Only Ask What You Don't Know: Grounded Delta Planning for Efficient Multi-step RAG

Wei-Chieh Chou[1*] **Xuanjun Chen**[2*] **Jian-Ren Lin**[3†] **Claire Lin**[4†]
Hung-yi Lee[2,5], **Jyh-Shing Roger Jang**[1]
[1]Dept. of Computer Science and Information Engineering, National Taiwan University
[2]Graduate Institute of Communication Engineering, National Taiwan University
[3]Department of Economics, National Taiwan University
[4]Department of Information Management, National Taiwan University
[5]NTU Artificial Intelligence Center of Research Excellence (NTU AI-CoRE)

Abstract

Multi-hop question answering remains challenging for Retrieval-Augmented Generation (RAG) because existing approaches either propagate errors across iterative retrieval rounds or over-generate reasoning steps, increasing cost without improving accuracy. We propose Grounded Delta Planning RAG (GDP-RAG), a plan-based framework that targets only the information delta (Δ) based on three simple design choices: (1) preliminary retrieval to ground planning before execution, (2) a gap-conditioned planning prompt that asks only for missing information, and (3) a skeletal trajectory that pairs each subquery with a `Thought` capturing evidence from preliminary retrieval and carrying it through to the final answer. GDP-RAG focuses computation on unresolved gaps, yielding concise, reliable reasoning trajectories. Extensive experiments on HotpotQA, 2WikiMultiHopQA, and MuSiQue show that GDP-RAG achieves the highest accuracy (60.63%) among all compared systems while maintaining a cost-of-pass of 0.51, 22% lower than PAR-RAG (0.65) and 68% lower than KnowTrace (1.57), with no method achieving both higher accuracy and lower cost.

1 Introduction

Large Language Models (LLMs) exhibit strong reasoning capabilities (Kojima et al., 2022) but remain limited in knowledge-intensive scenarios, where reliance on parametric knowledge often leads to hallucinations. Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) alleviates this issue by grounding generation in external knowledge; however, it becomes insufficient for complex information needs such as multi-hop question answering (Talmor & Berant, 2018; Welbl et al., 2018; Lin et al., 2025), which require coordinated reasoning over distributed context. Consequently, modern RAG systems increasingly adopt question decomposition to integrate retrieval with reasoning (Gao et al., 2023b; Singh et al., 2025).

Existing approaches can be categorized by when decomposition is performed: step-wise and plan-based frameworks. Step-wise methods, such as ReAct (Yao et al., 2023) and Self-Ask (Press et al., 2023), dynamically interleave retrieval and reasoning by decomposing queries at each step. While flexible, this paradigm suffers from two critical limitations: (i) the accumulation of retrieved documents can exceed the model's effective context capacity, leading to the lost-in-the-middle problem (Liu et al., 2024a); and (ii) strong inter-step dependencies amplify error propagation, whereby early mistakes cascade through the entire reasoning process (Dziri et al., 2023). In contrast, plan-based frameworks such as ReWOO (Xu et al., 2023) constructing static global plans before the execution. This design alleviates context overflow and reduces cascading errors by isolating sub-steps in advance,

[*]Contributed equally as first authors.
[†]Contributed equally as second authors.

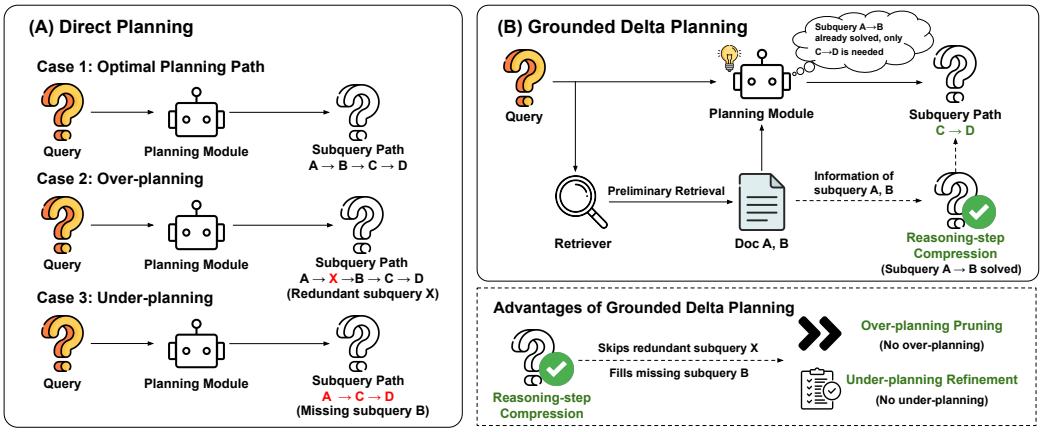


Figure 1: (A) Direct Planning decomposes queries using only parametric knowledge, producing redundant subqueries (over-planning, e.g., X in Case 2) or missing ones (under-planning, e.g., B in Case 3). (B) Grounded Delta Planning retrieves relevant documents first, then generates subqueries only for missing information (Δ), enabling: Reasoning-step Compression that skips resolved steps (Case 1), Over-planning Pruning that eliminates X (Case 2), and Under-planning Refinement that recovers B and preserves it in the `Thought` field (Case 3).

providing structural stability. However, this poses the risk of over or under-planning. Furthermore, strictly following such rigid plans leads to severe efficiency issues as the cost of each execution step becomes increasingly substantial (Singh et al., 2025).

To address these challenges, we propose **Grounded Delta Planning RAG (GDP-RAG)**, a plan-based framework that plans only for the information delta (Δ), the knowledge still missing after an preliminary retrieval. GDP-RAG realises this idea through three design choices: (1) preliminary retrieval to ground the planner in available evidence before decomposition; (2) a gap-aware query decomposition that instructs the planner to identify what is already known and generate sub-questions only for remaining gaps; and (3) a skeletal trajectory that pairs each subquery with a `Thought` capturing evidence from preliminary retrieval and carrying it through to the final answer. Together, these choices yield compact plans that target genuine information gaps, avoiding both the over-planning of conventional plan-based methods and the error propagation of step-wise approaches. As illustrated in Figure 1, Direct Planning decomposes queries using only parametric knowledge, producing redundant or missing subqueries. Grounded Delta Planning conditions the planner on preliminary retrieval, enabling three structural advantages: Reasoning-step Compression that skips steps already resolved by retrieved context, Over-planning Pruning that eliminates redundant subqueries, and Under-planning Refinement that recovers missing dependencies through broader retrieval semantics. Our contributions are summarized as follows:

- We propose GDP-RAG, a plan-based multi-step RAG framework built on Grounded Delta Planning that retrieves first and plans only for the information delta, yielding concise reasoning trajectories without sacrificing answer quality.

- Experiments on HotpotQA, 2WikiMultiHopQA, and MuSiQue show that GDP-RAG achieves the highest accuracy (60.63%) among compared systems while maintaining a cost-of-pass of 0.51, 22% lower than PAR-RAG (0.65) and 68% lower than KnowTrace (1.57), with no method achieving both higher accuracy and lower cost.

- Ablation and effectiveness analyses confirm that each design choice contributes, and that Grounded Delta Planning compresses reasoning steps, prunes redundant sub-questions, and refines incomplete plans.

2 Related Work

We categorize prior work on complex QA into (1) Step-wise Approaches, (2) Plan-based Approaches, (3) Retrieval and Verification, and (4) Improving the Planner and Reasoner.

2.1 Step-wise Approaches

Step-wise methods interleave retrieval and reasoning, generating one subquery at a time conditioned on previously retrieved context. ReAct (Yao et al., 2023), IRCoT (Trivedi et al., 2023), and Iter-RetGen (Shao et al., 2023) integrate retrieval into Chain-of-Thought reasoning (Wei et al., 2022). RAT (Wang et al., 2024b) and Verify-and-Edit (Zhao et al., 2023) further refine reasoning chains through iterative retrieval and verification. Search-o1 (Li et al., 2025b) extends this paradigm by interleaving agentic search with reasoning while maintaining a condensed summary of retrieved documents as its knowledge state. Beyond such textual representations, several methods introduce structured graph-based representations to manage growing context. SG-Prompt (Li & Du, 2023) and ERA-CoT (Liu et al., 2024b) adopt structured prompting strategies for multi-step reasoning. KnowTrace (Li et al., 2025a) summarizes each step's knowledge into a structured graph, keeping reasoning state compact. RAS (Jiang et al., 2025) maintains a graph-structured knowledge state with iterative planning, while MultiCube-RAG (Shi et al., 2026) employs hierarchical multi-dimensional indexing for iterative subquery retrieval. Despite these advances, step-wise methods face two limitations: they lack global foresight, deciding the next subquery based only on the current state without a coherent plan, and they rely on the LLM to determine when to stop, so intermediate errors can propagate undetected.

2.2 Plan-based Approaches

Plan-based methods construct a global reasoning blueprint before execution, fixing the number of steps and thus determining when to stop at planning time. PlanRAG (Lee et al., 2024) and Plan*RAG (Verma et al., 2024) generate linear or DAG-structured plans from parametric knowledge without external context, while ReWOO (Xu et al., 2023) and LPKG (Wang et al., 2024a) enhance reliability via in-context learning and task-specific training. PAR-RAG (Zhang et al., 2025), ComposeRAG (Wu et al., 2025), and MA-RAG (Nguyen et al., 2025) adopt modular architectures with conditional or selective execution. Grounding planning in retrieval has been explored by STORM (Shao et al., 2024) and Godbole et al. (2024) for article generation. However, these grounded methods use preliminary retrieval to enrich the plan rather than to prune it, producing a complete reasoning trajectory regardless of whether the information is already covered. While grounding in retrieval is well-explored in step-wise methods, it remains underexplored in plan-based frameworks. GDP-RAG introduces gap-aware query decomposition: it identifies what is already known from preliminary retrieval and generates subqueries only for the information delta (Δ), yielding a minimal skeletal trajectory rather than a complete reasoning chain. In contrast to these methods, including Godbole et al. (2024) which grounds planning in retrieval yet still produces a complete reasoning path, GDP-RAG generates subqueries only for the missing information, so the plan itself is already minimal before execution begins.

2.3 Retrieval and Verification

Beyond query decomposition, multi-hop reasoning depends on accurate retrieval and answer verification at each step. Standard RAG (Lewis et al., 2020) suffers from query ambiguity, motivating query expansion (Gao et al., 2023a; Xia et al., 2025) and structured retrieval via knowledge graphs (Edge et al., 2024; Gutierrez et al., 2024; Li et al., 2026). Robustness is further improved through filtering and verification (Yan et al., 2024; Asai et al., 2024; Dhuliawala et al., 2024). GDP-RAG builds on the Act-Review-Update mechanism from PAR-RAG (Zhang et al., 2025). Act retrieves documents and generates a provisional answer, Review cross-verifies it via secondary retrieval, and Update commits the verified result to a persistent Trajectory Memory and refines the next sub-question based on accumulated evidence, providing runtime adaptability so that execution is not locked to the initial plan.

2.4 Improving the Planner and Reasoner

Methods for improving the planner and reasoner in multi-step RAG fall into training-free and training-based approaches. Training-free approaches include prompting-based methods such as ReAct (Yao et al., 2023) and IRCoT (Trivedi et al., 2023), and few-shot learning with planning demonstrations as in ReWOO (Xu et al., 2023). Training-based methods instead fine-tune the LLM, either via supervised learning, including LPKG (Wang et al., 2024a) and Self-RAG (Asai et al., 2024), or via reinforcement learning. Search-R1 (Jin et al., 2025) introduces an RL framework that optimizes LLM reasoning with multi-turn search interactions through outcome-based rewards, R1-Searcher (Song et al., 2025) proposes a two-stage outcome-based RL approach, and FrugalRAG (Java et al., 2026) explores efficiency-oriented RL fine-tuning. GDP-RAG is training-free, leveraging preliminary retrieval and gap-aware instructions to guide the planner toward a concise plan.

3 Method

3.1 Problem Formulation

We study efficient multi-step RAG. Given a query Q and a document collection $\mathcal{D}$, the goal is to produce an answer A by constructing a reasoning trajectory $\mathcal{T}$ that integrates information from retrieved documents across multiple steps, such that:

$$A = \text{Answer}(Q, \mathcal{T}), \tag{1}$$

where Trajectory $\mathcal{T} = \langle (q_1, d_1, a_1), \ldots, (q_N, d_N, a_N) \rangle$ is defined as a sequence of N steps. The plan-based method first decomposes Q into a set of subqueries $\{q_1, \ldots, q_N\}$. Subsequently, at each step $t \in \{1, \ldots, N\}$, the system utilizes q_t to retrieve a document set $d_t \subset \mathcal{D}$ and generates an intermediate answer a_t. The objective is to maximize the final answer A quality, while minimizing computational cost, captured by minimizing the trajectory length N. This requires a minimal yet sufficient set of subqueries that uncover the information required to answer Q, while avoiding redundant retrieval and reasoning.

3.2 Framework of GDP-RAG

As illustrated in Figure 2, GDP-RAG consists of three stages: Planning (§ 3.2.1), Trajectory Generation (§ 3.2.2), and Answering (§ 3.2.3). Planning performs preliminary retrieval to obtain D_{rel} and outputs a skeletal trajectory $\mathcal{T}_{skel}$ containing only essential subqueries. Trajectory Generation executes and verifies each subquery in $\mathcal{T}_{skel}$, yielding a fully grounded trajectory $\mathcal{T}$. Answering synthesizes the final answer A from $\mathcal{T}$.

3.2.1 Planning Phase

The Planning Phase decomposes Q into a minimal set of subqueries that target only the information delta not yet covered by D_{rel}. This is realised through three design choices.

Preliminary Retrieval. Without retrieved context, the planner must rely solely on parametric knowledge, risking redundant or misguided subqueries. To address this, the system first retrieves query-related context $D_{rel} \subset \mathcal{D}$ via coarse retrieval on Q. D_{rel} grounds the planner in concrete evidence before decomposition and defines what information is already available so that planning targets only the remaining gaps. We ablate passage quality in Section 6 and show that relevant passages improve both step efficiency and accuracy.

Gap-aware Query Decomposition. Even with retrieved documents, the planner may still decompose the full query without recognizing what is already answered. To focus on the delta, the planning prompt enforces two instructions: (1) identify information available in D_{rel}, and (2) only create steps for information not in D_{rel}. This ensures the planner generates sub-questions only for Δ. We ablate this in Section 6 and show it primarily drives efficiency. The Direct Planning baseline omits D_{rel} and the gap instructions. Full prompts for Direct and GDP-RAG are in Appendix Figures 5 and 6.

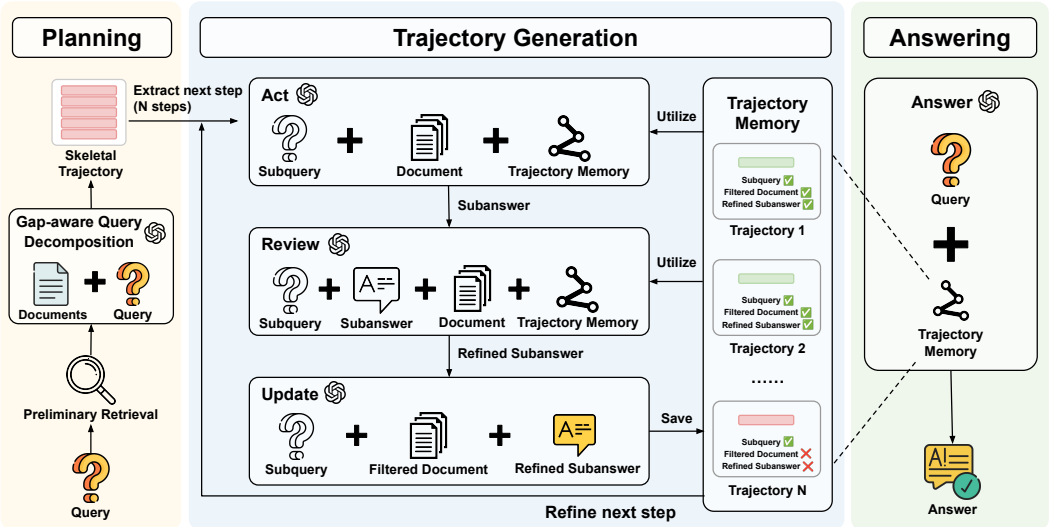


Figure 2: GDP-RAG framework. The workflow proceeds in three phases: (1) **Planning Phase**: Utilizes Grounded Delta Planning to generate a concise skeletal trajectory $\mathcal{T}_{skel}$, leveraging context from preliminary retrieval to prune redundant steps. (2) **Trajectory Generation Phase**: Executes an iterative Act-Review-Update loop, where each step interacts with Trajectory Memory and undergoes unconditional verification to transform the skeleton into a fully verified trajectory. (3) **Answering Phase**: Synthesizes the final response based on this comprehensive trajectory $\mathcal{T}$.

Skeletal Trajectory. To organize what is already known and what still needs to be retrieved into a unified structure, the planner outputs a skeletal trajectory:

$$\mathcal{T}_{skel} = \text{Plan}(Q, D_{rel}) = \langle (\theta_1, q_1), \ldots, (\theta_n, q_n) \rangle, \tag{2}$$

where each `Thought` θ_i preserves what evidence from D_{rel} has already been obtained, and each subquery q_i targets the information that is still missing. This extends the standard trajectory tuple from (q_t, d_t, a_t) to $(\theta_t, q_t, d_t, a_t)$. Preserving θ_t serves two purposes: the planner must justify every subquery, which discourages unnecessary steps, and the evidence in θ_t is carried through Trajectory Memory to the final answering stage, so it is not lost during execution. PAR-RAG also generates Thoughts but discards them afterward. We ablate the Thought component in Section 6 and show it is the primary driver of accuracy.

Advantages of Grounded Delta Planning. Together, these three choices yield three structural benefits (empirically validated in Section 7): (1) *Reasoning-step Compression*: D_{rel} often already contains facts that would otherwise require dedicated steps. Since θ_t preserves this evidence through to the answering stage, those steps can be safely skipped without information loss. (2) *Over-planning Pruning*: Without retrieved evidence, planners rely on parametric knowledge and tend to hallucinate unnecessary steps. Grounding on D_{rel} lets the planner verify assumptions against actual documents, eliminating subqueries with no factual basis. (3) *Under-planning Refinement*: Direct Planning decomposes the query into narrow, targeted subqueries that may miss relevant documents. Because D_{rel} is retrieved using the original query, its broader semantics can retrieve entities and relations that targeted subqueries would miss, helping the planner capture dependencies that they would miss.

3.2.2 *Trajectory Generation Phase*

The skeletal plan $\mathcal{T}_{skel} = \langle (\theta_1, q_1), \ldots, (\theta_N, q_N) \rangle$ defines the planned sequence of subqueries before execution. During execution, the system progressively grounds each subquery through an Act, Review, Update cycle adopted from PAR-RAG (Zhang et al., 2025), storing results in Trajectory Memory $\mathcal{M}_t = \langle (\theta_1, q_1, d_1, a_1), \ldots, (\theta_t, q_t, d_t, a_t) \rangle$, which records the grounded execution state up to step t. In PAR-RAG, the plan module decides whether

each subquery goes through the full Act, Review, Update cycle or is answered directly from Trajectory Memory. GDP-RAG instead executes every step through the full cycle, since Grounded Delta Planning already ensures that only genuine information gaps remain in $\mathcal{T}_{skel}$ (Appendix A.2). Furthermore, Trajectory Memory $\mathcal{M}_{t-1}$ is provided to all three modules at each step, ensuring consistency across the reasoning process.

Each subquery targets a specific information gap that requires dedicated retrieval to resolve. At each step t, the **Act** module retrieves relevant documents from $\mathcal{D}$ for the planned subquery q_t and produces a provisional answer $\hat{a}_t$ conditioned on the query, the retrieved documents, and the current state of Trajectory Memory. Formally:

$$\hat{a}_t = \text{Act}\big(q_t, \mathcal{D}, \mathcal{M}_{t-1}\big), \tag{3}$$

where $\mathcal{M}_{t-1}$ denotes the Trajectory Memory up to step $t-1$. For factual transparency, we adopt a citation-enhanced generation strategy (Li et al., 2024), in which $\hat{a}_t$ is generated by explicitly citing supporting documents from the retrieved set. At this stage, $\hat{a}_t$ remains provisional and has not yet been committed to the final reasoning trajectory.

However, a single retrieval pass may miss relevant documents or introduce hallucinations that propagate through subsequent steps. To mitigate this, the **Review** module validates $\hat{a}_t$ before it is finalized. The system constructs a verification query by concatenating q_t and $\hat{a}_t$, and performs a secondary retrieval over $\mathcal{D}$ to obtain targeted documents. The provisional answer is then examined against the newly retrieved documents to identify and correct factual inconsistencies. Formally, the Review operation is defined as:

$$a_t = \text{Review}\big(q_t, \hat{a}_t, \mathcal{D}, \mathcal{M}_{t-1}\big), \tag{4}$$

where a_t denotes the validated intermediate answer.

Because static plans cannot adapt to information discovered during execution, later subqueries may be redundant or poorly targeted. The **Update** module addresses this by committing $(\theta_t, q_t, d_t, a_t)$ to Trajectory Memory, where d_t is documents cited during Act and Review, and refines the next subquery for runtime adaptability:

$$(\mathcal{M}_t, q_{t+1}) = \text{Update}(q_t, a_t, d_t, \mathcal{M}_{t-1}) \tag{5}$$

where $\mathcal{M}_t$ represents the updated Trajectory Memory and q_{t+1} is the refined subquery for the next step. Using the newly updated trajectory memory, the system refines q_{t+1} to incorporate the latest findings before advancing to the subsequent cycle. The cycle repeats until all N steps are completed, yielding the fully grounded trajectory:

$$\mathcal{T} = \mathcal{M}_N = \langle (\theta_1, q_1, d_1, a_1), \ldots, (\theta_N, q_N, d_N, a_N) \rangle \tag{6}$$

This verified trajectory, containing both the reasoning structure and supporting citations, then serves as the comprehensive context for the final stage.

3.2.3 *Answering Phase*

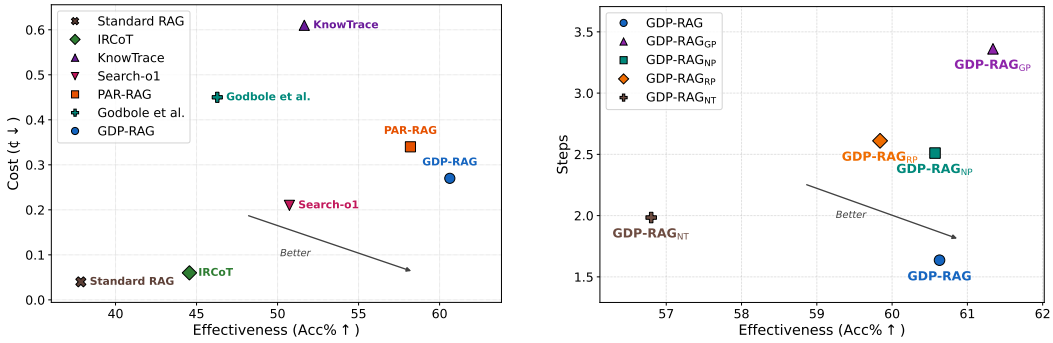
The system synthesizes the completed trajectory $\mathcal{T}$ with query Q to generate the final answer A (Section 3.1), leveraging the verified tuples $(\theta_t, q_t, d_t, a_t)$ in Trajectory Memory to produce a response that is logically consistent and strictly supported by retrieved documents.

4 Experiment Setup

Datasets. We evaluate on three multi-hop QA benchmarks, denoted l1–l3 in all tables: HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), and MuSiQue (Trivedi et al., 2022). We use GPT-4.1-mini, the same model used by all methods for generation, to filter out questions answerable by parametric knowledge alone, and sample a balanced set across complexities. We additionally report results on the full unfiltered splits in Appendix A.3, where the method ranking is preserved, confirming the comparison is not an artifact of filtering. The resulting evaluation sets contain 200 questions per 2/3/4-hop category for HotpotQA and MuSiQue, and a 300/0/300 (2-hop/3-hop/4-hop) distribution for 2WikiMultiHopQA. For HotpotQA the hop label denotes the number of supporting documents, since HotpotQA's questions are inherently 2-hop. Details in Appendix A.4.

Method	Efficiency				Effectiveness				Cost							
	cost-of-pass↓				Acc./%↑				Cost (¢)↓				#Tokens↓			
	all	l1	l2	l3	all	l1	l2	l3	all	l1	l2	l3	all	l1	l2	l3
Standard RAG	**0.14**	**0.05**	**0.09**	**0.29**	37.86	56.41	31.89	25.26	**0.04**	**0.03**	**0.03**	**0.07**	**1077**	**696**	**726**	**1809**
IRCoT	0.17	0.07	0.11	0.31	44.56	54.11	52.99	26.58	0.06	0.04	0.06	0.08	1284	810	1204	1839
KnowTrace	1.57	0.68	0.51	3.52	51.65	55.66	68.31	30.99	0.61	0.38	0.35	1.09	13519	8296	7692	24569
Search-o1	0.49	0.30	0.34	0.81	50.73	51.03	**72.66**	28.51	0.21	0.16	0.25	0.23	4095	2881	5172	4233
PAR-RAG	0.65	0.39	0.43	1.14	58.20	62.79	69.10	42.70	0.34	0.24	0.30	0.49	7112	4856	5956	10523
Godbole et al.	1.12	0.54	0.64	2.16	46.28	59.42	44.83	34.59	0.45	0.32	0.29	0.75	9606	6511	5964	16343
GDP-RAG	0.51	0.25	0.30	0.98	**60.63**	**64.75**	72.62	**44.53**	0.27	0.16	0.22	0.44	5717	3271	4460	9420

Table 1: Main results. all = average of the three per-dataset values, applied uniformly to all metrics. l1/l2/l3 = HotpotQA/2WikiMultiHopQA/MusiQue, each averaged over hops. **Bold** = best, underline = second best.



(a) Baseline Comparison

(b) Planning Ablation

Figure 3: Experimental results of the proposed GDPRAG framework. (a) shows the trade-off between effectiveness and efficiency, while (b) illustrates the impact of planning steps.

Evaluation Metrics. Following the methodology of previous work Wang et al. (2025), we evaluate the RAG system along three axes. For effectiveness, we report Accuracy (Acc), the average of Exact Match (EM), F1 score (F1), and Substring Match (SM). EM checks whether the prediction exactly matches the ground truth, F1 measures token-level overlap, and SM checks whether the ground truth appears as a substring of the prediction. Per-metric definitions are in Appendix A.5, and per-dataset EM/F1/SM scores in Table 6. For cost, we report Cost (¢) and #Tokens, the monetary cost and total tokens consumed per query. For efficiency, we report Cost-of-pass (¢), the expected cost to produce one correct answer, jointly capturing cost and accuracy in a single figure and serving as our primary metric. Detailed formulas and pricing are provided in Appendix A.6.

Baselines. We evaluate against five baselines in two categories. *Step-wise methods* interleave retrieval and reasoning one step at a time: IRCoT (Trivedi et al., 2023) with chain-of-thought reasoning, KnowTrace (Li et al., 2025a) with an iteratively built knowledge graph, and Search-o1 (Li et al., 2025b) with a condensed summary of retrieved documents as its knowledge state. *Plan-based methods* build a plan before execution: PAR-RAG (Zhang et al., 2025) generates all sub-questions upfront from parametric knowledge, and Godbole et al. (Godbole et al., 2024) derive queries from an outline grounded in initial retrieval. We reproduce KnowTrace in its inference-time KG-tracing form only, without the self-bootstrapping fine-tuning, to keep all comparisons training-free. As Godbole et al. (2024) was designed for article writing, we adapt it to QA by keeping only its first planning stage and pairing it with our execution and answering modules, which isolates the planning strategy: it constructs full reasoning paths, whereas GDP-RAG plans only for the information delta.

Implementation Details. All methods use GPT-4.1-mini (OpenAI, 2024) for planning and generation. Documents are encoded with BAAI/bge-m3 (Chen et al., 2024b); top-30

Method	Efficiency				Effectiveness				Cost							
	cost-of-pass↓				Acc./%↑				Cost (¢)↓				#Tokens↓			
	all	l1	l2	l3	all	l1	l2	l3	all	l1	l2	l3	all	l1	l2	l3
Planning Ablation																
GDP-RAG	**0.51**	**0.25**	**0.30**	**0.98**	60.63	64.75	72.62	44.53	**0.27**	**0.16**	0.22	**0.44**	**5717**	**3271**	**4460**	**9420**
GDP-RAG$_{GP}$	0.93	0.57	0.51	1.72	**61.34**	**65.63**	**75.31**	43.07	0.50	0.37	0.38	0.74	10512	7504	7862	16169
GDP-RAG$_{NP}$	0.59	0.33	0.44	1.00	60.57	64.90	71.20	**45.61**	0.33	0.21	0.32	0.46	6857	4288	6442	9841
GDP-RAG$_{RP}$	0.69	0.39	0.48	1.21	59.84	63.37	71.21	44.95	0.38	0.25	0.34	0.54	7962	5013	7008	11866
GDP-RAG$_{NT}$	0.60	0.27	0.31	1.23	56.80	62.40	67.19	40.82	0.29	0.17	**0.21**	0.50	6491	3593	4518	11362
Trajectory Generation Ablation																
GDP-RAG	0.51	0.25	0.30	0.98	**60.63**	**64.75**	72.62	**44.53**	0.27	0.16	0.22	0.44	5717	3271	4460	9420
GDP-RAG$_{NR}$	0.41	0.21	0.24	**0.78**	59.41	62.99	71.50	43.75	0.21	0.13	0.17	0.34	4512	2669	3426	7440
GDP-RAG$_{NU}$	0.51	0.25	0.27	1.01	59.77	64.08	**73.89**	41.33	0.26	0.16	0.20	0.42	5511	3251	4177	9106

Table 2: Ablation study. Planning ablation (top) and Trajectory Generation ablation (bottom).

candidates are reranked by BAAI/reranker-large-v2-m3 (Chen et al., 2024a) to yield the final top-10. Step-wise methods (IRCoT, KnowTrace, Search-o1) are capped at 10 reasoning steps. All experiments run on a single NVIDIA RTX 3090 GPU.

5 Main Results

Table 1 and Figure 3a present the main results. GDP-RAG attains the highest overall accuracy (60.63%), surpassing the strongest baseline PAR-RAG by 2.43 percentage points. It attains the best accuracy on HotpotQA (64.75%) and MusiQue (44.53%), and ranks a close second on 2Wiki (72.62%, just behind Search-o1's 72.66%). The gains confirm that gap-conditioned planning not only reduces unnecessary retrieval but also improves answer quality by focusing on genuinely missing information. A paired significance test (paired t-test and a 10,000-sample bootstrap) confirms that GDP-RAG's improvement over the strongest baseline PAR-RAG is statistically significant on EM and F1 ($p < 0.001$); the full per-baseline analysis is reported in Appendix A.7.

On efficiency, GDP-RAG lies on the Pareto frontier: no method reaches both higher accuracy and lower cost, while the only cheaper methods, Standard RAG (0.04¢), IRCoT (0.06¢), and Search-o1 (0.21¢), trail it by 10 to 23 accuracy points. Cost-of-pass, our primary metric, summarizes this: GDP-RAG scores 0.51, and the only lower scores (Standard RAG 0.14, IRCoT 0.17, Search-o1 0.49) belong to these low-accuracy methods, placing GDP-RAG in the high-accuracy, low-cost corner of Figure 3a. Finally, we examine robustness to model scale: under the weaker GPT-4.1-nano, every method loses accuracy and GDP-RAG's lead narrows, yet it stays cheaper than the strongest baseline PAR-RAG (0.07 vs. 0.09¢) at comparable accuracy (40.00% vs. 41.38%; Appendix A.8).

6 Ablation Study

To analyze each component in GDP-RAG, we conduct an ablation study of both the Planning and Trajectory Generation phases. In the Planning phase, we ablate the three design choices introduced in § 3.2.1: the quality of preliminary D_{rel}, the gap-conditioning instruction, and the skeletal trajectory format. In the Trajectory Generation phase, we ablate the two-stage retrieval (Review) mechanisms and sub-question refinement (Update).

6.1 Planning Phase

Since all planning variants share the same trajectory generation phase, differences reflect purely planning-phase effects (Table 2). Figure 3b visualizes the step count vs. accuracy trade-off, where GDP-RAG occupies the bottom-right (fewest steps, high accuracy).

Preliminary Retrieval. We compare GDP-RAG against two variants: GDP-RAG$_{NP}$ (No Passages), which removes all passages from the planner input, and GDP-RAG$_{RP}$ (Random Passages), which replaces retrieved passages with 10 randomly sampled documents. Without passages (NP), accuracy is nearly unchanged (60.57%) but cost rises 22% (0.33 vs. 0.27 ¢/query) and steps increase to 2.51. With random passages (RP), accuracy drops further (59.84%) and cost is even higher (0.38 ¢/query, 2.61 steps). Notably, RP is worse than NP on both axes, showing that irrelevant passages are worse than no passages at all.

Gap-aware Query Decomposition. We compare GDP-RAG against GDP-RAG$_{GP}$ (Grounded Planning), which removes the gap-conditioning instruction. The planner still receives passages and generates `Thoughts`, but without gap instructions. Without gap-conditioning instruction, accuracy (+0.71%) is marginally higher but cost rises 85% (0.50 vs. 0.27 ¢), generating 3.36 steps versus 1.64 due to redundant sub-questions. Even GDP-RAG$_{NP}$ generates fewer steps (2.51) than GP (3.36), confirming that gap instruction drives step reduction independent of passages.

Skeletal Trajectory. We compare GDP-RAG against GDP-RAG$_{NT}$ (No Thought), which removes the Thought component (θ_t) from the skeletal trajectory, reducing each entry to only q_t. Without θ_t, evidence from D_{rel} is not preserved through execution. NT causes the largest accuracy drop among all planning ablations (56.80% vs. 60.63%), while cost remains similar (0.29 vs. 0.27 ¢). The skeletal trajectory format is critical for accuracy: it improves sub-question quality rather than merely reducing step count.

6.2 Trajectory Generation Phase

To evaluate the trajectory generation phase, we ablate the Review and Update modules while keeping the planning stage constant (Table 2). This setup isolates the effects of each module on reasoning quality and computational cost.

Review. We compare GDP-RAG against GDP-RAG$_{NR}$ (No Review), which uses the provisional answer directly without secondary retrieval or verification. While NR reduces cost-of-pass by 20% (0.41 vs. 0.51 ¢) and tokens by 21%, it causes only a marginal accuracy loss (59.41% vs. 60.63%). This indicates that Review is the dominant cost component in trajectory generation but contributes modestly to final accuracy.

Update. We compare GDP-RAG against GDP-RAG$_{NU}$ (No Update), which disables the refinement of planned sub-questions based on previous trajectory steps. NU yields negligible cost savings (around 4%) but results in a consistent accuracy drop (59.77% vs. 60.63%). Since sub-question refinement requires only one additional LLM call per step, Update is essentially a low-cost yet effective mechanism for maintaining runtime adaptability.

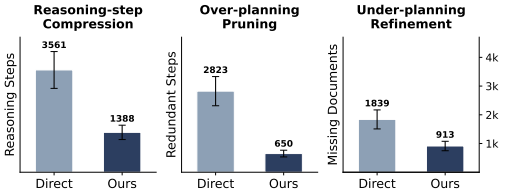
Update should always be included as it requires only one additional LLM call per step while providing runtime adaptability. Review presents a cost-accuracy trade-off: it dominates cost but contributes modestly to accuracy, exposing two operating points — full GDP-RAG for higher accuracy and GDP-RAG$_{NR}$ for lower cost.

7 Effectiveness Analysis of GDP-RAG

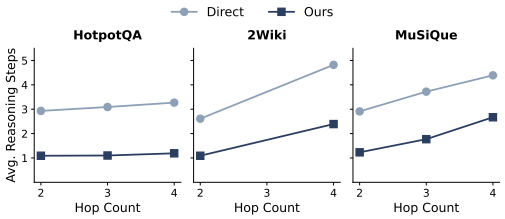
Beyond aggregate accuracy, we examine how Grounded Delta Planning reshapes the planning process itself. We compare against Direct Planning (without preliminary retrieval and gap-aware instruction). A qualitative case study is provided in Appendix A.12, Table **??**.

7.1 Compression, Pruning, and Refinement.

Figure 4a quantifies the impact: an LLM-based evaluator labels each planned step as a Reasoning Step that targets a gold document, a Redundant Step that over-plans, or a Missing Document not covered by the plan, with 82% agreement against human review on a 10% sample. Classification criteria are detailed in Appendix A.10.



(a) Reasoning Steps, Redundant Steps, and Missing Documents for Direct vs. Ours, evaluated by LLM-based alignment with 82% human-verified accuracy (details in Appendix A.10).



(b) Average reasoning steps across 2- to 4-hop queries for Direct Planning vs. Grounded Delta Planning on three datasets. The gap widens as hop count increases.

Figure 4: Comparison of planning statistics (a) and reasoning complexity by query hops (b).

GDP-RAG improves on all three dimensions: Reasoning-step Compression cuts valid steps from 3,561 to 1,388 (61.0%), Over-planning Pruning cuts redundant steps from 2,823 to 650 (77.0%), and Under-planning Refinement lowers missing gold documents from 1,839 to 913 (50.4%), confirming that grounding the planner both reduces overhead and strengthens coverage. To prove the gains are not an artifact of a weak planner, we use GPT-5.5 as the LLM in Appendix A.11 and preserve the compression and pruning improvements.

The compression effect is governed by how much gold evidence the preliminary retrieval already covers. Its recall@10 declines as the hop count grows, averaging 0.61 across datasets (per-dataset, per-hop breakdown in Appendix Table 11). Because each reasoning step typically resolves a single passage, the fraction of steps the planner can compress tracks this coverage: the 61.0% Reasoning-step Compression closely matches the 0.61 average recall.

7.2 Efficiency under Increasing Query Complexity.

Figure 4b compares average reasoning steps across 2- to 4-hop queries. Grounded Delta Planning consistently requires fewer steps, with the gap widening as complexity increases. On HotpotQA and 2Wiki, Grounded Delta Planning maintains nearly constant step count, amortizing additional hops. On MuSiQue, although steps increase linearly, GDP-RAG maintains an efficiency advantage. These results show that Grounded Delta Planning adapts to task complexity, pruning redundant steps while retaining necessary reasoning.

8 Conclusion

We presented GDP-RAG, a plan-based multi-hop RAG framework that retrieves first and plans only for the information delta (Δ) through preliminary retrieval, gap-aware query decomposition, and a skeletal trajectory. An Act-Review-Update cycle grounds each subquery with runtime adaptability. Experiments on HotpotQA, 2WikiMultiHopQA, and MuSiQue show that GDP-RAG achieves the highest accuracy among all compared systems with a cost-of-pass 22% lower than PAR-RAG and 68% lower than KnowTrace. Ablations confirm that each component contributes: preliminary retrieval improves step efficiency, the gap-aware query decomposition drives efficiency, and the skeletal trajectory drives accuracy.

9 Limitations and Future Work

Our evaluation targets short-form multi-hop QA, where information gaps are defined over factual entities and relations and answers are short and verifiable. Open-ended generation, long-form synthesis, and multi-modal retrieval would require a different notion of the information delta and a different evaluation metric. Extending GDP-RAG to such open-ended and agentic settings is a natural direction for future work.

Acknowledgments

This work was supported by the National Science and Technology Council (NSTC), Taiwan, under Grant No. 113-2740-H-002-001-MY3, "TAIHUCAIS: TAIwan HUmanities Conversational AI Knowledge Discovery System." This work was also financially supported by the National Science and Technology Council (NSTC) in Taiwan, under Grant 114-2628-E-002-022. This work was supported by the Ministry of Education (MOE) of Taiwan under the project Taiwan Centers of Excellence in Artificial Intelligence, through the NTU Artificial Intelligence Center of Research Excellence (NTU AI-CoRE). We also thank the National Center for High-performance Computing (NCHC) of the National Applied Research Laboratories (NARLabs), Taiwan, for providing computational and storage resources.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *ICLR 2024*, 2024. URL https://openreview.net/forum?id=hSyW5go0v8.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. BGE-Reranker-v2-M3: A multilingual cross-encoder reranker, 2024a. URL https://huggingface.co/BAAI/bge-reranker-v2-m3.

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Findings of ACL 2024*, pp. 2318–2335, 2024b. doi: 10.18653/v1/2024.findings-acl.137. URL https://doi.org/10.18653/v1/2024.findings-acl.137.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. Chain-of-verification reduces hallucination in large language models. In *Findings of ACL 2024*, pp. 3563–3578, 2024. doi: 10.18653/v1/2024.findings-acl.212. URL https://doi.org/10.18653/v1/2024.findings-acl.212.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Xiang Ren, Allyson Ettinger, Zaïd Harchaoui, and Yejin Choi. Faith and fate: Limits of transformers on compositionality. In *Advances in Neural Information Processing Systems (NeurIPS 2023)*, 2023. URL https://papers.nips.cc/paper_files/paper/2023/hash/deb3c28192f979302c157cb653c15e90-Abstract-Conference.html.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. From local to global: A graph RAG approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024. doi: 10.48550/arXiv.2404.16130. URL https://arxiv.org/abs/2404.16130.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise zero-shot dense retrieval without relevance labels. In *ACL 2023*, pp. 1762–1777, 2023a. doi: 10.18653/v1/2023.acl-long.99. URL https://doi.org/10.18653/v1/2023.acl-long.99.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023b. doi: 10.48550/arXiv.2312.10997. URL https://arxiv.org/abs/2312.10997.

Ameya Godbole, Nicholas Monath, Seungyeon Kim, Ankit Singh Rawat, Andrew McCallum, and Manzil Zaheer. Analysis of plan-based retrieval for grounded text generation. In *EMNLP 2024*, pp. 13101–13119, 2024. doi: 10.18653/v1/2024.emnlp-main.727. URL https://doi.org/10.18653/v1/2024.emnlp-main.727.

Bernal Jimenez Gutierrez, 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 (NeurIPS*

2024), 2024. URL https://papers.nips.cc/paper_files/paper/2024/hash/6ddc001d07ca4f319af96a3024f6dbd1-Abstract-Conference.html.

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 COLING 2020*, pp. 6609–6625, 2020. doi: 10.18653/v1/2020.coling-main.580. URL https://doi.org/10.18653/v1/2020.coling-main.580.

Abhinav Java, Srivathsan Koundinyan, Nagarajan Natarajan, and Amit Sharma. Frugalrag: Less is more in rl finetuning for multi-hop question answering. In *The Fourteenth International Conference on Learning Representations (ICLR 2026)*, 2026. URL https://arxiv.org/abs/2507.07634.

Pengcheng Jiang, Lang Cao, Ruike Zhu, Minhao Jiang, Yunyi Zhang, Jimeng Sun, and Jiawei Han. RAS: retrieval-and-structuring for knowledge-intensive LLM generation. *arXiv preprint arXiv:2502.10996*, 2025. doi: 10.48550/arXiv.2502.10996. URL https://arxiv.org/abs/2502.10996.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O. Arık, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. In *Conference on Language Modeling (COLM 2025)*, 2025. URL https://arxiv.org/abs/2503.09516.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *NeurIPS 2022*, 2022. URL https://papers.nips.cc/paper_files/paper/2022/hash/8bb0d291acd4acf06ef112099c16f326-Abstract-Conference.html.

Myeonghwa Lee, Seonho An, and Min-Soo Kim. PlanRAG: A plan-then-retrieval augmented generation framework for large language models as decision makers. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: NAACL 2024*, pp. 6537–6555, 2024. doi: 10.18653/v1/2024.naacl-long.364. URL https://aclanthology.org/2024.naacl-long.364.

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 (NeurIPS 2020)*, 2020. URL https://papers.nips.cc/paper_files/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Cheng-Yen Li, Xuanjun Chen, Claire Lin, Wei-Yu Chen, Wenhua Nie, Hung-Yi Lee, and Jyh-Shing Roger Jang. Codarag: Connecting the dots with associativity inspired by complementary learning. *arXiv preprint arXiv:2604.10426*, 2026. URL https://arxiv.org/abs/2604.10426.

Rui Li, Quanyu Dai, Zeyu Zhang, Xu Chen, Zhenhua Dong, and Ji-Rong Wen. Knowtrace: Bootstrapping iterative retrieval-augmented generation with structured knowledge tracing. In *KDD 2025*, pp. 1470–1480, 2025a. doi: 10.1145/3711896.3737015. URL https://doi.org/10.1145/3711896.3737015.

Ruosen Li and Xinya Du. Leveraging structured information for explainable multi-hop question answering and reasoning. In *Findings of EMNLP 2023*, pp. 6779–6789, 2023. doi: 10.18653/v1/2023.findings-emnlp.452. URL https://doi.org/10.18653/v1/2023.findings-emnlp.452.

Weitao Li, Junkai Li, Weizhi Ma, and Yang Liu. Citation-enhanced generation for llm-based chatbots. In *ACL 2024*, pp. 1451–1466, 2024. doi: 10.18653/v1/2024.acl-long.79. URL https://doi.org/10.18653/v1/2024.acl-long.79.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. In *EMNLP 2025*, pp. 5420–5438, 2025b. doi: 10.18653/v1/2025.emnlp-main.276. URL https://doi.org/10.18653/v1/2025.emnlp-main.276.

Claire Lin, Bo-Han Feng, Xuanjun Chen, Te-Lun Yang, Hung-yi Lee, and Jyh-Shing Roger Jang. A preliminary study of RAG for taiwanese historical archives. *arXiv preprint arXiv:2511.07445*, 2025. doi: 10.48550/arXiv.2511.07445.

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*, 12:157–173, 2024a. doi: 10.1162/tacl_a_00638. URL https://doi.org/10.1162/tacl_a_00638.

Yanming Liu, Xinyue Peng, Tianyu Du, Jianwei Yin, Weihao Liu, and Xuhong Zhang. Eracot: Improving chain-of-thought through entity relationship analysis. In *ACL 2024*, pp. 8780–8794, 2024b. doi: 10.18653/v1/2024.acl-long.476. URL https://doi.org/10.18653/v1/2024.acl-long.476.

Thang Nguyen, Peter Chin, and Yu-Wing Tai. MA-RAG: multi-agent retrieval-augmented generation via collaborative chain-of-thought reasoning. *arXiv preprint arXiv:2505.20096*, 2025. doi: 10.48550/arXiv.2505.20096. URL https://arxiv.org/abs/2505.20096.

OpenAI. Gpt-4.1 mini, 2024. URL https://openai.com/.

Ofir Press, Muru 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*, pp. 5687–5711, 2023. doi: 10.18653/v1/2023.findings-emnlp.378. URL https://doi.org/10.18653/v1/2023.findings-emnlp.378.

Yijia Shao, Yucheng Jiang, Theodore A. Kanell, Peter Xu, Omar Khattab, and Monica S. Lam. Assisting in writing wikipedia-like articles from scratch with large language models. In *NAACL 2024*, pp. 6252–6278, 2024. doi: 10.18653/v1/2024.naacl-long.347. URL https://doi.org/10.18653/v1/2024.naacl-long.347.

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*, pp. 9248–9274, 2023. doi: 10.18653/v1/2023.findings-emnlp.620. URL https://doi.org/10.18653/v1/2023.findings-emnlp.620.

Jimeng Shi, Wei Hu, Runchu Tian, Bowen Jin, Wonbin Kweon, SeongKu Kang, Yunfan Kang, Dingqi Ye, Sizhe Zhou, Shaowen Wang, and Jiawei Han. Multicube-rag for multi-hop question answering. *arXiv preprint arXiv:2602.15898*, 2026. doi: 10.48550/arXiv.2602.15898. URL https://arxiv.org/abs/2602.15898.

Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. Agentic retrieval-augmented generation: A survey on agentic RAG. *arXiv preprint arXiv:2501.09136*, 2025. doi: 10.48550/arXiv.2501.09136. URL https://arxiv.org/abs/2501.09136.

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

Alon Talmor and Jonathan Berant. The web as a knowledge-base for answering complex questions. In *NAACL 2018*, pp. 641–651, 2018. doi: 10.18653/v1/n18-1059. URL https://doi.org/10.18653/v1/n18-1059.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022. doi: 10.1162/tacl_a_00475. URL https://doi.org/10.1162/tacl_a_00475.

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 the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10014–10037. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-long.557.

Prakhar Verma, Sukruta Prakash Midigeshi, Gaurav Sinha, Arno Solin, Nagarajan Natarajan, and Amit Sharma. Plan-RAG: Planning-guided retrieval-augmented generation. *arXiv preprint arXiv:2410.20753*, 2024. doi: 10.48550/arXiv.2410.20753. URL https://arxiv.org/abs/2410.20753.

Junjie Wang, Mingyang Chen, Binbin Hu, Dan Yang, Ziqi Liu, Yue Shen, Peng Wei, Zhiqiang Zhang, Jinjie Gu, Jun Zhou, Jeff Z. Pan, Wen Zhang, and Huajun Chen. Learning to plan for retrieval-augmented large language models from knowledge graphs. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 7813–7835, 2024a. doi: 10.18653/v1/2024.findings-emnlp.459. URL https://aclanthology.org/2024.findings-emnlp.459.

Ningning Wang, Xavier Hu, Pai Liu, He Zhu, Yue Hou, Heyuan Huang, Shengyu Zhang, Jian Yang, Jiaheng Liu, Ge Zhang, Changwang Zhang, Jun Wang, Yuchen Eleanor Jiang, and Wangchunshu Zhou. Efficient agents: Building effective agents while reducing cost. *arXiv preprint arXiv:2508.02694*, 2025. doi: 10.48550/arXiv.2508.02694. URL https://arxiv.org/abs/2508.02694.

Zihao Wang, Anji Liu, Haowei Lin, Jiaqi Li, Xiaojian Ma, and Yitao Liang. RAT: retrieval augmented thoughts elicit context-aware reasoning in long-horizon generation. *arXiv preprint arXiv:2403.05313*, 2024b. doi: 10.48550/arXiv.2403.05313. URL https://arxiv.org/abs/2403.05313.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS 2022*, 2022. URL https://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. Constructing datasets for multi-hop reading comprehension across documents. *TACL*, 6:287–302, 2018. doi: 10.1162/tacl_a_00021. URL https://doi.org/10.1162/tacl_a_00021.

Ruofan Wu, Youngwon Lee, Fan Shu, Danmei Xu, Seung won Hwang, Zhewei Yao, Yuxiong He, and Feng Yan. Composerag: A modular and composable RAG for corpus-grounded multi-hop question answering. *arXiv preprint arXiv:2506.00232*, 2025. doi: 10.48550/arXiv.2506.00232. URL https://arxiv.org/abs/2506.00232.

Yu Xia, Junda Wu, Sungchul Kim, Tong Yu, Ryan A. Rossi, Haoliang Wang, and Julian J. McAuley. Knowledge-aware query expansion with large language models for textual and relational retrieval. In *NAACL 2025*, pp. 4275–4286, 2025. doi: 10.18653/v1/2025.naacl-long.216. URL https://doi.org/10.18653/v1/2025.naacl-long.216.

Binfeng Xu, Zhiyuan Peng, Bowen Lei, Subhabrata Mukherjee, Yuchen Liu, and Dongkuan Xu. Rewoo: Decoupling reasoning from observations for efficient augmented language models. *arXiv preprint arXiv:2305.18323*, 2023. doi: 10.48550/arXiv.2305.18323. URL https://arxiv.org/abs/2305.18323.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884*, 2024. doi: 10.48550/arXiv.2401.15884. URL https://arxiv.org/abs/2401.15884.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. 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 (EMNLP 2018)*, pp. 2369–2380, 2018. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations (ICLR 2023)*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Ningning Zhang, Chi Zhang, Zhizhong Tan, Xingxing Yang, Weiping Deng, and Wenyong Wang. Credible plan-driven RAG method for multi-hop question answering. *arXiv preprint arXiv:2504.16787*, 2025. doi: 10.48550/arXiv.2504.16787. URL https://arxiv.org/abs/2504.16787.

Ruochen Zhao, Xingxuan Li, Shafiq Joty, Chengwei Qin, and Lidong Bing. Verify-and-edit: A knowledge-enhanced chain-of-thought framework. In *ACL 2023*, pp. 5823–5840, 2023. doi: 10.18653/v1/2023.acl-long.320. URL https://doi.org/10.18653/v1/2023.acl-long.320.

A Appendix

A.1 Prompt Design

This section details the transition from the baseline Direct Planning prompt to our proposed Grounded Planning formulation.

```
Think carefully, build a step-by-step plan for this question: {question} using JSON format.
  For each step in the plan, generate a sub-question.

# Format instructions
Use the following Strict format, only choose an action from the list:[Retrieve, Answer]:
[
  {{
    "Thought":"[your thought about the current step]",
    "Question":"[the question you generated for the current step]",
    "Action": "[the action you chose]"
  }}
]
Return only the JSON array, no explanation.
```

Figure 5: Prompt of Direct Planning.

```
Think carefully, build a step-by-step plan for this question: {question} using JSON format.
  For each step in the plan, generate a sub-question.
{formatted_passages}

IMPORTANT: Before creating steps, analyze what information you already have:
1. Identify what specific information is already available in the passages
2. Only create steps for information that is NOT already in the passages
For each step in the plan:
- Write down known information from the passages in "Thought" so it is carried forward
 - Only ask for information that is genuinely missing
# Format instructions
[
  {{
      "Thought": "[summarize what is already known from the passages and explain what this step still needs to find]",
      "Question": "[the sub-question for the missing information]",
      "Action": "Retrieve"
   }}
]
Return only the JSON array, no explanation.
```

Figure 6: Prompt of Grounded Delta Planning.

Direct Planning Prompt As shown in Figure 5, our Direct Planning prompt is adapted from PAR-RAG, requiring the LLM to generate a structured JSON plan. For each reasoning step, the model must produce a `Thought` process and a specific subquery, while restricting its choice of actions to a predefined list: [`Retrieve`, `Answer`]. These generated actions serve as the foundational candidates for our subsequent conditional execution mechanism, allowing the system to determine which steps require external grounding.

Retrieve: This action triggers the full Act-Review-Update cycle, where the system actively gathers new information and verifies its relevance to the reasoning trajectory.

Answer: This action indicates that the current trajectory memory contains sufficient information; the model then formulates a response based on it without further searching.

Grounded Delta Planning Prompt As shown in Figure 6, the Grounded Delta Planning prompt incorporates preliminary retrieval results and enforces two key instructions: (1) identify what information is already available in the retrieved passages, and (2) only create steps for information not already present. The resulting JSON output requires a Thought field to justify why specific information is not yet available and a Question field to define the missing details. Unlike the Direct Planning prompt, no predefined action list is provided.

A.2 Comparison with PAR-RAG

Although GDP-RAG builds upon PAR-RAG (Zhang et al., 2025), the two frameworks differ in a key design choice: what to execute. PAR-RAG generates a full reasoning trajectory from parametric knowledge and conditionally decides, at each step, whether to retrieve externally or answer from previously generated intermediate results (see Appendix A.1). GDP-RAG inverts this logic: because Grounded Delta Planning already eliminates redundant and trivial steps during planning, every subquery in $\mathcal{T}_{skel}$ corresponds to a genuine information gap. The system therefore passes through the full Act–Review–Update loop every planned step, increasing reliability by ensuring that no intermediate answer relies on unverified parametric reasoning. GDP-RAG further strengthens consistency by sharing Trajectory Memory across the Act and Review modules and by committing all cited documents from both phases. Only documents that directly support each intermediate answer a_t are stored as d_t, yielding a compact and fully verifiable trajectory $\mathcal{T}$.

A.3 Filtered vs. Unfiltered Evaluation

To verify that knowledge-intensive filtering does not drive our results, we additionally evaluate on the full unfiltered splits (600 questions per dataset). Table 3 reports accuracy and cost on both settings, averaged over the three datasets. The method ranking is identical across the two splits: GDP-RAG attains the highest accuracy on both, and every method's relative ordering is preserved. We adopt the filtered set as the main setting because it removes questions answerable without retrieval, yielding cleaner ablations. The unfiltered split confirms the comparison is not an artifact of filtering.

Method	Filtered		Unfiltered	
	Acc%↑	Cost ¢↓	Acc%↑	Cost ¢↓
Standard RAG	37.85	0.04	46.17	0.04
IRCoT	44.56	0.06	50.25	0.06
KnowTrace	51.65	0.61	52.47	0.57
Godbole et al.	46.28	0.45	52.16	0.45
PAR-RAG	58.20	0.34	62.87	0.35
GDP-RAG	**60.63**	0.27	**64.09**	0.27

Table 3: Filtered vs. unfiltered evaluation. Each cell is accuracy (%) and cost (¢/query), averaged over the three datasets and 600 questions each on the unfiltered split. The method ranking is identical across both settings.

A.4 Dataset Curation Two Stage Processing

Knowledge-Intensive Filtering. To ensure retrieval dependency, we exclude questions solvable by the LLM's parametric knowledge (Table 4). We identify such cases via Substring Match and evaluate only the remaining "Wrong" subset.

QA Dataset	Wrong(%)	Correct(%)
HotpotQA	62.38	37.62
2Wiki	63.80	36.20
MuSiQue	88.95	11.05

Table 4: Justification for Dataset Filtering.

Hop-Balanced Sampling. We sample 600 questions per dataset. HotpotQA and MuSiQue use a uniform 200/200/200 split for 2/3/4 hops. 2Wiki uses 300/0/300 due to lack of 3-hop data. This prevents lower-hop dominance and enables scalability evaluation.

A.5 Effectiveness Metrics

Question: Whose father is older, Amy or Bob? **Ground Truth :** Bob				
Predictions	**Tokenization**	**EM**	**F1**	**SM**
1) Bob.	Bob	100%	100%	100%
2) Bob's father is older.	Bob/'s/father/is/older	0%	33%	100%
3) Bob's father is older than John's.	Bob/'s/father/is/older/than/John/'s	0%	22%	100%

Table 5: Example of Quality Metric Calculation for Question Answering.

Exact Match (EM) It tests the final synthesis quality. This metric tells us if the LLM successfully converted all the retrieved and reasoned facts into the final answer. This is a binary metric that scores 100% for a perfect match and 0% otherwise. As seen in Table 5, Prediction 2 and 3 both scores 0% EM; only Prediction 1 scores 100%.

F1 score It reflects partial correctness while simultaneously penalizing the model for extraneous content. It is calculated by determining the token-level Precision and Recall of the predicted answer against the ground truth. The final score is the harmonic mean of these two values, calculated using the function:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

As shown in Table 5, the score drops as prediction length increases. For instance, while Recall remains perfect, the Precision drops from 1 (Prediction 1) to 1/5 (Prediction 2) to 1/8 (Prediction 3), causing F1 to decline from 100% to 22%.

Substring Match (SM) It relaxes the need for perfect phrasing and rewards factual completeness, even if the prediction includes additional context or words. This is a forgiving, binary metric that scores 100% if the entire ground-truth answer is a substring within the predicted answer, and 0% otherwise. As shown in Table 5, all three predictions contain the ground truth ("Bob"), so they all achieve an SM score of **100**%.

A.6 Cost and Efficiency Metrics

Cost (¢). The monetary cost of a single inference attempt, computed as $C = n_{in} \cdot c_{in} + n_{out} \cdot c_{out}$, where n_{in} and n_{out} are the input and output token counts, and c_{in}, c_{out} are the per-token prices (GPT-4.1-mini: \$0.40/1M input, \$1.60/1M output).

#Tokens. The total tokens (input + output) consumed per question, reflecting the overall computational demand.

Cost-of-pass (¢). The expected monetary cost to produce one correct answer, defined as $Cost - of - pass = C \ / \ Acc$ (Wang et al., 2025). By normalizing cost by accuracy, this metric penalizes both expensive systems and inaccurate ones, serving as our primary efficiency–effectiveness metric.

Method	HotpotQA			2Wiki			MusiQue		
	EM	F1	SM	EM	F1	SM	EM	F1	SM
IRCoT	41.67	58.83	61.83	44.00	51.81	63.16	16.33	29.24	34.17
KnowTrace	41.50	59.47	66.00	55.16	69.93	**79.83**	23.33	34.65	35.00
PARRAG	47.67	68.03	**72.67**	60.16	69.48	77.66	30.17	46.93	**51.00**
Godbole et al.	47.00	64.58	66.67	33.00	40.32	61.16	25.17	39.44	39.17
GDPRAG	**52.33**	**70.40**	71.50	**66.67**	**73.53**	77.67	**34.67**	**48.43**	50.50

Table 6: Effectiveness breakdown by dataset. We report EM, F1, and SM (all in %) for each method on HotpotQA, 2Wiki, and MusiQue.

A.7 Statistical Significance

We test whether GDP-RAG's improvements are statistically significant using a paired one-sided t-test, with a paired bootstrap of 10,000 resamples as a robustness check. Tests are run per question over the full filtered set ($n = 1800$). Table 7 reports, for each metric, the mean per-question difference Δ (in points), the win rate (Win%, the fraction of non-tied questions on which GDP-RAG scores higher), and the p-value. GDP-RAG significantly outperforms every external baseline on all three metrics ($p < 0.001$). Against the strongest baseline PAR-RAG, the gains on EM and F1 are significant while SM saturates (Win% $\approx$ 50), so including SM in the Accuracy average makes our headline conservative rather than inflated. Among the ablations, the accuracy-oriented components (skeletal Thought, Review, Update) yield significant gains, whereas the efficiency-oriented planning ablations (GP, NP) are not significant, consistent with their role of reducing steps rather than improving accuracy.

GDP-RAG vs.	EM			F1			SM		
	Δ	Win%	p	Δ	Win%	p	Δ	Win%	p
Baselines									
Standard RAG	+20.61	85.2	<.001	+22.58	80.0	<.001	+26.72	88.4	<.001
IRCoT	+17.72	82.0	<.001	+17.97	75.9	<.001	+19.44	81.1	<.001
KnowTrace	+11.72	74.0	<.001	+9.91	69.1	<.001	+6.89	65.2	<.001
Godbole et al.	+16.67	83.5	<.001	+16.48	73.2	<.001	+11.50	74.0	<.001
PAR-RAG	+5.72	65.7	<.001	+3.11	58.7	<.001	+0.06	50.2	.48
Planning-stage ablations									
GDP-RAG$_{GP}$	+0.33	51.4	.34	+0.03	51.1	.49	−0.89	46.7	.85
GDP-RAG$_{NP}$	+0.78	52.8	.19	+0.77	51.3	.17	+0.22	50.8	.40
GDP-RAG$_{RP}$	+1.72	56.7	.02	+1.34	53.0	.04	+0.89	53.1	.16
GDP-RAG$_{NT}$	+3.89	63.4	<.001	+4.20	61.0	<.001	+4.95	65.1	<.001
Execution-stage ablations									
GDP-RAG$_{NR}$	+1.94	58.7	.007	+2.13	56.1	.001	+1.17	54.8	.08
GDP-RAG$_{NU}$	+1.44	56.4	.03	+1.62	55.7	.01	+1.11	54.5	.09

Table 7: Statistical significance of GDP-RAG's per-question improvements over each baseline and ablation, on EM, F1, and SM. Δ: mean score difference (points); Win%: fraction of non-tied questions GDP-RAG wins; p: one-sided paired t-test (the bootstrap agrees throughout).

A.8 Robustness to Model Scale (GPT-4.1-nano)

To test whether our findings depend on a particular backbone, we rerun every method with the weaker GPT-4.1-nano. Table 8 reports macro accuracy and cost at both scales. Every method loses accuracy at the smaller scale, confirming the benchmark remains

discriminative. GDP-RAG leads on accuracy at the mini scale; at the nano scale its accuracy lead narrows to a close second behind PAR-RAG (40.00 vs. 41.38), though it stays cheaper than PAR-RAG (0.07 vs. 0.09¢ per query).

Method	GPT-4.1-mini		GPT-4.1-nano	
	Acc%↑	Cost ¢↓	Acc%↑	Cost ¢↓
Standard RAG	37.86	0.04	24.11	0.01
IRCoT	44.56	0.06	35.78	0.01
Godbole et al.	46.28	0.45	32.84	0.11
KnowTrace	51.65	0.61	32.50	0.20
PAR-RAG	58.20	0.34	**41.38**	0.09
GDP-RAG	**60.63**	0.27	40.00	0.07
GDP-RAG$_{NR}$	59.41	0.21	37.79	0.06

Table 8: Robustness to model scale. Macro accuracy and cost per query under GPT-4.1-mini and GPT-4.1-nano, averaged over the three datasets. **Bold** = best accuracy at each scale. Every method loses accuracy at the smaller scale.

To locate where the nano gap arises, Table 9 compares GDP-RAG against PAR-RAG, the strongest baseline, on every split at both scales. GDP-RAG matches or beats PAR-RAG on six of the eight nano splits. The two substantive deficits, 2WikiMultiHopQA 4-hop and MuSiQue 4-hop, are precisely the splits where GDP-RAG most outperforms PAR-RAG at the mini scale (+6.87 and +4.59). This reflects a tradeoff inherent to compression: GDP-RAG distills the plan to the missing facts and leaves the final compositional step to the answering model, which a capable model completes but a tiny one does not, whereas PAR-RAG's longer uncompressed trajectory carries spare steps that absorb a weak model's mistakes.

Split	GPT-4.1-mini			GPT-4.1-nano		
	PAR-RAG	GDP-RAG	Δ	PAR-RAG	GDP-RAG	Δ
HotpotQA 2-hop	62.85	63.11	+0.26	54.82	57.12	+2.30
HotpotQA 3-hop	60.16	65.20	+5.04	48.11	48.44	+0.33
HotpotQA 4-hop	65.35	65.93	+0.58	49.97	49.50	−0.47
2Wiki 2-hop	63.34	63.51	+0.17	54.72	55.40	+0.69
2Wiki 4-hop	74.87	81.74	+6.87	32.60	19.72	−12.88
MuSiQue 2-hop	57.40	54.52	−2.88	39.79	41.24	+1.45
MuSiQue 3-hop	39.44	43.24	+3.80	21.18	29.71	+8.53
MuSiQue 4-hop	31.25	35.84	+4.59	27.56	21.33	−6.23

Table 9: Per-split accuracy (%) of GDP-RAG vs. PAR-RAG at both model scales. Δ = GDP-RAG − PAR-RAG. GDP-RAG matches or beats PAR-RAG on six of eight nano splits; the two deficits (2Wiki 4-hop, MuSiQue 4-hop) are the deepest-hop splits where GDP-RAG most outperforms PAR-RAG at mini.

A.9 Carry-forward Control: Delta Planning vs. Evidence Access

A natural question is whether GDP-RAG's gains come from carrying preliminary evidence forward rather than from delta planning itself. To isolate this, we give the Direct Planning baseline (PAR-RAG) the same preliminary passages D_{rel}, while leaving its planner unchanged—Direct Planning, with no gap-conditioning and no skeletal Thought. D_{rel} is injected as side context either only into the final Answer module, or into every execution step (Act, Review, Update, and Answer); the retriever, model, and decoding all match the main experiments. As Table 10 shows, both variants fall below PAR-RAG alone while costing more, and neither approaches GDP-RAG. Simply carrying D_{rel} forward, without delta planning to distill it into the skeletal Thought, adds noise rather than signal: the benefit lies in *how* the evidence is used, not in whether it is available.

Method	cost-of-pass↓				Acc./%↑				Cost (¢)↓			
	all	l1	l2	l3	all	l1	l2	l3	all	l1	l2	l3
PAR-RAG	0.65	0.39	0.43	1.14	58.20	62.79	69.10	42.70	0.34	0.24	0.30	0.49
PAR-RAG + D_{rel} (answer only)	0.73	0.41	0.46	1.31	57.32	62.38	68.74	40.85	0.37	0.26	0.32	0.54
PAR-RAG + D_{rel} (every step)	0.92	0.46	0.54	1.77	55.22	64.31	61.07	40.28	0.45	0.30	0.33	0.71
GDP-RAG	**0.51**	**0.25**	**0.30**	**0.98**	**60.63**	**64.75**	**72.62**	**44.53**	**0.27**	**0.16**	**0.22**	**0.44**

Table 10: Carry-forward control. Direct Planning (PAR-RAG) given the same preliminary retrieval D_{rel}, injected at the final answer only or at every execution step, with all other settings matched to the main experiments. Adding D_{rel} without delta planning does not reach GDP-RAG and even falls below PAR-RAG alone. **Bold** = best.

A.10 Grounded Delta Planning Advantages Analysis

To quantify the efficacy of Grounded Delta Planning, we perform an LLM-based alignment. We feed the reasoning steps generated by both Direct Planning and Grounded Delta Planning alongside the Gold Documents into an LLM to identify valid mappings. Based on this alignment, the metrics in Figure 4a are defined as follows:

Reasoning-step Compression. Under the same alignment criteria, we measure valid reasoning steps by counting generated queries that match gold documents. The reduction in this count from Direct to Grounded Planning represents necessary steps saved because the planner recognizes information already available in the retrieved context.

Dataset	2-hop	3-hop	4-hop	Avg
HotpotQA	0.86	0.77	0.71	0.78
2WikiMultiHopQA	0.66	—	0.50	0.58
MuSiQue	0.70	0.44	0.24	0.46

Table 11: Preliminary retrieval recall@10 by dataset and hop (macro average 0.61). Recall declines with hop count and closely tracks the 61.0% Reasoning-step Compression (Section 7).

Over-planning Pruning. Under the same alignment criteria, we quantify redundant planning by counting steps that fail to align with any gold document. The difference between the two settings represents redundant queries eliminated through grounding, as the planner avoids generating steps for irrelevant or already-covered information.

Under-planning Refinement. This metric quantifies missed information by counting gold documents not covered by the system. While Direct Planning is evaluated against generated reasoning steps alone, Grounded Delta Planning is measured against the union of preliminary retrieval (D_{rel}) and the generated plan. The reduction in missing documents represents information recovered by incorporating initial retrieval into the planning process, ensuring that essential details overlooked by the planner are still accounted for.

Step-matching Validation We validated LLM accuracy via human review on a 10% sample (Table 12). Four researchers, each holding at least a bachelor's degree, conducted this audit to ensure the expertise required for evaluating complex multi-hop reasoning. For both settings, we verified whether each step-matching output correctly matched a gold document or was accurately identified as redundant.

A.11 Plan-level Analysis under a Frontier Planner

A natural question is whether a stronger planner already produces concise plans on its own, making Grounded Delta Planning unnecessary. To test this, we repeat the plan-level analysis of Section 7 with the frontier model GPT-5.5 as the planner, reporting the necessary and

Dataset	Total	Correct	Acc (%)
HotpotQA 2hop	84	83	98.81
HotpotQA 3hop	77	60	77.92
HotpotQA 4hop	87	78	89.66
MuSiQue 2hop	83	68	81.93
MuSiQue 3hop	103	77	74.76
MuSiQue 4hop	145	125	86.21
2Wiki 2hop	118	101	85.59
2Wiki 4hop	219	159	72.60
Total	**916**	**751**	**81.99**

Table 12: LLM Step-matching Accuracy

redundant step counts behind Reasoning-step Compression and Over-planning Pruning. As Table 13 shows, a stronger Direct planner does write sub-questions that align to more gold passages (necessary steps rise from 3,561 to 4,525), yet it continues to over-plan (2,609 redundant steps). Grounded Delta Planning still compresses the necessary steps to 1,673 and prunes the redundant steps to 506. Both advantages are therefore structural: compression comes from preliminary retrieval resolving facts before planning, and over-planning is a property of planning without sight of the retrieved context, neither of which a stronger planner removes on its own. For this study the GPT-5.5 plans are generated in a planning-only setting, using the same preliminary retrieval D_{rel} and the same fixed judge (GPT-4.1-mini) as Figure 4a, so the comparison isolates the effect of planner strength. The GPT-4.1-mini column therefore reproduces the values of that figure.

Step type	Direct Planning		Grounded Delta (Ours)	
	GPT-4.1-mini	GPT-5.5	GPT-4.1-mini	GPT-5.5
Necessary	3,561	4,525	1,388	1,673
Redundant	2,823	2,609	650	506

Table 13: Plan-level analysis under a frontier planner. Necessary and Redundant step counts from Direct Planning and Grounded Delta Planning under GPT-4.1-mini and GPT-5.5. The GPT-4.1-mini columns are the values of Figure 4a.

A.12 Qualitative Case Analysis

Tables 14–16 illustrate the three structural advantages of Grounded Delta Planning on concrete examples. In each, **GD** denotes a Gold Document, **S** a Subquery, colors indicate supporting context, and [NULL] marks a redundant step matched to no gold document.

Reasoning-step Compression. Direct Planning issues multiple subqueries to resolve intermediate facts. In the first example, it generates two steps: one to identify the academy and another to locate it. Grounded Delta Planning, having already retrieved both gold documents (GD1, GD2) during preliminary retrieval, compresses these into a single step that directly asks for the location (Table 14).

Over-planning Pruning. Direct Planning generates redundant steps that do not align with any gold document. In the second example, it produces three steps: two to retrieve the founding years and a third comparison step (marked [NULL]) whose inputs are already available. Grounded Delta Planning identifies that both founding years are already in D_{rel} and generates only a single comparison step, eliminating the redundant action (Table 15).

Under-planning Refinement. Direct Planning decomposes the query into narrow, targeted subqueries that may miss relevant documents. In the third example, Direct Planning generates three steps but only covers two of the three gold documents (GD1, GD2), missing

GD3 entirely. Because Grounded Delta Planning retrieves using the original query's broader semantics, it recovers all three gold documents (GD1, GD2, GD3) during preliminary retrieval, capturing dependencies that the targeted subqueries would miss (Table 16).

Question: Where is the academy, for which Joseph D. Stewart was appointed Superintendent, located?
Gold Documents: GD1: Joseph D. Stewart … was appointed as Superintendent of the United States Merchant Marine Academy … GD2: The United States Merchant Marine Academy … located in Kings Point, New York.
Direct Planning: S1: For which **academy was Joseph D. Stewart appointed Superintendent [GD1]**? S2: **Where is the academy**, for which Joseph D. Stewart was appointed Superintendent, **located [GD2]**?
Grounded Delta Planning: *Retrieved Context*: **[GD1]**, **[GD2]** S1: Where is the United States Merchant Marine Academy located?

Table 14: Reasoning-step Compression. Direct Planning uses two steps to identify and then locate the academy. Grounded Delta Planning, already holding GD1 and GD2 from preliminary retrieval, compresses them into a single step.

Question: Was Vanderbilt University or Emory University founded first?
Gold Documents: GD1: Vanderbilt University – Founded in 1873, … GD2: Emory University … was founded as Emory College in 1836 in Oxford …
Direct Planning: S1: What year was **Vanderbilt University founded [GD1]**? S2: What year was **Emory University founded [GD2]**? S3: Which was founded first … [NULL]
Grounded Delta Planning: *Retrieved Context*: **[GD1]**, **[GD2]** S1: Based on the founding years provided, which university was founded first: Vanderbilt University or Emory University?

Table 15: Over-planning Pruning. Direct Planning adds a redundant comparison step whose inputs are already retrieved. Grounded Delta Planning prunes it to a single comparison.

Question: What is another name for a forest which has the hamlet of Oakenclough on the edge of it?
Gold Documents: GD1: Oakenclough is an English hamlet located on the edge of the Forest of Bowland in Lancashire. GD2: The Forest of Bowland, also known as the Bowland Fells, … GD3: It was once described as the "Switzerland of England".
Direct Planning: S1: Where is the **hamlet of Oakenclough located [GD1]**? S2: What forest **lies on the edge of the hamlet Oakenclough [GD2]**? S3: What is **another name for the forest [GD2]** that has Oakenclough on its edge?
Grounded Delta Planning: *Retrieved Context*: **[GD1]**, **[GD2]**, **[GD3]** S1: What is another name for the forest which has the hamlet of Oakenclough on the edge of it?

Table 16: Under-planning Refinement. Direct Planning's targeted subqueries miss GD3. Grounded Delta Planning's broader preliminary retrieval recovers all three gold documents.

A.13 Workflow

Algorithm 1 summarizes the full GDP-RAG procedure, following the three phases of Section 3.2. In the Planning phase, the system performs preliminary retrieval to obtain query-

relevant context D_{rel} and calls the gap-conditioned planner to produce a skeletal trajectory $\mathcal{T}_{skel}$ of (θ_t, q_t) pairs that target only the information delta. The Trajectory Generation phase then grounds each planned step through the Act–Review–Update cycle: Act retrieves documents d_t for q_t and drafts a provisional answer $\hat{a}_t$ from the current Trajectory Memory $\mathcal{M}_{t-1}$; Review issues a second, answer-conditioned retrieval d_t^* and cross-verifies $\hat{a}_t$ into a validated answer a_t; and Update commits the grounded tuple to $\mathcal{M}_t$ and refines the next subquery q_{t+1} from the accumulated evidence. Providing $\mathcal{M}_{t-1}$ to all three modules keeps the steps consistent. Once all n steps are grounded, the Answering phase synthesizes the final answer A from the completed trajectory $\mathcal{T} = \mathcal{M}_n$.

Algorithm 1 Grounded Delta Planning RAG

Require: Question Q, Document Collection $\mathcal{D}$, Trajectory Memory $\mathcal{M}_0$
Ensure: Final Answer A

```
D_rel = Retrieve(Q, D)                                  ▷ Preliminary Retrieval
T_skel = Plan(Q, D_rel)                                 ▷ Gap-aware Decomposition
for t = 1, ..., n do
    (θ_t, q_t) ← T_skel
    d_t = Retrieve(q_t, D)
    â_t = Act(q_t, d_t, M_{t-1})                        ▷ Provisional answer
    d*_t = Retrieve(q_t + â_t, D)
    a_t = Review(q_t, â_t, d*_t, M_{t-1})               ▷ Cross-verify
    (M_t, q_{t+1}) = Update(q_t, a_t, d_t ∪ d*_t, M_{t-1})
end for
T = M_n
A = Answer(Q, T)
```