both Node U and Node V.
(5) [common one_hop neighbour] short-eared owl - asio flammeus - little rann of kutch, gujarat, india This neighbour links both Node U and Node V.
<|end_of_documents|>

They share a dense set of direct common neighbors, all of which are the same type of birding posts from the same west-India region (Kutch / Gujarat) and the same naming template. This is a strong co-post motif in the Reddit graph, indicating U and V belong to the same closely connected cluster.
</think>
<answer>yes</answer>

Figure 6: NC prompt template (core). Placeholders in the prompt use {{...}}.

Prompt Template: Node Classification (Core)

```
<|im_start|>system
You are a reasoning assistant with the ability to perform graph searches to help you answer the
      user's question accurately.
{{TASK_LINE}}
You can use graph search to retrieve neighbor node information to support your prediction.

THINK/ANSWER FORMAT:
- Do ALL internal reasoning inside <think>...</think>.
- Provide ONLY the final category label inside <answer>...</answer>.
- Never reveal your chain-of-thought outside <think>...</think>.
- When confident, output exactly one label as required.

GRAPH SEARCH POOLS (for context retrieval):
{{POOLS}}

SEARCH FORMAT ({{SEARCH_LIMITS_DESC}}):
- All searches MUST be performed INSIDE the <think>...</think> block.
- Use one of the pool-specific tags below:
  * 1-hop:    <|begin_of_query|>1-hop:QUERY<|end_of_query|>
  * 2-hop:    <|begin_of_query|>2-hop:QUERY<|end_of_query|>
  * pagerank: <|begin_of_query|>pagerank:QUERY<|end_of_query|>
  * similar:  <|begin_of_query|>similar:QUERY<|end_of_query|>
- Results are returned as:
  <|begin_of_documents|> ... <|end_of_documents|>

GUIDELINES:
- One search per round; multiple rounds allowed.
- Max total searches = {{MAX_SEARCH_LIMIT}}.
- Output only <think>...</think> and a single <answer>...</answer>.

OUTPUT REQUIREMENT:
{{OUTPUT_REQ}}
<|im_end|>

<|im_start|>user
{{ANCHOR_HDR}}Anchor Node Information:
{{SUMMARY_SNIPPET}}
<|im_end|>

<|im_start|>assistant
```

Figure 7: Dataset-specific inserts for Arxiv (NC).

Dataset Inserts: Arxiv

```
TASK_LINE:
Your task is to predict the category of arXiv computer science (cs) papers.

OUTPUT_REQ:
- Final output must be exactly: <answer>cs.XX</answer>

POOLS:
- 1-hop: direct neighbors (papers that directly cite or are cited by the anchor). Returns up to
      {{TOPK_ONE_HOP}} nodes.
- 2-hop: neighbors of neighbors expanding the local region. Returns up to {{TOPK_TWO_HOP}}
     nodes.
- pagerank: globally influential nodes selected by PageRank. Returns up to {{TOPK_PAGERANK}}
     nodes.
- similar: globally most semantically similar nodes by embedding similarity. Returns up to {{
     TOPK_SIMILAR}} nodes.

ANCHOR_HDR:
Now please predict the category of the anchor node paper:

SEARCH_LIMITS_DESC:
per-pool limits -> 1-hop {{TOPK_ONE_HOP}}, 2-hop {{TOPK_TWO_HOP}}, pagerank {{TOPK_PAGERANK}},
     similar {{TOPK_SIMILAR}}
```

Figure 8: Dataset-specific inserts for PubMed (NC).

Dataset Inserts: PubMed

```
TASK_LINE:
Your task is to predict the category of PubMed biomedical papers.

OUTPUT_REQ:
- Final output must be exactly one of the listed PubMed categories inside <answer>...</answer>.

POOLS:
- 1-hop: direct neighbors (papers that directly cite or are cited by the anchor). Returns up to
      {{TOPK_ONE_HOP}} nodes.
- 2-hop: neighbors of neighbors expanding the local region. Returns up to {{TOPK_TWO_HOP}}
     nodes.
- pagerank: globally influential nodes selected by PageRank. Returns up to {{TOPK_PAGERANK}}
     nodes.
- similar: globally most semantically similar nodes by embedding similarity. Returns up to {{
     TOPK_SIMILAR}} nodes.

ANCHOR_HDR:
Now please predict the category of the anchor node paper:
Available PubMed categories:
- {{CATEGORY_LIST}}

SEARCH_LIMITS_DESC:
per-pool limits -> 1-hop {{TOPK_ONE_HOP}}, 2-hop {{TOPK_TWO_HOP}}, pagerank {{TOPK_PAGERANK}},
     similar {{TOPK_SIMILAR}}
```

Figure 9: Dataset-specific inserts for Amazon (NC).

Dataset Inserts: Amazon

```
TASK_LINE:
Your task is to predict the category of Amazon products.

OUTPUT_REQ:
- Final output must be exactly one of the listed Amazon categories inside <answer>...</answer>.

POOLS:
- 1-hop: direct neighbors (products that are frequently co-purchased with the anchor). Returns
    up to {{TOPK_ONE_HOP}} nodes.
- 2-hop: neighbors of neighbors expanding the local region in the co-purchasing graph. Returns
    up to {{TOPK_TWO_HOP}} nodes.
- pagerank: globally influential products selected by PageRank. Returns up to {{TOPK_PAGERANK}}
     nodes.
- similar: globally most semantically similar products by embedding similarity. Returns up to
    {{TOPK_SIMILAR}} nodes.

ANCHOR_HDR:
Now please predict the category of the anchor node product:
Available Amazon categories:
- {{CATEGORY_LIST}}

SEARCH_LIMITS_DESC:
per-pool limits -> 1-hop {{TOPK_ONE_HOP}}, 2-hop {{TOPK_TWO_HOP}}, pagerank {{TOPK_PAGERANK}},
    similar {{TOPK_SIMILAR}}
```

Figure 10: Dataset-specific inserts for Reddit (NC).

Dataset Inserts: Reddit

```
TASK_LINE:
Your task is to predict the subreddit category of a Reddit post.

OUTPUT_REQ:
- Final output must be exactly one of the listed subreddit categories inside <answer>...</
    answer>.

POOLS:
- 1-hop: other posts from the same author (also_posted set). Returns up to {{TOPK_ONE_HOP}}
    nodes.
- 2-hop: same as 1-hop in this dataset (duplicates allowed). Returns up to {{TOPK_TWO_HOP}}
    nodes.
- pagerank: globally influential posts selected by PageRank over the author cliques. Returns up
     to {{TOPK_PAGERANK}} nodes.
- similar: globally most semantically similar posts by embedding similarity. Returns up to {{
    TOPK_SIMILAR}} nodes.

ANCHOR_HDR:
Now please predict the category of the anchor post:
Available subreddit categories:
- {{CATEGORY_LIST}}

SEARCH_LIMITS_DESC:
per-pool limits -> 1-hop {{TOPK_ONE_HOP}}, 2-hop {{TOPK_TWO_HOP}}, pagerank {{TOPK_PAGERANK}},
    similar {{TOPK_SIMILAR}}
```

Figure 11: Link prediction (LP) prompt template (core). Placeholders in the prompt use {{...}}.

Prompt Template: Link Prediction (Core)

```
<|im_start|>system
You are a reasoning assistant with access to graph searches.
Determine whether two nodes from the {{DATASET}} dataset should be connected.
{{RELATION_DESC}}
Treat the task as binary classification and return 'yes' if the edge should exist and 'no'
    otherwise.

THINK/ANSWER FORMAT:
- Perform all reasoning and search planning inside <think>...</think>.
- Output ONLY the final judgment as <answer>yes</answer> or <answer>no</answer>.
- Never leak your chain-of-thought outside of <think>...</think>.

GRAPH SEARCH POOLS:
- 1-hop: prioritize common direct neighbours (connected to BOTH endpoints U and V). If
    insufficient, fill with non-common 1-hop neighbours from U and/or V (balanced when possible
    ).
- 2-hop: prioritize common 2-hop neighbours (reachable from BOTH endpoints within two hops). If
     insufficient, fill with non-common 2-hop neighbours from U and/or V (balanced when
    possible).
- pagerank: list globally influential reference edges selected offline using PageRank, as
    complementary structural priors.
- similar: retrieve Top-K node pairs most similar to the current pair, including their edge
    status as reference.

SEARCH FORMAT ({{SEARCH_LIMITS_DESC}}):
- Every search must happen inside <think>...</think>.
- To launch a search, emit exactly one of the following tags with an optional free-form hint:
  * <|begin_of_query|>1-hop:Query<|end_of_query|>
  * <|begin_of_query|>2-hop:Query<|end_of_query|>
  * <|begin_of_query|>pagerank:Query<|end_of_query|>
  * <|begin_of_query|>similar:Query<|end_of_query|>
- Retrieved blocks arrive as <|begin_of_documents|> ... <|end_of_documents|>.
- Use at most one search per round and no more than {{MAX_SEARCH_LIMIT}} total searches before
    answering.

GUIDELINES:
- Start by analyzing the two node descriptions.
- Prefer covering multiple pools (shared neighbours, unique structure, global priors).
- Keep the final judgement concise.
- Final answer MUST be <answer>yes</answer> or <answer>no</answer>.
<|im_end|>

<|im_start|>user
We are investigating whether the following two nodes should be linked:
- Node U (id={{NODE_U}}): {{SUMMARY_U}}
- Node V (id={{NODE_V}}): {{SUMMARY_V}}

Use up to {{MAX_SEARCH_LIMIT}} searches to gather evidence from the graph retriever.
- Label 1 corresponds to <answer>yes</answer>.
- Label 0 corresponds to <answer>no</answer>.
Remember to answer strictly with yes/no enclosed by <answer> tags.
<|im_end|>

<|im_start|>assistant
```

Figure 12: Dataset-specific relation descriptions used in LP prompts.

Dataset Inserts: Relation Descriptions (LP)

```
[arxiv/pubmed]
Nodes are research papers. An edge represents a citation linkage between the two papers.

[amazon/products]
Nodes are Amazon products. An edge indicates strong co-purchase relationships between the items
    .

[reddit]
Nodes are Reddit posts. An edge indicates strong co-post relationships between the posts.

[default]
Nodes come from the same graph dataset; an edge captures the canonical relation defined for
    that dataset.
```

Figure 13: Per-pool search limits description used in LP prompts.

Dataset Inserts: Search Limits (LP)

```
SEARCH_LIMITS_DESC:
per-pool limits -> 1-hop {{TOPK_ONE_HOP}}, 2-hop {{TOPK_TWO_HOP}}, pagerank {{TOPK_PAGERANK}},
    similar {{TOPK_SIMILAR}}
```