Efficient Retrieval-Augmented Generation via Grounded Planning

Anonymous ACL submission

Abstract

Large Language Models (LLMs) remain limited in knowledge-intensive and multi-hop reasoning scenarios, where reliance on parametric knowledge often leads to hallucinations. Although Retrieval-Augmented Generation (RAG) mitigates this issue by grounding generation in external knowledge, existing RAG frameworks remain inefficient and unreliable for complex multi-step inference. In this work, we propose Grounded Planning Strategy RAG (GPS-RAG), a plan-based framework that integrates retrieval into planning through Grounded Planning. Rather than constructing plans blindly and directly, GPS-RAG performs a preliminary retrieval to build a grounded plan that reflects available knowledge and explicitly targets the remaining knowledge gaps. This approach optimizes multi-hop reasoning, yielding concise and reliable reasoning trajectories. Extensive experiments on the HotpotQA, 2WikiMultiHopQA, and MuSiQue benchmarks demonstrate that GPS-RAG consistently outperforms baselines, surpassing the strongest competitor, PAR-RAG, with improvements of +2.1 in EM and +0.31 in F1, while reducing the average number of subqueries by 54.4%. These results validate its ability to execute minimal yet highly effective reasoning plans.

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 and 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., 2022) 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., 2024); 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, 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 efficiency and quality challenges, we propose **Grounded Planning Strategy RAG (GPS-RAG)**, a plan-based framework that tightly integrates retrieval into planning. GPS-RAG follows a three-phase pipeline consisting of *Planning*, *Trajectory Generation*, and *Answering*. Its core innovation lies in *Grounded Planning*: rather than generating plans blindly and directly, the system performs a preliminary retrieval to construct a *grounded plan* that explicitly reflects available context and targets remaining knowledge gaps. This grounded plan enables more efficient and reliable multi-step reasoning, yielding compact reasoning trajectories for complex queries. Figure 1 contrasts conventional Direct Planning with our Grounded

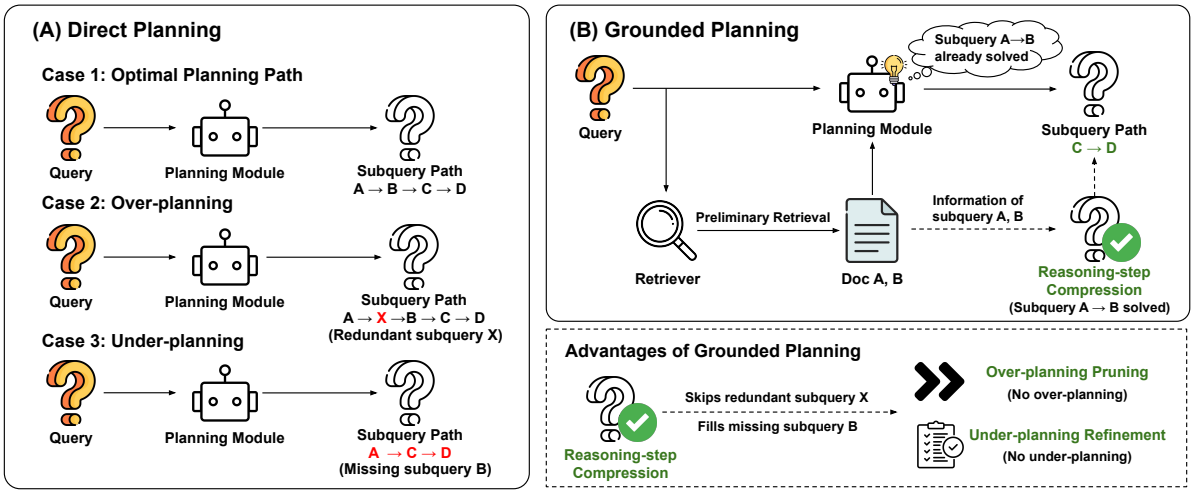

Figure 1: (A) Direct Planning: The planner decomposes queries solely based on the input question and parametric knowledge, leading to hallucinations such as over-planning (redundant subquery X in Case 2) and under-planning (missing subquery B in Case 3). (B) Grounded Planning (Ours): The planner conditions decomposition on context from preliminary retrieval (Docs A, B), enabling Reasoning-step Compression (resolved A, B), Over-planning Pruning (skipped X), and Under-planning Refinement(filled B), and producing a minimal plan (C→D).

Planning strategy, illustrating the resulting advantages for multi-hop reasoning. Our contributions are summarized as follows:

- We propose GPS-RAG, a new plan-based framework for multi-hop RAG that integrates retrieval into planning through Grounded Planning, enabling substantially more efficient and reliable multi-step reasoning.
- Extensive experiments on the HotpotQA, 2WikiMultiHopQA, and MuSiQue show that GPS-RAG outperforms the strongest baseline, PAR-RAG, by +2.1 EM and +0.31 F1, while reducing the average number of subqueries by 54.4%, yielding minimal yet highly effective reasoning trajectories.

2 Related Work

We categorize prior work on complex query answering into (1) *Global Reasoning and Decomposition* and (2) *Local Retrieval and Refinement*.

2.1 Global Reasoning and Decomposition

Existing approaches differ in *when* and *how* decomposition is performed. Specifically, **step-wise** approaches decompose tasks *during execution*, whereas **plan-based** approaches perform decomposition *prior to execution*. Step-wise approaches interleave reasoning and retrieval, generating one subquery at a time conditioned on previously retrieved context. Representative frameworks include ReAct (Yao et al., 2022), IRCoT (Trivedi et al., 2022a), and IteRetGen (Shao et al., 2023), which integrate retrieval into Chain-of-Thought reasoning (Wei et al., 2022). Subsequent work further improves robustness through test-time scaling (Snell et al., 2024), multi-path decoding, and self-correction (Shinn et al., 2023). In contrast, plan-based approaches explicitly construct a global reasoning blueprint prior to execution. PlanRAG (Lee et al., 2024) and PAR-RAG (Zhang et al., 2025) adopt linear plans, while Plan*RAG (Verma et al., 2024) generalizes planning into directed acyclic graphs. ReWOO (Xu et al., 2023) and LPKG (Wang et al., 2024) further enhance planning reliability via in-context learning and task-specific training.

While step-wise methods dynamically adapt to intermediate feedback, they lack global foresight. Conversely, plan-based methods provide a structured reasoning roadmap before execution but rely solely on parametric knowledge during planning. Distinct from both, our **Grounded Planning** integrates context from preliminary retrieval into the planning phase, combining the adaptability of step-wise reasoning with the structural advantages of plan-based decomposition.

2.2 Local Retrieval and Refinement

Beyond global decomposition, the success of multi-hop reasoning critically depends on accurate *local execution* at each step. This line of work focuses on improving the quality of single-step retrieval

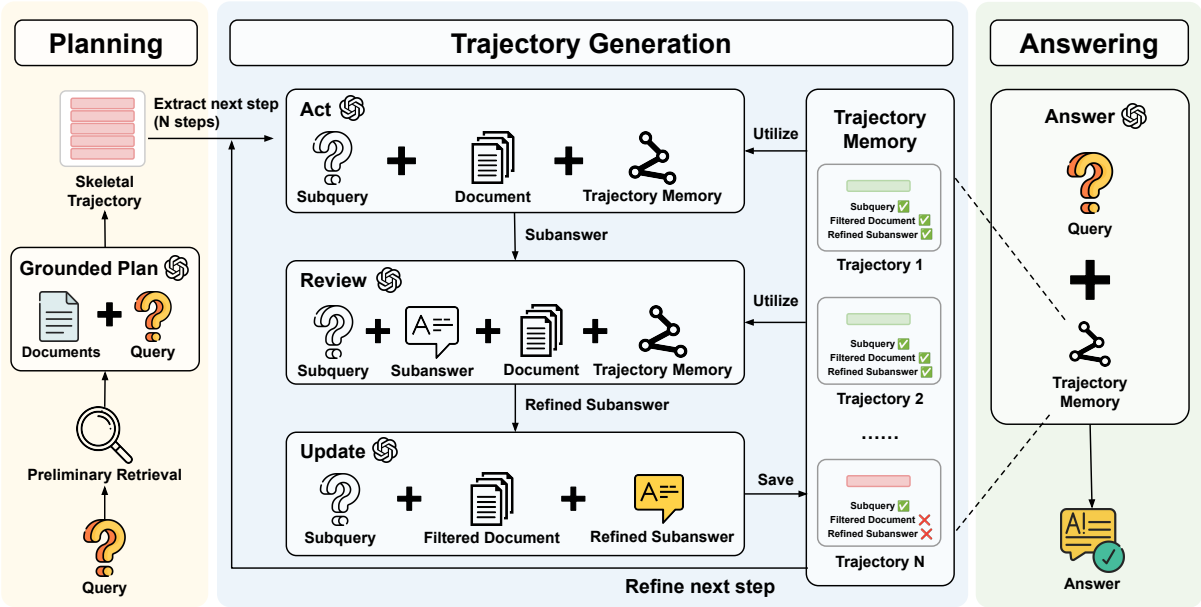

Figure 2: GPS-RAG framework. The workflow proceeds in three phases: (1) **Planning Phase**: Utilizes Grounded Planning to generate a concise skeletal trajectory $\mathcal{T}_{skel}$, leveraging context from preliminary retrieval to prune redundant steps. (2) **Trajectory Generation Phase**: Executes 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) **Answer Phase**: Synthesizes the final response based on this comprehensive trajectory $\mathcal{T}$.

and answer refinement, which directly determines whether intermediate reasoning states remain reliable. Standard RAG (Lewis et al., 2020) suffers from query ambiguity, motivating query expansion (Gao et al., 2023a; Xia et al., 2024) and structured retrieval via knowledge graphs (Edge et al., 2024; Jiménez Gutiérrez et al., 2024). Robustness is further improved through filtering and verification mechanisms (Yan et al., 2024; Asai et al., 2024; Dhuliawala et al., 2023). Building on these advances, GPS-RAG incorporates the Act-Review mechanism from PAR-RAG (Zhang et al., 2025), while introducing an unconditional execution flow driven by Grounded Planning and a persistent Trajectory Memory, enabling reliable and verifiable reasoning throughout the execution process.

3 Method

3.1 Problem Formulation

We study efficient multi-hop RAG under the following setting. 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 the trajectory is defined as a sequence of N steps $\mathcal{T} = \langle (q_1, d_1, a_1), \ldots, (q_N, d_N, a_N) \rangle$. 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 GPS-RAG

As illustrated in Figure 2, our proposed GPS-RAG consists of three stages: *Planning* (§ 3.2.1), *Trajectory Generation* (§ 3.2.2), and *Answering* (§ 3.2.3). Given a query Q, the *Planning* stage first performs preliminary retrieval to obtain query-related context D_{rel} and outputs a skeletal trajectory $\mathcal{T}_{skel}$ containing only the essential subqueries. The *Trajectory Generation* stage then executes and verifies each subquery in $\mathcal{T}_{skel}$ to populate it with documents and intermediate answers, yielding a fully grounded trajectory $\mathcal{T}$. Finally, the *Answering* stage synthesizes the final answer A from $\mathcal{T}$.

3.2.1 Planning Phase

This phase decomposes the query into a minimal subqueries sufficient for deriving the final answer, grounded in context from preliminary retrieval.

Grounded Plan Module. Given an input query Q, the system first retrieves query-related context $D_{rel} \subset \mathcal{D}$. Conditioned on D_{rel}, the planner produces a skeletal trajectory $\mathcal{T}_{skel} = \langle q_1, \ldots, q_n \rangle$, where each subquery q_i requests information not covered by the retrieved context:

$$\mathcal{T}_{skel} = \mathrm{Plan}\big(Q, D_{rel}\big). \tag{2}$$

In our prompt design, each sub-query is paired with a justification (labeled as `Thought`), enforcing groundedness and preventing speculative or redundant decomposition. The full prompt details are provided in Appendix A.6.

Rationale of Grounded Planning. Conventional planners rely on Direct Planning based solely on the LLM's parametric knowledge, which fundamentally limits their reliability and makes them vulnerable to hallucinations, often resulting in both over-planning and under-planning. By contrast, Grounded Planning explicitly conditions decomposition on retrieved context, yielding several structural benefits. In particular, this design yields three key advantages. (1) *Reasoning-step Compression*: D_{rel} often resolves multiple potential intermediate steps directly, eliminating the need to generate subqueries for them and significantly shortening the trajectory. (2) *Over-planning Pruning*: The planner evaluates candidate steps against D_{rel}, discarding subqueries that request unnecessary information already present in the context. (3) *Under-planning Refinement*: D_{rel} uncovers underlying entities and relations, supplying the necessary context to resolve implicit dependencies.

3.2.2 Trajectory Generation Phase

After constructing the skeletal trajectory $\mathcal{T}_{skel}$ during the Grounded Planning phase, the system enters Trajectory Generation, where the remaining d_t and a_t slots are populated, as illustrated in Figure 2. The detailed operational procedure of the Act–Review–Update cycle is described below.

Act Module. Each cycle begins with the Act module, whose objective is to generate a provisional intermediate answer $\hat{a}_t$. Given the planned subquery q_t, the system retrieves a set of relevant documents from the collection $\mathcal{D}$, and then produces $\hat{a}_t$ conditioned on the query, the retrieved documents, and the current state of Trajectory Memory. Formally, the Act operation is defined as:

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

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.

Review Module. The Review module is responsible for validating the provisional answer $\hat{a}_t$ and mitigating potential hallucinations before it is finalized. The system constructs a verification query by concatenating the subquery q_t and the provisional answer $\hat{a}_t$, and performs a secondary retrieval over the document collection $\mathcal{D}$ to obtain targeted documents. The provisional answer is then examined against the newly retrieved documents to identify and correct any factual inconsistencies. Formally, the Review operation is defined as:

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

where a_t denotes the validated intermediate answer. During this process, we again adopt a citation-enhanced generation strategy: a_t is produced by explicitly citing the supporting documents from the secondary retrieval. This cross-verification procedure ensures that the finalized intermediate answer is strictly grounded in verifiable context.

Update Module. The Update module finalizes each cycle by committing validated information to Trajectory Memory. Specifically, the system first stores the verified tuple (q_t, d_t, a_t), where d_t denotes the union of documents cited during the Act and Review stages. By committing only cited documents at the end of each cycle, the system preserves a compact and coherent reasoning context. Formally, the Update operation is defined as:

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

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 system repeats the Act–Review–Update cycle for each subquery until all N steps in

$\mathcal{T}_{skel}$ are completed. Through this iterative process, the initial skeletal trajectory is transformed into a fully grounded reasoning trajectory $\mathcal{T} = \langle (q_1, d_1, a_1), \ldots, (q_n, d_n, a_n) \rangle$. This verified trajectory, containing both the reasoning structure and supporting citations, then serves as the comprehensive context for the final stage.

Comparison with PAR-RAG. Although GPS-RAG builds upon the PAR-RAG framework (Zhang et al., 2025), the two methods differ in execution strategy, most notably in the treatment of *Conditional* versus *Unconditional* execution. PAR-RAG adopts a conditional execution scheme in which the planner decides, for each step, whether external retrieval is necessary or whether the step can be completed solely from previously generated intermediate results (e.g., for final comparisons; see Appendix A.5 for the full prompt). In contrast, GPS-RAG enforces an *unconditional* execution policy: every planned step in $\mathcal{T}_{skel}$ is executed through the full Act–Review–Update loop. This design is enabled by our Grounded Planning process (§ 3.2.1), which eliminates redundant or trivial operations during planning; as a result, the remaining sub-queries correspond to genuine information gaps that must be externally verified. Consequently, unconditional execution in GPS-RAG increases reliability by preventing error propagation from unverified intermediate states. Beyond execution control, GPS-RAG further strengthens consistency by introducing Trajectory Memory as shared context for both the Act and Review modules, and by extending citation storage to include supporting documents from both phases. All documents that directly supports each intermediate answer a_t is committed to Trajectory Memory as d_t, yielding a compact and fully verifiable trajectory $\mathcal{T}$.

3.2.3 Answering Phase

In the final Answering phase, the system synthesizes the fully constructed reasoning trajectory $\mathcal{T}$ with query Q to generate the final answer A, following the formulation defined in Section 3.1. The system directly leverages the verified tuples (q_t, d_t, a_t) maintained in the Trajectory Memory, integrating the accumulated intermediate result into a coherent and grounded response. This step completes the GPS-RAG pipeline by producing an answer that is both logically consistent with the reasoning process and strictly supported by retrieved documents.

4 Experiment Setup

Datasets. We evaluate our method on three multi-hop QA benchmarks: HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), and MuSiQue (Trivedi et al., 2022b). To focus on retrieval and reasoning, we apply a two-stage filtering procedure, as a substantial portion of original questions can be answered by the model using parametric knowledge alone. We first remove such questions, then sample the remaining ones to obtain a balanced evaluation set across 2-hop, 3-hop, and 4-hop complexities. The resulting evaluation sets contain 200 questions per 2/3/4-hop category for HotpotQA and MuSiQue, compared to a 300/0/300 distribution for 2WikiMultiHopQA. Detailed statistics and the full filtering procedure are provided in Appendix A.1.

Evaluation Metrics. Our goal is to determine if GPS-RAG can enhance efficiency while maintaining the quality of standard RAG baselines. To evaluate answer quality, we report Exact Match (EM), F1 Score, and Substring Match (SM) to quantify the quality of the final answer A against the ground truth. In terms of efficiency, we measure the average number of reasoning Steps (Steps), which reflects the conciseness of the generated trajectory $\mathcal{T}$. Further details and formal definitions for each metric are provided in Appendix A.2.

Baseline. To evaluate GPS-RAG, we compare it against established RAG paradigms. Vanilla RAG (Lewis et al., 2020) serves as the naive single-step retrieval baseline, representing systems without multi-step planning capabilities. IRCoT (Trivedi et al., 2022a) represents standard step-wise RAG systems that dynamically generate the reasoning trajectory through interleaved retrieval and reasoning. PAR-RAG (Zhang et al., 2025) represents standard plan-based systems that employ Direct Planning, formulating the plan steps upfront without access to document context. Finally, the Gold Documents setting serves as an ideal upper-bound benchmark, as it uses the ground-truth documents required to answer the query, effectively eliminating retrieval errors.

Implementation Details. All experiments utilize GPT-4.1-mini (OpenAI, 2024) as the backbone model for planning and generation. For the retrieval pipeline, documents are encoded using BAAI/bge-m3 (Chen et al., 2024b). We retrieve the top-30 candidate documents, which are

Method	HotpotQA				2Wiki				MuSiQue				Average			
	SM	F1	EM	Steps	SM	F1	EM	Steps	SM	F1	EM	Steps	SM	F1	EM	Steps
Gold Documents	83.00	62.33	41.50	-	85.00	50.19	34.00	-	69.67	45.16	28.83	-	79.22	52.56	34.78	-
Vanilla RAG	56.33	40.89	24.83	-	35.67	19.37	8.00	-	19.50	16.45	7.50	-	37.17	25.57	13.44	-
IRCoT	67.33	46.71	40.17	1.40	79.17	32.92	17.00	2.05	34.33	23.55	17.50	2.74	60.28	34.39	24.89	2.06
PAR-RAG	**71.83**	**67.71**	48.33	3.09	79.83	67.73	60.17	3.72	**50.83**	**45.92**	**30.67**	3.67	**67.50**	60.45	46.39	3.49
GPS-RAG	70.50	67.04	**50.67**	**1.11**	**80.67**	**72.50**	**67.00**	**1.74**	47.33	42.68	27.83	**1.92**	66.17	**60.74**	**48.50**	**1.59**

Table 1: Performance comparison on multi-hop datasets. Higher SM, F1, and EM indicate better performance, while lower Steps reflect greater efficiency. Bold marks the best results (excluding the Gold Document upper bound).

subsequently reranked using the cross-encoder BAAI/reranker-large-v2-m3 (Chen et al., 2024a) to yield the final top-10 documents. To align with PAR-RAG's native knowledge graph integration, we augment the pipeline such that every retrieval operation combines these dense retrieval results with knowledge graph information using HippoRAG2 (Jiménez Gutiérrez et al., 2025). Finally, all experiments were conducted on a single NVIDIA RTX 4090 GPU.

5 Main Results

We evaluate GPS-RAG against competitive baselines to assess answer quality and reasoning efficiency. Table 1 shows GPS-RAG demonstrates competitive performance across three multi-hop benchmarks. In terms of answer quality, GPS-RAG attains the highest in both F1 (60.74) and EM (48.50), outperforming the strongest baseline, PAR-RAG. These gains indicate that improved planning does not compromise answer quality, but rather preserves general quality and, in terms of F1 and EM, even improves it. More importantly, GPS-RAG delivers a substantial improvement in reasoning efficiency. As shown in the *Steps* column, GPS-RAG reduces the average number of steps to 1.59, compared to 3.49 for PAR-RAG, a 54.4% reduction. This efficiency gain is a direct result of the proposed Grounded Planning, which prunes unnecessary subqueries at the planning stage and identifies concise reasoning paths before execution. These results demonstrate that GPS-RAG achieves superior overall answer quality while significantly compressing reasoning trajectories, confirming that high-quality multi-hop reasoning can be obtained with minimal steps.

6 Ablation Study

To analyze the contribution of each component in GPS-RAG, we conduct an ablation study addressing three research questions: (1) the necessity of preliminary retrieval, (2) the impact of Grounded Planning, and (3) the effect of Trajectory Generation strategies.

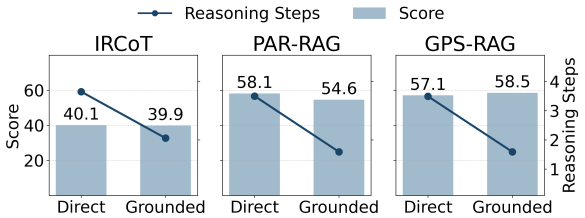

Figure 3: Effect of preliminary retrieval on three methods. Left: Score (avg. SM/F1/EM). Right: Steps.

Phase		Score			Efficiency	
Planning	Trajectory	SM	F1	EM	Steps	Time
Direct	Conditional	**67.50**	60.45	46.39	3.49	14.97
	Unconditional	65.83	59.08	46.28	3.50	31.26
Grounded	Conditional	61.44	57.25	45.17	**1.59**	**7.79**
	Unconditional	66.17	**60.74**	**48.50**	**1.59**	9.86

Table 2: Ablation study on component contributions. **Planning**: Comparison between Direct and Grounded Planning. **Trajectory**: Comparison between Conditional and Unconditional trajectory generation.

Necessity of Preliminary Retrieval. We investigate whether preliminary retrieval provides consistent benefits across different RAG paradigms. Specially, we evaluate three frameworks: the stepwise IRCoT, the plan-based PAR-RAG, and our proposed GPS-RAG. For each framework, we compare a *Direct* setting, in which the model initiates reasoning or planning without preliminary retrieval, with a *Grounded* setting that incorporates a preliminary retrieval step (Figure 3). IRCoT adopts the *Grounded* strategy by default; we build a *Direct* variant by removing the preliminary retrieval. For PAR-RAG and GPS-RAG, we contrast Direct Planning with Grounded Planning. We report the number of reasoning *Steps* and the overall *Score*, defined as the average of SM, F1, and EM.

As shown in Figure 3, the *Grounded* strategy re-

duces reasoning steps across all three frameworks while maintaining performance, with GPS-RAG even achieving a slight performance gain. These results answer our first research question: preliminary retrieval acts as a universal form of grounding mechanism that shortens reasoning steps without sacrificing quality, thereby reducing computational cost across RAG architectures.

Impact of Grounded Planning. To understand the benefits of our approach, as reported in Table 2, replacing Direct Planning with Grounded Planning reduces the average number of reasoning steps from approximately 3.50 to 1.59, which in turn significantly lowers the total inference time across both execution strategies. The improvement stems from incorporating query-related context, D_{rel}, as an explicit reference during planning. This mechanism constrains the search space and aligns the plan with available context, thereby optimizing the reasoning trajectory before execution begins.

Effect of Trajectory Generation strategies. We evaluate the Trajectory Generation phase as reported in Table 2. Specifically, in Grounded Planning, our Unconditional strategy significantly outperforms the Conditional approach, yielding substantial gains of +4.73 SM, +3.49 F1, and +3.33 EM. This superiority stems from strictly executing every planned action alongside a double-verification mechanism, which effectively enhances reasoning reliability. While this robust execution incurs a marginal latency overhead of 2.07s, the total inference time remains substantially lower than any configuration under Direct Planning, which requires at least 14.97s. This confirms that the Unconditional strategy ensures superior answer integrity without compromising the efficiency gains achieved through Grounded Planning.

Taken together, these results reveal a clear and complementary interaction between the two components: Grounded Planning minimizes the reasoning complexity during the planning phase, while Unconditional execution ensures correctness and robustness throughout the process. This synergy forms the core mechanism underlying the effectiveness of GPS-RAG.

7 Effectiveness Analysis of GPS-RAG

In this section, we further examine the effectiveness of GPS-RAG beyond performance-only, focusing on its efficiency under increasing query complexity and its ability to mitigate planning inefficiencies.

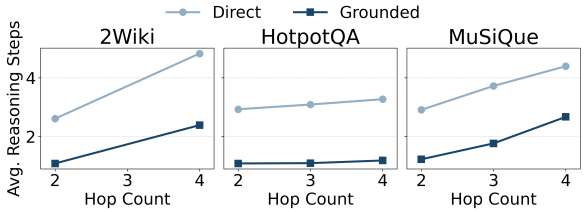

Figure 4: Efficiency under Increasing Query Complexity. Average reasoning steps of Direct and Grounded Planning across 2- to 4-hop queries on different datasets.

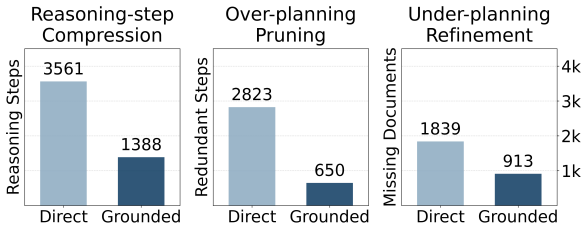

Figure 5: Impact of Grounded Planning on reducing planning suboptimalities. Metrics include: Reasoning Steps (valid steps matching gold documents), Redundant Steps (irrelevant steps unsupported by gold documents), and Missing Documents (gold documents not covered by the generated plan).

Efficiency under Increasing Query Complexity. We evaluate GPS-RAG under varying question complexity by comparing the average reasoning steps of Direct and Grounded Planning across 2- to 4-hop queries, as shown in Figure 4. We can observe that Grounded Planning requires fewer steps than Direct Planning, with the gap widening as query complexity increases. On HotpotQA and 2Wiki, Grounded Planning maintains a nearly constant step count, amortizing the cost of additional hops. On MuSiQue, although steps increase linearly with task difficulty, GPS-RAG maintains an efficiency advantage. These results show that Grounded Planning adapts its reasoning to task complexity, pruning redundant steps while retaining necessary reasoning for harder queries.

Compression, Pruning, and Refinement. Figure 5 illustrates the impact of our approach. To derive these metrics, we utilized an LLM-based evaluator to determine if each step in the plan targets a gold document. The analysis decomposes the planning behavior into: Reasoning Steps (valid reasoning), Redundant Steps (over-planned actions), while identifying Missing Documents by checking for gold documents not covered by the plan. We

	Question	Gold Document	Direct	Grounded
Reasoning-step Compression	Where is the academy, for which Joseph D. Stewart was appointed Superintendent, located?	**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.	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]**?	*Retrieved Context*: **[GD1]**, **[GD2]** S1: Where is the United States Merchant Marine Academy located?
Over-planning Pruning	Was Vanderbilt University or Emory University founded first?	**GD1**: Vanderbilt University - Founded in 1873, ... **GD2**: Emory University ... was founded as Emory College in 1836 in Oxford ...	S1: What year was **Vanderbilt University founded [GD1]**? S2: What year was **Emory University founded [GD2]**? S3: Which was founded first ... [NULL]	*Retrieved Context*: **[GD1]**, **[GD2]** S1: Based on the founding years provided, which university was founded first: Vanderbilt University or Emory University
Under-planning Refinement	What is another name for a forest which has the hamlet of Oakenclough on the edge of it?	**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".	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?	*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 3: Qualitative case analysis illustrating the three advantages of Grounded Planning over Direct Planning: *Reasoning-step Compression*, *Over-planning Pruning*, and *Under-planning Refinement*. **GD**: Gold Documents; **S**: Subquery; *Retrieved Context*: documents retrieved before execution. Colors indicate supporting context. [NULL] indicates a redundant step not matched to any gold document.

validated this automated protocol via human review on a 10% data sample, yielding an alignment accuracy of 82%. Further details on the classification criteria are available in Appendix A.4.

As shown in Figure 5, GPS-RAG improves across all three dimensions. First, Reasoning-step Compression reduces reasoning steps from 3,561 to 1,198 (66.4% reduction), indicating that many of the reasoning steps is absorbed by query-related context D_{rel}. Second, Over-planning Pruning decreases redundant steps from 2,823 to 650 (77.0% reduction), effectively mitigating hallucinated or unnecessary actions. Finally, Under-planning Refinement reduces the missing gold documents from 1,839 to 736 (60.0% reduction), showing that Grounded Planning both reduces computational overhead and strengthens reasoning reliability.

Qualitative Case Analysis. Table 3 illustrates how Grounded Planning improves planning efficiency and reliability across the three analyzed dimensions. For Reasoning-step Compression, Direct Planning requires multiple explicit subqueries to resolve intermediate facts, whereas Grounded Planning produces subqueries only that is not in the query-related context D_{rel} saving necessary reasoning steps. For Over-planning Pruning, Direct Planning exhibits redundant steps. In contrast, Grounded Planning leverages D_{rel} to identify and prune these redundancies, effectively avoiding unnecessary exploration. For Under-planning Refinement, Direct Planning often omits essential reasoning paths. In contrast, Grounded Planning leverages D_{rel} to bridge these information gaps, recovering necessary gold documents to guarantee a fully supported reasoning trajectory. Overall, these cases confirm that GPS-RAG achieves efficiency gains by aligning planning with available information: compressing essential reasoning, pruning redundant steps, and refining incomplete plans.

8 Conclusion

We propose GPS-RAG, a Grounded Planning framework for multi-hop RAG that integrates retrieval into planning to produce concise and reliable reasoning trajectories. Experiments on three benchmarks show that GPS-RAG outperforms strong baselines in answer quality while reducing reasoning length by over 50%, demonstrating that effective multi-hop reasoning can be achieved with minimal steps when planning is properly grounded. Further analysis confirms that these gains are primarily driven by the Grounded Planning mechanism, which enables three distinct advantages: Reasoning-step Compression, Over-planning Pruning, and Under-planning Refinement. By minimizing computational overhead while bridging informational gaps, our framework ensures a fully supported reasoning trajectory without unnecessary exploration. These results establish Grounded Planning as a practical and scalable design paradigm for complex RAG systems.

9 Limitations

We identify two primary areas that constrain the direct interpretation of our results:

(1) Our study relied on a very strong reasoning model, GPT-4.1-mini. While this was necessary to test the full potential of the informed planning signal, it also creates a constraint: The efficiency and robustness gains we observed may not apply equally to weaker language models. We assume that models with less reasoning ability might struggle to use the planning information effectively. Therefore, our results are primarily applicable to LLMs that have a comparable level of strong multi-step reasoning capacity.

(2) While we have thoroughly assessed performance using Substring Match, Exact Match, and F1, the inclusion of the Gold Document baseline highlights a limitation in our current retrieval analysis. Regrettably, we could not calculate a specific Precision score for the retrieved documents, despite successfully determining whether the final answer was derivable. Since the underlying mechanism relies on citation-based answering, future work should leverage the saved trajectory documents as a key indicator of retrieval quality. This would enable a direct evaluation of the retrieval component's precision and recall, offering a more complete picture of the framework's performance.

10 Ethical considerations

We confirm our commitment to following ethical guidelines and institutional policies. This work uses only publicly available datasets, HotpotQA, 2WikiMultiHopQA, and MuSiQue; no new data were collected, no human subjects were involved, and no personally identifiable information was processed. All datasets comply with their research licenses and are properly cited.

The manual audit described in this work was conducted by the authors (four researchers holding at least a bachelor's degree) as part of the internal research process. Consequently, no external crowdworkers were recruited or compensated, consistent with standard practices for author-led assessments.

Potential risks include dataset bias and misinformation propagation in retrieval-augmented generation. We report limitations, and recommend bias and safety filtering before any deployment. GPT-4.1-mini was used for prompting and analysis; all outputs were verified by the authors, and AI tools are not considered authors.

References

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

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024a. BGE-Reranker-v2-M3: A multilingual cross-encoder reranker. Hugging Face model card.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024b. M3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *Preprint*, arXiv:2402.03216.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Celina Asghari, and Jason Fairbanks. 2023. Chain-of-verification reduces hallucination in large language models. *arXiv preprint arXiv:2309.11495*.

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, Zaid Harchaoui, and Yejin Choi. 2023. Faith and fate: Limits of transformers on compositionality. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023a. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1762–1777, Toronto, Canada. Association for Computational Linguistics.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2023b. Retrieval-augmented generation for large language models: A survey. *arXiv preprint*, arXiv:2312.10997.

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

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. Hipporag: Neurobiologically inspired long-term memory for large

language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. 2025. From rag to memory: Non-parametric continual learning for large language models. *arXiv preprint arXiv:2502.14802*. ICML 2025.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *NeurIPS*.

Myeonghwa Lee, Seonho An, and Min-Soo Kim. 2024. Planrag: A plan-then-retrieval augmented generation for generative large language models as decision makers. *arXiv preprint arXiv:2406.12430*. NAACL 2024.

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. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Weitao Li, Junkai Li, Weizhi Ma, and Yang Liu. 2024. Citation-enhanced generation for LLM-based chatbots.

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

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

OpenAI. 2024. Gpt-4.1 mini. `https://openai.com/`. Large language model.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Singapore. Association for Computational Linguistics.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9248–9274, Singapore. Association for Computational Linguistics.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. 2025. Agentic retrieval-augmented generation: A survey on agentic rag. *arXiv preprint arXiv:2501.09136*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *NAACL*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022a. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *arXiv preprint arXiv:2212.10509*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022b. Musique: Multi-hop questions via single-hop question composition. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1996–2012.

Prakhar Verma, Sukruta Prakash Midigeshi, Gaurav Sinha, Arno Solin, Nagarajan Natarajan, and Amit Sharma. 2024. Plan*rag: Efficient test-time planning for retrieval augmented generation. *arXiv preprint arXiv: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. 2024. Learning to plan for retrieval-augmented large language models from knowledge graphs. In *Findings of the Association for Computational Linguistics: EMNLP 2024*. Association for Computational Linguistics.

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

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. In *TACL*.

Yu Xia, Junda Wu, Sungchul Kim, Tong Yu, Ryan A. Rossi, Haoliang Wang, and Julian McAuley. 2024. Knowledge-aware query expansion with large language models for textual and relational retrieval.

Binfeng Xu, Zhiyuan Peng, Bowen Lei, Subhabrata Mukherjee, Yuchen Liu, and Dongkuan Xu. 2023. Rewoo: Decoupling reasoning from observations for efficient augmented language models. In *NeurIPS*.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. 2024. Corrective retrieval augmented generation.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Ningning Zhang, Chi Zhang, Zhizhong Tan, Xingxing Yang, Weiping Deng, and Wenyong Wang. 2025. Credible plan-driven rag method for multi-hop question answering. *arXiv preprint arXiv:2504.16787*.

A Appendix

A.1 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.

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 a 300/0/300 split due to lack of 3-hop data. This balanced distribution prevents lower-hop dominance and enables evaluation of scalability.

A.2 Metric

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, computed using the function:

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

. 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%.

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.

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 $\mathbf{100}\%$.

Step To ensure a fair comparison of efficiency, we quantify the number of generated reasoning actions for step-wise IRCoT, plan-based PAR-RAG, and our GPS-RAG. For IRCoT, we count the sub-queries generated within its Chain-of-Thought process prior to the final answering step. For PAR-RAG and GPS-RAG, we count the steps contained in the generated reasoning trajectory.

Time To provide a holistic view of system latency beyond discrete reasoning steps, we report the inference time which is measured as the wall-clock duration from query input to final answer generation on a single RTX 4090 GPU. This captures the complete lifecycle of the reasoning process, including retrieval, planning, and generation steps.

A.3 Efficiency and Architectural Trade-off

Beyond individual metrics, we analyze the balance between performance and cost as summarized in Table 6. While IRCoT achieves the lowest latency (3.84s) primarily due to its simpler execution mechanism, this simplicity comes at a significant cost to answer quality (Score 39.85). In contrast, GPS-RAG strikes an optimal balance: it achieves top-tier performance comparable to the complex PAR-RAG baseline (Score 58.47 vs. 58.11), yet reduces the inference time by approximately 34% (14.97s $\rightarrow$ 9.86s). This confirms that Grounded Planning is a highly cost-effective strategy, delivering superior reasoning quality with significantly reduced inference time.

A.4 Grounded Planning Advantages Analysis

To quantify the efficacy of Grounded Planning, we perform an LLM-based alignment. We feed the

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.

Method	**HotpotQA**			**2Wiki**			**MuSiQue**			**Average**		
	Score	Steps	Time	Score	Steps	Time	Score	Steps	Time	Score	Steps	Time
IRCoT	51.40	1.40	**2.79**	43.03	2.05	**3.96**	25.13	2.74	**4.77**	39.85	2.06	**3.84**
PAR-RAG	62.63	3.09	16.27	69.24	3.72	17.91	**42.47**	3.67	22.20	58.11	3.49	14.97
GPS-RAG	**62.73**	**1.11**	7.31	**73.39**	**1.74**	9.98	39.28	**1.92**	14.14	**58.47**	**1.59**	9.86

Table 6: Efficiency and architectural trade-off. Score represent the average of SM, F1, and EM.

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 7: LLM Step-matching Accuracy

reasoning steps generated by both the Direct Planning baseline and our Grounded Planning approach alongside the Gold Documents into an LLM to identify valid mappings. Based on this alignment, the metrics in Figure 5 are defined as follows:

Reasoning-step Compression. This metric measures the total number of valid reasoning steps required to answer the query. *Direct Planning* counts the generated steps that successfully align with a gold document in the baseline, while *Grounded Planning* counts the corresponding valid steps in our approach. The difference represents the total number of necessary steps saved.

Over-planning Pruning. This metric measures the total number of redundant steps. *Direct Planning* counts the generated steps that fail to align with any supporting gold document in the baseline, while *Grounded Planning* counts the remaining unaligned steps in our approach. The difference represents the total number of redundant steps eliminated via the pruning mechanism.

Under-planning Refinement. This metric measures the total number of missed ground truth documents. *Direct Planning* counts the ground truth documents that weren't covered by the plan in the baseline, while *Grounded Planning* counts the documents that remain missing in our approach. The difference represents the total number of documents recovered via the refinement mechanism.

Step-matching Validation We validated LLM's accuracy via human review on a 10% data sample, with results presented in Table 7. This manual audit was conducted by four researchers, each holding at least a bachelor's degree, to ensure the necessary linguistic and logical expertise for evaluating complex multi-hop reasoning. For both Direct and Grounded Planning, we manually verified the LLM's step-matching outputs whether each step correctly matched a corresponding gold document or was accurately identified as redundant.

A.5 Prompt Design

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

Direct Planning Prompt As shown in Figure 6, 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 sub-query, while restricting its choice of actions to a predefined list: [`Retrieve`, `Answer`]. These generated actions serve as the foundational candidates

```
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]"
  }}
]
```

Figure 6: 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
# Format instructions
[
 {{
   "Thought":"[explain what information this step seeks and why it's not already available in the passages]",
   "Question":"[the sub-question for missing information]",
  }}
]
```

Figure 7: Prompt of Grounded Planning.

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 solely on the existing state without further searching.

Grounded Planning Prompt As shown in Figure 7, the Grounded Planning prompt incorporates preliminary retrieval results to guide the model. Unlike the direct prompt, which requires selecting from actions such as [Retrieve, Answer], this version provides no predefined action list. Instead, the LLM is tasked with analyzing the existing context to identify remaining information gaps. 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.

A.6 Workflow

Based on the proposed architecture, we show the algorithmic description in Algorithm 1.

Algorithm 1 Grounded Planning Strategy RAG

Input: Question Q, Document Collection D, Trajectory Memory $\mathcal{M}_0$
Output: Final Answer A
$t \leftarrow 1$
$D_{rel} = \text{Retrieve}(Q)$
$\mathcal{T}_{skel} = Plan(Q, D_{rel})$
repeat
  $q_t \leftarrow T_{skel}$
  $d_t = \text{Retrieve}(q_t, D)$
  $\hat{a}_t = \text{Answer}(q_t, d_t, \mathcal{M}_{t-1})$
  $d_t^* = \text{Retrieve}(q_t + \hat{a}_t, D)$
  $a_t = \text{Review}(q_t, d_t^*, \mathcal{M}_{t-1})$
  $(\mathcal{M}_t, q_{t+1}) = \text{Update}(q_t, a_t, d_t, \mathcal{M}_{t-1})$
  $t = t + 1$
until $t > n$
$A = \text{Answer}(Q, \mathcal{T})$