Can Knowledge-Graph-based Retrieval Augmented Generation Really Retrieve What You Need?

Junchi Yu[1], **Yujie Liu**[2], **Jindong Gu**[1], **Philip Torr**[1], **Dongzhan Zhou**[2*]
[1]Department of Engineering Science, University of Oxford, UK
[2]Shanghai Artificial Intelligence Laboratory, China
junchi.yu@eng.ox.ac.uk, liuyujie1@pjlab.org.cn, jindong.gu@eng.ox.ac.uk,
philip.torr@eng.ox.ac.uk, zhoudongzhan@pjlab.org.cn

Abstract

Retrieval-Augmented Generation (RAG) based on knowledge graphs (KGs) enhances large language models (LLMs) with structural and textual external knowledge. Yet, existing KG-based RAG methods struggle to retrieve accurate and diverse information when handling complex queries. By modeling KG-based retrieval as a multi-step decision process, Process Reward Models (PRMs) offer a promising solution to align the retrieval behavior with the query-specific knowledge requirements. However, PRMs heavily rely on process-level supervision signals that are expensive and hard to obtain on KGs. To address this challenge, we propose **GraphFlow**, a framework that efficiently retrieves *accurate and diverse* knowledge required for complex queries from text-rich KGs. GraphFlow employs a detailed balance objective with local exploration to jointly optimize a retrieval policy and a flow estimator. The flow estimator factorizes the outcome reward of the retrieval results into the intermediate retrieval steps. Such reward factorization guides the retrieval policy to retrieve candidates from KGs in proportion to their outcome reward. This allows GraphFlow to explore relevant regions of KGs that yield diverse and accurate results. We evaluate GraphFlow on STaRK benchmark, which includes real-world queries from multiple domains over text-rich KGs. GraphFlow outperforms strong KG-based RAG baselines including GPT-4o by 10% performance gain on both retrieval accuracy and diversity metrics. GraphFlow also shows strong generalization by effectively retrieving information from unseen KGs to support new-domain queries, highlighting its effectiveness and robustness [2].

1 Introduction

Retrieval-Augmented Generation (RAG) [36] has emerged as a promising framework to reduce the hallucination of Large Language Models (LLMs) by mitigating the gap between model knowledge and factual knowledge [28, 89, 26]. Traditional RAG usually employs an unstructured vector-indexed database as the external knowledge source, where the text corpus is indexed using pretrained encoders to support retrieval. Recent work has explored knowledge graphs (KGs) [68] as a structural alternative to the external knowledge source of RAG [59]. KGs enjoy several advantages over the vector-indexed database in traditional RAG, such as representing relational information with graph structures [88], integrating knowledge from heterogeneous resources [59], and enhancing interpretability by neural-symbolic reasoning [87]. Thus, KG-based RAG has demonstrated great potential in enhancing LLMs in many domains, including medical diagnosis [67], biochemistry [77], and physics [72].

*Corresponding Author
[2]Code is available https://github.com/Samyu0304/GraphFlow

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

 https://doi.org/10.52202/085713-3200

Recent KG-based RAG methods employ two approaches to retrieve information from KGs when receiving an input query [47, 19, 48]. Retrieval-based approaches [21, 37, 80] leverage pretrained language models [62] to encode KG text into embeddings, and use retriever models [31] to identify relational triplets or subgraphs in KGs that most support the query. And agent-based methods treat LLMs as searching agents to navigate across KGs and retrieve a relational path with supporting information for a given query [68, 47, 50].

While KG-based RAG methods show promise in retrieving structural information for simple relational queries, their effectiveness is limited in more complex ones. As shown in Figure 1, structural information in KGs alone is often sufficient for many relational queries. For example, retrieving the relation triplet $(Alice, daughter, Bob)$ adequately answers the question "Who is the father of Alice?". However, complex queries typically require leveraging both structural and textual information during retrieval [50]. Consider the paper searching query "Please list the papers published by University A relevant to research topic B". Addressing such a query requires understanding authorship and affiliation relationships, as well as text descriptions of paper content, research topics, institutions, and authors. The fusion of relational and text knowledge presents a significant challenge for accurate retrieval with KG-based RAG methods. Another challenge is the diversity of retrieval targets in complex queries. Unlike relational queries corresponding to a single deterministic retrieval target (e.g., $Alice \xrightarrow{daughter} Bob$), complex queries require retrieving a diverse set of candidates. For example, the paper searching query in Figure 1 corresponds to multiple retrieval targets. Therefore, KG-based RAG must emphasize both retrieval accuracy and diversity when supporting complex queries. Unfortunately, our empirical findings reveal that existing KG-based RAG methods face challenges in achieving this goal.

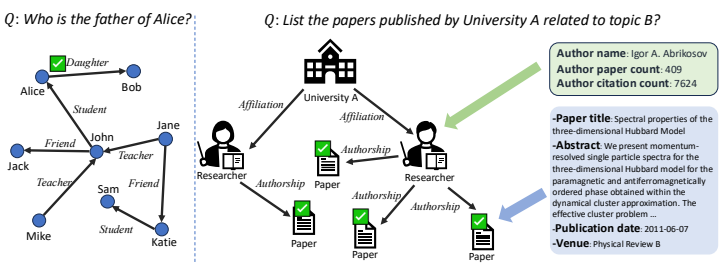

Figure 1: Comparison of the retrieval tasks between relational and complex queries.

To overcome the above challenge, it is essential to align the retrieval process of KG-based RAG with the diversity and accuracy demands for complex queries. Process Reward Models (PRMs) [54, 90, 92, 85] offer a promising framework for this goal. By providing step-wise guidance, PRMs have been widely used in LLM alignment [40], reasoning [7] and planning [6] when treating these tasks as the multi-step decision. In KG-based RAG, the retrieval process can be naturally viewed as a multi-step decision process, where an agent traverses a KG and expands its retrieval trajectory at each step. PRMs can provide step-wise guidance for the agent to retrieve desired information for complex queries. However, training a PRM needs high-quality preference datasets with fine-grained and process-level reward signals [57, 76, 27]. In KG-based RAG, assessing the process-level reward at each step in retrieval trajectories is expensive. Only the outcome reward is easily available (i.e., whether the retrieval trajectory can support a query or not).

Proposed Work. We present GraphFlow, a novel framework for supporting complex queries by retrieving accurate and diverse knowledge from knowledge graphs (KGs), without relying on process-level reward supervision. Inspired by GFlowNet [3], GraphFlow formulates the problem of retrieving from KGs as learning a retrieval policy that generates retrieval trajectories with probabilities proportional to their outcome rewards. Thus, the retrieval trajectory that better supports the query is retrieved with a higher probability, leading to diverse and accurate retrieval results. To achieve this, GraphFlow jointly trains the retrieval policy with a flow estimator, which assigns non-negative flow values to partial trajectories. These flow values decompose the final outcome reward across intermediate retrieval steps, thereby providing rich supervision signals without requiring explicit process-level rewards. The retrieval policy is guided by these flow values and receives process-level supervision "for free". We adopt the detailed balance objective [64] to co-train the retrieval policy and the flow estimator. To further enhance training efficiency, we introduce a local exploration strategy that reduces visits to low-reward regions of the KG. Thus, GraphFlow effectively explores high-reward regions of the retrieval space, leading to more accurate and diverse retrievals that better support complex query.

We evaluate the effectiveness of GraphFlow on the STaRK benchmark [74], which involves retrieving from text-rich KGs for real-world queries across multiple domains. Extensive experiments demonstrate that GraphFlow consistently produces high-quality and diverse retrieval results, outperforming both Process Reward Models (PRMs) and existing KG-based RAG methods. Notably, GraphFlow surpasses strong KG-based RAG baselines instantiated with GPT-4o, achieving an average improvement of 10% in both retrieval accuracy and diversity metrics. In addition, GraphFlow enjoys strong generalization capabilities and can retrieve from unseen KGs to support queries in new domains.

2 Preliminary and Notations

2.1 KG-based RAG

We denote a knowledge graph (KG) as $\mathcal{G} = \{\mathbb{V}, \mathbb{E}\}$ where $\mathbb{V}$ and $\mathbb{E}$ are the sets of nodes and edges. The node V_i is associated with a short text description of an entity (e.g. $V_i =$ 'Alice'). And the edge e_{ij} denotes the relationship between node V_i and V_j. For example, $e_{ij} =$ 'daughter' represents the relationship between $V_i =$ 'Alice' and $V_j =$ 'Bob'.

Retrieval-based Approach. For an input query $\mathcal{Q}$, the retrieval-based method [21, 37, 13, 48, 80, 4] first encodes the texts in nodes and edges into embeddings using a pretrained LM [62]. Then, a retriever $\mathrm{Ret}(\cdot)$ is employed to retrieve a subgraph $\mathcal{G}_{\text{sub}}$ from $\mathcal{G}$ [84] that can support answering $\mathcal{Q}$:

$$\max_{G_{\text{sub}}} P(\mathcal{A}|\mathcal{Q}, G_{\text{sub}}), \quad G_{\text{sub}} = \mathrm{Ret}(\mathcal{G}). \tag{1}$$

Here $\mathcal{A}$ is the answer to the input query. Some works employ non-parametric retrievers, such as dense retriever with vector similarity [1] and Prize-Collecting Steiner Tree (PCST) algorithm [21] to retrieve from KGs. Other works train parameterized retriever models based on Multi-layer Perceptron (MLP) and Graph Neural Network (GNN) [82] to retrieve from KGs.

Agent-based Approach. Recent work formulates retrieval on KGs as a multi-step decision process [68, 47, 38] and employs LLM agents to search on KGs due to their superior capability in planning. For an input query $\mathcal{Q}$, the agent $\mathrm{LLM}(\cdot)$ starts from an initial node V_0 and searches T steps incrementally in a KG to produce a retrieval trajectory $\tau = V_0 \to \cdots \to V_{T-1} \to V_T$ to support $\mathcal{Q}$:

$$\max_{\tau \in \mathcal{T}} P(\mathcal{A}|\mathcal{Q}, \tau), \quad \tau = \mathrm{LLM}(\mathcal{G}). \tag{2}$$

The inital node V_0 can be identified using Entity name recognition (ENR) [18] or vector similarity matching [63]. At step t, the searching agent expands the trajectory conditioned on the input query $\mathcal{Q}$, the partial trajectory at t step $\tau_{\leq t} = V_0 \to \cdots \to V_{t-1} \to V_t$:

$$V_{t+1} \sim P_{\text{LLM}}(V_{t+1}|\mathcal{Q}, \tau_{\leq t}), \quad V_{t+1} \in \mathcal{N}(V_t). \tag{3}$$

Here P_{LLM} is the policy instantiated by an LLM, and $\mathcal{N}(V_t)$ is the neighborhood node set of V_t.

2.2 Process Reward Models

Process Reward Models (PRMs) have emerged as a promising framework for aligning large language models (LLMs) with human preferences [85, 73, 40]. For a multi-step decision problem, denote $s \in \mathcal{S}$ as the state and $a \in \mathcal{A}$ as the action. Training a PRM requires a preference dataset $\mathcal{D} = \{(a_i^+, a_i^-, s_i) \mid i = 1, \cdots, N\}$, where a_i^+ and a_i^- are positive and negative actions at state s_i, respectively. Such datasets can be constructed through human supervision [40], rule-based heuristics [55], or LLM-generated annotations [76]. The goal of PRM is to learn a scoring function $r_\theta(a, s) : \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ that assigns real-valued preference scores to action–state pairs by minimizing the following objective:

$$\mathcal{L}_{\text{PRM}} = -\mathrm{E}_{(a_i^+, a_i^-, s_i) \sim \mathcal{D}} \log[\sigma(r_\theta(a_i^+, s_i) - r_\theta(a_i^-, s_i))], \tag{4}$$

where $\sigma(\cdot)$ denotes the sigmoid function. Once trained, the PRM can be used to provide step-wise preference signals for LLM alignment [40, 73]. Additionally, it can directly guide the multi-step decision with a soft policy $P(s_{t+1}|s_t) \propto e^{r_\theta(s_t, a_t)}$ [7, 73, 76, 92].

 https://doi.org/10.52202/085713-3200

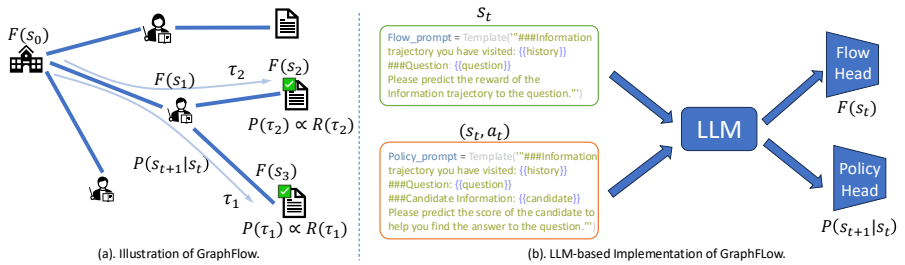

Figure 2: An overview of the proposed GraphFlow framework. (a). GraphFlow employs a flow estimator $F(\cdot)$ to factorize the outcome reward $R(\tau)$ of a retrieval trajectory τ to flow value $F(s_t)$. The flow value guides to learn a policy $P(s_{t+1}|s_t)$ that leads to accurate and diverse retrieval results for complex queries. (b) We introduce an LLM-based implementation of GraphFlow to enhance KG-based RAG on text-rich KGs.

3 Method

3.1 KG-based RAG as Multi-step Decision

Problem Formulation. Given an input query $\mathcal{Q}$, our goal is to retrieve a set of K target nodes $\mathbb{V}^* = \{V_i \mid i = 1, \cdots, K\}$ from a text-rich knowledge graph (KG) $\mathcal{G} = \{\mathbb{V}, \mathbb{E}, \mathbb{D}\}$ such that the associated documents $\mathbb{D}^* = \{D_i^* \mid i = 1, \cdots, K\}$ can support answering $\mathcal{Q}$. Here, $V_i \in \mathbb{V}$ is a node, $E_{ij} \in \mathbb{E}$ is an edge between V_i and V_j indicating their relation. Each D_i denotes the textual document associated with node V_i (e.g., the content of a paper).

Agent-based Retrieval as a Multi-step Decision Process. To effectively leverage both relational and text information in the KG, we employ the agent-based approach initiated with an LLM due to its superior text understanding and planning ability. We formulate the agent-based retrieval as a multi-step decision problem, consisting of the following components:

- **State.** The agent starts retrieval from the initial state $s_0 = (\mathcal{Q}, \{D_0\})$, where D_0 is the document associated with the source node V_0 from which the retrieval process is initiated. At step t, the agent arrives at node V_t and the current state is defined as $s_t = (\mathcal{Q}, \{D_j\}_{j=0}^t)$, where $\{D_j\}_{j=0}^t$ is the set of documents collected along the partial retrieval trajectory $\tau_{\leq t} = V_0 \to \cdots \to V_t$.
- **Action.** Given state s_t, the agent selects an action $a_t \in \mathcal{A}(s_t)$, corresponding to moving from V_t to a adjacent node $V_{t+1} \in \mathrm{N}(V_t)$ along edge $E_{t,t+1}$. The agent then retrieves the documents D_{t+1} associated with V_{t+1}.
- **Transition.** The agent transits to state $s_{t+1} = (\mathcal{Q}, \{D_j\}_{j=0}^{t+1})$. This process continues until either the document D_{t+1} is deemed sufficient to support the query $\mathcal{Q}$, or a predefined maximum number of steps is reached.
- **Reward.** Upon termination, the agent receives a reward $R(\tau)$ for the retrieval trajectory τ. The reward is calculated whether the document D_{T} associated with the terminal node V_{T} of trajectory τ can the query (i.e. $D_{\mathrm{T}} \in \mathbb{D}^*$).

Energy-based Modeling for Accurate and Diverse Retrieval. As shown in Figure 2 (a), the goal of GraphFlow is to learn the policy $P(s_{t+1} \mid s_t)$ that can effectively retrieve accurate and diverse information from a knowledge graph (KG) to support answering an input query. To this end, we formulate the retrieval process as an energy-based distribution over trajectories:

$$P(\tau) = \prod_{t=0}^{\mathrm{T}} P(s_{t+1} \mid s_t) \propto R(\tau). \tag{5}$$

The equality in Eq. 5 is due to the Markov property of the state transition.

In contrast to the objectives of prior KG-based RAG methods in Eq. 1 and Eq. 2 that maximize the likelihood of the most relevant information in KG, the objective of GraphFlow in Eq. 5 reflects the intuition that high-reward retrieval trajectories (i.e., trajectories ending in high-quality supporting documents) should be sampled more frequently. Thus, GraphFlow naturally promotes diverse yet

Table 1: Performance of retrieval accuracy of KG-based RAG methods on STaRK benchmark. Our GraphFlow outperforms baselines with higher hit rates and MRR scores. GraphFlow also surpasses strong baselines implemented with GPT-4o on most metrics.

Method	Dataset	STaRK-AMAZON			STaRK-MAG			STaRK-PRIME		
	Metric	Hit@1 ↑	Hit@5↑	MRR↑	Hit@1↑	Hit@5↑	MRR↑	Hit@1↑	Hit@5↑	MRR↑
Retrieval -based	DenseRetriver	6.10	15.85	10.61	24.44	40.23	32.41	5.43	13.07	8.99
	G-Retriever	6.10	11.59	8.54	24.44	31.95	28.08	5.43	8.94	6.95
	SubgraphRAG	8.03	12.43	9.90	9.30	25.59	16.11	4.82	8.00	6.17
Agent-based (w/o Rerank)	ToG+LLaMA3	4.21	6.16	5.25	12.0	14.09	12.67	21.92	34.0	26.61
	ToG+GPT4o	**20.67**	41.38	30.90	23.33	56.67	36.38	16.67	39.77	27.02
	SFT	8.16	15.30	13.54	26.53	28.57	29.10	27.5	40.07	33.06
	PRM	20.09	26.25	28.16	26.05	28.0	28.52	21.01	46.72	31.25
	GraphFlow	19.63	**44.17**	**31.66**	**29.32**	**58.64**	**41.32**	**39.84**	**71.71**	**54.58**
Agent-based (Rerank)	ToG+LLaMA3	4.21	6.16	5.25	12.0	14.09	12.67	21.92	34.0	26.61
	ToG+GPT4o	27.58	51.72	39.08	26.67	56.67	39.65	**53.33**	63.73	57.78
	SFT	12.24	30.61	21.54	27.55	44.89	36.37	23.75	52.5	35.98
	PRM	21.25	42.50	31.97	27.31	44.09	33.69	22.86	28.24	26.94
	GraphFlow	**47.85**	**65.03**	**55.49**	**39.09**	**57.51**	**47.82**	51.39	**72.11**	**61.37**

accurate retrieval results since the retrieval trajectories resulting in highly relevant documents are more likely to be explored. Moreover, GraphFlow also enjoys good generalization ability by avoiding strict likelihood maximization and does not overfit to a few dominant candidates.

3.2 Flow Estimation as Credit Assignment

A major challenge in learning the policy $P(s_{t+1} \mid s_t)$ to satisfy Eq. 5 is the lack of process-level supervision. When collecting retrieval trajectories $\tau \in \mathcal{T}$ for training, only the outcome reward $R(\tau)$ is observable, indicating whether the final retrieved document supports answering the query. Annotating the process-level reward signals for every intermediate state and action is expensive. This gives rise to the credit assignment problem [56, 91], which attributes the terminal reward of a trajectory back to the intermediate decisions. To address this, we adopt the GFlowNet framework [3], which implicitly performs credit assignment by estimating a non-negative flow value for each state.

Rather than directly maximizing the reward or value of a full trajectory, GFlowNets introduce a flow function $F(s) : \mathcal{S} \to \mathbb{R}_{\geq 0}$ for each intermediate state s. The learning objective enforces a local consistency constraint between transitions, which is known as the detailed balance condition:

$$F(s_t) \cdot P(s_{t+1} \mid s_t) = F(s_{t+1}) \cdot P_{\mathrm{B}}(s_t \mid s_{t+1}), \tag{6}$$

where $P(s_{t+1} \mid s_t)$ is the forward policy we want to obtain, and $P_{\mathrm{B}}(s_t \mid s_{t+1})$ is the backward policy. When this condition holds for all transitions, the retrieval trajectory induced by the policy $P(s_{t+1} \mid s_t)$ satisfies the objective in Eq. 5, leading to diverse and accurate retrieval results on KGs.

While alternative GFlowNet objectives, such as trajectory balance [52] or subtrajectory balance [51], can also promote diversity, they require computation over entire trajectories or sub-trajectories. In KG-based RAG, the retrieval trajectory involves multi-hop transitions, and each node is associated with long documents. These objectives are computation-intensive and often lead to out-of-memory (OOM) issues. To ensure scalability and efficiency, we thus adopt the detailed balance objective that operates on state transitions.

Detailed Balance with Local Exploration. Enforcing the detailed balance condition globally across all transitions in a knowledge graph (KG) is computationally inefficient, since the vast state space makes many nodes and transitions unreachable during training. To address this, we introduce a local exploration strategy that focuses the detailed balance objective on the neighborhoods of states observed in sampled trajectories.

For a retrieval trajectory $\tau = V_0 \to \cdots \to V_T$ with reward $R(\tau)$, we apply local exploration to each non-terminal state $s_t = (\mathcal{Q}, \{D_j\}_{j=0}^t)$ where $t \neq T$. Specifically, the agent takes an exploratory action $a'_t \in \mathcal{A}(s_t)$ that moves from node V_t to a neighboring node $V'_{t+1} \in \mathcal{N}(V_t)$ different from the original next node V_{t+1}. This results in a new exploratory state $s'_{t+1} = (\mathcal{Q}, \{D_j\}_{j=0}^t \cup \{D'_{t+1}\})$, corresponding to the partial trajectory $\tau'_{\leq t+1} = V_0 \to \cdots \to V_t \to V'_{t+1}$.

https://doi.org/10.52202/085713-3200

Table 2: Performance of retrieval diversity of KG-based RAG methods. GraphFlow retrieves more correct documents to support queries with high diversity.

Method	Dataset	STaRK-AMAZON		STaRK-MAG		STaRK-PRIME	
	Metric	R@20↑	D-R@20↑	R@20↑	D-R@20↑	R@20↑	D-R@20↑
Retrieval -based	DenseRetriver	13.63	13.63	41.80	41.80	13.92	13.92
	G-Retriever	5.35	5.35	25.37	25.37	6.75	6.75
	SubgraphRAG	6.53	6.53	27.83	26.95	6.49	6.49
Agent -based	ToG+LLaMA3	2.61	2.61	6.77	6.77	33.84	33.84
	ToG+GPT4o	25.81	23.70	48.03	47.71	54.35	54.35
	SFT	25.22	24.97	37.48	35.90	47.72	45.36
	PRM	35.72	18.94	36.73	36.73	45.97	45.97
	GraphFlow	**36.15**	**36.15**	**57.18**	**57.18**	**79.71**	**79.59**

By performing k such explorations, we generate k exploratory actions $\{a'_{t,1}, \cdots, a'_{t,k}\}$ and their resulting states $\{s'_{t+1,1}, \cdots, s'_{t+1,k}\}$. With the ground-truth next state denoted as $s'_{t+1,0} = s_{t+1}$, we obtain $k+1$ transitions from s_t to candidate next states for optimizing the detailed balance objective. The forward policy is then defined as $P(s_{t+1} = s'_{t+1,i}|s_t) = \frac{e^{r_\theta(s_t, a'_{t,i})}}{\sum_{i=0}^{k} e^{r_\theta(s_t, a'_{t,i})}}$. Here $r_\theta(s, a)$ is a learned process reward function parameterized by a neural network with parameters θ. Since the retrieval process is inherently irreversible (i.e., backtracking is disallowed), we follow prior work [22] and set the backward policy $P_{\mathrm{B}}(s_t \mid s_{t+1}) = 1$ in Eq. 5. We yield the following objective for state s_t by taking the $\log$ function to both sides of Eq. 5:

$$\begin{aligned}\mathcal{L}_{\text{DBLE}}(s_t) &= \sum_{i=0}^{k} [\log F(s_t) - \log F(s'_{t+1,i}) + \log P(s_{t+1} = s'_{t+1,i}|s_t)]^2 \\ &= \sum_{i=0}^{k} [\log F(s_t) - \log F(s'_{t+1,i}) + r_\theta(s_t, a'_{t,i}) - \log \sum_{i=0}^{k} e^{r_\theta(s_t, a'_{t,i})}]^2.\end{aligned} \tag{7}$$

Boundary Condition. We impose boundary conditions on the initial and terminal states to ensure proper propagation of flow values along the retrieval trajectory: $\log F(s_0) = \log F(s_T) = 1$. Here s_0 is the initial state and s_T is the terminal state. The reason is that we only collect retrieval trajectories that reach target documents during model training. With such boundary conditions, we ensure that the total incoming and outgoing flow is consistent across the trajectory and enable the flow estimator to correctly distribute the outcome reward of the terminal state to the intermediate states.

Termination Condition. To allow the retrieval policy $P(s_{t+1} \mid s_t)$ to decide when to stop, we introduce a special self-loop action that retrieves the current node again. At each step, this action is included among the candidate actions in Eq. 7. Hence, Eq. 7 can also be applied for the terminal state. If the policy chooses to retrieve the current document (i.e., selects the self-loop), the trajectory is terminated, indicating that the current document is relevant to the query. Otherwise, the policy continues to explore the KG. This mechanism enables the agent to adaptively determine when to stop retrieval based on its experience, rather than relying on a fixed number of steps.

Difference Between GraphFlow, SFT, and PRM. SFT and PRM learn the retrieval policy by treating the action leading to the ground-truth next state s_{t+1} as a positive sample, and exploratory actions leads to $\{s'_{t+1,1}, \cdots, s'_{t+1,k}\}$ as negative ones, akin to behavior cloning [70]. GraphFlow generalizes this by learning state-dependent flow values $F(s)$ and factorizes the outcome reward via Eq. 7. When setting $\log F(s_t) = 1$ and $\log F(s'_{t+1,i}) = 0$, GraphFlow reduces to behavior cloning. However, such as a hard objective limits generalization and cannot learn a policy leading to diverse and accurate retrieval results for complex queries.

3.3 Instantiating GraphFlow with LLMs

We implement GraphFlow with an LLM due to its ability of text understanding and decision-making, as shown in Figure 2 (b). The state and state-action pair are decorated with a flow prompt and policy prompt template, which are encoded using a shared LLM. The embeddings of the final tokens are used as representations of the state and the state–action pair, respectively. On top of the shared

Table 3: Quantitative retrieval quality of different KG-based retrieval methods.

Method	STaRK-Amazon		STaRK-MAG		STaRK-PRIME	
	Step-Δ_{Seper} ↑	Answer-Δ_{Seper} ↑	Step-Δ_{Seper} ↑	Answer-Δ_{Seper} ↑	Step-Δ_{Seper} ↑	Answer-Δ_{Seper} ↑
ToG+GPT-4o	0.031 ± 0.109	0.092 ± 0.128	0.041 ± 0.172	0.065 ± 0.150	0.065 ± 0.125	0.148 ± 0.165
ToG+LLaMA3	0.010 ± 0.118	0.046 ± 0.149	0.068 ± 0.108	0.105 ± 0.146	0.009 ± 0.102	0.021 ± 0.106
SFT	0.079 ± 0.151	0.141 ± 0.160	0.035 ± 0.101	0.084 ± 0.095	0.062 ± 0.132	0.158 ± 0.183
PRM	0.029 ± 0.089	0.071 ± 0.112	0.037 ± 0.117	0.060 ± 0.115	0.057 ± 0.106	0.131 ± 0.174
G-Retriever	—	0.024 ± 0.110	—	0.012 ± 0.089	—	0.029 ± 0.117
SubgraphRAG	—	0.021 ± 0.093	—	0.039 ± 0.076	—	0.046 ± 0.083
GraphFlow	$\mathbf{0.097 \pm 0.158}$	$\mathbf{0.219 \pm 0.257}$	$\mathbf{0.081 \pm 0.137}$	$\mathbf{0.145 \pm 0.112}$	$\mathbf{0.091 \pm 0.147}$	$\mathbf{0.206 \pm 0.192}$

encoder, we employ two separate multi-layer perceptrons (MLPs) as the policy head and the flow head, respectively. The policy head predicts the forward transition probability, while the flow head estimates logarithm of the flow value of the state. During model training, we apply LoRA [23] to inject learnable adapters into the frozen backbone of the LLM, and update the parameters of the flow head and the policy head. This design enables joint optimization of policy learning and flow estimation in a parameter-efficient manner, while also capturing rich contextual information through the LLM encoder. We present detailed implementation in Supplementary Material due to space limit.

4 Experiment

4.1 Dataset

We employ the STaRK [74] benchmark to validate the retrieval quality of the proposed GraphFlow to support complex queries. STaRK is a recently proposed benchmark designed to evaluate the retrieval performance of KG-based RAG methods on text-rich KGs spanning three domains:

- **STaRK-AMAZON** is an e-commerce KG where the nodes contain detailed product information and the edges denotes the properties of products and co-purchase between products. The retrieval task is to retrieve the diverse products to satisfy the recommendation query.
- **STaRK-MAG** is an academic graph constructed based on OGB [24] and Microsoft Academic Graph [66]. The nodes contain author information, institute, and publications. The retrieval task is to address academic queries such as paper searching.
- **STaRK-PRIME** is a biomedical KG where the nodes are associated with the detailed description of drugs, disease, genes, and pathways, and the edges are their relationship. The retrieval task is to address the biomedical query.

The StaRK benchmark challenges KG-based RAG methods by complex queries corresponding to diverse retrieval targets and fusion of text and structure information that complicates accurate retrieval.

4.2 Baseline and Evaluation Metrics

Baseline. We choose representative retrieval-based and agent-based baselines with explicit retrieval results (i.e., the retrieved node index) on the STaRK benchmark [74]. All detailed implementations are shown in Supplementary Material due to space limit.

For retrieval-based baselines, we consider **Dense-Retriever** [30], **G-Retriever** [21], and **SubgraphRAG** [37]. Dense-Retriever is implemented with SentenceBERT [62] to encode both questions and the documents of KG nodes into dense embeddings and retrieve the documents with top vector similarity. G-Retriever employs the Prize-Collecting Steiner Tree (PCST) [2] algorithm to extract a subgraph from KGs relevant to the query. Since computing PCST on STaRK benchmark is infeasible, we follow the hybrid setting [33, 34] that first identifies a source node in KG via Dense-Retriever and only computes PCST around the ego-graph up to 2 hops around the identified node. We also adopt the same hybrid setting for other baselines to ensure computational feasibility on STaRK. SubgraphRAG integrates a learnable subgraph retrieval module to retrieve from KG.

For agent-based methods, we consider **ToG** [68], **SFT**, and **PRM** [40] as baselines. ToG employs an LLM agent to search from the KG to retrieve supporting documents. We instantiate ToG using both LLaMA3-8B-Instruct and GPT-4o as backbone models, denoted as ToG+LLaMA3 and ToG+GPT4o,

 https://doi.org/10.52202/085713-3200

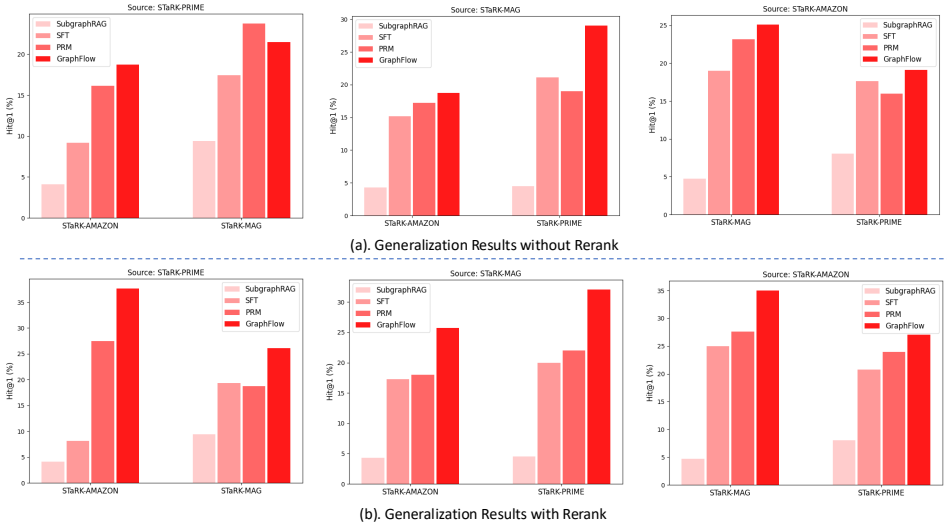

Figure 3: Generalization Performance of KG-based RAG methods. GraphFlow shows superior cross-domain generalization performance, especially under the rerank setting (best viewed in color).

respectively. SFT and PRM are two popular approaches that fine-tune the LLM agent to enhance RAG in recent works [76, 12, 25]. SFT, PRM, and GraphFlow are deployed with LLaMA3-8B-Instruct [17]. For agent-based methods, we use the agent to rerank all the retrieved results.

Evaluation Metrics. All KG-based RAG methods retrieve every input query 20 times and generate 20 retrieval results for diversity and accuracy evaluation following the standard setting in STaRK [74]. We employ the following metrics to evaluate the retrieval performance. Hit@k denotes whether the ground truth is retrieved in the top-k results. We employ **Hit@1 and Hit@5** to measure the retrieval precision of the different KG-based RAG methods. **Mean Reciprocal Rank (MRR)** measures the average of reciprocal ranks of the first ground-truth item in the retrieval results and encourages the ground-truth item to be retrieved in a higher rank. **Recall@k (R@k)** is a standard metric to measure the percentage of ground-truth items that appear in the top-k retrieved results. We employ Recall@20 (R@20) for evaluation. **De-duplicate Recall@k (D-R@k)** measures the percentage of *unique* ground-truth items that appear in the top-k retrieved results. This metric is used to evaluate the diversity of the correctly retrieved results. We use De-duplicate Recall@20 (D-R@20).

4.3 Main Results

Accuracy. Table 1 presents the retrieval accuracy of various KG-based RAG methods on STaRK. GraphFlow consistently outperforms other KG-based RAG approaches on most metrics. In particular, it achieves higher Hit rates and MRR scores than the strong baseline ToG+GPT-4o with an average 10% improvement in retrieval accuracy. Interestingly, ToG's performance is highly sensitive to the choice of backend model. When instantiated with LLaMA3-8B, ToG shows a significant drop in performance compared to using GPT-4o. Additionally, rerank has no effect in the ToG+LLaMA3-8B setup, as all retrieved results receive equally high scores, leaving the ranking unchanged.

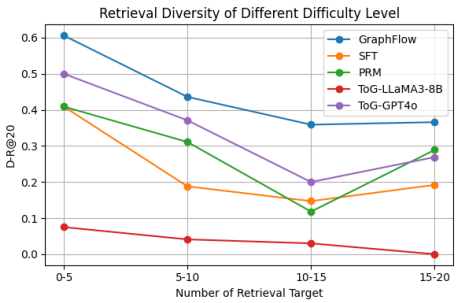

Figure 4: GraphFlow shows improved retrieval diversity on different difficulty levels of retrieval queries on STaRK-PRIME.

Two finetuned agent-based baselines, SFT and PRM, outperform ToG without finetuning, but still fall short of GraphFlow. While PRM training can leverage curated preference datasets with fine-grained process-level rewards, such labeling is prohibitively expensive. Instead, GraphFlow achieves high-quality retrieval with only outcome rewards of retrieval trajectories. For retriever-based approaches, DenseRetriever, G-Retriever, and SubgraphRAG show moderate performance but are generally inferior to agent-based

methods. Overall, retriever-based methods remain more lightweight but trail behind agent-based approaches in retrieval accuracy.

Diversity. Table 2 reports the retrieval diversity of different KG-based RAG methods on the STaRK benchmark. We evaluate both Recall@20 (R@20) and its de-duplicated variant (D-R@20), which better captures retrieval diversity. GraphFlow achieves the highest retrieval diversity across all datasets, outperforming both retriever-based and agent-based baselines. Its results not only match more ground-truth contents but also avoid redundancy. Notably, GraphFlow exceeds the strongest baseline (ToG+GPT-4o) by a large margin on the STaRK-PRIME dataset, highlighting its ability to retrieve results that are both relevant and diverse. Compared with PRM and SFT, GraphFlow also demonstrates superior diversity. In contrast, retriever-based methods retrieve less diverse content and cover fewer retrieval targets.

4.4 Quantifying Retrieval Quality

We further employ the Seper score (Δ_{Seper}) [9] to quantify the retrieval quality of different KG-based retrieval methods. The Seper score is a recently proposed metric for evaluating retrieval utility by measuring semantic perplexity reduction after retrieval: $\Delta_{Seper} = p_M(a|q,d) - p_M(a|q)$. Here, q is the question, d is the document associated with the retrieval item, and M is an LLM used for question answering. In our case, we use LLaMA3–8B–Instruct to instantiate M to keep consistent with the retrieval model. Since there is no ground-truth answer for the questions in the STaRK benchmark, we use the title or summarized description of the ground-truth retrieval item as a. We design the following metrics for comprehensive evaluation and report their mean and standard deviation (std).

- Step-Δ_{Seper}: the Seper score that quantity the retrieval quality of intermediate retrieval.
- Answer-Δ_{Seper}: the Seper score that quantity the retrieval quality of the final result.

As shown in Table 3, GraphFlow consistently achieves higher Step-Δ_{Seper} and Answer-Δ_{Seper} than all the baselines, demonstrating stronger information utility during retrieval. These results further confirm that GraphFlow can significantly improve the information utility when retrieving from text-rich KGs. Notice that all the methods have high variance in Step-Δ_{Seper} and Answer-Δ_{Seper}. The reason is the high variance in natural language entailment when calculating Seper scores. Moreover, we observe that some retrieval samples have negative Seper scores for all methods, indicating a negative impact on question answering when retrieving bad contents.

4.5 Further Discussion

Cross-domain Generalization. Figure 3 reports the cross-domain generalization ability of different KG-based RAG methods. We use Hit@1 to evaluate the retrieval accuracy Compared with SubgraphRAG using a small model for retrieval, SFT, PRM, and GraphFlow that finetune the LLM show better cross-domain generalization ability due to the *over-parameterization* [35, 15]. GraphFlow demonstrates superior cross-domain generalization, since it avoids from likelihood maximization objectives used by SFT and PRM. Instead, GraphFlow adaptively assigns the outcome reward of the retrieval trajectory to the flow values of intermediate states and guides the retrieval policy, leading to better generalization ability. More results are shown in Supplementary Material due to space limit.

Performance on Hard Cases. We categorize the retrieval queries with different numbers of retrieval targets into 4 difficulty levels. Figure 4 shows the D-R@20 scores of KG-based RAG methods on retrieval queries on STaRK-PRIME at different levels. GraphFlow outperforms the other agent-based approaches by a large margin by covering more diverse and accurate retrieval targets, especially on the hard cases containing more than 15 retrieval targets. The performance on hard cases shows the superior performance of GraphFlow in retrieving more relevant and diverse results. More results of hard cases are shown in Supplementary Materials due to space limit.

5 Related Work

KG-based RAG. Knowledge graphs (KGs) are widely used as knowledge sources in retrieval-augmented generation (RAG) [20**?**] systems to enhance large language models (LLMs) with both relational and textual information for answering complex queries [14, 13, 8, 75]. A core challenge in

 https://doi.org/10.52202/085713-3200

KG-based RAG lies in retrieving relevant knowledge from KGs in response to a given query. Recent methods addressing this problem can be broadly categorized into two approaches. Retrieval-based methods [80, 53, 21, 61, 93] leverage pretrained language models to encode the textual content in KGs into embeddings and use small models, such as MLP and GNN, to retrieve relevant information. In contrast, agent-based methods [68, 88, 50, 47, 45] employ LLMs as agents that iteratively traverse the KG to locate supporting evidence. While both paradigms have shown promise in knowledge graph question answering (KGQA) [37], their effectiveness in retrieving diverse and high-quality candidates for complex queries remains limited. Furthermore, complex queries often require retrieval from text-rich KGs, necessitating the joint consideration of both relational structure and text content. Although recent studies [50, 34] have begun to explore this setting and tackle complex queries, enhancing the diversity and accuracy of KG-based RAG is still underexplored.

Process Reward Models. Process Reward Models (PRMs) [40, 32] have shown great promise in guiding LLMs with process supervision and have been adopted in many domains such as complex reasoning [7], alignment [58], and planning [6]. The key to PRMs is to construct a preference dataset with process supervision [60]. Previous works obtain the process supervision from human feedback and LLM evaluation. However, fine-grained process-level supervision is expensive for KG-based RAG due to the potentially vast search space of KGs and the difficulty of accessing the intermediate state during retrieval. Although early explorations are made to use PRM to guide the retrieval process of RAG on unstructured knowledge bases [39, 65, 25, 76], they still need a preference dataset with process-level supervision. How to guide the retrieval process of KG-based RAG on structured KGs without process supervision data is still challenging.

GFlowNet. GFlowNet [3] aims to sample diverse and high-quality candidates from an unnormalized density and has received increasing attention in sampling from discrete and vast spaces [16, 46, 11, 10, 49]. The goal of GFlowNet is to learn a policy that can lead to the terminal states with the likelihood in proportion to their rewards [86]. Some objects are proposed to optimize GFlowNet by regularizing the state flows and their transitions [69, 52, 51]. Recently, GFlowNet has also been introduced to improve the generative performance of LLMs and diffusion models by promoting diversity in the decoding process [71, 22, 81, 29, 43] . Differently, our work focuses on aligning the retrieval results of KG-based RAG with the knowledge required for real-world queries by estimating the state flow in multi-step retrieval. Moreover, we introduce a local exploration strategy to avoid visiting less-valued states, thus efficiently optimizing the detailed balance.

6 Conclusion

We introduce GraphFlow, a novel framework that enhances existing KG-based RAG methods by enabling accurate and diverse retrieval from text-rich KGs. By jointly optimizing a retrieval policy and a flow estimator via a detailed balance objective, GraphFlow effectively aligns the retrieval process with query-specific knowledge demands without explicit process-level reward. Extensive evaluation on the STaRK benchmark demonstrates that GraphFlow not only surpasses strong baselines deployed with GPT-4o, but also generalizes well to unseen KGs. These findings underscore the effectiveness and robustness of GraphFlow in supporting complex queries using textual and structured knowledge. Our future work will incorporate causality into KG-based RAG to improve the reasoning ability of LLMs [44, 5, 83], reduce forgetting [41, 42], and explore their scientific applications [78].

Acknowledgments and Disclosure of Funding

This work is supported by the UKRI grant: Turing AI Fellowship EP/W002981/1. This work is jointly supported by the Shanghai Municipal Science and Technology Major Project and Shanghai Artificial Intelligence Laboratory. The authors would like to thank PCs, ACs, and all the reviewers for their insightful comments and suggestions.

References

[1] Micheal Abaho and Yousef H Alfaifi. Select and augment: enhanced dense retrieval knowledge graph augmentation (abstract reprint). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 22690–22690, 2024.

[2] Aaron Archer, MohammadHossein Bateni, MohammadTaghi Hajiaghayi, and Howard Karloff. Improved approximation algorithms for prize-collecting steiner tree and tsp. *SIAM journal on computing*, 40(2):309–332, 2011.

[3] Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J Hu, Mo Tiwari, and Emmanuel Bengio. Gflownet foundations. *Journal of Machine Learning Research*, 24(210):1–55, 2023.

[4] Weijie Chen, Ting Bai, Jinbo Su, Jian Luan, Wei Liu, and Chuan Shi. Kg-retriever: Efficient knowledge indexing for retrieval-augmented large language models. *arXiv preprint arXiv:2412.05547*, 2024.

[5] Yongqiang Chen, Yonggang Zhang, Yatao Bian, Han Yang, MA Kaili, Binghui Xie, Tongliang Liu, Bo Han, and James Cheng. Learning causally invariant representations for out-of-distribution generalization on graphs. *Advances in Neural Information Processing Systems*, 35:22131–22148, 2022.

[6] Sanjiban Choudhury. Process reward models for llm agents: Practical framework and directions. *arXiv preprint arXiv:2502.10325*, 2025.

[7] Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

[8] Yuanning Cui, Zequn Sun, and Wei Hu. A prompt-based knowledge graph foundation model for universal in-context reasoning. *Advances in Neural Information Processing Systems*, 37:7095–7124, 2024.

[9] Lu Dai, Yijie Xu, Jinhui Ye, Hao Liu, and Hui Xiong. Seper: Measure retrieval utility through the lens of semantic perplexity reduction. In *The Thirteenth International Conference on Learning Representations*, 2025.

[10] Tristan Deleu, António Góis, Chris Emezue, Mansi Rankawat, Simon Lacoste-Julien, Stefan Bauer, and Yoshua Bengio. Bayesian structure learning with generative flow networks. In *Uncertainty in Artificial Intelligence*, pages 518–528. PMLR, 2022.

[11] Tristan Deleu, Mizu Nishikawa-Toomey, Jithendaraa Subramanian, Nikolay Malkin, Laurent Charlin, and Yoshua Bengio. Joint bayesian inference of graphical structure and parameters with a single generative flow network. *Advances in Neural Information Processing Systems*, 36:31204–31231, 2023.

[12] Guanting Dong, Yutao Zhu, Chenghao Zhang, Zechen Wang, Ji-Rong Wen, and Zhicheng Dou. Understand what llm needs: Dual preference alignment for retrieval-augmented generation. In *Proceedings of the ACM on Web Conference 2025*, pages 4206–4225, 2025.

[13] Yuxin Dong, Shuo Wang, Hongye Zheng, Jiajing Chen, Zhenhong Zhang, and Chihang Wang. Advanced rag models with graph structures: Optimizing complex knowledge reasoning and text generation. In *2024 5th International Symposium on Computer Engineering and Intelligent Communications (ISCEIC)*, pages 626–630. IEEE, 2024.

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

[15] Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pages 10835–10866. PMLR, 2023.

[16] Pouya M Ghari, Alex Tseng, Gökcen Eraslan, Romain Lopez, Tommaso Biancalani, Gabriele Scalia, and Ehsan Hajiramezanali. Generative flow networks assisted biological sequence editing. In *NeurIPS 2023 Generative AI and Biology (GenBio) Workshop*, 2023.

[17] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

 https://doi.org/10.52202/085713-3200

[18] Tiezheng Guo, Qingwen Yang, Chen Wang, Yanyi Liu, Pan Li, Jiawei Tang, Dapeng Li, and Yingyou Wen. Knowledgenavigator: Leveraging large language models for enhanced reasoning over knowledge graph. *Complex & Intelligent Systems*, 10(5):7063–7076, 2024.

[19] Haoyu Han, Harry Shomer, Yu Wang, Yongjia Lei, Kai Guo, Zhigang Hua, Bo Long, Hui Liu, and Jiliang Tang. Rag vs. graphrag: A systematic evaluation and key insights. *arXiv preprint arXiv:2502.11371*, 2025.

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

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

[22] Edward J Hu, Moksh Jain, Eric Elmoznino, Younesse Kaddar, Guillaume Lajoie, Yoshua Bengio, and Nikolay Malkin. Amortizing intractable inference in large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

[23] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

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

[25] Jerry Huang, Siddarth Madala, Risham Sidhu, Cheng Niu, Julia Hockenmaier, and Tong Zhang. Rag-rl: Advancing retrieval-augmented generation via rl and curriculum learning. *arXiv preprint arXiv:2503.12759*, 2025.

[26] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55, 2025.

[27] Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

[28] Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. Challenges and applications of large language models. *arXiv preprint arXiv:2307.10169*, 2023.

[29] Haoqiang Kang, Enna Sachdeva, Piyush Gupta, Sangjae Bae, and Kwonjoon Lee. Gflowvlm: Enhancing multi-step reasoning in vision-language models with generative flow networks. *arXiv preprint arXiv:2503.06514*, 2025.

[30] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781, 2020.

[31] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[32] Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, et al. Rewardbench: Evaluating reward models for language modeling. *arXiv preprint arXiv:2403.13787*, 2024.

[33] Meng-Chieh Lee, Qi Zhu, Costas Mavromatis, Zhen Han, Soji Adeshina, Vassilis N Ioannidis, Huzefa Rangwala, and Christos Faloutsos. Hybgrag: Hybrid retrieval-augmented generation on textual and relational knowledge bases. *arXiv preprint arXiv:2412.16311*, 2024.

[34] Yongjia Lei, Haoyu Han, Ryan A Rossi, Franck Dernoncourt, Nedim Lipka, Mahantesh M Halappanavar, Jiliang Tang, and Yu Wang. Mixture of structural-and-textual retrieval over text-rich graph knowledge bases. *arXiv preprint arXiv:2502.20317*, 2025.

[35] Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871*, 2018.

[36] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[37] Mufei Li, Siqi Miao, and Pan Li. Retrieval or reasoning: The roles of graphs and large language models in efficient knowledge-graph-based retrieval-augmented generation. In *The Thirteenth International Conference on Learning Representations*, 2025.

[38] Vincent Li, Yule Fu, Tim Knappe, Kevin Han, and Kevin Zhu. Automating mathematical proof generation using large language model agents and knowledge graphs. *arXiv preprint arXiv:2503.11657*, 2025.

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

[40] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

[41] Runqi Lin, Chaojian Yu, Bo Han, and Tongliang Liu. On the over-memorization during natural, robust and catastrophic overfitting. In *The Twelfth International Conference on Learning Representations*, 2024.

[42] Runqi Lin, Chaojian Yu, and Tongliang Liu. Eliminating catastrophic overfitting via abnormal adversarial examples regularization. *Advances in Neural Information Processing Systems*, 36:67866–67885, 2023.

[43] Vijay Lingam, Behrooz Omidvar Tehrani, Sujay Sanghavi, Gaurav Gupta, Sayan Ghosh, Linbo Liu, Jun Huan, and Anoop Deoras. Enhancing language model agents using diversity of thoughts. In *The Thirteenth International Conference on Learning Representations*, 2025.

[44] Chenxi Liu, Yongqiang Chen, Tongliang Liu, Mingming Gong, James Cheng, Bo Han, and Kun Zhang. Discovery of the hidden world with large language models. *Advances in Neural Information Processing Systems*, 37:102307–102365, 2024.

[45] Hao Liu, Zhengren Wang, Xi Chen, Zhiyu Li, Feiyu Xiong, Qinhan Yu, and Wentao Zhang. Hoprag: Multi-hop reasoning for logic-aware retrieval-augmented generation. *arXiv preprint arXiv:2502.12442*, 2025.

[46] Stephen Zhewen Lu, Ziqing Lu, Ehsan Hajiramezanali, Tommaso Biancalani, Yoshua Bengio, Gabriele Scalia, and Michał Koziarski. Cell morphology-guided small molecule generation with gflownets. In *ICML 2024 Workshop on Structured Probabilistic Inference $\{\backslash\&\}$ Generative Modeling*, 2024.

[47] LINHAO LUO, Yuan-Fang Li, Reza Haf, and Shirui Pan. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *The Twelfth International Conference on Learning Representations*, 2024.

[48] Linhao Luo, Zicheng Zhao, Gholamreza Haffari, Dinh Phung, Chen Gong, and Shirui Pan. Gfm-rag: Graph foundation model for retrieval augmented generation. *arXiv preprint arXiv:2502.01113*, 2025.

 https://doi.org/10.52202/085713-3200

[49] Pouya M Ghari, Alex Tseng, Gokcen Eraslan, Romain Lopez, Tommaso Biancalani, Gabriele Scalia, and Ehsan Hajiramezanali. Gflownet assisted biological sequence editing. *Advances in Neural Information Processing Systems*, 37:106841–106869, 2024.

[50] Shengjie Ma, Chengjin Xu, Xuhui Jiang, Muzhi Li, Huaren Qu, Cehao Yang, Jiaxin Mao, and Jian Guo. Think-on-graph 2.0: Deep and faithful large language model reasoning with knowledge-guided retrieval augmented generation. *arXiv preprint arXiv:2407.10805*, 2025.

[51] Kanika Madan, Jarrid Rector-Brooks, Maksym Korablyov, Emmanuel Bengio, Moksh Jain, Andrei Cristian Nica, Tom Bosc, Yoshua Bengio, and Nikolay Malkin. Learning gflownets from partial episodes for improved convergence and stability. In *International Conference on Machine Learning*, pages 23467–23483. PMLR, 2023.

[52] Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in gflownets. *Advances in Neural Information Processing Systems*, 35:5955–5967, 2022.

[53] Costas Mavromatis and George Karypis. Gnn-rag: Graph neural retrieval for large language model reasoning. *arXiv preprint arXiv:2405.20139*, 2024.

[54] Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a reference-free reward. *Advances in Neural Information Processing Systems*, 37:124198–124235, 2024.

[55] Tong Mu, Alec Helyar, Johannes Heidecke, Joshua Achiam, Andrea Vallone, Ian D Kivlichan, Molly Lin, Alex Beutel, John Schulman, and Lilian Weng. Rule based rewards for language model safety. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[56] Duc Thien Nguyen, Akshat Kumar, and Hoong Chuin Lau. Credit assignment for collective multiagent rl with global rewards. *Advances in neural information processing systems*, 31, 2018.

[57] Thang Nguyen, Peter Chin, and Yu-Wing Tai. Reward-rag: Enhancing rag with reward driven supervision. *arXiv preprint arXiv:2410.03780*, 2024.

[58] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[59] Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*, 36(7):3580–3599, 2024.

[60] Miao Peng, Nuo Chen, Zongrui Suo, and Jia Li. Rewarding graph reasoning process makes llms more generalized reasoners. *arXiv preprint arXiv:2503.00845*, 2025.

[61] Zhangchi Qiu, Linhao Luo, Zicheng Zhao, Shirui Pan, and Alan Wee-Chung Liew. Graph retrieval-augmented llm for conversational recommendation systems. *arXiv preprint arXiv:2503.06430*, 2025.

[62] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, 2019.

[63] Diego Sanmartin. Kg-rag: Bridging the gap between knowledge and creativity. *arXiv preprint arXiv:2405.12035*, 2024.

[64] Max W Shen, Emmanuel Bengio, Ehsan Hajiramezanali, Andreas Loukas, Kyunghyun Cho, and Tommaso Biancalani. Towards understanding and improving gflownet training. In *International conference on machine learning*, pages 30956–30975. PMLR, 2023.

[65] Yucheng Shi, Tianze Yang, Canyu Chen, Quanzheng Li, Tianming Liu, Xiang Li, and Ninghao Liu. Searchrag: Can search engines be helpful for llm-based medical question answering? *arXiv preprint arXiv:2502.13233*, 2025.

[66] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June Hsu, and Kuansan Wang. An overview of microsoft academic service (mas) and applications. In *Proceedings of the 24th international conference on world wide web*, pages 243–246, 2015.

[67] Xiaorui Su, Yibo Wang, Shanghua Gao, Xiaolong Liu, Valentina Giunchiglia, Djork-Arné Clevert, and Marinka Zitnik. Knowledge graph based agent for complex, knowledge-intensive qa in medicine. In *The Thirteenth International Conference on Learning Representations*, 2025.

[68] Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *The Twelfth International Conference on Learning Representations*, 2024.

[69] Alexander Tong, Kilian FATRAS, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, 2022.

[70] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4950–4957, 2018.

[71] Siddarth Venkatraman, Moksh Jain, Luca Scimeca, Minsu Kim, Marcin Sendera, Mohsin Hasan, Luke Rowe, Sarthak Mittal, Pablo Lemos, Emmanuel Bengio, et al. Amortizing intractable inference in diffusion models for vision, language, and control. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[72] Haiyuan Wan, Chen Yang, Junchi Yu, Meiqi Tu, Jiaxuan Lu, Di Yu, Jianbao Cao, Ben Gao, Jiaqing Xie, Aoran Wang, et al. Deepresearch arena: The first exam of llms' research abilities via seminar-grounded tasks. *arXiv preprint arXiv:2509.01396*, 2025.

[73] Weiyun Wang, Zhangwei Gao, Lianjie Chen, Zhe Chen, Jinguo Zhu, Xiangyu Zhao, Yangzhou Liu, Yue Cao, Shenglong Ye, Xizhou Zhu, et al. Visualprm: An effective process reward model for multimodal reasoning. *arXiv preprint arXiv:2503.10291*, 2025.

[74] Shirley Wu, Shiyu Zhao, Michihiro Yasunaga, Kexin Huang, Kaidi Cao, Qian Huang, Vassilis Ioannidis, Karthik Subbian, James Y Zou, and Jure Leskovec. Stark: Benchmarking llm retrieval on textual and relational knowledge bases. *Advances in Neural Information Processing Systems*, 37:127129–127153, 2024.

[75] Yu Xia, Junda Wu, Sungchul Kim, Tong Yu, Ryan A Rossi, Haoliang Wang, and Julian McAuley. Knowledge-aware query expansion with large language models for textual and relational retrieval. *arXiv preprint arXiv:2410.13765*, 2024.

[76] Guangzhi Xiong, Qiao Jin, Xiao Wang, Yin Fang, Haolin Liu, Yifan Yang, Fangyuan Chen, Zhixing Song, Dengyu Wang, Minjia Zhang, et al. Rag-gym: Optimizing reasoning and search agents with process supervision. *arXiv preprint arXiv:2502.13957*, 2025.

[77] Cheng Yang, Jiaxuan Lu, Haiyuan Wan, Junchi Yu, and Feiwei Qin. From what to why: A multi-agent system for evidence-based chemical reaction condition reasoning. *arXiv preprint arXiv:2509.23768*, 2025.

[78] Zonglin Yang, Wanhao Liu, Ben Gao, Tong Xie, Yuqiang Li, Wanli Ouyang, Soujanya Poria, Erik Cambria, and Dongzhan Zhou. Moose-chem: Large language models for rediscovering unseen chemistry scientific hypotheses. *arXiv preprint arXiv:2410.07076*, 2024.

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

 https://doi.org/10.52202/085713-3200

[80] Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 535–546, 2021.

[81] Fangxu Yu, Lai Jiang, Haoqiang Kang, Shibo Hao, and Lianhui Qin. Flow of reasoning: Efficient training of llm policy with divergent thinking. *arXiv preprint arXiv:2406.05673*, 2024.

[82] Junchi Yu, Jie Cao, and Ran He. Improving subgraph recognition with variational graph information bottleneck. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19396–19405, 2022.

[83] Junchi Yu, Jian Liang, and Ran He. Mind the label shift of augmentation-based graph ood generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11620–11630, 2023.

[84] Junchi Yu, Tingyang Xu, Yu Rong, Yatao Bian, Junzhou Huang, and Ran He. Graph information bottleneck for subgraph recognition. In *International Conference on Learning Representations*, 2021.

[85] Thomas Zeng, Shuibai Zhang, Shutong Wu, Christian Classen, Daewon Chae, Ethan Ewer, Minjae Lee, Heeju Kim, Wonjun Kang, Jackson Kunde, et al. Versaprm: Multi-domain process reward model via synthetic reasoning data. *arXiv preprint arXiv:2502.06737*, 2025.

[86] Dinghuai Zhang, Nikolay Malkin, Zhen Liu, Alexandra Volokhova, Aaron Courville, and Yoshua Bengio. Generative flow networks for discrete probabilistic modeling. In *International Conference on Machine Learning*, pages 26412–26428. PMLR, 2022.

[87] Jing Zhang, Bo Chen, Lingxi Zhang, Xirui Ke, and Haipeng Ding. Neural, symbolic and neural-symbolic reasoning on knowledge graphs. *AI Open*, 2:14–35, 2021.

[88] Qinggang Zhang, Junnan Dong, Hao Chen, Daochen Zha, Zailiang Yu, and Xiao Huang. Knowgpt: Knowledge graph based prompting for large language models. *Advances in Neural Information Processing Systems*, 37:6052–6080, 2024.

[89] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. Siren's song in the ai ocean: a survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*, 2023.

[90] Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*, 2025.

[91] Meng Zhou, Ziyu Liu, Pengwei Sui, Yixuan Li, and Yuk Ying Chung. Learning implicit credit assignment for cooperative multi-agent reinforcement learning. *Advances in neural information processing systems*, 33:11853–11864, 2020.

[92] Jiachen Zhu, Congmin Zheng, Jianghao Lin, Kounianhua Du, Ying Wen, Yong Yu, Jun Wang, and Weinan Zhang. Retrieval-augmented process reward model for generalizable mathematical reasoning. *arXiv preprint arXiv:2502.14361*, 2025.

[93] Deyu Zou, Yongqiang Chen, Mufei Li, Siqi Miao, Chenxi Liu, Bo Han, James Cheng, and Pan Li. Weak-to-strong graphrag: Aligning weak retrievers with large language models for graph-based retrieval augmented generation. *arXiv preprint arXiv:2506.22518*, 2025.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claim in the abstract and introduction accurately reflect the paper's contribution and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

https://doi.org/10.52202/085713-3200

Justification: We discuss the limitation in a separate PDF file due to the space limitation of the submission.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: This paper does not contribute new theory or new proof.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We describe the details of the proposed GraphFlow framework in Method section. We also provide implementation including training details and dataset pre-processing.

   Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We use publicly available benchmark in our experiment. Moreover, we provide details on how we use this benchmark in Supplementary Material.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.

 https://doi.org/10.52202/085713-3200

- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Due to space limit, we provide these details in Supplementary Material.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The error bar is not accessible in the standard metrics of the benchmark used in our paper. However, our main results have shown a significant performance gain over the baselines.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide these details in Supplementary Material.

   Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in this paper conform with the NeruIPS Code of Ethics in every respect.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The broader impacts are discussed in Supplementary Material.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

https://doi.org/10.52202/085713-3200

Answer: [NA]

Justification: This paper does not pose such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We properly cite the open-sourced LLM model and dataset in our paper.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: The paper does not release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: This paper does not involve crowd-sourcing nor research with human objects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowd-sourcing nor research with human objects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: We confirm that the core method development in this research does not involve LLMs.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Introduction of GFlowNet

Generative Flow Networks (GFlowNets) [3] aim to learn a stochastic policy that generates objects $x \in \mathcal{X}$ through sequential decisions, such that the marginal probability of generating x is proportional

 https://doi.org/10.52202/085713-3200

to a reward function $R(x) > 0$. Given a complete trajectory $\tau = (s_0, a_1, s_1, \ldots, a_T, s_T = x)$ that terminates in object x, the forward trajectory probability is:

$$P_F(\tau) = \prod_{t=0}^{T-1} P_F(a_{t+1}|s_t)$$

and the backward probability (used to reverse the trajectory) is:

$$P_B(\tau) = \prod_{t=1}^{T} P_B(a_t|s_t)$$

Trajectory Balance (TB) Loss [52].. The Trajectory Balance objective ensures the ratio of forward to backward probability matches the reward:

$$\frac{P_F(\tau)}{P_B(\tau)} = \frac{R(x)}{Z} \quad \Longleftrightarrow \quad \log P_F(\tau) - \log P_B(\tau) = \log R(x) - \log Z$$

where Z is the global partition function. The loss function is then defined as:

$$\mathcal{L}_{\text{TB}} = \left(\log P_F(\tau) - \log P_B(\tau) - \log R(x) + \log Z\right)^2$$

In practice, $\log Z$ is treated as a learnable scalar parameter.

Subtrajectory Balance (SubTB) Loss [51].. To enable learning from partial trajectories, the Subtrajectory Balance loss generalizes TB to arbitrary subpaths. For any subtrajectory $\tau_{i:j} = (s_i, a_{i+1}, \ldots, s_j)$ from state s_i to s_j, the balance condition becomes:

$$\frac{P_F(\tau_{i:j})}{P_B(\tau_{i:j})} = \frac{Z(s_j)}{Z(s_i)} \quad \Longleftrightarrow \quad \log P_F(\tau_{i:j}) - \log P_B(\tau_{i:j}) = \log Z(s_j) - \log Z(s_i)$$

This leads to the Subtrajectory Balance loss:

$$\mathcal{L}_{\text{SubTB}} = \left(\log P_F(\tau_{i:j}) - \log P_B(\tau_{i:j}) - \log Z(s_j) + \log Z(s_i)\right)^2$$

Here, $Z(s)$ denotes the flow or partition function at state s, typically parameterized by a neural network as $F_\phi(s) = \log Z(s)$. SubTB enables more flexible and sample-efficient training, especially for long-horizon generation tasks. However, directly implementing GFlowNet on KG-based RAG faces several challenges. First, the objectives such as Trajectory balance and sub-trajectory balance are computed on the whole trajectories, leading to computational burden in KG-based RAG where entities are associated with long texts. Second, many states and transitions in KGs are less-valued and not visited, making the traditional GFlowNet objective inefficient. Second, the discrete and symbolic nature of KGs poses difficulty in defining state transitions and flow dynamics, especially when integrating pretrained language models to interpret semantic relevance. These factors collectively make it challenging to directly apply GFlowNet to KG-based retrieval without significant adaptations in trajectory design, reward shaping, and exploration strategy.

B Implementation of GraphFlow with LLMs

Model Architecture. We use LLaMA3-8B-Instruct as the backbone LLM to implement GraphFlow. Specifically, we first employ the following flow prompt template to wrap the retrieval trajectory $\tau_{\leq t}$ at state s_t into a text sequence for flow estimation.

> ###Information trajectory you have visited: {history}
> ###Question: {question}
> Please predict the reward of the Information trajectory to the question:

Here {history} is the concatenation of documents of previously visited entities. {question} is the input complex query. The backbone LLM encodes the above wrapped text sequence. The embedding of the last token is treated as the representation of the wrapped sequence used for flow estimation. We employ a 1-layer MLP as the flow head, which receives the representation of the wrapped sequence and outputs the log value of the estimated flow $\log s_t$.

Then we employ the following policy prompt template to warp the retrieval trajectory $\tau_{\leq t}$ at state s_t into a text sequence for policy learning.

Figure 5: Training dynamics of GraphFlow.

###Information trajectory you have visited: {history}
###Question: {question}
###Candidate Information: {candidate}
Please predict the score of the candidate to help you find the answer to the question:

Here {history} is the concatenation of documents of previously visited entities. {question} is the input complex query. And {candidate} is one action a_t that leads to the next state s_{t+1}. The backbone LLM encodes the above wrapped text sequence. The embedding of the last token is treated as the representation of the wrapped sequence used for learning $P(s_{t+1}|s_t)$. Specifically, we employ a 1-layer MLP with a ReLU function to parameterize $\sigma_\theta(s_t, a_t)$. The forward policy $P(s_{t+1}|s_t)$ is calculated as below:

$$P(s_{t+1}|s_t) = \frac{\sigma_\theta(s_t, a_t)}{\sum_{a_t} \sigma_\theta(s_t, a_t)}. \tag{8}$$

Training Configuration. we apply LoRA [23] to inject learnable adapters into the frozen backbone of the LLM, and update the parameters of the flow head and the policy head. This design enables joint optimization of policy learning and flow estimation in a parameter-efficient manner, while also capturing rich contextual information through the LLM encoder. The parameters of these modules are trained by optimizing the detailed balance with local exploration (DBLE) objective:

$$\begin{aligned}\mathcal{L}_{\text{DBLE}}(s_t) &= \sum_{i=0}^{k} [\log F(s_t) - \log F(s'_{t+1,i}) + \log P(s_{t+1} = s'_{t+1,i}|s_t)]^2 \\ &= \sum_{i=0}^{k} [\log F(s_t) - \log F(s'_{t+1,i}) + r_\theta(s_t, a'_{t,i}) - \log \sum_{i=0}^{k} e^{r_\theta(s_t, a'_{t,i})}]^2.\end{aligned} \tag{9}$$

Experimental Settings. To facilitate training the LoRA module and the flow head and the policy head on the STaRK benchmark, we first collect training dataset consisting of transitions between states. For a given question $\mathcal{Q}$ with the set of ground truth retrieval entities V_T in the training set, we first identify the initial entity V_0 using vector similarity between the embedding of $\mathcal{Q}$ and V_0. Then, we sample the trajectory $\tau_{\leq T} = V_0 \to \cdots \to V_T$ staring from V_0 and ending at V_T. We collect the all the transitions between s_t to s_{t+1} in the example the trajectory $\tau_{\leq T} = V_0 \to \cdots \to V_T$

https://doi.org/10.52202/085713-3200

Table 4: Parameters of GraphFlow training on STaRK benchmark.

	STaRK-AMAZON	STaRK-MAG	STaRK-PRIME
Accumulation steps	2		
alpha	16		
batch_size	1		
num_gpu	8		
depth_cutoff	6		
doc_cutoff	400		
eval_ratio	0.8		
eval_step	100		
lora_dropout	0.05		
lr	1.00E-05		
max_length	1024		
n_epochs	1		
num_exploration	4		
r	32		
window_size	3		

Table 5: We provide data statistics of STaRK. The statistics are from the STaRK benchmark [74].

	entity type	relation type	avg. degree	entities	relations	tokens
STARK-AMAZON	4	5	18.2	1,035,542	9,443,802	592,067,882
STARK-MAG	4	4	43.5	1,872,968	39,802,116	212,602,571
STARK-PRIME	10	18	125.2	129,375	8,100,498	31,844,769

to implement local exploration as introduced in Section 3.2 in the main paper. For every training step, we construct mini-batch of traditions between states to calculate the loss in Eq. 9. The training dynamic is shown in Figure 5. Here, training transition loss is calculated using the transition between non-terminal states. And training starting loss and training end loss are calculated using boundary condition $F(s_0) = F(s_T) = 0$. Training total loss and eval loss are calculated on all the transitions between states on the training and evaluation dataset. Eval policy accuracy is the accuracy of policy $P(s_{t+1}|s_t)$ on the evaluation dataset. We training GraphFlow on these dataset for one epoch, other important parameters are shown in Table 4.

C Implementations of Baselines

To the best of our knowledge, few KG-based RAG methods are implemented on the text-rich STaRK benchmark. Instead, many KG-based RAG methods employ simple KGQA datasets such as CWQ, WEBQSP. Thus, we choose representative retrieval-based and agent-based baselines with explicit retrieval results (i.e., the retrieved node index) on the STaRK benchmark [74]. We provide the implementation details of the used baseline methods as below.

Dense-Retriever is implemented with SentenceBERT [62] to encode both questions and the documents of KG nodes into dense embeddings and retrieve the documents with top vector similarity. We choose SentenceBERT as the text document to be consistent with prior works [21], where SentenceBERT is used to encode the text information in KGs. Although STaRK benchmark provide the pre-processed text embedding of entities and relationships in KGs using text-embedding-ada-002 model, we find the inconsistency between the entities IDs and the entities embeddings. Some entities in KGs are not converted into embeddings. Thus, we rerun the encoding model using SentenceBERT to obtain the full entities embeddings. After encoding the text information into embeddings, we employ the vector similarity between the question embedding and text embeddings for retrieval. We evaluate the retrieval performance on top 20 retrieval results.

G-Retriever [21] is a two-stage method for KG-based RAG. It first employs the Prize-Collecting Steiner Tree (PCST) [2] algorithm to retrieve a subgraph from KGs relevant to the query. Then, the

retrieved subgraph is encoded into the token space of LLM using a GNN for question answering (QA). To further improve the QA performance, G-Retriever also applies LoRA module to fine-tune LLM. Since we focus on the evaluating the retrieval performance of different KG-based RAG methods, we do not fine-tune the GNN and LLM for QA. To make PCST algorithm feasible on STaRK benchmark, we adopt a hybrid approach that first identify the 20 seed nodes and implement the PCST algorithm to extract the subgraphs around 2-hop ego graph around the seed nodes. We drop the seed nodes with dense neighborhoods to avoid computation overhead [33, 34].

SubgraphRAG [37] integrates a learnable subgraph retrieval module to retrieve from KGs. Since training the subgraph retrieval module on the STaRK benchmark is infeasible, we employ the ego-graph setting similar to G-Retriever. We identify the up-to 2 hop neighbor hood graph around the seed node to construct the training and testing set for SubgraphRAG. We also drop the the seed node that has dense neighborhood to avoid computation overhead. This ego-graph setting is also employed to construct the test set for the other KG-based RAG models. We follow the default setting of SubgraphRAG to reproduce it on STaRK benchmark.

ToG [68] employs an LLM agent to search from the KG to support KG-based question anwsering. ToG is implemented with frozen LLMs by prompt engineering instead of fine-tuning. Specifically, ToG employs tree-based search [79] to transverse the KG and search the relevant information for KG-based QA. Since we focus on evaluating the retrieval performance of KG-based RAG models, we modify ToG to retrieve the relevant document at each searching steps instead of incorporating the retrieved document to update the question answering results. Since running ToG on the whole KGs in STaRK is infeasible, we identify the seed node for ToG searching using vector similarity and constrain the searching area around the 2-hop neighborhood of the seed node. We instantiate ToG using both LLaMA3-8B and GPT-4o as backbone models, denoted as ToG+LLaMA3 and ToG+GPT4o, respectively.

We also implement SFT and PRM as two fine-tuning baselines build upon ToG and LLaMA3-8B-Instruct. We use the sample training dataset to train ToG using SFT and PRM as GraphFlow for a fair comparison. We employ the **TRL** (Transformer Reinforcement Learning) package to fir SFT and PRM fine-tuning. We apply LoRA funetuning to improve the efficiency.

Other potential baselines but hard to implement on STaRK. There are alternative KG-based RAG baseline methods for evaluation. However, we find it hard to implemented these baseline on STaRK, mostly due to the compatibility issues. We list some examples as below.

QAGNN [80] is designed for improving the QA performance on KG-based QA task. Although its retrieval performance is reported on STaRK benchmark, detailed implementation code on STaRK is not publicly available. Although recent concurrent work [34, 33] tried to implement QAGNN on STaRK, the reported performances of QAGNN diverge from the reported results on STaRK benchmark.

RoG [47] adopts similar approach as it finetunes the LLM to search from KGs. It first employs an LLM to generate retrieval trajectories for the input queries and use the generated retrieval trajectories to construct a training dataset to fine-tune the retrieval agent by SFT. However, we find that LLM usually generate invalid retrieval trajectory, leading to low quality training datasets for SFT fine-tuning. Thus, we finetune the retrieval agent using the valid retrieval trajectories by SFT in the main paper.

ToG-2.0 [50] is a recently proposed method to retrieve from the structured database and unstructured database. The key to ToG-2.0 is to identify the topic entitiies for a given questions. However, the implementation of topic entity recognition is absent, making it difficult to reproduce ToG-2.0 on STaRK benchmark.

HybridRAG [33], Mixture of RAG [34], and KAR [75] are recent pre-prints on Arxiv focusing on retrieving from text-rich KGs. However, their codes are not available yet, making it difficult for us to reproduce these methods.

D More results of Cross-domain Generalization

We show more generalization performance in terms of Hit@5 in Figure 6.

 https://doi.org/10.52202/085713-3200

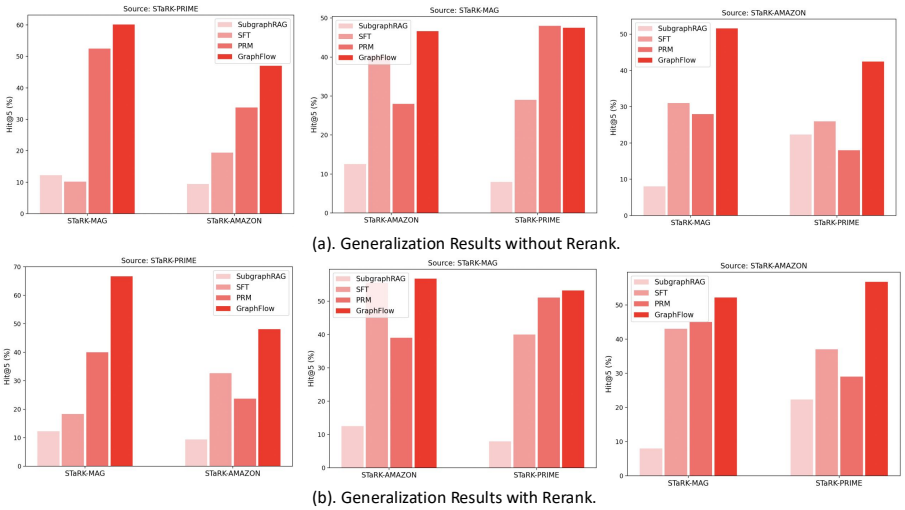

Figure 6: Generalization Performance (Hit@5) of KG-based RAG methods. GraphFlow shows superior cross-domain generalization performance, especially under the rerank setting (best viewed in color).

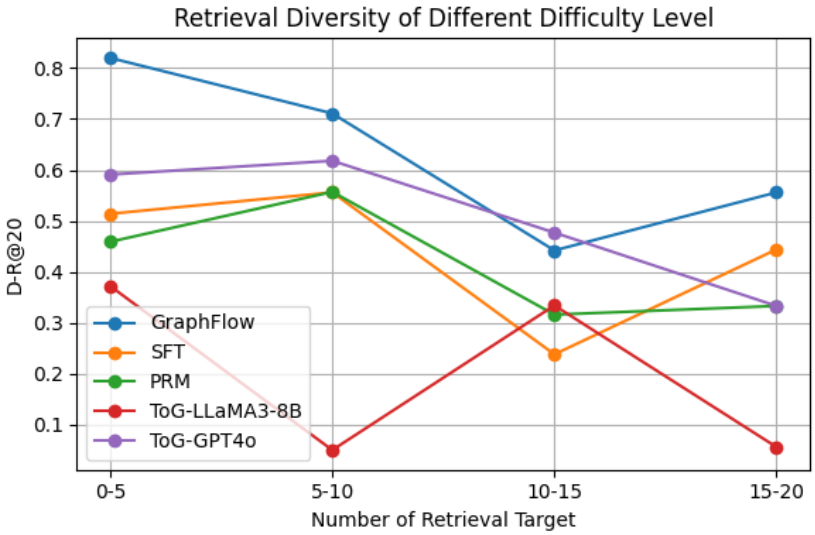

Figure 7: GraphFlow shows improved retrieval diversity on different difficulty levels of retrieval queries on STaRK-PRIME.

E More results of Hard Cases

We categorize the retrieval queries with different numbers of retrieval targets into 4 difficulty levels. We provide the performance of different KG-based RAG on STaRK-PRIME, STaRK-MAG, and STaRK-AMAZON at different difficulty levels. The results are shown in Figure 7, Figure 8, and Figure 9.

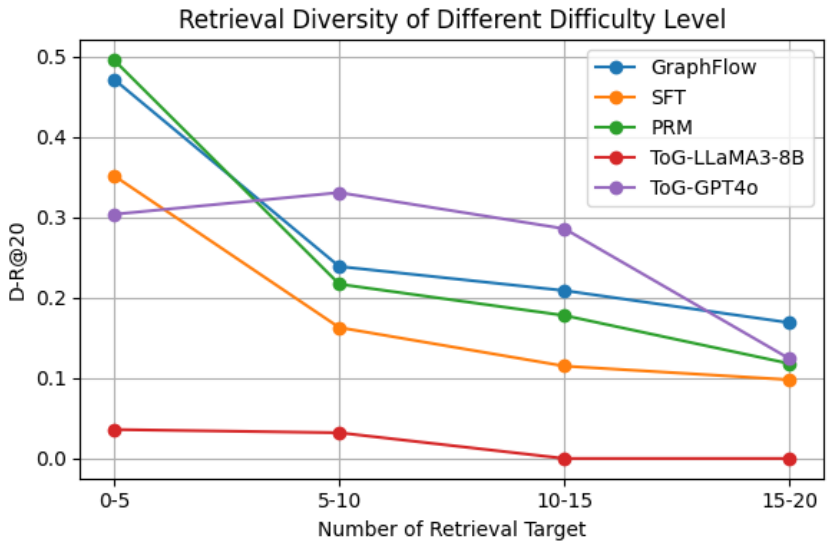

Figure 8: GraphFlow shows improved retrieval diversity on different difficulty levels of retrieval queries on STaRK-AMAZON.

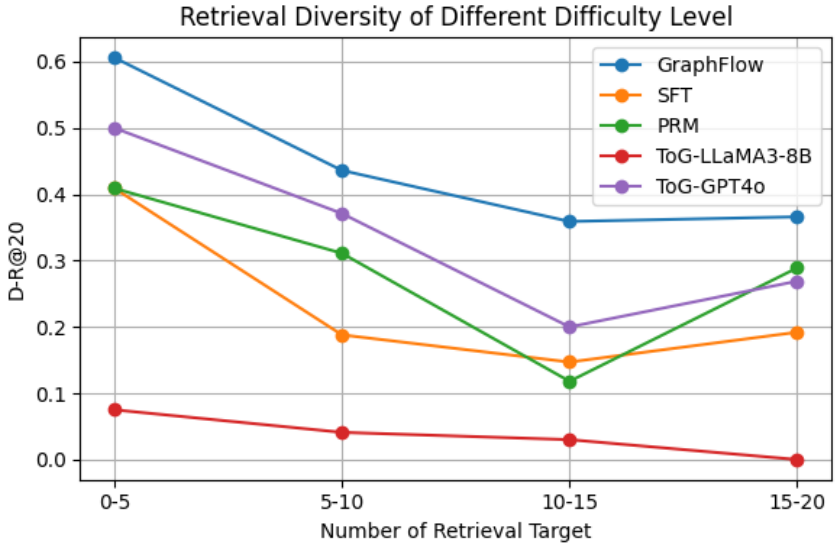

Figure 9: GraphFlow shows improved retrieval diversity on different difficulty levels of retrieval queries on STaRK-MAG.

F Benchmark Information

We provide benchmark information in Table 5.

G Computing Resources

We run all experiments on 8/16 NVIDIA-A800-SXM4-80GB GPUs and 56 Intel(R) Xeon(R) Platinum 8336C CPUs.

https://doi.org/10.52202/085713-3200

H Broader Impact and Limitations

GraphFlow introduces a novel framework for retrieval-augmented generation over text-rich knowledge graphs, enabling Large Language Models (LLMs) to reason more effectively through process supervision using GFlowNets. By modeling retrieval as a generative process that balances diverse and relevant paths, GraphFlow promotes both interpretability and coverage in knowledge-based reasoning. This has broad implications for applications such as scientific discovery, open-domain question answering, and medical decision support, where combining structured knowledge with free-text reasoning is crucial. Moreover, GraphFlow can serve as a foundation for future research in integrating generative decision-making with symbolic structures, thereby pushing forward the synergy between LLMs and knowledge graphs. One potential limitation is that we only evaluate the generalization ability of GraphFlow on two new domains.

https://doi.org/10.52202/085713-3200