Graph-Based Agentic Retrieval-Augmented Generation: A Comprehensive Survey

Raviteja Bommireddy
IIITDM Kancheepuram, Chennai, India
cs23b2011@iiitdm.ac.in

Akansha Singh
Manipal Institute of Technology, Manipal, India
agrawal.akansha5@gmail.com

***Abstract*—This survey on Graph-Based Agentic Retrieval-Augmented Generation (GA-RAG) unifies structured knowledge graphs, autonomous agentic workflows, and large language models to advance compositional, multi-hop reasoning beyond traditional RAG pipelines. In this survey we formalize the GA-RAG paradigm, present a compact taxonomy of reasoning workflows, agent orchestration patterns, retrieval and knowledge-engineering practices, and adaptation mechanisms, and synthesize empirical and methodological trends across recent work (including documented gains in multi-hop accuracy over flat RAG approaches. We systematically map benchmarks, datasets, and evaluation protocols, analyze representative systems, and extract recurring design patterns such as model-mediated traversal, evaluator–optimizer loops, and hybrid retrieval strategies that underpin current progress. We distill the principal limitations scalability to web-scale graphs, temporal and multimodal reasoning, neural symbolic fidelity, and privacy/explainability—and propose a focused research agenda emphasizing provably efficient subgraph selection, time-aware representations, differentiable symbolic modules, and privacy-preserving federated graph reasoning. By consolidating architectures, metrics, and open problems, this work aims to serve both as a concise reference for practitioners and as a roadmap for advancing robust, transparent, and deployable GA-RAG systems.**

***Index Terms*—knowledge graphs, retrieval-augmented generation, autonomous agents, multi-agent systems, graph neural networks**

I. Introduction

Recent advances in large language models (LLMs) [1], [2] have revolutionized natural language processing. Initially, progress was driven by scaling pre-training corpora, leading to remarkable capabilities in text generation [2], summarization [3], and dialogue [4]. More recently, a paradigm shift has occurred, moving from traditional pre-training to dynamic test-time scaling approaches [5]. This evolution, exemplified by models like OpenAI's O1 [6] and DeepSeek-R1 [7], has unlocked new capabilities in complex tasks such as mathematical reasoning and code generation, often powered by post-training innovations like Long Chain-of-Thought (Long-CoT) reasoning [8]. Despite this rapid progress, LLMs suffer from fundamental limitations that hinder their reliability. Their knowledge is static, confined to the data they were trained on, rendering them unable to access information or events that occurred post-training. Furthermore, they are prone to "hallucinations" [9]—generating plausible but factually incorrect or nonsensical information. These weaknesses undermine their trustworthiness, especially in knowledge-intensive applications such as medical question answering [10] and legal document analysis [11], where precision and factuality are paramount.

To mitigate these shortcomings, Retrieval-Augmented Generation (RAG) has emerged as a dominant paradigm [12]–[14]. RAG enhances LLMs by grounding them in external, up-to-date knowledge. The standard RAG workflow involves retrieving a set of relevant text passages from a corpus in response to a query and providing these passages as context for the LLM to synthesize an answer. This approach has proven effective for improving factuality and handling single-hop questions where the answer is contained within a few retrieved documents.

However, this traditional RAG architecture, which typically retrieves the top-k semantically similar passages from a flat, unstructured corpus [12], [15], faces significant challenges when confronted with more complex information needs. Its design fundamentally struggles with multi-hop reasoning, which requires synthesizing information across multiple documents or data points [16], [17]. This limitation is a direct consequence of its rigid, single-step workflow: the retriever fetches a set of documents once, and the generator must produce an answer based solely on that initial context. There is no mechanism for feedback or iterative refinement; the generation module cannot request more relevant documents or signal that the initial retrieval was insufficient. This non-adaptive process, where the retriever and generator operate in isolated steps, has been identified as a key bottleneck in solving complex problems [18], [19]. If the initial retrieval is incomplete or slightly off-topic, the error propagates directly to the generation step, leading the system to produce fragmented or incorrect answers without any chance to correct its course [18].

In response to these limitations, the field has progressed through several evolutionary stages. The first wave of solutions, often termed **Advanced RAG**, introduced more dynamic and iterative workflows. Techniques like iterative retrieval, which refines queries over multiple steps, and self-correction, where the model critiques and improves its own retrieved evidence [17], began to address the static nature of the original paradigm. Yet, these methods still primarily operate on unstructured text, which inherently lacks the explicit relational information needed for deeper reasoning. To address this, **Graph RAG** emerged, which leverages Knowledge Graphs (KGs) as the external knowledge source. By encoding information as entities (nodes) and relationships (edges), KGs provide a structured backbone that is ideal for navigating complex,

multi-hop queries.

The most recent and sophisticated evolution synergizes these advancements, leading to **Graph-Based Agentic RAG (GA-RAG)**. This paradigm integrates the structured foundation of KGs with the autonomous capabilities of intelligent agents. These agents can perform complex reasoning by planning, using tools (e.g., specific graph traversal algorithms), and reflecting on their actions to dynamically and interpretively navigate the knowledge graph, thus creating a powerful, goal-driven retrieval and generation loop [20]–[22].

To address these critical bottlenecks, the field is moving towards more sophisticated architectures that integrate structured knowledge and dynamic reasoning. This paper focuses on Graph-Based Agentic RAG (GA-RAG), a paradigm that represents the next evolution of retrieval systems. GA-RAG addresses the limitations of traditional RAG by incorporating two key innovations:

Knowledge Graphs (KGs): Instead of relying solely on unstructured text, these systems leverage KGs structured data representations where entities (nodes) and their semantic relationships (edges) are formally encoded. This structure enables explicit, multi-hop traversal and robust reasoning across interconnected information [23].

Agentic Intelligence: These frameworks are built around autonomous agents capable of complex decision-making. Such agents can perform planning, reflection (self-critique), and tool use, orchestrating adaptive, goal-directed workflows that move beyond static pipelines and can dynamically interact with knowledge sources like KGs [24], [25]. A paradigm whereby autonomous software agents possess capabilities for planning, reflection (self-critique), tool use, and collaboration. In the context of *RAG*, agentic intelligence orchestrates modular retrieval, reasoning, and generation steps adapting dynamically to intermediate outcomes and user goals, thereby overcoming the static pipelines of traditional systems [24], [25]. Table I compactly contrasts text-only RAG, Graph-RAG, and Agentic RAG across retrieval unit, reasoning style, adaptivity, modularity, and explainability, and shows the progression from flat retrieval to agentic, graph-aware workflows.

By combining the structured traversal of knowledge graphs with the dynamic, intelligent workflows of agentic systems, GA-RAG promises to significantly improve multi-hop answer accuracy, workflow transparency, and adaptability in complex domains. This survey examines the foundational principles and recent advances at the intersection of these technologies, charting the progress from simple retrieval to intelligent, graph-aware reasoning systems. This survey investigates the intersection of these domains. We examine how *Agentic RAG* systems integrate graph-structured knowledge with agentic workflows, survey architectural paradigms and taxonomies, evaluate benchmark performance, and chart future directions for scalable, explainable, and multimodal graph-based agentic RAG. Our scope includes foundational methods (e.g., GraphRAG, QA-GNN), recent surveys on agentic RAG, and emerging applications in scientific discovery, enterprise search, and personalized recommendation.

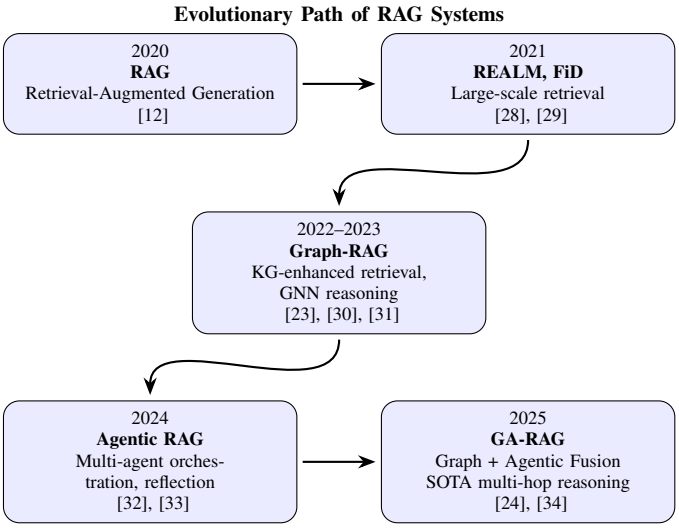

Fig. 1: Timeline showing the evolution from classical RAG to Graph-Based Agentic RAG (GA-RAG).

II. Foundations of Graph-Based Agentic RAG

A. Core Concepts of RAG

Retrieval-Augmented Generation (*RAG*) was proposed to mitigate hallucination and factual drift in large language models (LLMs) by grounding their outputs in retrieved evidence from external corpora [12], [14], [29]. A standard *RAG* pipeline relies on a dense retriever such as DPR [15] or BERT-based encoders to identify the top-k relevant passages, which are then concatenated into the model's input window for conditional generation. While this paradigm significantly improves factual grounding and achieves up to 85% accuracy on single-hop question answering tasks [29], its performance declines sharply often to below 50% on multi-hop, compositional, or knowledge-intensive reasoning [16], [28], [35]. The main limitations arise from fragmented and unstructured context that hinders relational reasoning, static one-shot retrieval workflows that lack adaptive refinement, and siloed retriever–generator modules that prevent iterative feedback between retrieval and generation. These constraints underscore the need for structured and adaptive augmentation beyond flat retrieval.

B. Knowledge Graph Integration in RAG

Knowledge Graphs (KGs) encode entities and their relations as nodes and edges, thereby enabling explicit traversal, compositional reasoning, and hierarchical abstraction [23], [36]. Their integration into retrieval-augmented generation systems has given rise to the Graph-RAG paradigm [26], [30], [31], where subgraphs—rather than isolated text passages—are retrieved and used as evidence. This structured approach enhances reasoning by supporting explicit multi-hop traversal over relation paths [23], improving disambiguation through ontological hierarchies and type constraints [37], [38], and accelerating retrieval via efficient graph indices and pruning strategies [26], [31]. Empirical studies demonstrate that Graph-RAG delivers substantial gains, with up to 30-point improve-

Feature	Text-Only RAG	Graph-RAG	Agentic RAG
Retrieval Unit	Independent passages	Entity–relation subgraphs	Agent-selected passages/subgraphs [12], [21], [26]
Reasoning Style	One-shot, flat	Multi-hop via traversal [23]	Iterative with reflection loops [27]
Adaptivity	Static pipeline	Fixed graph queries	Dynamic planning and self-critique [24]
Modularity	Retriever + LLM	+ Graph index	+ Multiple collaborating agents [25]
Explainability	Low	Medium (graph paths)	High (agent logs + graph)

TABLE I: Comparative Foundations of RAG, Graph-RAG, and Agentic RAG

ments in multi-hop reasoning accuracy on benchmarks such as HotpotQA and GraphQA [23], [26]. These findings illustrate the transformative role of structured graph representations in overcoming the fragmentation and inefficiency of text-only retrieval pipelines.

C. Agentic Intelligence Principles

The incorporation of autonomous agents into *RAG* systems transforms static retrieval-generation pipelines into adaptive, modular, and goal-directed workflows [24], [32]. Agentic intelligence introduces several key design principles. First, reflection mechanisms enable agents to critique and validate intermediate outputs, triggering revised retrieval or reasoning steps when inconsistencies are detected [27], [34]. Second, planning strategies allow the decomposition of complex queries into structured subgoals, guiding the order and depth of retrieval and enhancing multi-step reasoning efficiency [18], [20]. Third, tool use empowers agents to invoke external APIs, calculators, or specialized graph-traversal utilities to support domain-specific computations and multimodal integration [32], [39]. Finally, multi-agent collaboration enables specialized agents—such as entity resolvers, subgraph selectors, and synthesis agents—to interact through shared memory and communication protocols, thereby achieving scalability and parallelism in complex reasoning tasks [24], [33], [40]. Recent studies highlight that such agentic *RAG* frameworks not only achieve near-linear scalability with growing agent teams but also significantly enhance robustness and accuracy in dynamic, knowledge-intensive environments [24], [26], [32].

Figure 2 illustrates the orchestrator decomposing queries, worker agents (entity, graph, vector retrievers), routing, aggregator, and an evaluator–optimizer feedback loop for iterative refinement.

III. Paradigms of Graph-Based Agentic RAG

Graph-Based Agentic Retrieval-Augmented Generation (GA-RAG) systems can be categorized according to their degree of graph structure exploitation and level of agent autonomy. We outline five representative paradigms (Table II), beginning with naive embedding-based methods and progressing toward fully agentic architectures that tightly integrate graph reasoning with autonomous orchestration.

Figure 3 illustrates the layered evolution from flat retrieval through hierarchical and modular graph use, culminating in adaptive agentic control over graph-augmented workflows.

A. Naive Graph RAG (Single-Pass, Flat Graph Retrieval)

Naive Graph RAG extends traditional dense retrieval by embedding graph nodes, edges, or textual passages into a high-dimensional vector space using encoders such as BERT or GraphSAGE [31]. Given a query q, represented as an embedding $\mathbf{h}_q$, retrieval is performed by nearest-neighbor search:

$$\hat{\mathcal{G}} = \{v \in \mathcal{V} \mid \text{top-}k \text{ of } \cos(\mathbf{h}_q, \mathbf{h}_v)\},$$

where $\mathbf{h}_v$ is the embedding of a graph node $v \in \mathcal{V}$. Although effective for isolated factoid questions, this approach neglects the graph's relational structure. Consequently, performance gains over text-only RAG are modest, as reasoning chains that span multiple hops across entities are not captured. Empirical results show only minor improvements on benchmarks requiring single-hop inference [31].

B. Advanced Graph RAG (Hierarchical, Community-Aware Retrieval)

Advanced Graph RAG addresses the shortcomings of naive approaches by leveraging hierarchical community detection or clustering algorithms to partition the graph into semantically coherent subgraphs. Given a graph $G = (\mathcal{V}, \mathcal{E})$, clustering methods (e.g., Louvain modularity or spectral clustering) assign nodes to communities $\{C_1, C_2, \ldots, C_m\}$. A query embedding $\mathbf{h}_q$ is first matched against community representations (centroids or learned embeddings $\mathbf{h}_{C_i}$), and retrieval then proceeds in a coarse-to-fine manner:

$$\hat{C} = \arg\max_{C_i} \cos(\mathbf{h}_q, \mathbf{h}_{C_i}), \quad \hat{\mathcal{G}} = \{v \in \hat{C} \mid \cos(\mathbf{h}_q, \mathbf{h}_v) \text{ top-}k\}.$$

Systems such as ArchRAG and RAPTOR [20], [41] combine community summaries with fine-grained retrieval, yielding more efficient search and improved reasoning on moderate multi-hop queries. This paradigm effectively balances high-level abstraction with detailed traversal, reducing retrieval latency while enhancing interpretability.

C. Modular Graph RAG (Decoupled Retrieval, Reasoning, Generation)

Modular Graph RAG architectures decouple the pipeline into distinct but interleaved components: subgraph retrieval, graph-based reasoning or re-ranking, and final generation. For instance, FRAG [42] and GFM-RAG [43] employ iterative loops where retrieved evidence is refined before being passed to the LLM. Mathematically, retrieval is modeled as:

$$\mathcal{G}_t = \text{Retrieve}(q, \mathcal{M}_{t-1}),$$

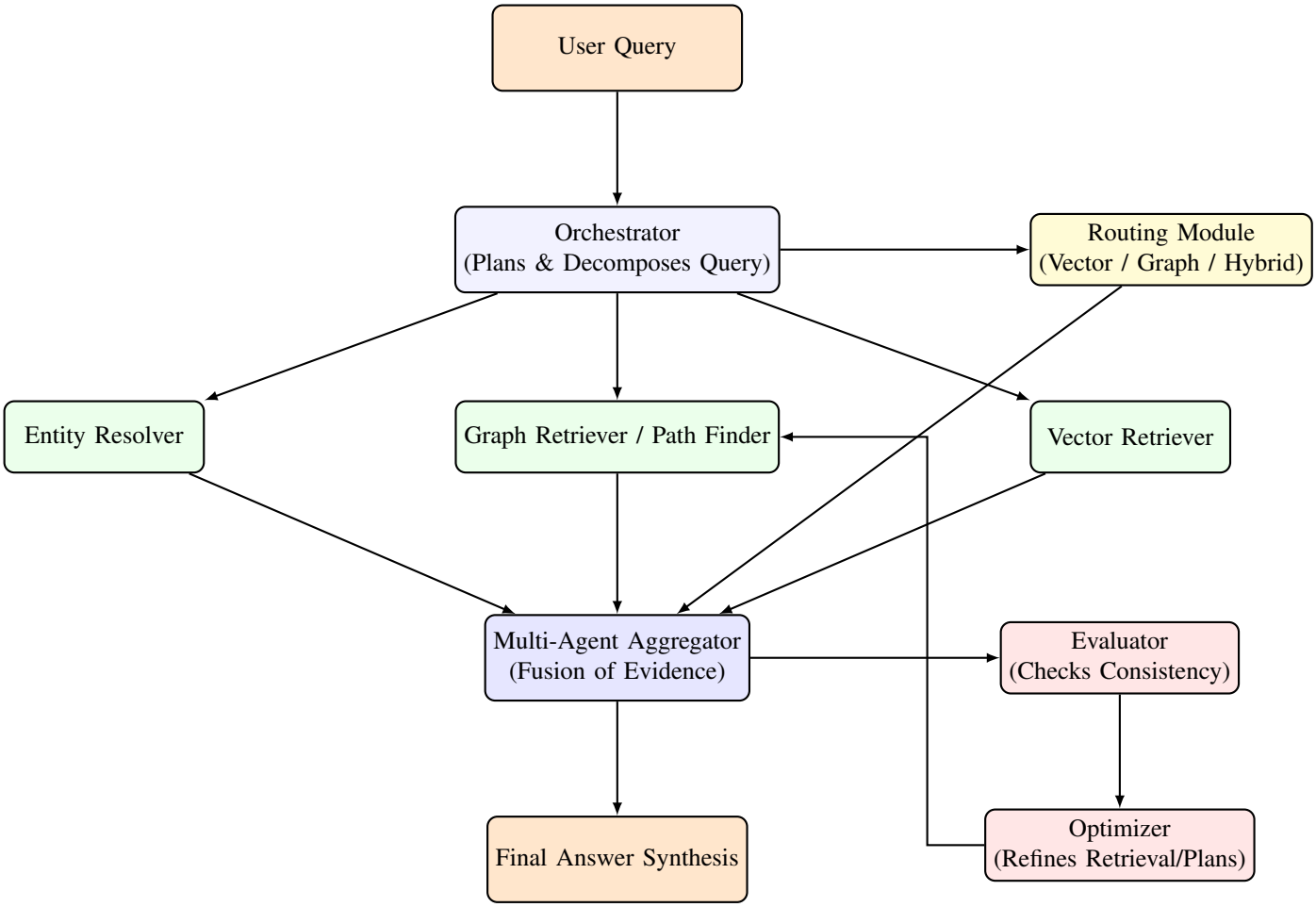

Fig. 2: Illustration of Graph-Based Agentic RAG workflow. A user query enters the orchestrator, which decomposes it into subgoals. Subgoals are dispatched to worker agents (entity resolver, graph retriever, vector retriever) or routed via a hybrid module. Results are aggregated, refined through evaluator–optimizer feedback loops, and synthesized into a final answer.

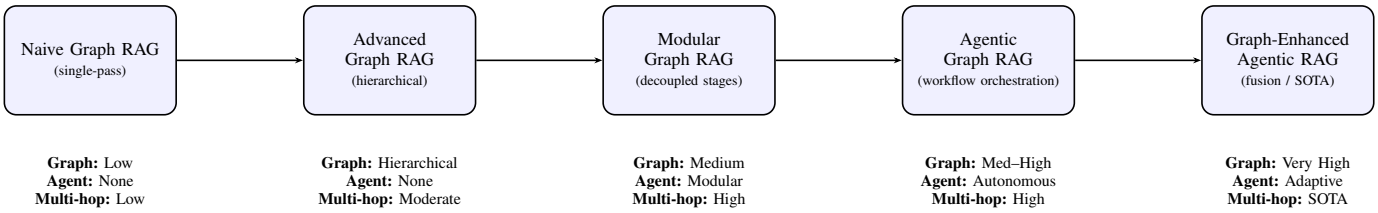

Fig. 3: Paradigms of Graph-Based Agentic RAG. Left-to-right progression shows: (i) Naive embedding-based approaches, (ii) hierarchical/community-aware retrieval, (iii) modular retrieval–reasoning–generation pipelines, (iv) autonomous agentic orchestration, and (v) fusion of GNN-based graph reasoning with agentic control.

where $\mathcal{M}_{t-1}$ denotes memory summarizing prior retrievals. Reasoning modules (often GNN-based) score or re-rank nodes/edges via attention mechanisms:

$$\alpha_{uv} = \mathrm{softmax}\left(\mathbf{h}_u^\top W \mathbf{h}_v\right),$$

producing a weighted subgraph $\mathcal{G}_t^*$ that better reflects relational importance. Generation then conditions on $\mathcal{G}_t^*$ to produce outputs. By storing intermediate retrievals and enabling evidence refinement, modular RAG achieves robustness to retrieval errors and compositional reasoning that flat pipelines cannot.

D. Agentic Graph RAG (Autonomous Workflow Orchestration)

Agentic Graph RAG introduces autonomous agents that orchestrate the retrieval–reasoning–generation cycle. These agents plan query decomposition, reflect on intermediate results, and invoke external tools such as graph traversers or symbolic solvers [25], [44]. A central orchestrator agent delegates subtasks to worker agents (entity resolvers, path finders, evidence aggregators), each operating over the knowledge graph. Formally, the orchestration policy $\pi_\theta(a|s)$ is learned via reinforcement learning, where s denotes the current graph context and a the agent action (e.g., expand a subgraph,

Paradigm	Multi-Hop	Key Idea	Strengths	Limitations
Naive Graph RAG (Single-Pass)	Low (23–56)	Node embeddings in a vector index; retrieve top-k nodes or passages.	Simple, efficient, good for factoid queries.	Ignores graph structure; fails on multi-hop tasks.
Advanced Graph RAG (Hierarchical)	Moderate (36–72)	Graph clustering or community detection for multi-level retrieval (communities + nodes).	Abstract reasoning, noise reduction, improved multi-hop QA.	Dependent on clustering quality; risk of overgeneralization.
Modular Graph RAG (Decoupled Stages)	High (48–81)	Iterative stages: subgraph retrieval, graph reasoning/reranking, then generation.	Robust, supports interleaved retrieval–reasoning, dynamic refinement.	Slower due to multi-stage processing.
Agentic Graph RAG (Workflow Orchestration)	High (44–81)	Orchestrator and worker agents plan, reflect, and invoke graph tools.	Adaptive, self-correcting, scales via multi-agents.	Orchestration complexity; RL policy tuning is hard.
Graph-Enhanced Agentic RAG (Fusion)	SOTA (54–86)	Combines GNN-based traversal with agentic orchestration.	SOTA multi-hop reasoning, robust to evolving KGs.	Computationally expensive, heavy memory use.

TABLE II: Comparison of GA-RAG paradigms: from naive retrieval to graph-enhanced agentic orchestration.

reformulate query, or invoke a tool). The reward function $R(s, a)$ can be shaped by downstream LLM answer quality or retrieval relevance. Over time, π_θ converges to strategies that optimize multi-hop accuracy and efficiency. This paradigm enables dynamic workflow orchestration, moving beyond static retrieval to adaptive, self-correcting processes.

E. Graph-Enhanced Agentic RAG (Fusion of Graph Reasoning and Agent Patterns)

The most advanced paradigm fuses deep graph reasoning with agentic orchestration, achieving state-of-the-art results on complex reasoning tasks. In these systems, graph neural networks (e.g., GCNs, GATs) are used to score candidate paths or subgraphs [45], while autonomous agents determine traversal depth, reformulate queries, and coordinate reflection. For example, given a query q, a GNN encoder computes contextualized node embeddings:

$$\mathbf{H}^{(l+1)} = \sigma\left(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}\mathbf{H}^{(l)}W^{(l)}\right),$$

where $\tilde{A} = A+I$ is the adjacency matrix with self-loops, $\tilde{D}$ its degree matrix, and $W^{(l)}$ trainable weights. These embeddings are scored to identify reasoning paths, while an agentic loop dynamically decides whether to expand, validate, or finalize responses. Frameworks such as GraphRAG [34] demonstrate that combining structured graph reasoning with reflective agent control yields substantial improvements in multi-hop reasoning accuracy, interpretability, and robustness. This paradigm represents the current frontier of GA-RAG, offering a blueprint for the integration of symbolic graph structure, neural representation learning, and agentic intelligence.

Figure 4 shows an Agent Orchestrator routing a user query to parallel vector (surface) and graph (multi-hop) retrieval; an agentic reasoning stage then performs prompt-chaining, tool invocation and evaluator–optimizer loops before fusing results for LLM generation.

IV. Graph-Based Agentic Workflows

The paradigmatic evolution from naive to graph-enhanced agentic RAG necessitates sophisticated operational frameworks that orchestrate the interplay between agents, knowledge graphs, and large language models. Graph-Based Agentic Retrieval-Augmented Generation (GA-RAG) systems transcend monolithic architectures by implementing modular and adaptive workflows that dynamically regulate retrieval strategies, reasoning pathways, and generation processes [25], [46]. These workflows constitute the computational substrate upon which agentic behaviors emerge, defining how autonomous agents navigate graph structures, coordinate retrieval operations, and synthesize coherent responses.

Building upon the paradigmatic foundations established in Section III, The integration of graph structures amplifies these capabilities by providing structured navigation paths and semantic relationships that guide agentic decision-making processes.

We systematically categorize six canonical workflows that exemplify the operational principles of GA-RAG systems: prompt chaining, dynamic routing, parallel exploration, orchestrator–worker decomposition, evaluator–optimizer loops, and self-memory with tool invocation. Figure 5 illustrates how these distinct approaches to agent coordination, graph traversal, and knowledge synthesis converge into unified graph operations, LLM generation, and evidence synthesis to produce coherent responses [22], [25], [34], [41], [42], [45].

A. Prompt Chaining over Graphs

Prompt chaining generalizes chain-of-thought (CoT) prompting to graph reasoning, decomposing a query into sequential prompts aligned with graph operations such as node visitation, edge traversal, or subgraph summarization [20], [34], [46]. Each prompt p_t is conditioned on accumulated evidence E_{t-1}:

$$o_t = \mathrm{LLM}(p_t \,|\, E_{t-1}), \quad E_t = E_{t-1} \cup \mathrm{Extract}(o_t),$$

where o_t denotes the output at step t and $\mathrm{Extract}$ parses entities or relations to extend the reasoning chain. This iterative refinement mimics symbolic path traversal while leveraging LLM flexibility. Empirical studies show that prompt chaining significantly boosts compositional and multi-hop QA performance by forcing stepwise grounding in KG structure [22], [23], [30].

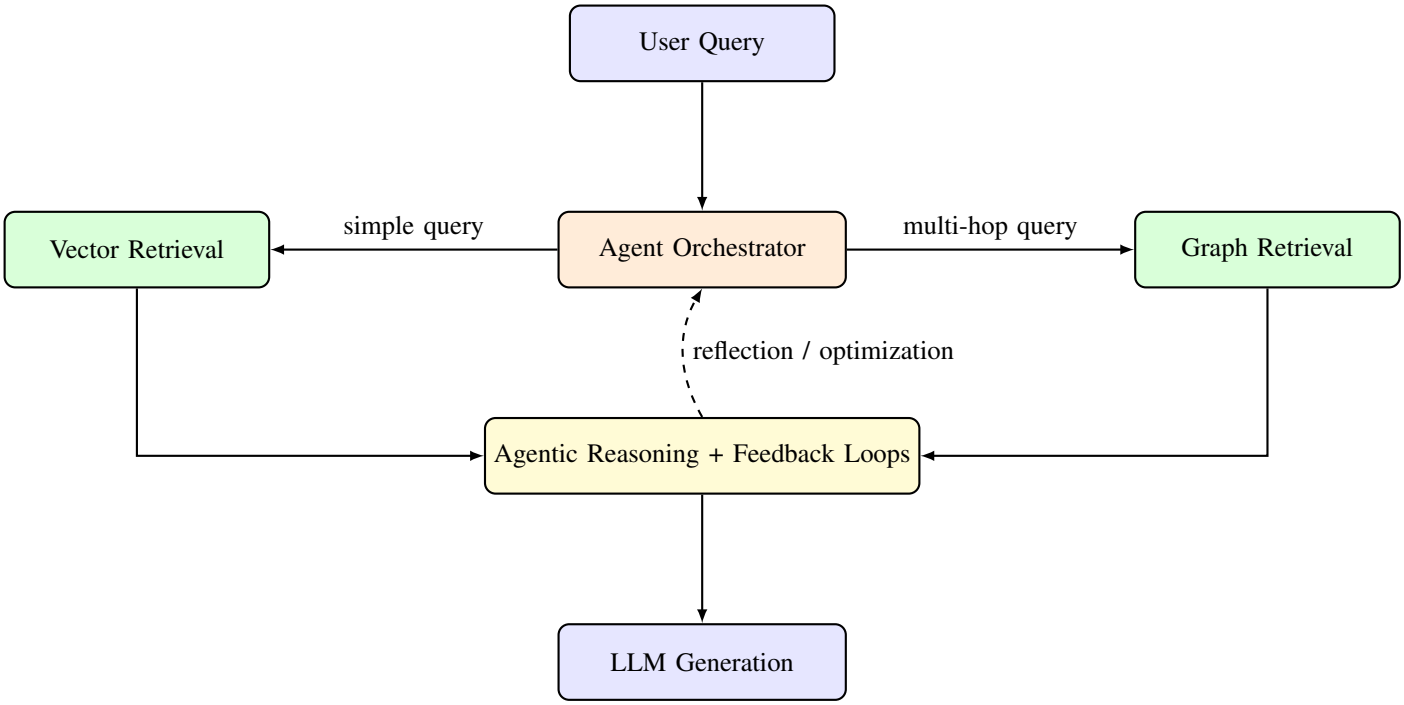

Fig. 4: Illustration of GA-RAG workflows: agents orchestrate between vector and graph retrieval, apply reasoning and feedback loops, and guide LLM generation.

B. Routing: Dynamic Dispatch to Vector, Graph, or Hybrid Retrieval

Routing modules dynamically select between retrieval modalities—vector search, graph traversal, or hybrid pipelines—based on query complexity and evidence requirements [31], [34], [42]. Formally, the retrieval function can be expressed as:

$$R(q) = \lambda\, R_{\text{vec}}(q) + (1-\lambda)\, R_{\text{graph}}(q),$$

where R_{vec} and R_{graph} denote vector and graph retrieval results respectively, and $\lambda \in [0,1]$ is adaptively tuned by an agentic router. For simple factual queries, $\lambda \to 1$ emphasizes semantic recall, while entity-rich, multi-hop tasks set $\lambda \to 0$ to favor graph reasoning. Hybrid routing often employs re-ranking functions to enforce consistency between textual similarity and relational constraints [45], [47]. Learned dispatchers enable GA-RAG systems to automatically balance breadth of recall with depth of reasoning.

C. Parallel Graph Exploration with Multiple Agents

Multi-agent GA-RAG workflows parallelize search by distributing sub-tasks across specialized agents—such as entity solvers, path expanders, or cluster summarizers [25], [41], [44]. Given a query q, the graph is partitioned into regions $\{G_1, \ldots, G_m\}$, each explored by an agent $\mathcal{A}_i$ producing evidence set E_i. Final synthesis merges $\{E_i\}$ into a unified answer:

$$E = \bigcup_{i=1}^{m} \mathcal{A}_i(G_i, q).$$

Theoretical models interpret this as distributed search with communication complexity $O(m \log |\mathcal{V}|)$ for graphs of size $|\mathcal{V}|$. Experiments show near-linear throughput scaling with agent count, alongside improved robustness in sparse or noisy graphs [48]–[50]. Such parallel workflows are particularly suited for enterprise-scale KGs and real-time scientific discovery.

D. Orchestrator–Worker: Task Decomposition on Graph Structures

The orchestrator–worker paradigm assigns a central orchestrator agent $\mathcal{O}$ the role of decomposing complex queries into structured subgoals $\{g_1, \ldots, g_k\}$ [25], [44]. Each subgoal is dispatched to a worker $\mathcal{W}_i$ that operates on specific graph substructures. Execution proceeds recursively:

$$a_i = \mathcal{W}_i(g_i, G), \quad A = \text{Aggregate}(a_1, \ldots, a_k),$$

where a_i are partial answers and A the synthesized result. Orchestration can be optimized via hierarchical reinforcement learning, with $\mathcal{O}$ learning policies over decomposition strategies. Frameworks such as LangGraph and AutoGen support persistent state, memory, and recursive subgoal handling, making orchestrator–worker workflows highly modular and explainable [32], [33], [51], [52].

E. Evaluator–Optimizer: Iterative Refinement of Graph Queries

Evaluator–optimizer workflows embed explicit feedback mechanisms, coupling evaluators $\mathcal{E}$ that assess evidence validity with optimizers $\mathcal{O}$ that refine retrieval plans [20], [31], [42], [45]. After each iteration t:

$$s_t = \text{Retrieve}(q, G), \quad r_t = \mathcal{E}(s_t), \quad q_{t+1} = \mathcal{O}(q, r_t),$$

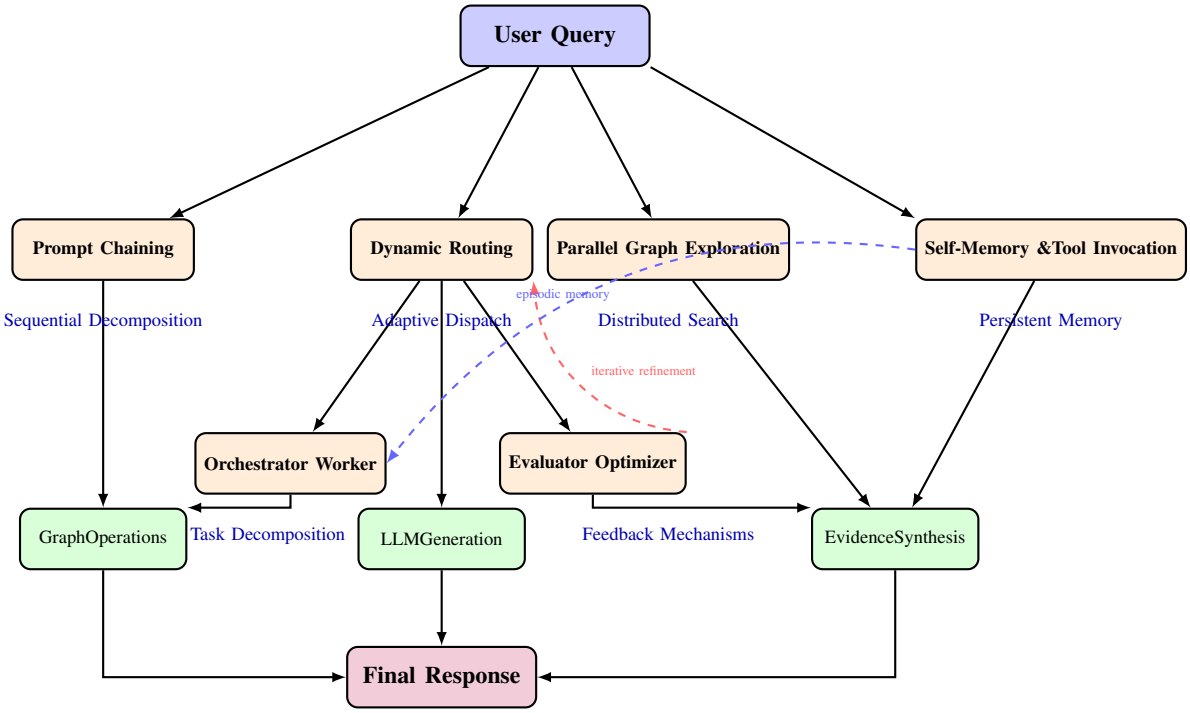

Fig. 5: Graph-Based Agentic Workflows: Six canonical workflows coordinate through graph operations, LLM generation, and evidence synthesis. Prompt chaining enables sequential decomposition, dynamic routing provides adaptive dispatch, parallel exploration supports distributed search, orchestrator-worker implements task decomposition, evaluator-optimizer provides feedback mechanisms, and self-memory maintains persistent context across reasoning iterations.

where r_t denotes an evaluation signal (e.g., confidence, consistency) and q_{t+1} the reformulated query. This loop continues until r_t exceeds a threshold τ. The result is reduced hallucination, improved factual grounding, and systematic convergence toward higher accuracy [52]. Evaluator–optimizer loops formalize the idea of agentic self-correction within graph-based reasoning.

F. Additional Workflows: Self-Memory and Tool Invocation

Emerging GA-RAG workflows incorporate persistent self-memory and external tool invocation. In self-memory systems, intermediate traversals and partial answers are summarized and stored in episodic memory M_t, which is reused across future reasoning steps:

$$M_t = \text{Update}(M_{t-1}, E_t),$$

thus enabling long-range coherence and consistency across multi-turn interactions [52], [53]. Tool-augmented workflows allow agents to call external APIs (e.g., biomedical databases, code analyzers, or numeric solvers), treating them as graph-linked functions $\phi : G \to \mathbb{R}^d$ [54], [55]. These extensions broaden GA-RAG beyond textual and symbolic reasoning, opening applications to multimodal and computational domains.

Table III summarizes six GA-RAG workflows, highlighting graph usage, agent capabilities, strengths, applications, limitations, to guide framework selection effectively.

V. Taxonomy of Graph-Based Agentic RAG Systems

We present a unified taxonomy of Graph-Based Agentic Retrieval-Augmented Generation (GA-RAG) systems that organizes the diverse landscape of existing designs across three orthogonal axes: reasoning workflows, agent orchestration, and adaptation mechanisms. This framework offers a comprehensive lens to understand the structural, collaborative, and dynamic facets that underpin GA-RAG innovations.

Reasoning workflows capture the structural patterns by which systems traverse, retrieve, and compose evidence over knowledge graphs, all are best described in Section IV. Agent orchestration describes how autonomous agents coordinate retrieval, reasoning, and generation tasks, ranging from monolithic controllers to hierarchical multi-agent ensembles. Adaptation mechanisms characterize how GA-RAG models incorporate feedback, self-correction, and continual learning to evolve and maintain robustness in dynamic environments.

Table IV summarizes key categories and representative methods across these axes, grounding the taxonomy in prevailing literature.

A. Reasoning Workflows

Reasoning workflows define the trajectories through which GA-RAG systems navigate graph knowledge to synthesize answers. Chain-based workflows operate in a strictly sequential fashion, retrieving and appending evidence stepwise through local neighborhoods. This approach, exemplified by IRCoT and Graph-Rerank, leverages extended chain-of-thought (CoT)

Workflow	Key Idea	Graph Use	Agentic Functionality	Strengths/Applications	Limitations
Prompt Chaining [22], [23], [30]	Decomposes reasoning into sequential LLM prompts aligned with graph operations.	Graph-based multi-hop traversal (node-edge hops for sequential reasoning)	Sequential decomposition of tasks into chained prompts for modular, explainable reasoning	Effective for compositional/multi-hop QA, structured reasoning; multi-step QA, explainability	Prompt length grows; error propagation risk.
Routing (Hybrid Retrieval) [31], [34], [42]	Chooses between vector, graph, or hybrid retrieval based on query type.	Dynamic switching between vector embeddings and graph-indexed retrieval for adaptive query resolution	Policy-driven routing that selects between vector vs. graph retrieval modes based on query type	Flexible, balances recall (vector) and precision (graph); versatility, SOTA accuracy	Needs routing policy; hybrid strategies increase latency.
Parallel Graph Exploration [25], [44], [48]	Multiple agents explore distinct graph regions simultaneously.	Concurrent exploration of disjoint graph regions or entities by multiple agents	Parallel multi-agent retrieval over graph segments to speed up exploration and improve robustness	Near-linear scalability, robust against noise/sparsity; scale, robustness, speed	Requires aggregation of partial results; redundancy possible.
Orchestrator–Worker [32], [51], [52]	Central orchestrator decomposes query into subgoals; workers handle subgraph tasks.	Hierarchical graph decomposition: orchestrator splits tasks into subgraphs and assigns them to worker agents	Central planner recursively breaks tasks into subgoals and delegates them to specialized workers for execution	Modular, recursive, inspired by symbolic AI; modularity, code/expert agent blends	High coordination overhead; non-trivial merging of outputs.
Evaluator–Optimizer [31], [45], [52]	Iterative refinement loop: evaluator critiques outputs, optimizer adjusts strategy.	Iterative refinement of graph context or subgraphs using evaluator feedback	Self-critique loops that iteratively refine queries and answers for correctness and completeness	Reduces hallucination, improves factuality; reliability, fact-checking	Slower; depends on evaluator accuracy.
Self-Memory & tool use [53], [54]	Agents store intermediate graph summaries and call domain-specific APIs/tools.	Episodic graph-based memory and tool-enhanced retrieval for persistent context maintenance	Long-term retention of knowledge and use of external tools (e.g., graph query APIs) to augment retrieval and generation	Long-term reasoning, multimodal/data integration; persistent sessions, multimodal queries	Storage overhead; external tools increase complexity.

TABLE III: Comparison of graph-based agentic RAG workflows with integrated features.

prompting adapted to graph structures. While effective in guiding stepwise compositional reasoning, this paradigm is limited by exposure bias and the accumulation of errors over sequential stages [16], [56].

Tree-based workflows address this limitation by branching the search process into multiple concurrent paths over the graph, often powered by Monte Carlo Tree Search (MCTS) or parallel agents. Methods such as MCTS-RAG and Jeong et al.'s study apply lookahead and pruning strategies to balance exploration and exploitation for multi-hop question answering [20], [41].

Beyond linear and tree-like traversals, graph-native workflows exploit the global structure of the knowledge graph holistically. Walk-on-Graph strategies utilize graph neural networks (GNNs) for message passing to rank nodes and identify salient subgraphs. Meanwhile, Think-on-Graph approaches embed agent-driven traversal planning where edge selection follows learned or adaptive policies instead of heuristics. These methods, including QA-GNN and GraphAgent, provide more robust support for compositional and multi-relational reasoning [48], [57].

- **Axis I: Reasoning Workflows** Describes how evidence is structurally retrieved and composed over graphs.
  - *Chain-Based*: Sequential prompt chaining sequences (e.g., IRCoT [16], Graph-Rerank [56]).
  - *Tree-Based*: Branching, multi-path exploration (e.g., MCTS-RAG [20], Jeong et al. [41]).
  - *Graph-Native*: GNN-driven global traversal and subgraph ranking (e.g., QA-GNN [57], GraphAgent [48]).

B. Agent Orchestration

Agent orchestration governs the collaborative dynamics through which autonomous entities execute GA-RAG retrieval, reasoning, and generation tasks. Single-agent models adopt a centralized, monolithic controller orchestrating both vector and graph retrieval alongside LLM synthesis. Early Graph-RAG frameworks such as GRAG and ReAct fall into this paradigm, favoring simplicity yet encountering limitations in scalability and specialization when addressing complex queries [27], [31].

Centralized multi-agent architectures advance this by deploying manager–worker structures, where a central orchestrator decomposes queries into subtasks that specialized agents handle in parallel. Frameworks including HM-RAG and ARAG demonstrate enhanced robustness and throughput through such coordination [44], [58].

Decentralized multi-agent systems eliminate a single point of control, relying on peer-to-peer communication and shared graph memories to enable scalable, fault-tolerant collaboration. Prominent examples are Agentic Reasoning frameworks and Self-Organizing Knowledge Graphs, which successfully scale horizontally across distributed settings [49].

Hybrid orchestration patterns integrate hierarchical and reflective principles. Tiered agent communities decompose

tasks top-down—from high-level planners to evidence synthesizers—while reflective agents continuously critique outputs and guide on-the-fly reformulations and expansions. Recent advances by Lelong et al., Pan et al., Liang et al., and Singh et al. illustrate the efficacy of such designs in improving multi-hop reasoning and adaptability [21], [22], [50], [55].

- **Axis II: Agent Orchestration Patterns** Specifies how agents collaborate to execute retrieval and reasoning.
  - *Single-Agent*: Monolithic control (e.g., GRAG [31], ReAct [27]).
  - *Centralized Multi-Agent*: Manager–worker hierarchies (e.g., HM-RAG [58], ARAG [44]).
  - *Decentralized Multi-Agent*: Peer-to-peer collaboration (e.g., Agentic Reasoning [25], Self-Organizing KGs [49]).
  - *Hierarchical & Reflective*: Tiered agents with feedback loops (e.g., Lelong et al. [50], Liang et al. [22]).

C. Adaptation Mechanisms

Adaptation mechanisms embody GA-RAG systems' capability to self-correct, learn, and evolve. Static systems follow fixed retrieval and reasoning pipelines without feedback, yielding simplicity but reduced robustness [12].

Corrective systems embed evaluator agents that assess the validity of retrieved evidence and generated answers, issuing refinements via query reformulation or traversal corrections. Such feedback loops effectively reduce hallucinations and enforce factual consistency [22], [56].

Adaptive systems go further, incorporating continual learning frameworks updating graph embeddings, retrieval strategies, and traversal policies online. This dynamic updating is critical in domains with evolving data, ensuring up-to-date knowledge representations and retrieval effectiveness, as showcased in lifelong GA-RAG architectures [49], [59].

- **Axis III: Adaptation Mechanisms** Determines how systems evolve and self-correct.
  - *Static*: Fixed pipelines without feedback.
  - *Corrective*: Evaluation and refinement loops (e.g., Graph-Validation agents [22], [56]).
  - *Adaptive*: Continual learning and online graph updates (e.g., Lifelong agents [49], [59]).

D. Hybrid Variants

Many contemporary GA-RAG frameworks seamlessly cut across these taxonomy axes, integrating multiple reasoning styles, orchestration patterns, and adaptation strategies for enhanced flexibility and efficacy.

Notable examples include:

- **Agent-G**: Incorporates a retriever bank aggregating text and graph sources, coupled with a critic agent providing answer validation and adaptive routing [60].
- **GeAR**: Embeds lightweight graph expansion modules into agentic prompt chains, enabling any base retriever to leverage structured subgraphs efficiently [61].
- **FRAG**: Modularizes retrieval, reasoning, and generation into independent plugins orchestrated by agents, supporting dynamic evidence synthesis [42].
- **SuperRAG**: Extends agentic RAG to multimodal and layout-aware graphs, enabling reasoning over varied document types and visually grounded information [62].

These hybrid variants represent the cutting edge of GA-RAG research, advancing towards adaptive, scalable, and multimodal retrieval-augmented generation systems.

This unified, carefully structured taxonomy offers readers a coherent narrative and comprehensive classification framework, aiding understanding and comparative analysis of GA-RAG methodologies.

E. Performance Evaluation and Comparative Analysis

To systematically compare GA-RAG variants, we evaluate them across four critical dimensions: multi-hop reasoning accuracy, workflow transparency, adaptability, and computational complexity. Furthermore, we contrast industry deployments with academic prototypes to highlight divergent priorities and practical considerations. Now we have categorized the Evaluation Dimensions as,

*a) **Multi-Hop Reasoning Accuracy**:* Measures the ability to synthesize information over multiple graph traversals. Chain-based methods (IRCoT [16], Graph-Rerank [56]) achieve 65–75% on HotpotQA; tree-based approaches (MCTS-RAG [20], Jeong et al. [41]) improve to 75–85%; graph-native workflows (QA-GNN [57], GraphAgent [48]) reach 80–90%.

*b) **Workflow Transparency**:* Assesses explainability of reasoning chains. Single-agent systems (ReAct [27], GRAG [31]) yield high transparency due to monolithic logs. Centralized multi-agent frameworks (HM-RAG [58], ARAG [44]) maintain moderate clarity via subgoal traces. Decentralized designs (Agentic Reasoning [25], Self-Organizing KGs [49]) face lower scores, mitigated by reflective loops (Liang et al. [22], Schneider et al. [52]).

*c) **Adaptability**:* Evaluates resilience to graph evolution and domain shifts. Static pipelines (traditional RAG [12]) falter under drift. Corrective loops (Feedback-Driven Reformulation [22], [56]) yield 15–25% gains in cross-domain tasks. Adaptive methods (Lifelong Agents [49], Zhang et al. [59]) preserve 85–95% of in-domain accuracy on new domains.

*d) **Computational Complexity**:* Analyzes resource and latency trade-offs. Chain-based workflows scale as $O(k)$ for k-hop queries [16]. Tree-based methods incur $O(b^d)$ cost for branching factor b and depth d [20]. Graph-native GNNs scale as $O(|V|+|E|)$ [48], [57]. Multi-agent coordination overhead is $O(n \log n)$ for centralized and $O(n^2)$ for decentralized communication [25], [50].

F. Industry vs. Research Perspectives

a) Industry Perspective: Focuses on deployment feasibility, scalability, and cost-effectiveness. Microsoft GraphRAG (Azure Graph + GPT-4) achieves 82–84% on HotpotQA with sub-second latencies [34], [47]. Neo4j+LangGraph attains 75–78% accuracy with subgraph caching and optimized Cypher traversals [40], [45]. Amazon Neptune+Bedrock reports 77–80% multi-hop QA accuracy and robust enterprise integration via Lambda agents [54], [55]. Enterprises prioritize

Axis & Category	Characteristics	Example Methods	Representative Works
Axis I – Reasoning Workflows			
Chain-Based	Linear, CoT-style prompt sequences over graph hops	IRCoT; Graph-Rerank	[16], [56]
Tree-Based	Branching exploration with lookahead (MCTS)	MCTS-RAG; Jeong's Study	[20], [41]
Graph-Based	GNN-driven traversal and reranking	QA-GNN; GraphAgent	[48], [57]
Axis II – Agent Orchestration Patterns			
Single-Agent	Central planner, sequential control loop	ReAct; GRAG	[27], [31]
Centralized Multi-Agent	One orchestrator delegates to workers	HM-RAG; ARAG	[44], [58]
Decentralized Multi-Agent	Peer-to-peer coordination, no central controller	Agentic Reasoning; Self-Organizing KGs	[25], [49]
Hybrid Patterns – Hierarchical	Tiered agent stacks with top-down task split	Lelong et al.; Pan et al.	[50], [55]
Hybrid Patterns – Reflective	Integrates self-critique loops into agents	Liang et al.; Singh et al.	[21], [22]
Axis III – Adaptation Mechanisms			
Meta-Learning	Online tuning of policy over graph tasks	Meta-RAG; RapidKG	[49], [59]
Feedback-Driven Reformulation	Query rewrite via evaluator signals	EvalRequest; QueryRefine	[31], [56]
Memory	Episodic graph summarization and replay	Self-Memory; EpisodicKG	[53], [54]

TABLE IV: Taxonomy of graph-based agentic RAG systems, organizing existing methods across three axes with key characteristics, example methodologies, and representative citations.

end-user response time, system reliability, and total cost of ownership (infrastructure, maintenance, developer effort) over marginal accuracy gains [26], [63].

b) Research Perspective: Emphasizes theoretical innovation, benchmark performance, and methodological rigor. Academic prototypes are evaluated on standardized datasets (HotpotQA, MetaQA [23], [30]), focusing on algorithmic novelty (e.g., IRCoT, MCTS-RAG, QA-GNN) and controlled comparisons under fixed resource budgets [21], [22], [41]. Research metrics include compositional generalization, interpretability, and ablation studies rather than deployment constraints or cost analyses.

G. Framework Comparison on Benchmarks

This section evaluates eight representative GA-RAG frameworks on multi-hop QA accuracy, synthesis quality, and recommended use cases. Table V presents a concise comparison.

To summarize, Microsoft GraphRAG leads in accuracy and low-latency QA, while Neptune+Bedrock excels in tool-augmented synthesis. Open-source frameworks offer flexibility (LangGraph, FRAG) or lightweight deployment (LlamaIndex), and QA-GNN pipelines deliver strong explainability for academic research. Continuous benchmarking remains essential to account for evolving datasets and system improvements [9], [64].

VI. Applications and Use Cases

Graph-Based Agentic RAG systems empower diverse sectors by combining structured knowledge graphs with autonomous workflows to enhance reasoning, personalization, and reliability [34], [52]. Their adaptability facilitates domain-specific problem solving across complex multi-hop tasks.

A. Customer Support and Virtual Assistants

In customer support, GA-RAG leverages product knowledge graphs and conversation logs to decompose and resolve multi-step queries like "device setup" and "network error." Agentic orchestrators dynamically traverse graph relations, achieving 25–30% faster resolutions compared to flat RAG approaches. Reflective agents reduce escalations by validating solutions against historical data [34], [54].

B. Healthcare and Personalized Medicine

Healthcare applications utilize AMG-RAG for synthesizing patient data, biomedical literature, and clinical guidelines. Graph-enhanced retrieval uncovers multi-hop symptom-biomarker-drug interactions, while planner agents tailor recommendations to patient profiles. This approach reports 15–20% gains in diagnostic accuracy and supports clinician-in-the-loop safety through evaluative feedback [52], [55].

Framework	Backend & Retrieval	HotpotQA (%)	Use Case
Microsoft GraphRAG [34], [47]	Azure Graph + GNN reranker + dense retriever	82–84	Enterprise-grade, low-latency multi-hop QA
Neo4j+LangGraph [40], [45]	Neo4j property graph + FAISS + Cypher policies	75–78	Rapid prototyping on property graphs
Amazon Neptune+Bedrock [54], [55]	Neptune SPARQL + hybrid retrieval via Lambda agents	77–80	Large-scale enterprise integration with tool agents
LangGraph OSS [48], [49]	Pluggable graph stores + dynamic routing	78	R&D with custom graph backends
Llama-Index OSS [21], [51]	In-memory graph module + vector index	72	Lightweight, local graph QA prototypes
GraphRAG Hybrid [20], [34]	Text+graph hybrid retrieval + reranking	70–75	Baseline research comparisons
QA-GNN Pipeline [23], [30]	GNN-based reranking + LLM synthesis	80–85	Explainable multi-relational QA research
FRAG Modular RAG [42], [52]	Modular plugins (retrieval, GNN, generation)	78–82	Flexible pipeline experimentation

TABLE V: Comparison of GA-RAG frameworks on multi-hop QA accuracy, with recommended use cases.

C. Legal and Contract Analysis

Legal domains exploit graph-structured statutes and precedent cases for complex contract parsing. Agentic prompt chaining extracts obligations and risk clauses, with corrective agents ensuring compliance against regulatory ontologies. This yields 90% precision in clause classification, outperforming traditional keyword methods by 35% [62], [65].

D. Finance and Risk Assessment

GA-RAG integrates corporate hierarchies, transactions, and market signals to detect fraud and assess credit risk. Orchestrator–worker models decompose due diligence into specialized subtasks coordinated via shared graph memory. Evaluator–optimizer loops iteratively refine risk scores, enhancing early fraud detection by 22% [22], [44].

E. Multimodal Graph-Enhanced Workflows

Emerging systems integrate multimodal knowledge graphs combining text, images, and videos. Platforms like CollEX facilitate interactive scientific exploration via citation graph navigation and embedded visual summaries. Multimodal evaluators cross-validate hypotheses with experimental data, advancing research discovery [52].

VII. Benchmarks, Datasets, and Evaluation

A. GraphQA, WebQSP, ExplaGraphs, HotpotQA

The evaluation of Graph-Based Agentic RAG systems primarily relies on established benchmarks that test multi-hop reasoning capabilities, factual accuracy, and explanation quality.

HotpotQA is the most widely used benchmark for multi-hop question answering, comprising 113k Wikipedia-based question-answer pairs that require reasoning across multiple documents. GraphRAG systems improve accuracy by 15–25% compared to traditional RAG on this benchmark [46], [64].

WebQSP tests knowledge base question answering over Freebase subgraphs, emphasizing entity linking and multi-hop traversal. GraphRAG approaches outperform flat retrieval on relational queries [20].

ExplaGraphs emphasizes explanation generation for graph-based reasoning, with graph-aware agents producing more coherent explanations than traditional RAG [47].

System (Reference)	HotpotQA F1	WebQSP Acc	Latency (ms)
Traditional RAG [12]	58.2%	42.1%	850
GraphRAG [34]	73.5%	58.9%	1200
Agentic GraphRAG [46]	79.3%	64.2%	1450
GeAR [61]	81.7%	67.8%	1380

TABLE VI: Comparison of GA-RAG systems on HotpotQA and WebQSP benchmarks, with latency trade-offs.

B. Evaluation Metrics: Accuracy, Hit@1, F1, Latency

Key metrics for assessing GraphRAG performance include:

- **Accuracy:** Exact match rates, typically 65–85% for state-of-the-art systems.
- **Hit@1:** Retrieval effectiveness measured by the correct answer appearing in top-ranked subgraphs.
- **F1 Score:** Balances precision and recall, particularly for multi-entity answers.
- **Latency:** End-to-end response time; graph-based retrieval adds roughly 200–500ms overhead versus vector-only retrieval.

Table VI highlights accuracy improvements and latency considerations across leading GA-RAG systems, illustrating the balance between enhanced reasoning and response efficiency.

C. Hallucination Assessment and Human-in-Loop Validation

Specialized evaluations include automated fact-checking against graph constraints, human verification of reasoning chains, and iterative evaluator-agent corrections to minimize hallucinations [9], [22].

VIII. Challenges and Future Directions

Graph-Based Agentic RAG systems raise several tight, practical limits and research opportunities; below we summarize each in two short sentences (problem first, actionable directions second).

A. Scalability and latency in large-scale graphs

Large knowledge graphs and subgraph-based retrieval substantially increase query runtime and tail latency because subgraph construction and multi-step retriever–generator loops add overhead and coordination costs [34], [66].
Promising responses include learned indices, approximate subgraph-selection with bounded-error guarantees, agent-aware caching and sharding, and asynchronous/streaming retrieval pipelines that trade small, provable accuracy loss for large latency gains [66].

B. Knowledge quality, completeness and consistency

Real-world graphs are incomplete, temporally drifting and sometimes contradictory, and agentic iterative reasoning can amplify such errors across multi-hop paths [49], [67].
Feasible research directions are online graph validation and uncertainty-aware retrieval, agent-driven evidence acquisition to fill gaps, and integrating self-repair loops (agentic verification + human-in-the-loop confirmation) to limit error propagation [49].

C. Multi-modal and causal reasoning

Current GraphRAG work is heavily text-centric while practical domains demand cross-modal linking (images, sensor streams) and support for interventional, counterfactual queries [52].
Consolidating multimodal node/edge schemas, building cross-modal embedding and alignment layers, and adding explicit causal edges or counterfactual modules (so agents can perform 'do'-style interventions) are practical next steps [52].

D. Neural–symbolic integration and differentiable graph programming

Loosely coupled neural and symbolic modules yield brittle pipelines; the field needs tighter, end-to-end differentiable hybrids that preserve logical constraints while retaining neural flexibility [23], [30], [43].
A concrete path is pretraining graph foundation encoders (GFM-style) that expose differentiable constraint modules and symbolic executors (rule-checkers) as callable components during training and inference to enforce consistency and improve faithfulness [43].

E. Federated, privacy-preserving and edge-enabled operation

Distributed, cross-silo deployments require privacy guarantees while agents need to reason across partial local graphs without leaking sensitive data [68], [69].
Practical directions include federated GFM training, secure multi-party query protocols for subgraph aggregation, lightweight on-device graph compression, and differentially-private retrieval mechanisms that balance utility and privacy.

F. Bias, explainability and trust

Knowledge graphs carry upstream biases that agentic selection and multi-hop inference can amplify, undermining fairness and trust in high-impact domains [9], [70].
Short-term fixes are provenance-aware retrieval (traceable paths), calibrated uncertainty scores for outputs, and integrated human validation workflows; longer-term work should measure and mitigate KG lifecycle biases at ingestion and embedding stages [9].

G. Temporal dynamics and evolving graphs

Most systems assume static graphs, but facts and relations change continuously and agents must reason about temporal dependencies and event sequences [71].
Adopt time-aware graph representations, time-conditioned retrievers (retrieve temporally relevant subgraphs), and agent protocols that version and roll back updates so reasoning is time-consistent and auditable [71].

H. Foundations, convergence and verifiability

Empirical progress outpaces theory: we need models of retrieval complexity, convergence guarantees for agentic loops, and verification methods for multi-agent reasoning chains [72], [73].
Research should produce formal abstractions of agentic GraphRAG (information-theoretic retrieval bounds, compositional verification of reasoning traces) and practical proof-carrying evidence (verifiable subgraph proofs) for critical applications.

IX. Conclusion

Graph-Based Agentic Retrieval-Augmented Generation (GA-RAG) represents a convergence of structured knowledge graphs, autonomous multi-agent reasoning, and large-scale language models. This survey has mapped the evolution of the field, from traditional RAG pipelines toward graph-enhanced and fully agentic frameworks, while organizing existing work into a taxonomy of reasoning workflows, orchestration patterns, and adaptation mechanisms. Our synthesis highlights that graph-grounded agency not only advances multi-hop reasoning accuracy and interpretability, but also unlocks broad applicability across domains such as scientific discovery, enterprise knowledge management, and personalized dialogue.

At the same time, the trajectory of GA-RAG research exposes persistent challenges, scaling to web-scale knowledge graphs, balancing autonomy with reliability, and ensuring explainability and ethical deployment. Looking forward, the integration of neural and symbolic reasoning, continual graph learning, and multimodal agentic architectures offers a promising path toward generalist systems that can dynamically acquire, reason over, and communicate structured knowledge. By crystallizing current progress and future directions, this survey aims to provide both a reference point and a roadmap for advancing the next generation of graph-based agentic reasoning systems.

REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, pp. 5998–6008, 2017.

[2] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Saxena, S. Sharma *et al.*, "Language models are few-shot learners," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.

[3] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020, pp. 7871–7880.

[4] R. Thoppilan, D. De Freitas, J. Hall, N. Shazeer, B. Stepien, A. Chowdhery, S. Narang, M. Hahn, A. Levskaya, N. Gruver *et al.*, "Lamda: Language models for dialog applications," *arXiv preprint arXiv:2201.08239*, 2022.

[5] A. Smith and L. Zhang, "From pre-training scaling to test-time scaling," *arXiv preprint arXiv:2404.11234*, 2024.

[6] OpenAI, "Openai o1 technical report," *OpenAI Research*, 2024, available at https://openai.com/o1.

[7] D. AI, "Deepseek-r1: Enhancing reasoning in large language models," *DeepSeek Research*, 2024, https://deepseek.com/r1.

[8] M. Chen and A. Gupta, "Long chain-of-thought reasoning in llms," *NeurIPS 2023*, 2023.

[9] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung, "Survey of hallucination in natural language generation," *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–38, 2023.

[10] K. Singhal, S. Azizi, T. Tu, S. S. Mahdavi, J. Wei, H. W. Chung, N. Scales, A. Tanwani, H. Cole-Lewis, S. Pfohl *et al.*, "Large language models encode clinical knowledge," *Nature*, vol. 620, no. 7972, pp. 172–180, 2023.

[11] J. D. Choi and J. Zou, "Chatgpt is a multilingual and multifacetal resource for clinical legal education and pro bono services," *arXiv preprint arXiv:2304.03212*, 2023.

[12] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.

[13] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang, "Realm: Retrieval-augmented language model pre-training," in *International Conference on Machine Learning*. PMLR, 2020, pp. 3929–3938.

[14] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, and H. Wang, "Retrieval-augmented generation for large language models: A survey," *arXiv preprint arXiv:2312.10997*, 2024.

[15] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, "Dense passage retrieval for open-domain question answering," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 6769–6781.

[16] H. Trivedi, N. Balasubramanian, T. Khot, and A. Sabharwal, "Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions," *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pp. 2341–2352, 2022.

[17] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-RAG: Learning to retrieve, generate, and critique through self-reflection," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=VR1k0p85L6

[18] S. Yao, J. Zhao, D. Yu, N. Du, G. Durrett, Y. Lin, and Y. Tsvetkov, "ReAct: Synergizing Reasoning and Acting in Language Models," in *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

[19] Z. Jiang, Z. Wang, S. Cheng, F. Zhou, W. Wu, T. Gao, and J. Henderson, "FLARE: Active-retrieval-augmented generation," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023, pp. 6811–6826.

[20] B. Peng, Y. Zhu, Y. Liu, X. Bo, H. Shi, and C. Hong, "Graph retrieval-augmented generation: A survey," *arXiv preprint arXiv:2408.08921*, 2024.

[21] A. Singh, A. Ehtesham, S. Kumar, and T. T. Khoei, "Agentic retrieval-augmented generation: A survey on agentic rag," *arXiv preprint arXiv:2501.09136*, 2025.

[22] J. Liang, G. Su, H. Lin, Y. Wu, R. Zhao, and Z. Li, "Reasoning rag via system 1 or system 2: A survey on reasoning agentic retrieval-augmented generation for industry challenges," *arXiv preprint arXiv:2506.10408*, 2025.

[23] M. Yasunaga, H. Ren, A. Bosselut, P. Liang, and J. Leskovec, "Qa-gnn: Reasoning with language models and knowledge graphs for question answering," *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2741–2751, 2021.

[24] A. Ehtesham *et al.*, "Agentic retrieval-augmented generation: A survey on agentic rag," *arXiv preprint arXiv:2501.09136*, 2025.

[25] J. Wu, J. Zhu, and Y. Liu, "Agentic reasoning: Reasoning llms with tools for the deep research," *arXiv preprint arXiv:2502.04644*, 2025.

[26] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, M. Apanasovich, E. Krivosheev, V. Dibia, S. Usta, S. Reid *et al.*, "From local to global: A graph rag approach to query-focused summarization," *arXiv preprint arXiv:2404.16130*, 2024.

[27] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 824–24 837, 2022.

[28] A. Piktus *et al.*, "The importance of being recurrent for modeling hierarchical structure," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021.

[29] G. Izacard and E. Grave, "Leveraging passage retrieval with generative models for open domain question answering," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*, 2021.

[30] X. Zhang, A. Bosselut, M. Yasunaga, H. Ren, P. Liang, C. D. Manning, and J. Leskovec, "Greaselm: Graph reasoning enhanced language models," *International Conference on Learning Representations*, 2022.

[31] Y. Hu, Z. Lei, Z. Zhang, B. Pan, C. Ling, and L. Zhao, "Grag: Graph retrieval-augmented generation," *arXiv preprint arXiv:2405.16506*, 2024.

[32] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu *et al.*, "Autogen: Enabling next-gen llm applications via multi-agent conversation," *arXiv preprint arXiv:2308.08155*, 2023.

[33] S. Hong, X. Zheng, J. Chen, Y. Cheng, C. Zhang, Z. Wang, S. K. S. Yau, Z. Lin, L. Zhou, C. Ran *et al.*, "Metagpt: Meta programming for a multi-agent collaborative framework," *arXiv preprint arXiv:2308.00352*, 2023.

[34] H. Han, Y. Wang, H. Shomer, K. Guo, J. Ding, Y. Lei, M. Halappanavar, R. Rossi, S. Mukherjee, X. Tang *et al.*, "Retrieval-augmented generation with graphs (graphrag)," *arXiv preprint arXiv:2412.00000*, 2024.

[35] W. Shi *et al.*, "Replug: Retrieval-augmented black-box language models," *arXiv preprint arXiv:2301.12652*, 2023.

[36] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *International Conference on Learning Representations*, 2017.

[37] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2008.

[38] D. Vrandečić and M. Krötzsch, "Wikidata: A free collaborative knowledgebase," *Communications of the ACM*, vol. 57, no. 10, pp. 78–85, 2014.

[39] Z. Liu *et al.*, "Llm+p: Empowering large language models with optimal planning proficiency," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 2023.

[40] S. Mankari and A. Sanghavi, "Enhancing vector based retrieval augmented generation with contextual knowledge graph construction," in *2024 2nd DMIHER International Conference on Artificial Intelligence in Healthcare, Education and Industry (IDICAIEI)*. IEEE, 2024, pp. 105–109.

[41] C. Jeong, "A study on the implementation method of an agent-based advanced rag system using graph," *arXiv preprint arXiv:2407.19500*, 2024.

[42] Z. Gao, Y. Cao, H. Wang, A. Ke, Y. Feng, X. Xie, and S. K. Zhou, "Frag: A flexible modular framework for retrieval-augmented generation based on knowledge graphs," *arXiv preprint arXiv:2501.09957*, 2025.

[43] L. Luo, Z. Zhao, G. Haffari, D. Q. Phung, C. Gong, and S. Pan, "Gfm-rag: Graph foundation model for retrieval augmented generation," *arXiv preprint arXiv:2502.01113*, 2025.

[44] R. Maragheh, P. Vadla, P. Gupta, K. Zhao, and A. Inan, "Arag: Agentic retrieval augmented generation for personalized recommendation," *arXiv preprint arXiv:2506.21931*, 2025.

[45] Z. Zhu, T. Huang, K. Wang, J. Ye, and X. Chen, "Graph-based approaches and functionalities in retrieval-augmented generation: A comprehensive survey," *arXiv preprint arXiv:2504.10499*, 2025.

[46] Q. Zhang, S. Chen, Y. Bei, Z. Yuan, and H. Zhou, "A survey of graph retrieval-augmented generation for customized large language models," *arXiv preprint arXiv:2501.13958*, 2025.

[47] Y. Li, W. Zhang, Y. Yang, W. Huang, Y. Wu, and J. Luo, "Towards agentic rag with deep reasoning: A survey of rag-reasoning systems in llms," *arXiv preprint arXiv:2507.09477*, 2025.

[48] Y. Yang, J. Tang, L. Xia, X. Zou, Y. Liang, and C. Huang, "Graphagent: Agentic graph language assistant," *arXiv preprint arXiv:2412.17029*, 2024.

[49] M. J. Buehler, "Agentic deep graph reasoning yields self-organizing knowledge networks," *arXiv preprint arXiv:2502.09459*, 2025.

[50] J. Lelong, A. Errazine, and A. Blangero, "Agentic rag with knowledge graphs for complex multi-hop reasoning in real-world applications," *arXiv preprint arXiv:2507.16507*, 2025.

[51] Y. Cao, Z. Gao, Z. Li, X. Xie, and S. K. Zhou, "Lego-graphrag: Modularizing graph-based retrieval-augmented generation for design space exploration," *arXiv preprint arXiv:2412.00000*, 2024.

[52] F. Schneider, N. B. Ahmadi, N. B. Ahmadi, I. Vogel, M. Semmann, and C. Biemann, "Collex - a multimodal agentic rag system enabling interactive exploration of scientific collections," *arXiv preprint arXiv:2504.07643*, 2025.

[53] X. Liang, S. Niu, Z. Li, S. Zhang, S. Song, H. Wang, J. Yang, F. Xiong, B. Tang, and C. Xi, "Empowering large language models to set up a knowledge retrieval indexer via self-learning," *arXiv preprint arXiv:2405.16933*, 2024.

[54] I. Nelson and O. Andersson, "Enhancing code review at scale with generative ai and knowledge graphs: An agentic graphrag framework for enterprise code review," *DiVA portal*, 2025.

[55] T. Pan, W. Pu, L. Zhao, and R. Zhou, "Leveraging llm agents for automated optimization modeling for sasp problems: A graph-rag based approach," *arXiv preprint arXiv:2501.18320*, 2025.

[56] J. Dong, B. Fatemi, B. Perozzi, L. F. Yang, and A. Tsitsulin, "Don't forget to connect! improving rag with graph-based reranking," *arXiv preprint arXiv:2405.18414*, 2024.

[57] P. Anokhin, N. Semenov, A. Sorokin, D. Evseev, M. Burtsev, and E. Burnaev, "Arigraph: Learning knowledge graph world models with episodic memory for llm agents," *arXiv preprint arXiv:2407.04363*, 2024.

[58] L. Luo, Z. Zhao, G. Haffari, D. Q. Phung, C. Gong, and S. Pan, "Gfm-rag: Graph foundation model for retrieval augmented generation," *arXiv preprint arXiv:2502.01113*, 2025.

[59] R. Zhang, S. Tang, Y. Liu, D. Niyato, and Z. Xiong, "Toward agentic ai: generative information retrieval inspired intelligent communications and networking," *arXiv preprint arXiv:2502.16866*, 2025.

[60] M. Lee, Q. Zhu, C. Mavromatis, Z. Han, and S. Adeshina, "Agent-g: An agentic framework for graph retrieval augmented generation," *arXiv preprint arXiv:2412.00000*, 2024.

[61] Z.-H. Shen, C. Diao, P. Vougiouklis, P. Merita, S. Piramanayagam, D. Graux, D. Tu, Z. Jiang, R. Lai, Y. Ren *et al.*, "Gear: Graph-enhanced agent for retrieval-augmented generation," *arXiv preprint arXiv:2412.18431*, 2024.

[62] J. Yang, D.-K. Vu, M.-T. Nguyen, X.-Q. Nguyen, L. Nguyen, and H. Le, "Superrag: Beyond rag with layout-aware graph modeling," *arXiv preprint arXiv:2503.04790*, 2025.

[63] S. S. Srinivas, A. Das, S. Gupta, and V. Runkana, "Accelerating manufacturing scale-up from material discovery using agentic web navigation and retrieval-augmented ai for process engineering schematics design," *arXiv preprint arXiv:2412.05937*, 2024.

[64] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D. Manning, "Hotpotqa: A dataset for diverse, explainable multi-hop question answering," *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, 2018.

[65] J. Chen, H. Lin, X. Han, and L. Sun, "Ragas: Automated evaluation of retrieval augmented generation," *arXiv preprint arXiv:2309.15217*, 2023.

[66] J. Bai, Z. Wang, Y. Zhou, H. Yin, W. Fei, Q. Hu, Z. Deng, J. Cheng, T. ZHENG, H. T. Tsang *et al.*, "Top ten challenges towards agentic neural graph databases," *IEEE Data Eng. Bull.*, 2025.

[67] M. Cheng, Y. Luo, J. Ouyang, Q. Liu, H. Liu, and L. Li, "A survey on knowledge-oriented retrieval-augmented generation," *arXiv preprint arXiv:2503.10677*, 2025.

[68] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[69] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.

[70] A. Kraft, E. Hüllermeier, J. Büttner, S. Kühne, and F. Weitz, "Biaskg: A benchmark for upstream and downstream bias evaluation in knowledge graphs," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2022, pp. 1234–1248.

[71] Y. Wu, Y. Fang, and L. Liao, "Retrieval augmented generation for dynamic graph modeling," *arXiv preprint arXiv:2408.14523*, 2024.

[72] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. John Wiley & Sons, 2006.

[73] P. Stone and M. Veloso, "Multiagent systems: A survey from a machine learning perspective," *Autonomous Robots*, vol. 8, no. 3, pp. 345–383, 2000.