ZERO-LLM TRAJECTORY SEARCH OVER LINE-LIFTED KNOWLEDGE GRAPHS: INTERNALIZING QUERY DECOMPOSITION FOR MULTI-HOP RAG

Anonymous authors
Paper under double-blind review

ABSTRACT

Retrieval-augmented generation (RAG) has become a cornerstone of LLM applications, underpinning vertical-domain knowledge construction and agent memory. As applications demand complex relational reasoning, retrieval has turned to graphs, yet graph-based RAG still relies almost entirely on LLM calls, one or more per query, making latency and cost a deployment bottleneck, as if precise retrieval required LLM guidance. We propose LiftRAG, which represents every extracted relation as a graph vertex carrying its evidence-sentence embedding, turning the knowledge graph into its line graph. Retrieval is then a multi-round beam search over entity–relation trajectories, where each step decides the next hop from the frontier reached so far and query decomposition emerges from the search itself, LLM-free. On the entity-grounded test sets of three multi-hop benchmarks, LiftRAG leads the published baselines on all four metrics. The accuracy margins are 5.7–10.0 points over LightRAG, which shares our extraction and index, and 0.9–3.9 over the strongest baseline, HippoRAG 2, at 13–25$\times$ lower retrieval latency in API deployment (per-call latency measured on our own LLM relay). A residual 7–20% of LLM-decomposed sub-question trajectories falls outside the coverage of a single retrieval by LiftRAG. This work establishes *Zero-LLM Decomposition*, query decomposition performed by the retrieval process itself without any LLM call. The part of LLM decomposition it does not cover defines the open problem of fusing the two.

1 INTRODUCTION

Retrieval-augmented generation (RAG) (Lewis et al., 2020) has become a core component of vertical-domain knowledge systems and agent memory. Answering a query there rarely stops at one document. Facts lie scattered across the corpus and must be connected by following chains of relations, which similarity-based vector retrieval (Karpukhin et al., 2020) cannot do. Graph-based RAG therefore rose to dominance, building a corpus-wide graph of relations and traversing it at query time.

Research on graph RAG has mainly concentrated on two dimensions, construction and retrieval. On construction, GraphRAG (Edge et al., 2024) established the prototype, LLM-extracted entities as nodes and relational triples as edges merged into a corpus-wide graph. Later systems refined extraction and decorated the graph, but the entity-node/triple-edge scheme never changed. Since a triple, fixed in edge form, cannot be embedded and retrieved the way a node can, only entities are efficiently retrievable, and the relation semantics extracted at such cost are *buried*: stored as edge labels, never addressable by retrieval. On retrieval, innovation has advanced along three goals. HippoRAG 2 (Jiménez Gutiérrez et al., 2025) pursues multi-hop precision, spreading activation with personalized PageRank behind LLM fact retrieval; PathRAG (Chen et al., 2025) asks an LLM to score and prune candidate paths; LightRAG (Guo et al., 2024) pursues efficiency, extracting query keywords with a single call and merging graph- and passage-level evidence. A second line abandons the graph altogether. IRCoT (Trivedi et al., 2023) and Self-ASK (Press et al., 2023) spend three to eight LLM calls per question decomposing it into sub-queries. The goals differ, and the costs are

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

shared. The LLM is called on every query, to extract, select, score, or decompose, while the graph itself never decides where to go next; *the LLM is the engine of retrieval, and the graph is only the road it drives on*. An implicit consensus has thus emerged that *precise retrieval requires LLM guidance*.

We challenge this consensus on both sides at once (Fig. 1): *line lifting* makes every extracted relation a retrievable *vertex* carrying its evidence-sentence embedding, so relation semantics become a directly retrievable object, on a par with entities; and *zero-LLM trajectory search* runs the query as a multi-round beam over entity–relation trajectories, each round resolving one hop from the frontier the search has reached, so query decomposition is performed by the search itself. Our claim is strictly scoped to retrieval. Extraction remains LLM work at indexing time, a cost every baseline pays.

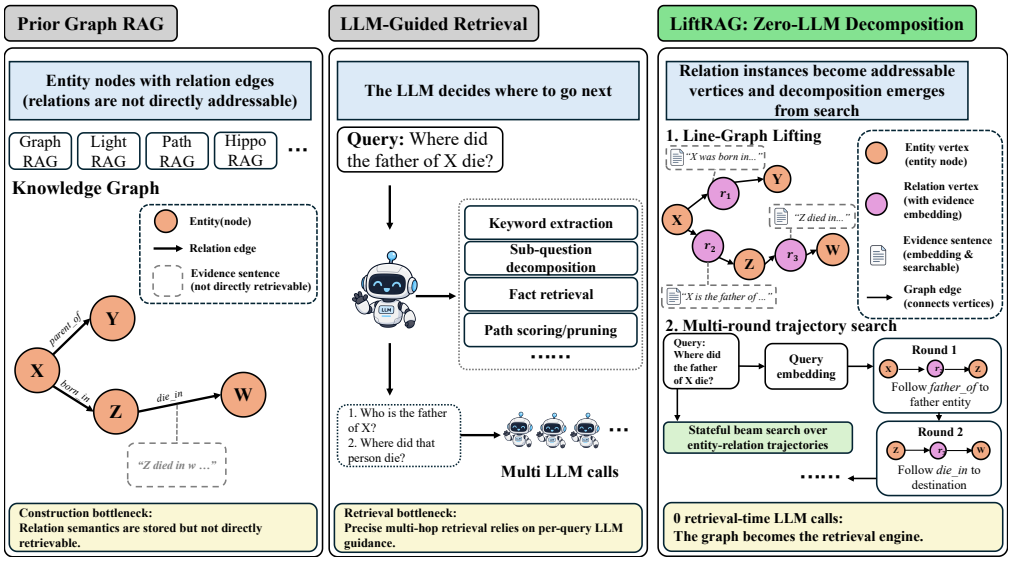

Figure 1: Motivation. Prior graph RAG stores relation semantics as edge labels never addressed by retrieval (left), while LLM-guided retrieval pays per-query calls for keywords, sub-questions, facts, and path scores (middle). LiftRAG lifts relation instances into vertices addressable by evidence-sentence embeddings and runs multi-round trajectory search, so decomposition emerges from the search at zero retrieval-time LLM calls (right).

We validate these designs with progressively tighter experiments. Full comparisons on three multi-hop benchmarks rule out a single-dataset accident. Swapping the reader between a 9B and a 27B model leaves the advantage unchanged. For the central claim that decomposition is performed by the search itself, we inject an LLM for reranking and decomposition and measure how much of it the search already covers.

This work establishes *Zero-LLM Decomposition*, the claim that decomposition can be carried out by the search itself rather than purchased from an LLM. Earlier pipelines, from Self-ASK to IRCoT, internalize decomposition in their loops yet pay an LLM call per sub-question. Here it is produced by the search dynamics at zero marginal cost, and this shift from purchased to emergent decomposition is what the term denotes. LiftRAG applies to today's vertical-domain knowledge systems and agent memory. The uncovered part defines the space for fusing the two. Our contributions are fourfold.

- **We propose line-lifted construction.** Every relation becomes a vertex carrying its evidence-sentence embedding, making triples addressable units in vector retrieval. Ablations confirm this representation is load-bearing exactly where multi-hop traversal is required.
- **We introduce zero-LLM trajectory search.** A multi-round beam search over entity–relation trajectories replaces per-query LLM guidance with fixed constants, achieving retrieval with zero LLM calls.

- **We validate on the entity-grounded test sets of three multi-hop benchmarks.** LiftRAG leads the published baselines on all four metrics. The margins are 5.7–10.0 accuracy points over LightRAG, which shares our extraction and index, and 0.9–3.9 over the strongest baseline, HippoRAG 2, at 13–25$\times$ lower retrieval latency in API deployment (per-call latency measured on our own relay).
- **We show that decomposition is internalized by the search.** Injecting an LLM for reranking or decomposition yields no net gain. The retrieval overlaps with LLM decomposition without coinciding with it, which establishes *Zero-LLM Decomposition* and leaves the fusion of LLM decomposition as its open problem.

2 Related Work

Graph-based RAG. For multi-hop questions whose answers span documents, graph-based systems augment flat corpora with knowledge graphs. GraphRAG (Edge et al., 2024) answers global questions with pre-computed community summaries. LightRAG (Guo et al., 2024) merges graph- and chunk-level evidence behind one keyword-extraction call. PathRAG (Chen et al., 2025) prunes traversal to high-value paths via LLM scoring. G-Retriever (He et al., 2024) prunes the graph into a question-guided subgraph. HippoRAG (Gutiérrez et al., 2024) runs personalized PageRank seeded by query entities, and HippoRAG 2 (Jiménez Gutiérrez et al., 2025) prepends LLM fact retrieval and a recognition-memory filter. They share a diagnosis: in every one of these systems, relations exist only as traversal structure, their semantics attached through entity-to-chunk associations, so relation-level evidence never re-enters retrieval as an addressable unit, and every query pays at least one retrieval-time LLM call. Outside retrieval, addressable relations have precedents. Knowledge-graph practice reifies statements as resources, as in RDF* (Hartig, 2017); hyper-relational embeddings such as StarE attach qualifiers to instances (Galkin et al., 2020); line-graph transformations elevate relations to nodes for message passing (Yao et al., 2025); link clustering makes edges the unit of analysis (Evans & Lambiotte, 2009; Ahn et al., 2010); and relation embeddings such as TransE map relation types to vectors (Bordes et al., 2013). Across these lines relations figure as typed abstractions, qualifier structures, or propagation targets, never as extracted instances whose evidence sentence is the retrieval key; LiftRAG makes them so, as addressable line-graph vertices navigated by a zero-LLM beam search.

Retrieval without LLM calls. MDR (Xiong et al., 2021) iteratively extends dense passage chains. SR (Zhang et al., 2022) learns to expand relation paths, and UniKGQA (Jiang et al., 2023a) unifies retrieval and reasoning across the knowledge base and text. GNN-RAG (Mavromatis & Karypis, 2024) ranks answer-bearing subgraphs with a trained graph network. SubgraphRAG (Li et al., 2025) scores triples in parallel with a trained MLP, and proposition-level retrieval (Chen et al., 2024) sharpens the retrieval unit on the same path. In all of them multi-hop behavior is carried by search structure rather than generated sub-questions, and every scorer is learned from annotated multi-hop QA, so each new domain re-enters the annotation loop graph RAG set out to remove. The closest of these, Beam Retrieval (Zhang et al., 2024), also runs a learned beam search over passage sets for multi-hop QA; LiftRAG differs in both the addressable unit and the scorer, navigating relation instances by their evidence-sentence embeddings under fixed constants. LiftRAG thus holds the training-free operating point over any extracted graph, the training-free test-time discipline of recent open-vocabulary perception (Qin et al., 2026).

LLM-guided retrieval and query decomposition. Self-ASK (Press et al., 2023) decomposes questions into sub-questions answered iteratively, and ReAct (Yao et al., 2023) interleaves reasoning with actions. IRCoT (Trivedi et al., 2023) alternates retrieval with reasoning steps. CoRAG (Wang et al., 2025) generates the missing information at each step and re-retrieves; ToG (Sun et al., 2024) and RoG (Luo et al., 2024) walk or plan relation paths on the knowledge graph under LLM guidance; Iter-RetGen (Shao et al., 2023), FLARE (Jiang et al., 2023b), and Self-RAG (Asai et al., 2024) interleave retrieval with generation, active querying, and self-critique. Beam-search variants scale this guidance with width and depth (Snell et al., 2025), and LLMs are also spent on evidence reranking and path scoring. These systems report strong multi-hop gains, but the gains are entangled with their retrievers. When the next hop depends on which entities earlier rounds actually reached, static decomposition adds nothing the search does not already generate; the benefit tracks the retrievers' single-step structure, not the LLM call itself.

Decomposition has a longer history independent of LLMs, from handwritten rules and semantic parsers (Ferrucci et al., 2010; Wolfson et al., 2020; Huang et al., 2023) to LLM-era sub-question generation (Zhou et al., 2023) and implicit CoT-style steps with self-consistency (Wei et al., 2022; Wang et al., 2023), each bought with additional calls. Within graph RAG, the assumption that decomposition must be purchased from an LLM has not been questioned; we break it. In LiftRAG decomposition is not a separate step at all. The multi-round beam expansion of trajectory search resolves one hop per round from the frontier the search has reached, so decomposition emerges from the structure of the search. This is what we call *Zero-LLM Decomposition*, and to our knowledge LiftRAG is the first instantiation of graph RAG whose retrieval is both training-free and LLM-free. The residual gap to LLM decomposition, 7–20% of sub-question trajectories, defines the space for fusing the two.

3 METHOD

We define *Zero-LLM Decomposition* as the principle that decomposition is performed by structured multi-round search at runtime, with no LLM calls at retrieval time. The LLM is confined to indexing and answer generation. The design takes its cue from relational memory, where transitive inference over stored relations is expressed flexibly (Konkel & Cohen, 2009; Eichenbaum & Cohen, 2014; Whittington et al., 2020; Behrens et al., 2018). Multi-hop QA is exactly such inference, and "where did the father of X die" *traverses stored relations* step by step. If retrieval itself walks relations one hop at a time, decomposition is a by-product of the walk, not an act of an external reasoner. LiftRAG instantiates this principle on LLM-extracted text knowledge graphs.

3.1 PRELIMINARIES

Task. Given a corpus chunked into passages $\mathcal{C}$, a pipeline indexes the corpus offline. At query time a retriever assembles a context of at most B characters from precomputed representations, and a reader LLM answers question q from it. The retrieval problem is to assemble the context that maximizes downstream accuracy; our claim is that this is achievable without spending test-time compute on retrieval-time LLM calls.

Index. An offline LLM reads the corpus and extracts entities $\mathcal{E}$ and relation instances $\mathcal{R}$, each $r = \langle u, \rho, v, s_r \rangle$ asserting relation ρ between u and v with its evidence sentence s_r. These objects form the graph index. An embedder $\phi(\cdot)$ maps text to unit vectors, and the query vector $\hat{\mathbf{q}} = \phi(q)$ is computed once per query. The query text is the full question by default. An alternative analytical view replaces it with the relation or action phrase extracted from the question, a pass the deployed default never needs. Both views instantiate the same formalism below.

3.2 LINE-GRAPH LIFTING

The base graph G realizes the index directly. Every entity $e \in \mathcal{E}$ is a vertex, and every relation instance $r \in \mathcal{R}$ is a labeled edge $u \xrightarrow{\rho} v$ between its endpoints $u, v \in \mathcal{E}$ (parallel edges allowed). Each entity vertex also links to its source passages, the entity-to-passage associations through which prior graph RAG attaches all evidence. In G an instance is reachable only through its endpoints, and its evidence sentence s_r is never consulted by retrieval. The semantics the extraction LLM attached to *what the instance says* is stored but not addressable. The construction below turns edges into vertices, the core of the line-graph transformation. Since alternating entity–relation walks on the lifted graph correspond exactly to walks on the line graph of G, we call the operation line-graph lifting.

The lifting places them in a second graph, the incidence graph $I(G)$ with vertex set $\mathcal{E} \cup \mathcal{R}$. Every entity $e \in \mathcal{E}$ remains a vertex, and every instance $r \in \mathcal{R}$ becomes a vertex joined to exactly its two endpoints u, v. The same r that was an edge of G is, after the lifting, a relation vertex of $I(G)$ carrying the same endpoints and evidence sentence. Trajectories then alternate entity vertices $e \in \mathcal{E}$ and relation vertices $r \in \mathcal{R}$ on $I(G)$. Entity and relation vertices thereby become the same kind of object, scoreable by a query alike, and relation vertices sharing an endpoint let partial trajectories be welded at common entity vertices. Entity-to-passage associations stay on entity vertices, feeding the passage channel at rendering time.

For retrieval, $I(G)$ is vectorized in three families. Each relation vertex $r \in \mathcal{R}$ carries $\mathbf{z}_r = \phi(s_r)$, the embedding of its evidence sentence. Each entity vertex $e \in \mathcal{E}$ carries $\mathbf{z}_e = \phi(\mathrm{desc}(e))$, where $\mathrm{desc}(e)$ is the description the extractor emitted for e. Each passage $c \in \mathcal{C}$ carries $\mathbf{z}_c = \phi(c)$. All three families are addressed by one primitive, the inner product,

$$\sigma_x(\hat{\mathbf{q}}) = \hat{\mathbf{q}}^\top \mathbf{z}_x, \quad x \in \mathcal{R} \cup \mathcal{E} \cup \mathcal{C}, \qquad A(q) = \text{top-}\, k_{e \in \mathcal{E}}\, \sigma_e(\hat{\mathbf{q}}), \tag{1}$$

where σ_r scores every relation vertex $r \in \mathcal{R}$ for the entry pass and $A(q) \subseteq \mathcal{E}$ selects the k anchor entities of the search. Retrieval thus begins with inner products over every family and no query-side LLM call.

3.3 Trajectory Search

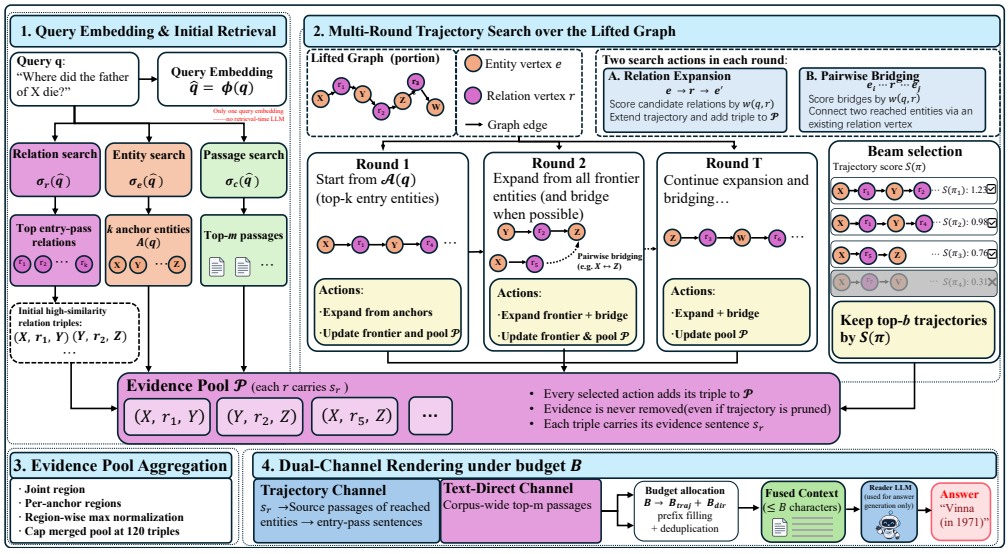

Figure 2: Online retrieval of LiftRAG. (1) One query embedding addresses the relation, entity, and passage families, and the entry pass populates the evidence pool. (2) Multi-round trajectory search over the lifted graph alternates relational expansion and pairwise bridging; beam selection prunes trajectories, never the pool. (3) The pool is aggregated into joint and per-anchor regions and normalized. (4) Dual-channel rendering splits the budget B adaptively and emits the context for the reader, with zero retrieval-time LLM calls.

Retrieval runs as a beam search over entity–relation trajectories on $I(G)$ (Fig. 2). An *entry pass* first computes $\sigma_r(\hat{\mathbf{q}})$ over all relation vertices and sends the highest-scoring triples directly into the evidence pool $\mathcal{P}$ as sentence-level entries, so high-confidence evidence never competes with later exploration. The k anchors $A(q)$ of Eq. equation 1 initialize the search, three by default. The search then starts from a set of seed states. One *joint* seed carries all k anchors as its entity set, with no edges between them yet, and each anchor also starts a single-entity seed of its own. Anchor-specific evidence survives in the per-anchor seeds even when the joint walk misses it.

Retrieval then runs T expansion rounds that grow the seeds into trajectories. A trajectory $\pi = (e_0, r_1, e_1, r_2, \dots)$ is a walk on $I(G)$ alternating entity vertices $e \in \mathcal{E}$ and relation vertices $r \in \mathcal{R}$. Each round, every live trajectory takes two kinds of actions. *Relational expansion* enumerates the relation vertices incident to a reached entity e; three fixed terms modulate each candidate relation $r \in \mathcal{R}$,

$$\mathrm{lex}(r, q) = \frac{|T(\rho_r) \cap T(q)|}{|T(\rho_r) \cup T(q)|}, \qquad \mathrm{idf}(r) = \log \frac{|\mathcal{R}|}{n(\rho_r)}, \qquad \mathrm{boost}(q, r) = \max_{\rho_{r'} = \rho_r} \sigma_{r'}(\hat{\mathbf{q}}), \tag{2}$$

with $T(\cdot)$ the token set and $n(\rho)$ the number of instances sharing the label ρ; boost reads the per-label maxima of the entry-pass scores, computed once per query. The traversal is scored by

$$w(q, r) = \underbrace{\hat{\mathbf{q}}^\top \mathbf{z}_r}_{\text{query–record sim}} \cdot \big(1 + \alpha\, \mathrm{lex}(r, q)\big) \cdot \mathrm{idf}(r) \; + \; \beta\, \mathrm{boost}(q, r), \tag{3}$$

and the top-K_A extensions per trajectory extend the walk. *Pairwise bridging* looks up the relation vertex joining each pair of entities the trajectory has already reached. Bridging candidates are scored by the same $w(q, r)$, and the top-K_B are kept each round; a trajectory carries at most six entities, so these pairwise lookups add only a bounded cost per round. Because the joint trajectory carries all anchors, a bridge welds partial chains seeded from different anchors into a closed one, as bridge questions require. Every action deposits its triple (e, r, e') with score $w(q, r)$ into the pool and extends the trajectory with cumulative score $S(\pi) \mathrel{+}= w(q, r)$.

At the end of each round, the candidate trajectories are deduplicated by their triple sets, scored by

$$S(\pi) = \textstyle\sum_{r \in \pi} w(q, r) \tag{4}$$

and truncated to the top-b (b the beam width). These b survivors become the trajectories of the next round, closing the loop. The two scores are thus decoupled. The cumulative $S(\pi)$ decides where to search next, while the single-action score deposited in $\mathcal{P}$ decides what enters the context. The beam prunes trajectories, never the pool. A pruned trajectory keeps every triple it already deposited. The constants α, β are fixed and the method contains no learned parameters.

After T rounds, the pool is reorganized for rendering so that no anchor drowns the others. Triples are split into regions by the anchors at their endpoints: triples connecting two different anchors form the joint region, and triples touching a single anchor form that anchor's region. Each region's scores are divided by its maximum, the regions are merged, and the merged pool is capped at a fixed size of 120 triples for rendering. The rounds are thus stateful, each one re-opening the frontier under the trajectories accumulated so far, so the search carries out query decomposition internally.

3.4 Dual-Channel Rendering under Budget

Rendering assembles the reader's context from the prepared pool in a fixed priority order (Liu et al., 2024). The trajectory channel opens with route sentences, each the evidence sentence s_r of a top pool triple; continues with the source passages of the entities the search reaches; and closes with the remaining high-confidence sentences from the entry pass. The text-direct channel adds the top-scoring passages from the whole corpus, reaching attribute-style evidence weakly connected in the graph. Within each channel, units are ranked by query similarity.

The context budget B, the same character limit every baseline answers under, is split between the channels in proportion to their evidence strength $s_{\text{traj}}, s_{\text{dir}}$, the mean top pool score versus the mean top passage similarity,

$$B_{\text{dir}} = \text{clip}\big(B \cdot s_{\text{dir}}/(s_{\text{traj}}{+}s_{\text{dir}}),\ B_{\min},\ B_{\max}\big), \tag{5}$$

so multi-hop questions route budget to trajectories and attribute-style questions to passages; a floor keeps the text-direct channel alive and a cap protects trajectory evidence ($B_{\min}$=1,500, $B_{\max}$=5,000 characters in all runs). Each channel fills its share along the priority order, the last unit truncated to the residual, so the context at any B is a prefix of one sequence. Deduplication is dynamic: a passage is skipped only if its text already entered the context, so one truncated away on the trajectory side remains reachable by the direct channel.

Algorithm A1 in Appendix A gives the complete pseudocode. Across both phases, retrieval pays dense vector passes over the entity, relation, and passage families (Eq. equation 1), as any dense retriever does, while the beam itself adds only $O(TbK_A|\mathcal{R}_e|)$ score evaluations per query, with b the beam width and $|\mathcal{R}_e|$ the relation instances incident to a reached entity, independent of corpus size. The defaults used throughout are beam width b=12, T=3 rounds, K_A=4, bridging width K_B=4, three anchors, m=20, α=1, and β=1.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate on the test splits of three multi-hop benchmarks with complementary hop structures, 2WikiMQA (Ho et al., 2020) (12,576; two-to-three-hop comparison and bridge questions), HotpotQA (Yang et al., 2018) (7,405; two-hop chains over adjacent paragraphs), and MuSiQue (Trivedi et al., 2022) (2,417; up-to-four-hop chains with paragraph-level distractors).

Protocol. All systems share one index, one reader, one budget, and one evaluation. The index is built by DeepSeek-V4-Flash extraction (the `deepseek-chat` API endpoint), embedded with bge-m3, yielding 251,698/351,470/119,697 entities, 317,881/473,101/145,725 relation instances, and 54,989/66,639/21,100 passages respectively. The reader is gemma2-9b at temperature 0, answering from the retriever's context of at most $B{=}8{,}000$ characters under the same prompt. All retrieval-time baseline LLM calls (LightRAG keyword extraction, PathRAG path scoring, HippoRAG 2 query NER and fact retrieval) share the same model and temperature. The n in each table counts questions whose gold answer resolves to the index's entity vocabulary (10,893/4,921/1,677), the entity-grounded test sets, evaluated in full without sampling; questions whose gold lies outside the entity space (yes/no answers, dates, non-entity spans) are unanswerable for every system sharing the index and are excluded uniformly.

Metrics. We report *Acc* (normalized bidirectional containment with gold), *Sub* (gold contained in prediction), token-level *F1* (HotpotQA-style), and *Gold* (answer coverage of the retrieved context, a retrieval-side metric invariant to the reader).

4.2 Main Results

Table 1: Main results on entity-grounded test sets. All rows share the reader, budget, and evaluation; LightRAG in three retrieval modes (naive, chunk retrieval bypassing the graph; local, entity-level graph retrieval; hybrid, entity- and relation-level keywords combined), PathRAG, Vector RAG, and ours share one index (DeepSeek-V4-Flash extraction, bge-m3). HippoRAG 2 normally re-extracts with its own schema; here it is re-indexed on our extraction and embeddings while keeping its official retrieval mechanism. GraphRAG runs hierarchical Leiden over the shared graph with community summaries generated by the same gemma2-9b. All values %; F1 in [0,1]; higher is better (↑) for all metrics. Δ is the gain of LiftRAG over the best prior system (HippoRAG 2).

	2WikiMQA (n=10,893)				HotpotQA (n=4,921)				MuSiQue (n=1,677)			
Method	Acc↑	Sub↑	F1↑	Gold↑	Acc↑	Sub↑	F1↑	Gold↑	Acc↑	Sub↑	F1↑	Gold↑
LightRAG hybrid	21.3	21.0	.218	52.5	33.9	32.0	.344	56.8	7.6	6.1	.107	32.9
LightRAG local	22.9	22.4	.239	55.6	32.5	30.2	.322	46.1	8.0	6.5	.101	24.0
GraphRAG	30.0	29.2	.297	33.0	36.5	33.4	.362	45.2	9.8	6.9	.119	29.2
PathRAG	35.5	34.6	.340	39.5	52.7	49.3	.517	63.6	17.1	14.2	.202	40.5
LightRAG-naive	42.9	41.5	.449	65.0	64.7	60.7	.635	80.8	20.0	17.1	.226	51.5
Vector RAG	44.9	43.3	.468	64.9	66.5	62.3	.652	81.2	23.3	19.7	.260	51.6
HippoRAG 2	48.7	46.9	.499	69.2	69.5	65.0	.679	82.1	26.6	23.0	.289	55.5
LiftRAG	**52.6**	**50.7**	**.530**	**74.9**	**70.4**	**66.2**	**.692**	**83.6**	**30.0**	**27.1**	**.327**	**62.0**
Δ *best prior*	*+3.9*	*+3.8*	*+.03*	*+5.7*	*+0.9*	*+1.2*	*+.01*	*+1.5*	*+3.4*	*+4.1*	*+.04*	*+6.5*

Main results. Table 1 compares against published baselines sharing our protocol; LiftRAG leads every cell. Against the strongest baseline, HippoRAG 2, accuracy margins are $+0.9$ to $+3.9$ points and coverage margins $+1.5$ to $+6.5$; a paired bootstrap puts 41 of the 42 margins at $p < 0.005$, the exception being this HotpotQA accuracy cell ($+0.9$, $p{=}0.076$): parity with the strongest LLM-guided baseline reached without its two retrieval-time LLM calls, its coverage margin there still significant ($+1.5$, $p{=}0.0002$). SQuAD-style exact match leaves the ranking unchanged, and the remaining baselines trail by $+3.9$ to $+37.9$ accuracy points, Vector RAG second strongest. The gap tracks where passage lookup fails, and Fig. A3 locates it by hop count on MuSiQue: the two strongest systems tie on 2-hop questions (66.0 versus 66.3), and the margin opens exactly on the 3- and 4-hop chains that require relation traversal ($+8.7$ and $+6.0$ coverage). GraphRAG's coverage falls below the Vector RAG floor on 2WikiMQA, as community summaries suit thematic rather than entity-anchored questions.

4.3 Robustness Analysis

Table 1's conclusions are robust along three axes: the reader, the hyperparameters, and the evidence budget.

Reader model robustness. All main results use gemma2-9b as the reader; re-answering every question with Qwen3.8-27B over the identical contexts tests whether a stronger reader compensates weaker retrieval. Gold coverage is identical across readers by construction (Table A1, Appendix), and decomposing accuracy as *coverage* $\times$ *extraction rate*, the reader moves only the second factor: LightRAG's extraction rate rises 11–23 points, yet our margins over LightRAG persist at 27B ($+10.1$ on 2WikiMQA, $+11.2$ on MuSiQue, $+3.8$ on HotpotQA). The stronger reader compensates for extraction weakness, not for retrieval weakness.

Hyperparameter sensitivity. The three core parameters are swept at full scale on every dataset (Fig. A1, Appendix); the remaining three (anchor count, α, bridging width K_B) are near-flat and tabulated there. Beam saturates by 8, K_A moves at most 1.3 points, and the round count is the only parameter with a consistent response: a single round, a single-step retriever, costs up to 2.3 points, while two rounds reach the plateau. Defaults are fixed once, never tuned per dataset.

Budget robustness. Shrinking the per-query budget from 8,000 characters, each system at its best response, our rendering re-allocates via Eq. equation 5 while LightRAG truncates (full curves in Appendix). Our coverage dominates at every budget on every dataset (never below $+6$ on 2WikiMQA, up to $+13.6$ on MuSiQue, $+1.7$ to $+2.8$ on HotpotQA), and re-rendering recovers up to $+17$ points over prefix truncation on HotpotQA as the 1k budget routes to the text-direct channel.

4.4 Ablations

Graph construction. We degrade every edge's information carrier from the relation-instance sentence vector to pure keyword token overlap (*kw-only*; the entry pass is disabled, so relation boost and all sentence-level evidence disappear), keeping the rest of the pipeline identical. Coverage drops 1.3–13.2 points (Table 2), concentrated on MuSiQue (-13.2), whose 4-hop chains can only be answered by traversing relations; on the other two datasets the text-direct channel largely compensates. The construction is thus indispensable exactly where the graph is needed, in multi-hop traversal, and redundant where passage lookup suffices.

Table 2: Ablations on entity-grounded test sets (answer coverage %); each row removes or degrades one element of the full system.

Variant	2WikiMQA	HotpotQA	MuSiQue
Full (LiftRAG)	74.9	83.6	62.0
Construction			
kw-only edges	67.4	82.3	48.8
Dual channel			
no text-direct (trajectory-only)	74.5	77.0	60.5
Search structure			
entry-pass only (no search)	69.0	80.3	55.8
Scoring components			
no lexical modulation	74.9	83.4	62.1
no relation boost	75.3	83.4	62.5
no relation idf	75.2	83.7	60.8
no region normalization	74.7	83.6	62.0

Components of the final system. Removing any scoring component (the lexical modulation, relation boost, relation idf, or region normalization) moves coverage by at most 1.2 points, and in the trajectory-only architecture by at most 0.9 across all twelve runs: the improvement is carried by the representation and the search structure, not the shape of the score, and the scoring terms stay as zero-cost fixed constants. Disabling the search itself, entry pass and text-direct channel kept, costs 3.3–6.2 points on every dataset (entry-pass only, the training-free analogue of learned parallel triple scorers). Removing the text-direct channel costs $+0.4$/$+6.6$/$+1.5$ on 2WikiMQA/HotpotQA/MuSiQue, largest on HotpotQA, whose attribute-style answers reside in passages.

4.5 EFFICIENCY

Retrieval-time cost. Our retrieval performs zero LLM calls: one query embed, dense vector passes over the three families, and the beam at 2 ms, totaling 38–76 ms per query under load. Wall-clock, LightRAG-naive's retrieval path takes 149 ms per question and HippoRAG 2's 96 ms, including their retrieval-time LLM calls; in API deployment each such call adds a measured 1.2 s of end-to-end latency, putting LightRAG at 13× our latency, HippoRAG 2 at 25×, and IRCoT-style pipelines (Trivedi et al., 2023) higher still (Appendix F).

4.6 MECHANISM ANALYSIS

Mechanism attribution. To verify that the advantage comes from the retrieval mechanism rather than index or reader fit, we re-run HippoRAG 2 (Jiménez Gutiérrez et al., 2025) on our own index, retrieval mechanism the only variable (the converted row of Table 1; the native pipeline falls between, Appendix E). LiftRAG wins all twelve cells by $+1.5$ to $+6.5$ coverage points (Table A5), as our construction predicts: diffusion carries a relation instance only as an edge weight, so a chain that must pass through a relation has no representation of that step, while trajectory search navigates the relation vertex directly.

LLM injection. Injecting an LLM at either canonical intervention point yields no gain (Table A6): evidence reranking loses 2.3 points, query decomposition loses 0.2–1.5 under all four fusion designs, and upgrading both to the frontier-grade DeepSeek-V4-Flash changes nothing (decomposition loses 0.3–1.3, frontier reranking loses 6.1). Within this protocol, multi-round beam expansion *is* runtime decomposition: each round re-opens the frontier earlier rounds actually reached, so a static decomposition is a subset of what the beam already generates and cannot react to what retrieval finds. The decomposition gains in the literature presuppose single-step retrievers; against a multi-round search the decomposition tax is pure overhead, whatever the strength of the decomposer.

Containment of LLM decomposition. Scoring the top evidence of every sub-question of a 9B decomposer against the pool a single LiftRAG retrieval returns, 80–93% of them fall within that pool, against a 0.01–0.03% random baseline. The random baseline calibrates the scale; the substantive control is the injection study above, where fusing the decomposer's actual output yields no gain, so the overlap reflects the search generating what the decomposer would ask rather than a permissive pool. Apart from the 7–20% residual, what an LLM decomposer contributes is already within reach of one retrieval.

Complementarity and the boundary. At full scale (Appendix E), the full question is the stronger single view everywhere, yet each view reaches gold evidence the other misses; interleaving the two pools adds at most $+0.5$, because under a fixed budget the views compete for the same slots. The complementarity is real but not cashable there.

Error decomposition. The retrieval residual (gold never entering the context) is 16.4%/25.1%/38.0% on HotpotQA/2WikiMQA/MuSiQue, and among covered questions the reader misses 15.8%/29.8%/51.6%; the 27B-reader bound on the second leaves the long-chain retrieval residual as the target for future work.

5 CONCLUSION

Precise multi-hop retrieval does not require an LLM call. Lifting each relation to a vertex carrying its evidence-sentence embedding restores semantics to the retrieval loop, and on the entity-grounded test sets trajectory search surpasses every compared baseline on all four metrics at 13–25× lower retrieval latency in API deployment. The injection experiments locate this advantage in the beam's runtime decomposition, and the remainder points to fusion.

AI USE STATEMENT

The authors used LLM-based tools (Claude Code and similar assistants) to assist with paper writing and the development of experiment code, including manuscript drafting, editing, LaTeX typesetting, and experiment scripts. All scientific content, including the method design, experimental protocols, analyses, and conclusions, was conceived, executed, and verified by the authors. All experimental numbers reported in this paper were produced by the authors' pipelines and checked against raw logs. The authors reviewed the final manuscript and take full responsibility for its content.

REFERENCES

Yong-Yeol Ahn, James P Bagrow, and Sune Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466:761–764, 2010.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *International Conference on Learning Representations*, 2024.

Timothy EJ Behrens, Thomas H Muller, James CR Whittington, Shirley Mark, Alon Baram, Kimberly L Stachenfeld, and Zeb Kurth-Nelson. What is a cognitive map? organizing knowledge for flexible behavior. *Neuron*, 100(2):490–509, 2018.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, 2013.

Boyu Chen, Zirui Guo, Zidan Yang, Yuluo Chen, Junze Chen, Zhenghao Liu, Chuan Shi, and Cheng Yang. Pathrag: Pruning graph-based retrieval augmented generation with relational paths. *arXiv preprint arXiv:2502.14902*, 2025.

Tong Chen, Hongwei Wang, Sihao Chen, Wenhao Yu, Kaixin Ma, Xinran Zhao, Hongming Zhang, and Dong Yu. Dense X retrieval: What retrieval granularity should we use? In *Proceedings of EMNLP*, 2024.

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

Howard Eichenbaum and Neal J Cohen. Can we reconcile the declarative memory and spatial navigation views on hippocampal function? *Neuron*, 83(4):764–770, 2014.

Thomas S Evans and Renaud Lambiotte. Line graphs, link partitions, and overlapping communities. *Physical Review E*, 80(1):016105, 2009.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. Building watson: An overview of the DeepQA project. *AI Magazine*, 31(3):59–79, 2010.

Mikhail Galkin, Priyansh Trivedi, Gaurav Maheshwari, Ricardo Usbeck, and Jens Lehmann. Message passing for hyper-relational knowledge graphs. In *Proceedings of EMNLP*, 2020.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. LightRAG: Simple and fast retrieval-augmented generation. *arXiv preprint arXiv:2410.05779*, 2024.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. HippoRAG: Neurobiologically inspired long-term memory for large language models. In *Advances in Neural Information Processing Systems*, 2024.

Olaf Hartig. Foundations of RDF* and SPARQL*. In *Alberto Mendelzon International Workshop on Foundations of Data Management (AMW)*, 2017.

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

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, 2020.

Xiang Huang, Sitao Cheng, Yiheng Shu, Yuheng Bao, and Yuzhong Qu. Question decomposition tree for answering complex questions over knowledge bases. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, and Ji-Rong Wen. UniKGQA: Unified retrieval and reasoning for solving multi-hop question answering over knowledge graph. In *International Conference on Learning Representations*, 2023a.

Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of EMNLP*, 2023b.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. From RAG to memory: Non-parametric continual learning for large language models. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025. arXiv:2502.14802.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of EMNLP*, pp. 6769–6781, 2020.

Aaron Konkel and Neal J Cohen. Relational memory and the hippocampus: Representations and methods. *Frontiers in Neuroscience*, 3:166–174, 2009.

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

Mufei Li, Siqi Miao, and Pan Li. Simple is effective: The roles of graphs and large language models in knowledge-graph-based retrieval-augmented generation. In *International Conference on Learning Representations*, 2025.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12, 2024.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *International Conference on Learning Representations*, 2024.

Costas Mavromatis and George Karypis. GNN-RAG: Graph neural retrieval for large language model reasoning. *arXiv preprint arXiv:2405.20139*, 2024.

Ofir Press, Merrill Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of EMNLP*, 2023.

Yitong Qin, Lihua Zhou, Jiwei Wei, Ran Ran, Shiyuan He, Zeyu Ma, Shuaifeng Li, Nianxin Li, and Heng Tao Shen. ViTPrompt: Training-free prompt refinement with visual tokens for open-vocabulary detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *Findings of EMNLP*, 2023.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. In *International Conference on Learning Representations*, 2025.

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

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multi-hop questions via single-hop question composition. In *Transactions of the Association for Computational Linguistics*, volume 10, pp. 539–554, 2022.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of ACL*, 2023.

Liang Wang, Haonan Chen, Nan Yang, Xiaolong Huang, Zhicheng Dou, and Furu Wei. Chain-of-retrieval augmented generation. *arXiv preprint arXiv:2501.14342*, 2025.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

James CR Whittington, Thomas H Muller, Shirley Mark, Guifen Chen, Caswell Barry, Neil Burgess, and Timothy EJ Behrens. The tolman-eichenbaum machine: Unifying space and relational memory through generalization in the hippocampal formation. *Cell*, 183(5):1249–1263, 2020.

Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. Break it down: A question understanding benchmark. *Transactions of the Association for Computational Linguistics*, 8:183–198, 2020.

Wenhan Xiong, Xiang Lorraine Li, Srini Iyer, Jingfei Du, Patrick Lewis, William Yang Wang, Yashar Mehdad, Wen-tau Yih, Sebastian Riedel, Douwe Kiela, and Barlas Oguz. Answering complex open-domain questions with multi-hop dense retrieval. In *International Conference on Learning Representations*, 2021.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, 2018.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

Tianjun Yao, Haoxuan Li, Zhiqiang Shen, Pan Li, Tongliang Liu, and Kun Zhang. Learning efficient and generalizable graph retriever for knowledge-graph question answering. *arXiv preprint arXiv:2506.09645*, 2025.

Jiahao Zhang, Haiyang Zhang, Dongmei Zhang, Yong Liu, and Shen Huang. End-to-end beam retrieval for multi-hop question answering. In *Proceedings of NAACL*, 2024.

Jing Zhang, Xiaokang Zhang, Jifan Yu, Jian Tang, Jie Tang, Cuiping Li, and Hong Chen. Subgraph retrieval enhanced model for multi-hop knowledge base question answering. In *Proceedings of ACL*, 2022.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. Least-to-most prompting enables complex reasoning in large language models. In *International Conference on Learning Representations*, 2023.

A ALGORITHM

Algorithm A1 formalizes the retrieval procedure of §3; the only query-side model call is the initial embedding, and no LLM is invoked anywhere in the loop.

Algorithm A1 Zero-LLM trajectory retrieval

Require: query q; index $\{\mathbf{z}_r, \mathbf{z}_e, \mathbf{z}_c\}$ over $I(G)$; budget B
Ensure: a context of at most B characters

$\hat{\mathbf{q}} \leftarrow \phi(q)$ ▷ the only query-side model call
$\sigma_r \leftarrow \hat{\mathbf{q}}^\top \mathbf{z}_r$ for all r; deposit top triples into $\mathcal{P}$ ▷ sentence-level entries of the entry pass, Eq. (1)
$\mathrm{boost}(\rho) \leftarrow \max_{r:\mathrm{type}(r)=\rho} \sigma_r$; $A(q) \leftarrow \text{top-}k\, \hat{\mathbf{q}}^\top \mathbf{z}_e$
$\Pi \leftarrow \{\mathrm{joint}(A(q))\} \cup \{\pi_e : e \in A(q)\}$ ▷ seed states
for $t = 1, \ldots, T$ **do**
  for $\pi \in \Pi$ **do**
    for the top-K_A relational actions over all reached entities of π **do** ▷ relational expansion
      $w \leftarrow w(q, r)$ by Eq. (3); $\mathcal{P}[(e, r, e')] \leftarrow w$; candidate $S \leftarrow S(\pi) + w$
    end for
    for the top-K_B pairs e_i, e_j reached in π **do** ▷ pairwise bridging
      $r \leftarrow$ relation vertex joining e_i, e_j; $\mathcal{P}[(e_i, r, e_j)] \leftarrow w(q, r)$
    end for
  end for
  $\Pi \leftarrow \text{top-}b$ candidates by S (dedup by triple set) ▷ Eq. (4); pool never pruned
end for
region-normalize $\mathcal{P}$ (joint region vs. per-anchor regions); cap the pool at 120 triples ▷ §3.3
split B into $B_{\mathrm{traj}}, B_{\mathrm{dir}}$ by Eq. (5); fill priority order with prefix filling and dynamic dedup ▷ §3.4
return $\mathcal{C}$

B READER ROBUSTNESS

Table A1: Reader robustness. Identical contexts answered by gemma2-9b and Qwen3.8-27B. Entity-grounded test sets; all values %.

	2WikiMQA		HotpotQA		MuSiQue	
Reader	Acc	Gold	Acc	Gold	Acc	Gold
LightRAG-naive 9B	42.9	65.0	64.7	80.8	20.0	51.5
LightRAG-naive 27B	52.9	65.0	73.8	80.8	31.7	51.5
LiftRAG9B	52.6	74.9	70.4	83.6	30.0	62.0
LiftRAG27B	**63.0**	74.9	**77.6**	83.6	**42.9**	62.0

C HYPERPARAMETER SWEEPS

Sweeps report answer coverage (%), the retrieval-side metric. The three core parameters are swept on the entity-grounded test sets of all three datasets (Fig. A1); the tables in this appendix report the 2WikiMQA column (10,893 gold-answerable questions). Each run varies one parameter while the rest stay at their defaults (beam 12, rounds 3, K_A=4, K_B=4, anchors 3, α=1, β=1); defaults are bold.

Core parameters. Figure A1 plots the sweeps of the three core search parameters on all three datasets, and Table A2 tabulates the 2WikiMQA column. Beam width is swept over $\{4, 8, 12, 16, 24\}$, rounds over $\{1, 2, 3, 4, 6\}$, and the per-round expansion width K_A over $\{2, 4, 6, 8\}$. Coverage saturates by beam 8 and round 2, and K_A moves at most 3.3 points across its range.

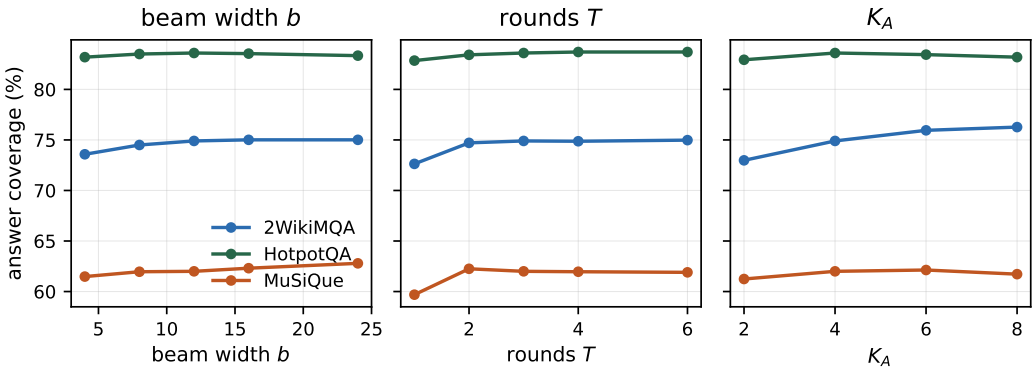

Figure A1: Hyperparameter sensitivity on the entity-grounded test sets of all three datasets (answer coverage %). Sweeps over beam width $b \in \{4, 8, 12, 16, 24\}$, rounds $T \in \{1, 2, 3, 4, 6\}$, and $K_A \in \{2, 4, 6, 8\}$, with defaults b=12, T=3, K_A=4.

Near-flat parameters. Table A3 tabulates the remaining three.

Table A2: Hyperparameter sensitivity (2WikiMQA entity-grounded test set, answer coverage %; defaults in bold).

Parameter	Value				
b	4	8	**12**	16	24
coverage	73.55	74.46	**74.92**	74.99	75.00
T	1	2	**3**	4	6
coverage	72.61	74.69	**74.92**	74.88	74.98
K_A	2	**4**	6	8	
coverage	73.00	**74.92**	75.92	76.24	

The anchor count is swept over $\{1, 2, 3, 5\}$ and the lexical weight α over $\{0, 0.5, 1, 2\}$. The anchor count is flat for reasonable values (2–5 within 0.5) with only the extreme of a single anchor costing 1.3 points; the text-direct channel makes anchor quality largely non-critical, since the evidence pool can be entered independently of the anchors. The lexical weight α is exactly flat (within 0.1), corroborating the no-lex ablation (§4.4). The bridging width K_B is exactly flat as well: across the three datasets, coverage moves at most 0.06 points between K_B=2 and K_B=8.

Table A3: Near-flat parameters (2WikiMQA entity-grounded test set, answer coverage %; defaults in bold).

Anchors	1	2	**3**	5
coverage	73.59	74.61	**74.92**	74.48
α	0	0.5	**1**	2
coverage	74.85	74.83	**74.92**	74.87
K_B	2	**4**	8	
coverage	74.93	**74.92**	74.92	

D Budget Sweeps and Rendering

LiftRAG re-renders its context under each budget (Eq. (5)); LightRAG's static context is prefix-truncated to B characters. Coverage is computed on the budgeted contexts.

Fig. A2 and Table A4 report how answer coverage changes as the per-query budget shrinks from the protocol value of 8,000 characters to 1,000, on entity-grounded test sets, each system given its best response to the smaller budget.

Table A4: Budget–coverage curves, entity-grounded test sets (answer coverage %). LiftRAGre-renders under each budget (Eq. (5)); LightRAG truncates its static context.

	1,000		2,000		4,000		8,000	
Dataset	LiftRAG	LightRAG	LiftRAG	LightRAG	LiftRAG	LightRAG	LiftRAG	LightRAG
2WikiMQA	**47.3**	41.1	**58.0**	50.7	**69.1**	58.7	**74.9**	65.0
HotpotQA	**62.8**	60.4	**71.6**	69.9	**78.1**	75.7	**83.6**	80.8
MuSiQue	**22.4**	21.3	**38.9**	31.7	**54.4**	40.8	**62.0**	51.5

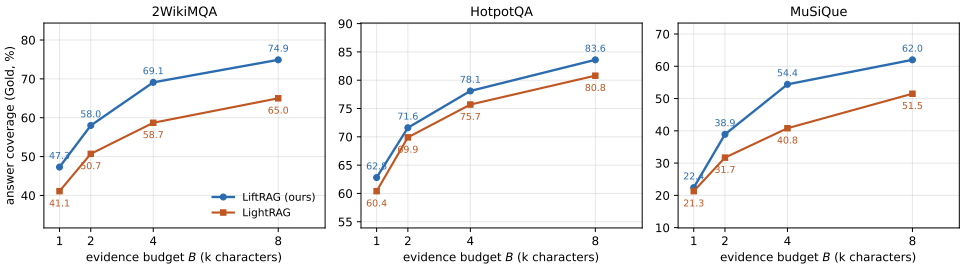

Figure A2: Answer coverage as the evidence budget shrinks from 8k to 1k characters (entity-grounded test sets). LiftRAG re-renders under each budget (Eq. (5)); LightRAG truncates its static context. The margin over LightRAG persists at every budget.

E Mechanism Details

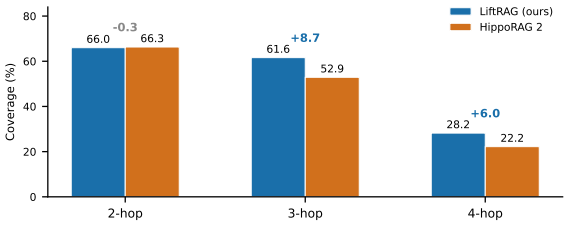

Figure A3: Answer coverage by hop count on MuSiQue (entity-grounded test set, n=1,677). The two strongest systems tie on 2-hop questions (66.0 versus 66.3); the margin opens exactly on the 3- and 4-hop chains that require relation traversal (+8.7 and +6.0 coverage), locating the gain of Table 1 in multi-hop traversal.

Hop-wise coverage. Fig. A3 groups the MuSiQue entity-grounded test set by the hop count encoded in its question ids and computes answer coverage per group from the identical contexts of the main protocol. On 2-hop questions the two strongest systems are tied; the advantage of trajectory search emerges precisely where chains lengthen and relation traversal becomes necessary, the pattern the injection and containment analyses below explain.

Case study. Fig. A4 shows one 4-hop question where LiftRAG retrieves the full chain and both baselines break it. The failure mode is the one our construction predicts: once the frontier must pass *through a relation* (state capital, shares border), systems that carry relations only as unaddressable edges lose the thread, while trajectory search navigates the relation vertices directly.

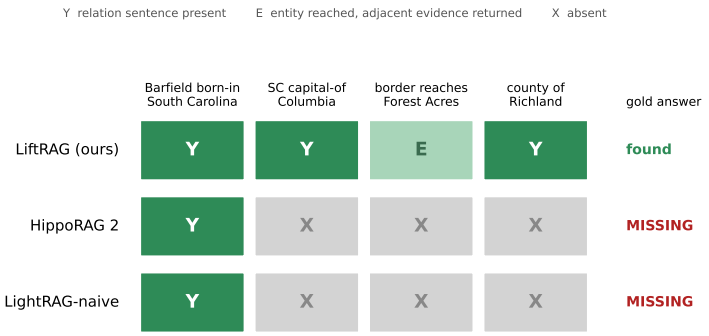

Figure A4: A 4-hop chain from the MuSiQue test set (question: *"What county is the city that shares a border with the state capital of the state where Velma Barfield was born located in?"*, answer *Richland County*). Cells mark whether the evidence of each hop enters the retrieved context (Y: relation sentence present; E: entity reached, adjacent evidence returned; X: absent). LiftRAG's trajectory carries the chain end to end: "Velma Barfield was born in rural South Carolina" → "Columbia is the capital city of the state of South Carolina" → the Forest Acres vertex is reached → "the city serves as the county seat of Richland County". In HippoRAG 2 and LightRAG-naive the chain breaks after the first hop: the passage around Barfield is retrieved, but diffusion and passage similarity never propagate through the capital-of and border relations, so the answer never enters the context.

Same-source conversion. HippoRAG 2's index (schema-4, manifest-bound) is normally built from its own extraction, which differs from ours in vocabulary and schema; comparing systems on different extractions would confound index and retriever. We therefore convert our extraction into HippoRAG 2's OpenIE cache format, in which each corpus chunk becomes a HippoRAG document, extracted entities are the chunk-covered entities of our graph, and extracted triples are exactly the relation instances whose two endpoints are both covered by the chunk, written as `[src, keywords, tgt]`. HippoRAG 2's index is then built from this cache with the same bge-m3 embedder, and its query pipeline (LLM fact retrieval, recognition-memory filter, dense passage scoring, PPR re-ranking) runs over it; retrieved documents are truncated to the same 8,000-character budget and answered with our unified protocol. HippoRAG's native QA templates are unused, since they are tuned to its own retrieval style; both systems therefore share extraction, corpus, embedder, reader, prompt, and budget; only the retrieval mechanism differs. A native-extraction control reruns HippoRAG 2's own OpenIE pipeline (deepseek-chat extraction over the raw chunks; 21,099 of 21,100 chunks, the one whose extraction call failed is skipped) and lands between its converted version and ours on MuSiQue (27.5/24.7/.304/56.9), ruling out conversion handicap.

Injection protocol. Both injections use gemma2-9b, the same model as the reader, so the comparison isolates the injection itself; a frontier condition repeats both with DeepSeek-V4-Flash as decomposer and reranker, on MuSiQue. Evidence reranking replaces the action-score ordering of the pool with the LLM's selection over the same candidate set (one call per question). Query decomposition asks the LLM for 2–3 sub-questions and fuses their evidence under four designs. *v1* lets sub-question chunks compete for the rendering budget; *v2* appends them without displacing main-question evidence; *v3* uses sub-questions to add intermediate-entity anchors (75% non-overlapping with main-query anchors, and indeed correct intermediate entities); *v4* adds downweighting that protects main-anchor rendering slots. Table A6 reports the verdict on 2WikiMQA (reranking) and MuSiQue (decomposition).

Query-view comparison. The two views of §3.1 retrieve with the same pipeline and differ only in the query text, the full question versus the relation or action phrase extracted from it by the reader-grade model (an analytical configuration; the deployed default needs no such pass); their union interleaves the two evidence pools before region normalization and rendering. All three runs are at full scale; the question view is an independent execution of the main configuration, and its coverage lands within 0.3 points of Table 1.

Table A5: Same-source comparison. HippoRAG 2 is re-indexed on our corpus with our extraction, our embedder, our reader, and our answering protocol. The only variable is the retrieval mechanism (HippoRAG 2's LLM fact retrieval, recognition memory, dense passage scoring, and PPR re-ranking vs. zero-LLM trajectory search). Entity-grounded test sets; all values %; F1 in [0,1]. The LiftRAG and HippoRAG 2 rows coincide with their Table 1 counterparts.

	2WikiMQA (n=10,893)				HotpotQA (n=4,921)				MuSiQue (n=1,677)			
Method	Acc	Sub	F1	Gold	Acc	Sub	F1	Gold	Acc	Sub	F1	Gold
HippoRAG 2	48.7	46.9	.499	69.2	69.5	65.0	.679	82.1	26.6	23.0	.289	55.5
LiftRAG	**52.6**	**50.7**	**.530**	**74.9**	**70.4**	**66.2**	**.692**	**83.6**	**30.0**	**27.1**	**.327**	**62.0**

Table A6: LLM injection study. Every injection point (reranking and decomposition under four fusion designs, at reader-grade and frontier grade) yields zero or negative gain; multi-round beam expansion already performs decomposition at runtime. Gold coverage, entity-grounded test sets; each injection is evaluated on the dataset shown, with Δ the change against the same-dataset Zero-LLM baseline.

Variant	Dataset	Coverage	
Zero-LLM baseline	2WikiMQA	74.9	
Zero-LLM baseline	MuSiQue	62.0	
+LLM reranking (1 call/q)	2WikiMQA	72.6	(-2.3)
+LLM decomposition v1 (competing)	MuSiQue	61.8	(-0.2)
+LLM decomposition v2 (no displacement)	MuSiQue	60.5	(-1.5)
+LLM decomposition v3 (anchor expansion)	MuSiQue	61.3	(-0.7)
+LLM decomposition v4 (+downweight)	MuSiQue	61.3	(-0.7)
+Frontier decomposition v1–v4 (DeepSeek)	MuSiQue	60.7–61.7	$(-0.3$ to $-1.3)$
+Frontier reranking (DeepSeek)	MuSiQue	55.9	(-6.1)

F Reproduction Configuration

All experiments run on local Linux servers; the reader and the embedder are served by a local Ollama deployment. *Index:* entities and relations extracted offline by DeepSeek-V4-Flash (the `deepseek-chat` API endpoint); all embeddings by bge-m3; relation instances lifted to line-graph vertices with evidence-sentence embeddings. *Retrieval hyperparameters:* beam width 12, rounds 3, action width K_A=4, bridging width K_B=4, anchors 3, lexical weight α=1, boost weight β=1, entry-pass width 50, pool size 120, relation idf, $\mathrm{idf}(r) = \log\big(|\mathcal{R}| \,/\, n(\rho_r)\big)$ with $n(\rho)$ the number of instances sharing the label ρ; lexical modulation, $\mathrm{lex}(r, q) = |T(\rho_r) \cap T(q)| \,/\, |T(\rho_r) \cup T(q)|$ with $T(\cdot)$ the token set (lowercased, non-alphanumeric characters stripped, tokens of length > 2 kept, stopwords removed); entry-pass boost, $\mathrm{boost}(q, r) = \max_{\rho_{r'}=\rho_r} \sigma_{r'}(\hat{\mathbf{q}})$ added with β=1; region normalization. The action score is floored at zero, so deposited scores are non-negative and division by the region maximum preserves order; entry-pass triples enter the pool at twice their sentence-level score. Beam actions rescore the cached relation embeddings of Eq. (1), so the beam cost counts scalar score evaluations rather than new vector passes. The evidence strengths of Eq. (5) are the mean of the top ten normalized pool scores and the mean of the top five passage similarities, both in $[0, 1]$; below the floor the clip degenerates to routing the whole budget to the text-direct channel. Rendering uses an 8,000-character budget with a 1,500-character floor and a 5,000-character cap on the text-direct channel. *Readers:* gemma2-9b (all tables) and Qwen3.8-27B (reader-robustness), temperature 0, 64-token answer cap, identical prompt. *Evaluation: Acc* (normalized bidirectional containment), *Sub* (normalized gold containment), token-level *F1* (HotpotQA-style), *Gold* (answer coverage of the retrieved context), on the entity-grounded test sets. *Query-side token measurement:* replaying each system's exact retrieval-time prompts on 100 real questions through the same local reader model measured 569.7 prompt and 35.3 completion tokens per query for LightRAG's keyword extraction (1 call/query) and 93.0/72.0 for HippoRAG 2's query NER (1 call/query).