Adaptive Retrieval Augmented Generation with Graph Based Knowledge Structures and Multi Agent Orchestration: A Comprehensive Review and Proposed Framework

Dr. Ankita V Karale, Aryan Yadav, Ashirwad Sharma, Hari Darade, Gaurav Malviya, Prof. Anmol Budhewar
Department of Computer Engineering, Sandip Institute of Technology and Research Centre
Mahiravani Trimbak Road, Nashik – 422213, Maharashtra, India

Abstract: **Large Language Models have become integral to enterprise knowledge systems, yet their propensity to produce factually ungrounded outputs termed hallucinations remains a critical obstacle in regulated deployments. Retrieval Augmented Generation grounds model outputs in external knowledge, but conventional vector based pipelines exhibit severe limitations in multi hop reasoning and deterministic verification. While recent literature explores Graph based RAG and Agentic RAG as independent solutions, their isolated deployments fail to resolve the underlying tradeoffs between structural rigidity and non deterministic overhead. This paper presents a comprehensive literature review of the current RAG landscape, mapping the architectural evolution from static retrieval to dynamic multi agent orchestration. A rigorous comparative analysis evaluates these paradigms across retrieval adaptivity, multi hop capability, and hallucination mitigation. Through critical analysis, this work identifies persistent research gaps regarding graph construction noise, static retrieval policies, and the self correction paradox in LLMs. To address these limitations, this paper introduces a technically rigorous Proposed Framework: An Adaptive Agentic GraphRAG architecture. The proposed methodology leverages a LangGraph state machine to orchestrate planner and verifier agents across a hybrid vector graph backend, introducing extrinsic algorithmic verification to solve the self correction paradox. This paper serves as a theoretical foundation and architectural proposal intended for future experimental validation in enterprise environments.**

Index Terms: **Retrieval Augmented Generation, Knowledge Graphs, GraphRAG, LLM Hallucination, Multi Agent Systems, Agentic RAG, Trustworthy AI, Research Proposal**

I. INTRODUCTION

The integration of Large Language Models into enterprise workflows represents a paradigm shift in automated reasoning and knowledge retrieval. However, deploying these systems in high stakes environments such as healthcare, legal services, and financial compliance is heavily constrained by the hallucination problem [5]. Hallucinations occur when LLMs generate plausible but factually incorrect information that contradicts source material or introduces unverified claims [22]. Zhang et al. [16] document that state of the art LLMs exhibit hallucination rates exceeding 27% on complex multi step reasoning tasks.

To ground generative outputs in factual reality, Retrieval Augmented Generation was introduced [7], fundamentally shifting the architecture to a retrieve then generate pipeline. Standard RAG relies on dense vector embeddings and cosine similarity to retrieve relevant text chunks. While highly effective for semantic matching, vector retrieval operates in a geometric space devoid of logical structure [10]. Consequently, standard RAG struggles with multi hop reasoning answering queries that require chaining scattered facts across disparate documents [14].

Two distinct architectural advancements have emerged in the literature to solve this reasoning deficit. First, GraphRAG [8] integrates Knowledge Graphs to map explicit entity relationship triplets, enabling structural traversal. Second, Agentic RAG [1], [31] delegates retrieval and verification processes to autonomous, LLM powered agents capable of iterative reflection [28].

Despite these advances, real world deployment reveals severe friction. Graph extraction remains noisy [20], and autonomous agents frequently suffer from the self correction paradox an inability to reliably evaluate their own logic without external grounding [34]. Furthermore, executing uncoordinated multi agent workflows introduces immense computational overhead and non determinism [1].

Motivation and Objectives: This research is motivated by the critical need for a deterministic, trustworthy, and structurally capable RAG architecture for regulated enterprise environments. The objectives of this paper are to:

1) Conduct a comprehensive literature review of RAG architectures.
2) Perform a comparative and critical analysis of existing methodologies.
3) Identify unresolved research gaps in the intersection of KGs and multi agent systems.
4) Propose a novel, theoretically sound framework Adaptive Agentic GraphRAG designed to overcome these specific limitations.

The expected contribution of this paper is a detailed, citation backed architectural proposal that resolves the tension between graph based structural reasoning and agentic flexibility, setting the stage for future empirical validation.

II. Literature Review

The evolution of RAG architectures can be categorized into four distinct technical themes: Traditional Pipeline RAG, Graph Augmented RAG, Agentic Orchestration, and Hallucination Verification.

A. Traditional and Advanced RAG Pipelines

The foundational RAG architecture [7], [17], categorized as Naive RAG [11], combines a generator with a retriever using semantic similarity search. To improve precision, Advanced RAG [11] adds pre retrieval methods like Hypothetical Document Embeddings [27] and post retrieval reranking. While efficient for simple fact finding, these static, single pass pipelines cannot adapt to query complexity, limiting their use in dynamic, reasoning heavy tasks [14].

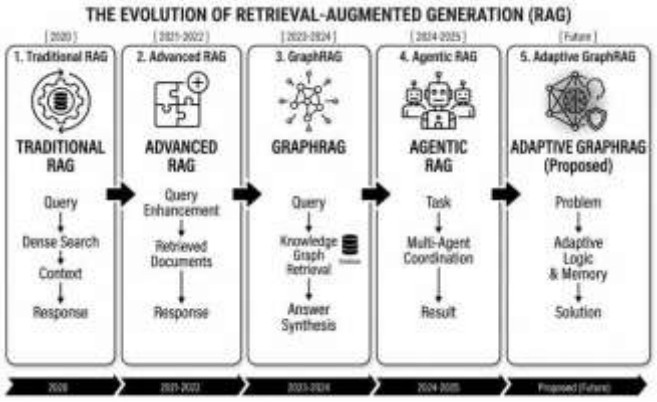


Fig. 1. Evolution of Retrieval Augmented Generation Architectures.

B. GraphRAG and Structural Knowledge Retrieval

To overcome vector space limits, Knowledge Graphs were integrated into RAG to store data as entity relationship triplets, enabling graph traversal queries [13]. Frameworks like Microsoft's GraphRAG [8] use community detection and map reduce to answer broad, corpus wide queries, excelling at multi hop reasoning and summarization. Similarly, QA GNN [18] and other hybrid approaches [20] successfully pair LLMs with Graph Neural Networks. However, these systems often suffer from extraction noise; LLMs can hallucinate false relationships that propagate through the graph and compromise its structural accuracy [8], [20].

C. Agentic Orchestration and Iterative Retrieval

Recent literature highlights a shift toward dynamic Agentic RAG [1], where autonomous LLM agents plan, retrieve, and reflect. Systems like Self RAG [28], Corrective RAG [29], and iterative retrieval with Chain of Thought approaches [14] significantly improve multi hop QA by dynamically triggering searches and evaluating confidence. Orchestration frameworks like AutoGen [31], CrewAI [32], and LangGraph [12] notable for modeling cyclic execution as a state machine enable these workflows. However, while agentic systems offer superior adaptivity, they suffer from nondeterminism, high token overhead, and complex coordination challenges [1].

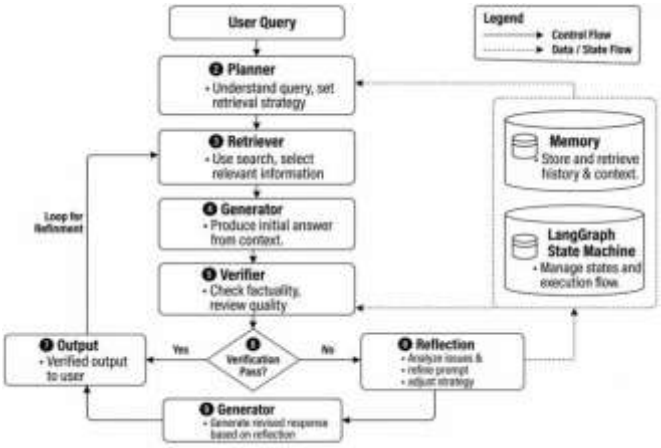


Fig. 2. Agentic GraphRAG Workflow with Verification and Reflection Loop.

D. Verification and the Self Correction Paradox

Strong verification is key to avoiding hallucinations. Although there are tools such as DSPy [23] for prompt optimization and frameworks such as RAGAS [9] and FActScore [25] as evaluation gates, LLMs cannot be their own verifiers. Huang et al. [34] established a correction paradox showing that LLMs cannot reliably self correct logical errors without external feedback. This is a serious limitation for agent architectures that rely on internal self reflection [33].

III. Comparative Analysis

This section presents a structured evaluation of the paradigms discussed in the literature review.

Analytical Discussion: Tables I and II illustrate a fundamental tradeoff in modern knowledge systems. Static vector architectures [7], [11] are highly scalable but computationally blind to logic. Conversely, GraphRAG [8] imposes logic but is computationally expensive to construct and query globally. Agentic systems [31] solve the adaptivity problem by orchestrating tools dynamically, but as shown in Table II, their reliance on intrinsic LLM judges exposes them to the self correction paradox [34]. The comparative data strongly indicates that a unified architecture must combine the structural determinism of KGs with the adaptive routing of agents, strictly gated by extrinsic algorithmic verification.

IV. Critical Analysis of Existing Literature

A critical evaluation of the literature reveals that isolated advancements in retrieval technology fail to solve the enterprise hallucination problem holistically.

The Failure of Static Retrieval: Lewis et al. [7] and Gao et al. [11] established vector based RAG as the industry standard. However, cosine similarity calculates spatial proximity in a latent space, which is fundamentally agnostic to causality, hierarchy, and logical dependency. When an enterprise query requires joining facts across documents e.g., matching a policy change in Document A to a patient record in Document B, vector retrieval fragments the context, leaving the LLM to hallucinate the missing links [14].

TABLE I
RETRIEVAL ARCHITECTURE AND SCALABILITY PROFILE

Approach	Knowledge Rep.	Multi Hop	Adaptivity	Comp. Cost	Scalability	Primary Strength	Primary Limitation
Traditional RAG [7]	Dense Embeddings	Weak	None	Low	High	Low latency, simple infrastructure	Fails cross document reasoning
Advanced RAG [11]	Embeddings + Metadata	Moderate	Preretrieval	Medium	High	Context precision	Rigid single pass pipeline
GraphRAG [8]	KGs Triplets	Strong	None	High	Moderate	Global summarization	Graph extraction noise
Self RAG [28]	Dense Embeddings	Moderate	Token triggered	Medium	High	Dynamic retrieval	Finetuned accuracy reliance
CRAG [29]	Embeddings + Web	Moderate	Binary Fallback	Medium	High	Recovers poor retrieval	Uncontrolled web data
IRCoT [14]	Dense Embeddings	Strong	Reasoning guided	Medium-High	Moderate	Complex QA gains	Cascading early errors
Agentic RAG [1], [31]	Flexible / Dual	Strong	Multi policy	Very High	Low-Moderate	Adaptive & self correct	High token cost

TABLE II
HALLUCINATION MITIGATION AND VERIFICATION MECHANISMS

Approach	Verification Method	Self Correction	Error Memory	Explainability	Key Vulnerability
Traditional RAG [7]	None	None	None	None	Blind to retrieval failures
GraphRAG [8]	None explicit	None	None	Path inspectable	Triplet hallucination systemic
CRAG [29]	Confidence Scorer	Partial Web fallback	None	Evaluator threshold	Same model verification bias
DSPy [23]	Prompt Compiler	Algorithmic tuning	None	Optimization trace	Metric overfitting
Agentic RAG [28], [31]	Algorithmic / Judge	Cyclic Iterative Loop	State based	Full agent trace	Self Correction Paradox [34]

The Graph Extraction Bottleneck: Edge et al. [8] demonstrated that GraphRAG significantly outperforms vector systems in comprehensiveness. Yet, critically analyzing their methodology reveals a heavy reliance on LLMs to perform the initial entity relationship extraction. Pan et al. [20] note that LLMs frequently hallucinate edges during this phase. Because the graph acts as the fundamental ground truth during retrieval, extraction noise is permanently codified into the system's architecture, undermining the entire premise of factual grounding.

The Complexity of Agentic Orchestration: The transition to Agentic RAG, championed by frameworks like AutoGen [31] and LangGraph [12], attempts to solve complex queries through workflow delegation. While Singh et al. [1] show that agents outperform fixed policies, the unconstrained nature of multi agent dialogue often leads to circular reasoning and infinite loops. Without a deterministic state machine to enforce execution boundaries, the computational cost token expenditure grows exponentially [12], rendering it unviable for high volume enterprise deployments.

The Verification Deficit: The most critical flaw in current agentic literature is the reliance on self reflection [33]. Huang et al. [34] empirically proved that LLMs struggle to detect their own reasoning errors. An agent that hallucinates a fact is highly likely to approve that same hallucination during its verification step. Therefore, agent architectures that do not incorporate extrinsic, deterministic verification scoring such as RAGAS [9] are fundamentally flawed for regulated applications.

V. RESEARCH GAP ANALYSIS

Based on the critical analysis, this paper identifies the following major, unresolved research gaps in the RAG literature:

Gap 1: Disjointed Modality Routing

- *Literature Origin:* Identified by Trivedi et al. [14] and Singh et al. [1].

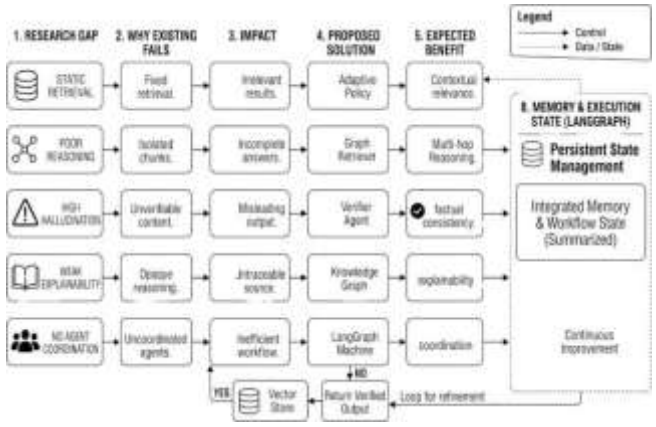


Fig. 3. Mapping of Research Gaps to Proposed Solutions in the Adaptive GraphRAG Framework.

- *Description:* Current systems force all queries through either a vector database [7] or a knowledge graph [8].
- *Consequence:* Simple queries waste massive compute traversing KGs, while complex queries fail due to vector fragmentation.
- *Research Opportunity:* There is a critical need for an intelligent policy selection module that dynamically routes queries to the optimal backend based on semantic complexity.

Gap 2: The Self Correction Paradox in Agent Workflows

- *Literature Origin:* Exposed by Huang et al. [34].
- *Description:* Agentic frameworks [28], [31] rely on intrinsic LLM self reflection to verify outputs.
- *Consequence:* Hallucinations bypass the verification stage because the LLM evaluator shares the same biases and blind spots as the LLM generator.
- *Research Opportunity:* The integration of extrinsic, non LLM algorithmic evaluators e.g., leveraging FActScore [25] or RAGAS [9] metrics directly into the agentic cyclic execution loop.

Gap 3: Unbounded Non Determinism in Multi Agent Orchestration

- *Literature Origin:* Discussed by Chase et al. [12] and Wu et al. [31].
- *Description:* Agent conversations in frameworks like AutoGen can easily derail, leading to context bloat and failure to terminate.
- *Consequence:* Unpredictable latency and computational costs that violate enterprise SLAs.
- *Research Opportunity:* Utilizing strict, state machine driven orchestration via Directed Acyclic Graphs or stateful cycles to enforce rigid execution paths and retry limits on autonomous agents.

Gap 4: Graph Construction and Temporal Invalidation Noise

- *Literature Origin:* Identified by Edge et al. [8] and Cai et al. [30].
- *Description:* KGs treat extracted knowledge as permanently true, ignoring the temporal decay of facts.
- *Consequence:* Obsolete information overrides current documents, leading to temporal hallucinations.
- *Research Opportunity:* Enhancing graph retrievers with temporal metadata awareness.

VI. Proposed Framework: Adaptive Agentic GraphRAG

Methodological Proposal: The following framework is a theoretical architectural design derived directly from the literature limitations identified in Section V. It constitutes the proposed methodology for future experimental implementation and validation.

To address the identified research gaps, this paper proposes Adaptive Agentic GraphRAG, a state machine orchestrated multi agent system operating over a hybrid knowledge backend. The framework completely decouples the generation of knowledge from its verification and dynamically adjusts its retrieval strategy based on query structure.

A. Design Objectives

Based on the literature review, the proposed framework is designed to satisfy the following objectives:

- Adaptive retrieval: Dynamically routing queries to the most efficient data modality.
- Reliable multi hop reasoning: Leveraging explicit graph structures rather than latent vectors for complex queries.
- Reduced hallucination: Enforcing extrinsic algorithmic verification.
- Explainability: Logging every step of the decision and retrieval process.
- Enterprise scalability: Preventing infinite loops and controlling computational overhead.
- Modular orchestration: Decoupling planning, generation, and verification into distinct state driven nodes.

B. Architectural Overview

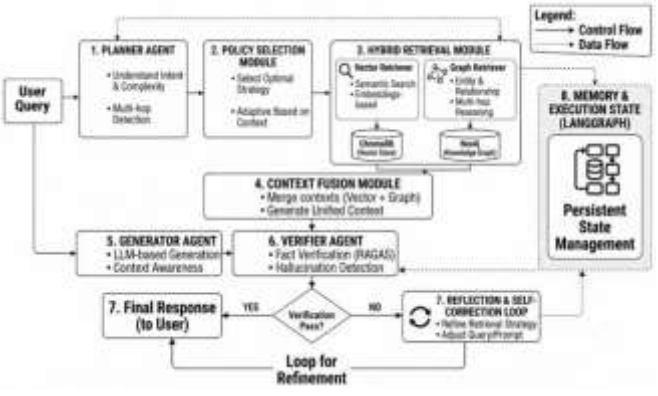


Fig. 4. Proposed Adaptive GraphRAG Framework with Multi Agent Orchestration.

The proposed framework is orchestrated using a strict LangGraph style state machine [12] to ensure deterministic

boundaries on agent execution. The architecture consists of three core phases: (1) Query Planning & Routing, (2) Hybrid Retrieval Execution, and (3) Generation & Extrinsic Verification.

C. Detailed Module Justification

1) User Query Interface:

- *Purpose:* Ingests the initial natural language prompt and maintains session context.
- *Addressed Limitation:* Semantic ambiguity in user inputs.
- *Literature Source:* Gao et al. [11].
- *Architectural Mechanism:* Implements pre retrieval query rewriting to expand acronyms and standardize terminology before hitting the agents.

2) Planner Agent:

- *Purpose:* Analyzes the rewritten query to break it down into atomic sub tasks.
- *Addressed Limitation:* Static, single pass pipeline failures on complex logic.
- *Literature Source:* Lewis et al. [7], Singh et al. [1].
- *Architectural Mechanism:* Utilizes Chain of Thought prompting [21] to explicitly map out the logical steps required to answer the query, executing Trivedi et al.'s [14] assertion that reasoning must precede retrieval.

3) Policy Selection Module:

- *Purpose:* Dynamically routes each atomic sub task to the optimal data modality.
- *Addressed Limitation:* The One Size Fits All modality routing.
- *Literature Source:* Pan et al. [20], Singh et al. [1].
- *Architectural Mechanism:* Acts as a classification router. If a sub task requires factual lookup e.g., What is the revenue, it routes to the Vector Retriever. If it requires relationship mapping e.g., Who owns the subsidiary that, it routes to the Graph Retriever. This optimizes computational cost.

4) Hybrid Retriever Vector + Graph Sub Modules:

- *Purpose:* Executes the retrieval queries against the underlying databases.
- *Addressed Limitation:* The tradeoff between semantic similarity and structural reasoning.
- *Literature Source:* Edge et al. [8], Lewis et al. [7].
- *Architectural Mechanism:* Vector Database & Retriever houses dense embeddings e.g., FAISS or Milvus for rapid semantic search. Knowledge Graph Database & Retriever houses Neo4j triplets. The Graph Retriever Cypher generation extract explicit subgraphs [13]. Incorporates temporal edge filtering to address Graph Quality issues identified by Cai et al. [30].

5) Generator Agent:

- *Purpose:* Synthesizes the retrieved context from both vector and graph sources into a coherent response.
- *Addressed Limitation:* LLM generation without structural context.
- *Literature Source:* Borgeaud et al. [17].
- *Architectural Mechanism:* Instructed strictly to ground its response only in the provided context window, utilizing the explicit logical paths provided by the Graph Retriever to construct its argument.

6) Verifier Agent & Reflection Loop:

- *Purpose:* Critiques the Generator's output and determines if a retry is necessary.
- *Addressed Limitation:* The Self Correction Paradox and intrinsic LLM bias.
- *Literature Source:* Huang et al. [34].
- *Architectural Mechanism:* The crucial Verifier Agent carries out extrinsic algorithmic evaluation rather than the usual LLM prompt. The agent carries out an automated test for the Faithfulness and Answer Relevance of RAGAS [9] to the source context. If the faithfulness measure falls below a certain threshold, a Reflection Loop is initiated.

7) Memory Module:

- *Purpose:* Stores the state of current iterations, retrieved context, and failed attempts.
- *Addressed Limitation:* Cyclic execution amnesia and infinite loops.
- *Literature Source:* Chase et al. [12], Madaan et al. [33].
- *Mitigation Strategy:* Maintains a strict log of previous retrieval paths to prevent the Planner Agent from repeating failed searches during a Reflection Loop.

8) LangGraph State Machine:

- *Purpose:* Orchestrates the entire workflow with enforced structural boundaries.
- *Addressed Limitation:* Unbounded non determinism in multi agent orchestration.
- *Literature Source:* Wu et al. [31].
- *Architectural Mechanism:* In this model, agents are represented by nodes Planner, Generator, Verifier while the transitions represent conditional edges in the deterministic graph. A maximum reflection depth limitation is set in order to guarantee reliability in the organization; failure of the Verifier continuously overrides the loop.

D. Mathematical Implementation Model

The mathematical architecture of the framework is governed by three primary equations that standardize query routing, temporal graph pruning, and extrinsic factual verification:

Query Complexity & Topology Routing Model: To dynamically route incoming queries without manual heuristics, the system calculates a multi hop complexity score $C(Q) \in [0, 1]$ using a logistic sigmoid transform over the embedded query vector $E(Q)$ combined with extracted entity depth E_Q:

$$C(Q) = \sigma(w_c^T \cdot E(Q) + b_c) \cdot \left(\frac{(E_Q \cdot depth_Q)}{(1 + log\ D_Q)}\right) \quad (1)$$

where:

- $C(Q) \in [0, 1]$ is the scalar query complexity score.
- $E(Q) \in R^d$ is the d-dimensional semantic dense query embedding vector.

- $w_c \in \mathrm{R}^d$ is a learnable projection weight vector.
- $(\cdot)^T$: Vector transpose operation converting weight w_c into a $1 \times d$ row vector.
- $b_c \in \mathrm{R}$ is a scalar bias term.
- $E_Q \in \mathrm{Z}^+$ is the count of explicit named entities detected in query Q.
- $\mathrm{depth}_Q \in \mathrm{Z}^+$ is the estimated relation traversal depth required.
- $D_Q \in \mathrm{R}^+$ is the local subgraph density around the detected entities.

The dynamic routing decision $P(Q)$ selects the retrieval backend based on threshold vectors τ_1, τ_2:

$$P(Q) = \begin{cases} Dense\ Vector\ Search & if\ C(Q) < \tau_1 \\ Knowledge\ Graph\ Traversal & if\ C(Q) \geq \tau_2 \\ Hybrid\ Fusion\ Search & if\ \tau_1 \leq C(Q) < \tau_2 \end{cases} \quad (2)$$

where:

- $P(Q)$ denotes the selected routing policy.
- τ_1 is the lower threshold isolating single hop lookups.
- τ_2 is the upper threshold triggering structural graph traversal.

Temporal Decay Graph Pruning Model: To mitigate graph extraction noise and temporal invalidation, each extracted knowledge graph edge $e = (u, v)$ is weighted using an exponential temporal decay function that scales structural confidence w_0 against age $t_0 - t_e$:

$$w(e,t) = w_0 \cdot exp(-\lambda(t_0 - t_e)) \quad (3)$$

where:

- $e = (u, v) \in E_G$: Directed edge connecting node u to node v in edge set E_G.
- $w(e, t) \in [0, 1]$ is the time decayed weight at current system execution time t_0.
- $w_0 \in (0, 1)$ is the initial extraction confidence weight.
- $\lambda > 0$ is the temporal decay rate constant.
- $t_0 \in \mathrm{R}^+$: Continuous epoch real timestamp at current system execution time.
- $t_e \in \mathrm{R}^+$: Continuous epoch real timestamp recording factual edge origin time.
- $(t_0 - t_e) \in \mathrm{R}_{\geq 0}$: Non negative real time elapsed since fact creation.
- $\tau_{\mathrm{decay}} \in (0, 1)$: Pruning scalar threshold below which weak edges are removed.

Edges falling below confidence threshold τ_{decay} are dynamically filtered out prior to response generation.

Composite Extrinsic Verification Score Model: To resolve the Self Correction Paradox, candidate generation Y is evaluated against context K using a weighted extrinsic composite matrix S_{total}:

$$S_{total} = \alpha \cdot S_{faith(Y,K)} + \beta \cdot S_{ans_rel(Y,Q)} + \gamma \cdot S_{temp(K)} \quad (4)$$

$$subject\ to\ \alpha + \beta + \gamma = 1$$

where:

- $S_{\mathrm{total}} \in [0, 1]$ is the composite factual verification score.
- $S_{\mathrm{faith}}(Y, K) \in [0, 1]$ measures programmatic context faithfulness via RAGAS and FActScore.
- $S_{\mathrm{ans_rel}}(Y, Q) \in [0, 1]$ measures answer relevance relative to prompt Q.
- $S_{\mathrm{temp}}(K) \in [0, 1]$ measures the average temporal validity of retrieved context K.
- $\alpha, \beta, \gamma \in [0, 1]$ are hyperparameter normalization coefficients.

E. Algorithmic Mechanics & Implementation Strategy

Rather than executing a rigid sequential pipeline, the framework is implemented as a stateful Graph Orchestration Engine leveraging LangGraph state machines.

State Engine & Graph Initialization: The core algorithm operates over a shared global state schema:

$S = \{Q_{\mathrm{rewritten}}, K_{\mathrm{fused}}, Y_{\mathrm{gen}}, S_{\mathrm{total}}, N_{\mathrm{retry}}\}$

Upon ingesting a user query, the state engine invokes pre retrieval normalization nodes before passing control to the Planner Agent.

Dynamic Modality Routing: The algorithm routes state transitions dynamically: queries classified under vector mode trigger dense semantic lookups via ChromaDB, graph queries execute parameterised Cypher queries across Neo4j, and hybrid queries combine results using Reciprocal Rank Fusion.

Extrinsic Gating & Reflection Mechanics: Response generation is passed to an extrinsic Verifier Node running programmatic RAGAS and FActScore evaluator suites.

If $S_{\mathrm{total}} \geq T_{\mathrm{verify}}$, the state machine terminates and emits the grounded answer. If verification fails, the Reflection Node updates the retry counter N_{retry}, appends negative feedback constraints to S, and loops back to the Planner Node. Bounded latency is guaranteed by hard stopping execution at $N_{\mathrm{max_retries}}$ and returning a deterministic fall back context.

VII. Expected Advantages of the Proposed Framework

While empirical validation is reserved for future work, the architectural design of the Adaptive Agentic GraphRAG framework is designed to offer several theoretical advantages over existing baseline systems [7], [8], [28]:

Deterministic Hallucination Mitigation: By explicitly addressing the self correction paradox [34] through the integration of extrinsic algorithmic gating Verifier Agent, the proposed framework aims to provide a reliable mechanism for trapping ungrounded claims before they reach the user.

Optimized Computational Efficiency: The Policy Selection Module prevents the framework from executing expensive graph community summarization [8] for simple semantic queries, reserving high cost operations only for multi hop necessity.

Bounded Agent Execution: The use of a strict state machine [12] ensures that enterprise Service Level Agreements regarding latency can be met, as the multi agent dialogue is structurally prevented from entering infinite hallucination loops [31].

Enhanced Explainability: Every step of the LangGraph execution from the Planner's CoT reasoning to the Graph Retriever's Cypher query and the Verifier's algorithmic score

is logged in the Memory Module. This provides a complete, auditable trace for enterprise compliance frameworks e.g., EU AI Act [38].

VIII. Conclusion

This paper is an attempt to discuss the development process of RAG technology starting from static vector pipelines up to dynamic self corrective multi agent GraphRAGs. The analysis of literature proves that neither the improvement of graphs nor agents separately solves the problem of nondeterminism, extraction noise, and self correctivily of LLMs.

To bridge these research gaps, this paper proposed the Adaptive Agentic GraphRAG framework. By integrating a hybrid vector graph retrieval backend with a state machine driven multi agent orchestrator, the proposed architecture dynamically routes queries to their optimal modality and enforces strict, algorithmic hallucination verification.

The proposed framework is intended for future implementation and experimental validation. This paper establishes the theoretical foundation, literature analysis, and architectural design necessary to construct next generation, trustworthy enterprise AI systems capable of deep reasoning without sacrificing deterministic safety.

Acknowledgments

The authors are grateful to Dr. Ankita V Karale for advising this review and to Prof. Anmol Budhewar for coordination support at the Department of Computer Engineering, Sandip Institute of Technology and Research Centre. This work is submitted toward the Bachelor of Engineering degree Computer Engineering, Academic Year 2026–27, under Savitribai Phule Pune University.

References

[1] A. Singh et al., "Agentic Retrieval Augmented Generation: A Survey on Agentic RAG," arXiv:2501.09136, Jan. 2025.
[2] P. Patil et al., "FinanceBench: A New Era to Evaluate Financial Context Aware Question Answering," arXiv:2311.11944, 2023.
[3] Y. Hu et al., "RAGTruth: A Hallucination Corpus for Developing Trustworthy Retrieval Augmented Language Models," Proc. ACL, pp. 1045-1060, 2024.
[4] M. Jin et al., "Search-R1: Reinforcement Learning for Agentic Retrieval Augmented Generation," arXiv:2501.12154, 2025.
[5] H. Alkaissi and S. I. McFarlane, "Artificial Hallucinations in ChatGPT: Implications in Scientific Writing," Cureus, vol. 15, no. 2, Feb. 2023.
[6] S. M. Tonmoy et al., "A Comprehensive Survey of Hallucination Mitigation Techniques in Large Language Models," arXiv:2401.01313, 2024.
[7] P. Lewis et al., "Retrieval Augmented Generation for Knowledge Intensive NLP Tasks," NeurIPS, vol. 33, pp. 9459–9471, 2020.
[8] D. Edge et al., "From Local to Global: A Graph RAG Approach to Query Focused Summarization," arXiv:2404.16130, Apr. 2024.
[9] S. Es et al., "RAGAS: Automated Evaluation of Retrieval Augmented Generation," arXiv:2309.15217, 2023.
[10] S. Ji et al., "A Survey on Knowledge Graphs: Representation, Acquisition, and Applications," IEEE Trans. Neural Networks and Learning Systems, vol. 33, no. 2, pp. 494–514, 2022.
[11] Y. Gao et al., "Retrieval Augmented Generation for Large Language Models: A Survey," ACM Computing Surveys, vol. 56, no. 7, Art. 182, pp. 1–24, May 2024.
[12] H. Chase et al., "LangGraph: Multi Agent Workflows," LangChain, 2024. [Online]. Available: https://github.com/langchain-ai/langgraph
[13] I. Robinson, J. Webber, and E. Eifrem, Graph Databases, 2nd ed. O'Reilly Media, 2023.
[14] H. Trivedi et al., "Interleaving Retrieval with Chain of Thought Reasoning for Multi Hop QA," Proc. 61st ACL, pp. 6012–6025, 2023.
[15] G. Mialon et al., "Augmented Language Models: A Survey of Mechanisms, Tools, and System Verification Layers," Trans. Machine Learning Research, vol. 2024, no. 3, pp. 412–445, Feb. 2024.
[16] Y. Zhang et al., "Siren's Song in the AI Ocean: A Survey on Hallucination in Large Language Models," Computational Linguistics, 2025.
[17] S. Borgeaud et al., "Improving Language Models by Retrieving from Trillions of Tokens," ICML, PMLR, pp. 2206–2240, 2022.
[18] M. Yasunaga et al., "QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering," Proc. NAACL, pp. 535–546, 2021.
[19] X. Wang et al., "Self Consistency Improves Chain of Thought Reasoning in Language Models," ICLR, 2023.
[20] S. Pan et al., "Unifying Large Language Models and Knowledge Graphs: A Roadmap for Hybrid Intelligence," IEEE Trans. KDE, vol. 36, no. 8, pp. 4012–4029, Aug. 2024.
[21] J. Wei et al., "Chain of Thought Prompting Elicits Reasoning in Large Language Models," NeurIPS, vol. 35, pp. 24824–24837, 2022.
[22] Z. Ji et al., "Survey of Hallucination in Natural Language Generation," ACM Computing Surveys, vol. 55, no. 12, pp. 1–38, Mar. 2023.
[23] O. Khattab et al., "DSPy: Compiling Declarative Language Model Calls into State of the Art Pipelines," ICLR, 2024.
[24] S. Wu et al., "Graph Neural Networks in Recommender Systems: A Survey," ACM Computing Surveys, vol. 53, no. 2, pp. 1–37, 2020.
[25] S. Min et al., "FActScore: Fine grained Atomic Evaluation of Factual Precision in Long Form Text Generation," Proc. EMNLP, pp. 12076–12091, 2023.
[26] N. F. Liu et al., "Lost in the Middle: How Language Models Use Long Contexts," Trans. ACL, vol. 12, pp. 157–173, 2024.
[27] L. Gao et al., "Precise Zero Shot Dense Retrieval without Relevance Labels (HyDE)," Proc. ACL, pp. 1762–1777, 2023.
[28] A. Asai et al., "Self RAG: Learning to Retrieve, Generate, and Critique through Self Reflection," ICLR, 2024.
[29] S. Yan et al., "Corrective Retrieval Augmented Generation (CRAG)," arXiv:2401.15884, 2024.
[30] C. Cai et al., "Temporal Knowledge Graph Completion: A Survey," arXiv:2201.08236, 2022.
[31] Q. Wu et al., "AutoGen: Enabling Next Gen LLM Applications via Multi Agent Conversation," arXiv:2308.08155, 2023.
[32] J. Moura, "CrewAI: Framework for Orchestrating Role Playing Autonomous AI Agents," GitHub/CrewAI, 2024.
[33] A. Madaan et al., "Self Refine: Iterative Refinement with Self Feedback," NeurIPS, vol. 36, 2023.
[34] J. Huang et al., "Large Language Models Cannot Self Correct Reasoning Yet," ICLR, 2024.
[35] S. Lin et al., "TruthfulQA: Measuring How Models Mimic Human Falsehoods," Proc. ACL, pp. 3214–3252, 2022.
[36] Z. Yang et al., "HotpotQA: A Dataset for Diverse, Explainable Multi hop Question Answering," Proc. EMNLP, pp. 2369–2380, 2018.
[37] NIST, "Artificial Intelligence Risk Management Framework (AI RMF 1.0)," National Institute of Standards and Technology, Jan. 2023.
[38] European Parliament, "Regulation (EU) 2024/1689: Artificial Intelligence Act," Official Journal of the European Union, Jul. 2024.