AirRAG: Autonomous Strategic Planning and Reasoning Steer Retrieval Augmented Generation

Anonymous ACL submission

Abstract

Leveraging the autonomous decision-making capabilities of large language models (LLMs) has demonstrated superior performance in reasoning tasks. However, despite the success of iterative or agentic retrieval-augmented generation (RAG) techniques, these methods are often constrained to a single solution space when confronted with complex problems. In this paper, we propose a novel thinking pattern in RAG that integrates autonomous strategic planning with efficient reasoning actions, significantly activating intrinsic reasoning capabilities and expanding the solution space of specific tasks via Monte Carlo Tree Search (MCTS), which we refer to as **AirRAG**. Specifically, our approach designs five fundamental reasoning actions, which are expanded to a broad tree-based reasoning space using MCTS. The approach also incorporates self-consistency verification to explore potential reasoning paths and inference scaling law. Additionally, computationally optimal strategies are employed to allocate more inference resources to key actions, thereby enhancing overall performance. Experimental results demonstrate the effectiveness of AirRAG, showing significant performance gains on complex question-answering datasets. Furthermore, AirRAG is flexible and lightweight, making it easy to integrate with other advanced technologies and models.

1 Introduction

Retrieval-Augmented Generation (RAG) has shown great potential in addressing the issue of generating factually incorrect content, especially in domain-specific or knowledge-intensive tasks (Kandpal et al., 2023). However, as task complexity increases, several new challenges emerge, such as the inability to retrieve sufficient knowledge with a single query and the difficulty of understanding the intricate reasoning logic inherent in the question. To tackle these challenges, it is crucial to harness the reasoning capabilities of large language models (LLMs) to improve RAG performance (Jiang et al., 2023; Jeong et al., 2024; Asai et al., 2024; Yu et al., 2024).

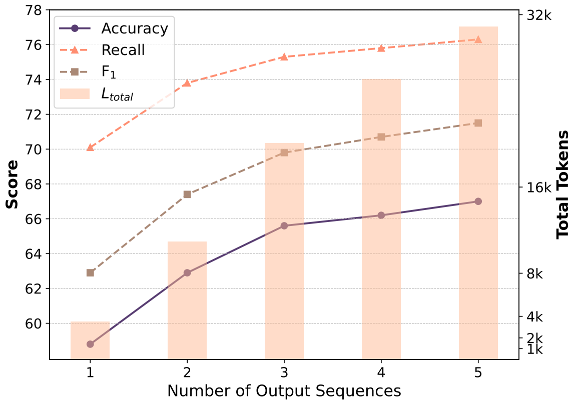

Figure 1: Comparison of average performance across three datasets with varying numbers of output sequences. L_{total} represents the total number of tokens consumed during the reasoning process. AirRAG leverages generation diversity and self-consistency to explore the potential solution space, significantly enhancing overall performance by scaling inference computation.

Previous research on complex query scenarios has primarily focused on optimizing the query and retrieval processes to obtain relevant information (Shi et al., 2023; Zhou et al., 2023; Gao et al., 2023; Jiang et al., 2023; Zheng et al., 2024; Asai et al., 2024; Yan et al., 2024). Iterative retrieval is frequently used to improve the depth and relevance of search results in information retrieval tasks. This process continuously updates intermediate queries and results to satisfy dynamic information needs during the complex task-solving process (Jeong et al., 2024; Yue et al., 2024). In addition, Li et al. (2025) integrates an agentic search workflow into the reasoning process, enabling dynamic retrieval when LLMs encounter uncertain knowledge points. The agentic LLMs are trained to learn step-by-step reasoning with search through reinforcement learning (Chen et al., 2025; Zheng et al., 2025).

However, these approaches face two significant issues. First, the single reasoning paradigm and the chain-like reasoning process often fail to effectively explore the solution space, particularly when reasoning relies on self-exploration. This process is vulnerable to low-quality intermediate reasoning steps and is easily trapped in a narrow solution space. Second, the agentic search workflow and guiding self-exploration become challenging when using relatively smaller language models (e.g., *Qwen2.5-7B-Instruct* (Yang et al., 2024)). Furthermore, trainable agentic LLMs require efficient reinforcement learning training data and are difficult to apply to models with hundreds of billions of parameters.

In response to these challenges, we propose AirRAG, a method that leverages autonomous strategic planning and reasoning capabilities and expands the solution space using Monte Carlo Tree Search (MCTS). We design five fundamental reasoning actions: system analysis, direct answer, retrieval-answer, query transformation, and summary-answer. These actions are the core and frequently used ones in the deep search scenarios, which can effectively address a wide range of problems in various scenarios, including those that require progressive or parallel queries. Importantly, these actions can be executed efficiently on LLMs of different scales. Additionally, we introduce MCTS and self-consistency to enable controllable reasoning path generation and efficient inference scaling. To accurately select the answer from multiple reasoning paths, we combine a voting mechanism with a process-supervised reward model. As inference computation increases, our approach demonstrates significant performance improvements as shown in Figure 1. Moreover, AirRAG features a flexible architecture that can easily integrate other advanced methods into the approach as additional action branches. In summary, our main contributions are as follows:

- We design five fundamental reasoning actions that can address most problem types in deep search scenarios, ensuring controllable reasoning processes.
- We introduce MCTS and self-consistency to effectively expand the solution space for complex tasks. Our approach improves generalization and performance through comprehensive inference scaling and a pluggable architecture.
- We show thorough experimental results that AirRAG outperforms current iterative or agentic methods, effectively activating the planning and reasoning capabilities of LLMs and flexibly expanding the solution space.

2 Related Work

Retrieval-Augmented Generation (RAG). RAG has demonstrated significant improvements in the performance of LLMs in knowledge-intensive tasks. Compared to vanilla RAG, optimizing the query and retrieval process enhances knowledge correlation and, consequently, improves reasoning performance. Several methods, such as query expansion and transformation, have been proposed to achieve better retrieval results (Zhou et al., 2023; Ma et al., 2023; Gao et al., 2023). However, as task complexity increases, retrieving sufficient knowledge in a single query becomes increasingly difficult. To address this, iterative retrieval techniques have been proposed to gather additional contextual references. For instance, IR-CoT (Trivedi et al., 2023) utilizes chain-of-thought (CoT) to guide the retrieval process, refining the CoT with the retrieved information. Similarly, ITER-RETGEN (Shao et al., 2023) combines retrieval and generation modules to promote a deeper understanding of specific tasks.

Autonomous Planning and Reasoning in RAG. In addition to optimizing retrieval, activating the planning and reasoning capabilities of LLMs can significantly improve the efficiency and relevance of the retrieved information. Self-RAG and its variants (Asai et al., 2024; Yan et al., 2024; Jeong et al., 2024) adopt a self-reflection mechanism that iteratively predicts reflection tokens during training, enabling better control during inference. Auto-RAG (Yu et al., 2024) systematically plans retrievals and refines queries to acquire valuable knowledge through multi-turn iterations. Iter-DRAG (Yue et al., 2024) explores inference scaling strategies in RAG, improving LLMs' ability to effectively acquire and utilize contextual information. Search-o1 (Li et al., 2025) designs an agentic search workflow to dynamically obtain effective knowledge. ReSearch (Chen et al., 2025) and Deep-Researcher (Zheng et al., 2025) train agentic LLMs to reason with search using reinforcement learning. Despite the progress made in these methods, they often struggle to explore the solution space effectively during reasoning. Self-exploration frequently leads to being trapped in a limited solution space, hindered by low-quality reasoning steps even after multiple iterations. This issue is often attributed to the chain reasoning pattern and the difficulty small-scale LLMs face when handling overly complex tasks in a single iteration.

Monte Carlo Tree Search (MCTS). To address these challenges, tree-based search algorithms, particularly Monte Carlo Tree Search (MCTS), have emerged as effective tools to expand search spaces and enhance reasoning capabilities (Silver et al., 2017; Chen et al., 2024; Qi et al., 2024; Zhang et al., 2024). MCTS has been shown to extend reasoning by exploring multiple branching queries, thus enabling the exploration of diverse reasoning paths (Yao et al., 2023; Besta et al., 2024). In the mathematical reasoning scenario, Zhang et al. (2024) and Chen et al. (2024) leverage MCTS to achieve more efficient exploration of solution spaces, while Qi et al. (2024) designs rich human-like reasoning actions to improve reasoning trajectories. Furthermore, recent research indicates that inference scaling (Yue et al., 2024) and self-consistency (Wang et al., 2023) can lead to substantial improvements. In this context, our approach samples diverse reasoning paths to achieve both inference scaling and self-consistency verification during the next expansion of the action space.

Unlike existing methods that focus on optimizing query and retrieval processes or leveraging LLMs' reasoning capabilities through iterative retrieval, AirRAG uniquely integrates MCTS and self-consistency to systematically expand the solution space and ensure the controllability of the reasoning process. Simultaneously, we design five fundamental reasoning actions that effectively address a broader range of question types, particularly in complex tasks. In the experiment, we thoroughly verify the performance gains brought by the inference scaling law and investigate how to rationally allocate inference resources.

3 Methodology

In order to effectively explore the solution space during reasoning, we propose a controllable tree-based framework of RAG. This framework combines Monte Carlo Tree Search (MCTS) with five distinct reasoning actions, enabling efficient and controlled expansion of the solution space. Meanwhile, we further implement more comprehensive inference scaling strategies based on Yue et al. (2024) and employ pruning techniques along with computationally optimal strategies to strike a balance between effectiveness and efficiency. The whole process is illustrated in Figure 2.

3.1 Define Fundamental Reasoning Actions

Relying solely on the autonomy of LLMs for iterative self-exploration often results in getting trapped in a solution space that is difficult to navigate, especially when dealing with different types of complex questions. IterDRAG (Yue et al., 2024) uses a single action type to generate the next reasoning step, which can lead to ineffective space exploration. The core of MCTS generation lies in the action space, which defines the scope of tree exploration. Based on advanced methods and reasoning language models, we summarize the most common actions in RAG, such as query transformation and retrieval answering. Meanwhile, the chain-of-thought in reasoning models has demonstrated superior performance in complex open-domain question answering. Therefore, simplifying human cognitive processes in complex reasoning is essential (Jaffe et al., 2023). Inspired by this, we introduce five fundamental human-like reasoning actions to bridge the gap between LLM reasoning and human cognition in RAG scenarios.

- A_1: *System Analysis* (SAY). This action analyzes the overall structure of the problem, followed by its decomposition or planning. It represents systematic and global thinking before problem-solving.

- A_2: *Direct Answer* (DA). This action leverages parametric knowledge of LLMs to answer questions directly, without relying on any external knowledge.

- A_3: *Retrieval-Answer* (RA). This action retrieves related knowledge from the external knowledge base to support subsequent reasoning.

- A_4: *Query Transformation* (QT). This action transforms human questions in order to improve retrieval performance. It supports various transformations, such as rewriting, step back prompting, follow-up questions and multi-query retrieval.

- A_5: *Summary-Answer* (SA). This action combines intermediate reasoning steps, answers and the initial questions to generate the final answer.

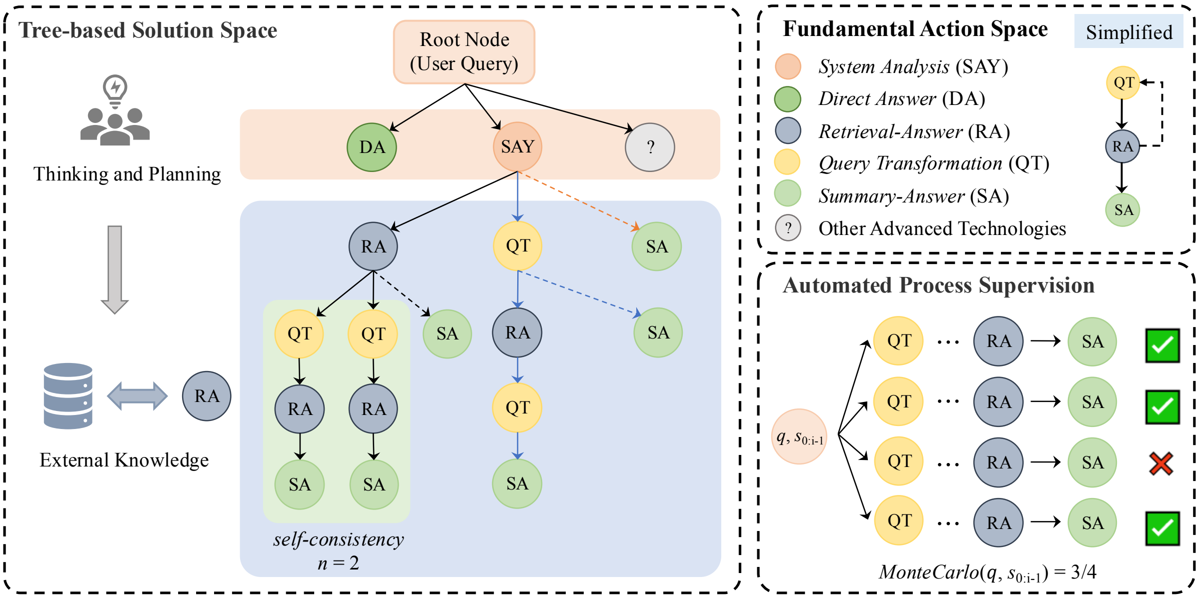

Figure 2: The schematic diagram of our proposed AirRAG. AirRAG implements a paradigm that combines system thinking with step-by-step reasoning. In the inference phase, we introduce MCTS and self-consistency to scaling computation, which significantly outperforms other strong baselines.

The above five actions define a highly diverse action space $\{A_1, A_2, A_3, A_4, A_5\}$. In the first step, the initial state is denoted as s_0 and then MCTS selects the action a_1 and a_2 to prompt the LLM to generate the next reasoning steps in parallel. Subsequent actions are performed sequentially to expand the reasoning path. It is important to note that there are sequential dependencies between different actions. For example, A_1 and A_2 can only be executed after the root question. Additionally, we incorporate the diverse sampling of self-consistency (Wang et al., 2023) for each action to expand the reasoning paths. Specifically, an action is more likely to generate the correct reasoning step if we sample multiple times in the current state. Finally, we can obtain multiple generated reasoning trajectories, such as $[s_0 \oplus s_{1:n}]$. To further improve inference efficiency, we choose the action $\{A_3, A_4, A_5\}$ as a simplified action space, referred to as *AirRAG-Lite*, which achieves a better balance between efficiency and effectiveness.

3.2 Perform Reasoning Processes via MCTS

3.2.1 Solution Generation

Based on the action space defined above, we introduce MCTS to generate candidate reasoning trajectories. The initial root node, s_0, represents the question without any reasoning steps. The policy is directly modeled by a language model as $\pi(a|s) = \text{LM}(a|s)$, and the state transition function combines preceding reasoning steps with current actions, i.e., $s_i = \text{Concat}(s_{0:i-1}, a_j)$. During each MCTS rollout, we execute multiple steps, including *selection*, *expansion*, *simulations*, and *backpropagation*. Multiple rollouts are performed to expand the solution space. To balance the exploration and exploitation, we adopt the well-known Upper Confidence Bounds applied to Trees (UCT) (Kocsis and Szepesvári, 2006) for node selection as follows:

$$\text{UCT}(s, p) = \frac{Q(s, a)}{N(s)} + w\sqrt{\frac{\log N_p(s)}{N(s)}}, \quad (1)$$

where $Q(s, a)$ is the reward value for node s and is updated through backpropagation. $N(s)$ denotes the number of visits to s, p is the parent node of s, and w is the weight to balance exploration and exploitation.

When the search reaches a terminal node, defined either by a terminal state or a predetermined maximum tree depth d, we obtain a trajectory from the root to the terminal node. All trajectories from the rollout iterations are collected as candidate solutions. Section 3.3 explains how we select the optimal answer node from these trajectories.

3.2.2 Inference Scaling

Numerous studies have demonstrated that scaling inference computation can significantly improve the performance of LLMs without additional training (Snell et al., 2024; Yue et al., 2024). Based on the above methods, we explore strategies to leverage inference computation scaling in AirRAG. One straightforward strategy is extending the *effective context length* (short for $L_{\max}$) during the document retrieval phase, allowing more related documents to supplement the knowledge base. Additionally, we perform multiple *rollouts* to thoroughly explore the solution space relying on the tree-based search. Adjusting *the number of output sequences* (n) generated during certain actions enables self-consistency verification and further inference scaling. These strategies provide flexibility for scaling inference computation in RAG, empowering LLMs to address complex knowledge-intensive queries more effectively.

To improve efficiency and minimize redundant computations, we implement an early pruning strategy for state nodes and reasoning paths. Deduplication is applied to the output sequence states generated by each action, ensuring the diversity of the subsequent path. Furthermore, if multiple rollouts select the same state sequence, only one valid reasoning path is retained.

3.2.3 Flexible Architecture

Our tree-based architecture provides the flexibility to integrate other advanced approaches. We reproduce the IterDRAG method based on the prompt design by Yue et al. (2024). Meanwhile, inspired by its iterative implementation, we simplify the fundamental action space to $\{A_3, A_4, A_5\}$, enabling a faster implementation while still achieving relatively good results. These methods serve as an exploratory extension of our approach and can be activated or deactivated as needed. Due to the training-free nature of our method, its generator LLM can be arbitrarily replaced with the strongest existing models for performance improvement.

3.3 Select the Optimal Answer Node

For common mathematical reasoning tasks, a simple consistency-based method can efficiently select the most precise reasoning path. For example, the most frequent number extracted from multiple candidate solutions in MATH (Hendrycks et al., 2021) can be chosen as the final answer. However, extracting precise answers and performing effective aggregation becomes more challenging for knowledge-intensive tasks. To address this, we design two self-consistency verification methods for such problems. *Jaccard similarity* and *text embeddings* are two different approaches used in natural language processing to measure the similarity between texts. We apply these methods to cluster text answers and compute answer scores as follows:

$$\text{jcdScore}_i = \frac{1}{N}\sum_{j=1}^{N}\frac{|A_i \cap A_j|}{|A_i \cup A_j|}, \tag{2}$$

$$\text{embScore}_i = \frac{1}{N}\sum_{j=1}^{N}\cos(E_i, E_j), \tag{3}$$

where N is the number of valid answer nodes, A_i is the word-level set of answer text i, and E_i denotes the embedding vector of answer text i.

In addition, we further investigate the *self-refine* and process-supervision *reward model* to identify the most accurate reasoning trajectory. Self-refinement uses the A_5 (Summary-Answer) action to refine the final answer from all candidate answer nodes. The reward modeling process consists of two steps: data synthesis and instruction tuning.

- **Data synthesis**: We leverage MCTS to perform multiple rollouts on partial training sets. Based on known ground truth, we sample positive and negative reasoning trajectories and use Monte Carlo estimation to evaluate intermediate state scores.

- **Instruction tuning**: Synthetic samples are used to fine-tune a relatively small LLM, such as *Qwen2.5-14B-Instruct*.

4 Experiments

In this section, we conducted experiments on complex QA benchmarks by answering the following research questions.

- **RQ1**: Does AirRAG outperform state-of-the-art baselines?

- **RQ2**: How does AirRAG perform when it comes to comprehensive inference scaling?

- **RQ3**: What is the performance benefit of AirRAG in optimizing the allocation of inference computation?

- **RQ4**: How does AirRAG perform for various verification methods for multiple candidate rollouts?

- **RQ5**: What is the intuitive performance of AirRAG in the reasoning process?

4.1 Experimental Settings

4.1.1 Datasets

To evaluate the effectiveness of AirRAG, we conduct experiments on various question-answering (QA) tasks, including both open-domain QA and multi-hop QA. The complex multi-hop QA datasets consist of HotpotQA (Yang et al., 2018), MuSiQue (Trivedi et al., 2022) and 2WikiMultiHopQA (2Wiki) (Ho et al., 2020). Other single-hop QA datasets include Natural Questions (NQ) (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), PopQA (Mallen et al., 2023) and WebQA (Berant et al., 2013).

4.1.2 Implementation Details

We use the hyperparameters reported for the existing models whenever available. Implementation details are available in the Appendix A.

4.1.3 Baselines and Metrics

To investigate the enhancement effects of thinking and planning on complex RAG tasks, we compare it with vanilla RAG, which performs only a single retrieval and generation process. We evaluate the naive generators of *Qwen2.5* (Yang et al., 2024) series instruction models and *Llama3-8B-Instruct* (Grattafiori et al., 2024). In the retrieval phase, we employ *multilingual-e5-base* (Wang et al., 2024) as the retriever. The prompt of vanilla RAG are shown in the Appendix D. For iterative retrieval, we compare AirRAG with IterRetGen (Shao et al., 2023), Self-RAG (Asai et al., 2024), Auto-RAG (Yu et al., 2024), and IterDRAG (Yue et al., 2024). For agentic retrieval, we compare AirRAG with Search-o1 (Li et al., 2025), ReSearch (Chen et al., 2025), and DeepResearcher (Zheng et al., 2025). To further explore RAG performance and inference computation scaling, we focus on a comparison with IterDRAG for a given budget on inference computation. For evaluation metrics, we report Exact Match (EM), F1 score (F1) and Accuracy (Acc) between the generated summary and gold answer, where accuracy measures whether the gold answer is covered in the generated answer.

4.2 Main Results (RQ1)

We first evaluate the performance of AirRAG on various complex QA datasets. Table 1 compares its accuracy and F1 scores with strong baselines based on LLMs of different scales. The optimal performance exhibits consistent gains as the LLMs scale up. For the *Qwen2.5-7b-instruct* model, our approach achieves the best performance, even surpassing the trainable approaches. To further validate its effectiveness on large-scale reasoning models, we also conduct experiments on *Qwen3-235B* in both the thinking mode and non-thinking mode. In thinking mode, our approach achieves state-of-the-art performance among all datasets.

4.3 Inference Scaling for RAG (RQ2)

Inference computation scaling can enable LLMs to improve their output performance (Snell et al., 2024). Self-consistency can also improve the robustness of the reasoning process (Wang et al., 2023). Therefore, we carry out a comprehensive experimental analysis on the inference computation scaling. Specifically, we can adjust both the number of retrieved documents in a single retrieval and the effective context length in all iterations. The average performance of three datasets exhibits consistent gains in Figure 3. In subsequent experiments, unless otherwise specified, the data presented represent the average performance across the HotpotQA, MuSiQue, and 2Wiki datasets. As shown in Figure 1, the average performance increases with the number of output sequences per action, demonstrating the effectiveness of self-consistency. We also investigate the number of effective reasoning paths under different rollouts in Figure 4. The performance improvement caused by the increase of effective reasoning paths in the early stage is relatively high. We provide additional dataset-specific results in Appendix B.

4.4 Ablation Studies

Effect of Computationally Optimal Strategies (RQ3). Extensive experiments show that the outputs of certain actions (e.g., RA, DA and SA) are almost consistent when performing multiple generations. Therefore, we only increase the number of output sequences (short for n) for the remaining actions (e.g., SAY and QT), which reduces invalid inference computation while maintaining good results. This also reflects that this kind of reasoning action, which effectively activates the creativity of LLMs, requires more diversified sampling strategies. We adjust the sampling parameters (top-p=1.0 and temperature=1.0) to improve the diversity of the model output. The complete experimental results in Table 2 show that the diversity of key actions can significantly improve performance.

From the aforementioned experiments, it is ob-

Method	**NQ**		**TriviaQA**		**HotpotQA**		**MuSiQue**		**2Wiki**		**Average**	
	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc
Qwen2.5-7B												
ZeroShot QA	38.4	37.4	57.7	56.3	36.1	34.8	9.1	7.5	45.0	44.1	37.3	36.0
Vanilla RAG	57.8	53.9	70.1	66.3	61.3	56.9	13.5	8.3	45.9	42.8	49.7	45.6
IterDRAG*	58.3	54.1	73.5	69.1	65.3	60.7	18.3	13.0	51.8	47.0	53.4	48.8
Search-o1*	57.8	54.3	72.6	69.8	57.3	54.2	20.2	18.5	56.9	53.6	53.0	50.1
ReSearch*	61.3	59.6	**76.2**	**73.4**	70.3	68.1	30.9	27.4	62.5	61.0	60.2	**57.9**
DeepResearcher	**62.4**	**59.8**	75.9	73.2	67.6	63.9	**33.9**	**27.8**	65.4	62.3	61.0	57.4
AirRAG-Lite	60.8	57.3	74.1	70.0	68.2	63.2	23.4	17.3	51.1	48.0	55.5	51.2
AirRAG	60.4	56.2	74.3	70.1	**74.9**	**70.0**	30.3	23.6	**65.7**	**63.2**	**61.1**	56.6
Qwen2.5-14B												
ZeroShot QA	47.1	46.3	70.7	69.5	42.5	41.3	13.5	12.1	48.2	47.3	44.4	43.3
Vanilla RAG	62.0	58.2	75.2	71.2	67.0	62.0	20.0	14.4	52.0	49.4	55.2	51.0
IterDRAG*	58.5	53.9	76.4	72.3	71.8	66.2	23.9	17.2	57.4	54.1	57.6	52.7
Search-o1*	61.2	59.7	74.3	72.6	71.4	67.4	23.2	20.4	58.1	55.8	57.6	55.2
AirRAG-Lite	64.8	60.9	**78.9**	**75.0**	77.2	72.5	30.3	24.3	70.3	68.0	64.3	60.1
AirRAG	**66.2**	**62.1**	78.1	73.8	**79.9**	**75.3**	**36.0**	**31.9**	**70.4**	**68.7**	**66.1**	**62.4**
Qwen2.5-32B												
ZeroShot QA	46.8	45.9	69.8	68.8	42.9	41.8	11.1	9.6	48.1	47.2	43.7	42.7
Vanilla RAG	60.6	56.7	75.4	71.6	66.5	62.5	19.5	13.8	51.8	49.6	54.7	50.8
IterDRAG*	61.1	56.7	76.9	73.0	72.0	67.3	23.3	17.9	58.2	55.4	58.3	54.1
Search-o1*	63.5	61.6	75.6	72.8	72.8	68.4	30.2	27.3	60.5	58.9	60.5	57.8
ReSearch*	63.8	61.2	74.6	72.3	76.2	72.6	**38.3**	**33.4**	66.8	62.8	63.9	60.5
AirRAG-Lite	65.6	61.9	75.8	71.6	79.3	74.8	33.3	32.5	**72.0**	**70.8**	65.2	62.3
AirRAG	**66.5**	**62.7**	**78.9**	**74.6**	**81.1**	**76.1**	36.5	32.7	71.9	70.6	**67.0**	**63.3**
Qwen3-235B (non-thinking)												
ZeroShot QA	64.1	63.7	77.1	76.3	53.9	52.9	17.1	15.7	56.8	56.1	53.8	52.9
Vanilla RAG	66.0	65.3	78.0	76.2	69.2	67.7	20.8	19.1	53.3	52.2	57.4	56.1
IterDRAG*	65.7	63.3	78.8	76.9	75.5	69.7	32.2	25.2	64.8	60.8	63.4	59.2
Search-o1*	67.3	66.2	77.3	76.4	75.9	73.8	36.7	34.2	71.3	68.2	65.7	63.7
AirRAG-Lite	**67.7**	**66.9**	77.6	75.8	78.3	76.8	43.7	36.5	74.9	74.1	68.4	65.7
AirRAG	66.4	65.6	**79.1**	**77.3**	**79.6**	**78.1**	**47.2**	**40.0**	**76.2**	**75.5**	**69.7**	**67.3**
Qwen3-235B (thinking)												
ZeroShot QA	66.1	65.6	79.3	78.6	54.2	53.5	21.3	19.9	56.6	56.3	55.5	54.8
Vanilla RAG	67.6	67.1	77.9	76.9	72.5	71.9	18.2	16.4	57.5	56.6	58.7	57.8
IterDRAG*	68.8	67.1	80.9	78.7	76.3	71.1	30.0	23.1	67.1	64.5	64.6	60.9
Search-o1*	67.2	66.4	78.1	77.5	75.2	72.4	33.4	28.9	69.2	66.3	64.6	62.3
AirRAG-Lite	73.2	72.7	81.1	80.1	**86.2**	**85.6**	44.2	37.8	76.4	75.6	72.2	70.3
AirRAG	**74.3**	**72.8**	**81.4**	**80.1**	84.7	84.0	**47.5**	**40.3**	**76.8**	**76.2**	**72.9**	**70.7**

Table 1: Overall evaluation results on the test sets of five datasets. * indicates the results reproduced by us. The best results for each model are in **bold**. The number of both rollouts and output sequences is set to 1. The number of documents for a single retrieval is set to 5.

served that the recall and accuracy of model are linearly correlated. Intuitively, the size of document database is also related to the recall score. By reducing the scale of the document database, we find a gradual improvement in model performance (shown in Figure 4). This observation provides experimental evidence for effective database partitioning in practical application.

Effect of Verification Methods (RQ4). The larger search space also generates more candidate reasoning trajectories. Therefore, how to select the optimal trajectory is crucial for the final performance. We compare multiple verification methods with the average scores of all candidates in Figure 5. These two self-consistency verification methods are always slightly better than the average score, but they are not nearly as good as the SA and QwenRM methods. The SA method uses the LLM to further refine the final answer from all candidate rollouts, which is simple and effective. Finally, the reward model achieves the most competitive results due to the introduction of supervised information on key intermediate reasoning steps in the training process. However, collecting process-supervised train-

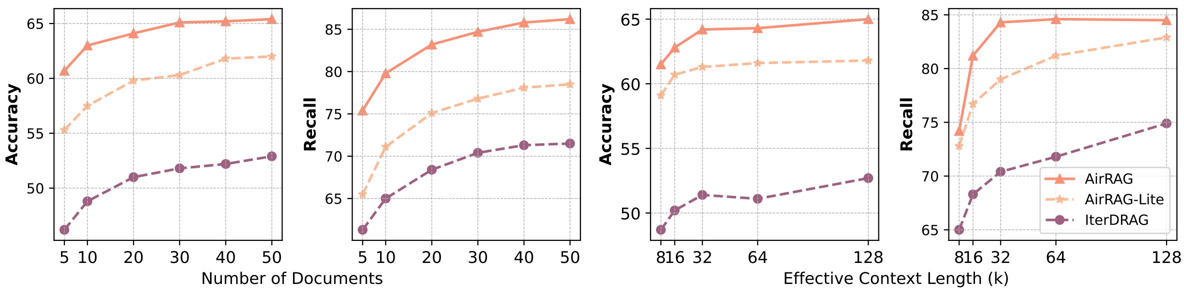

Figure 3: Impact of the retrieved document number scaling (**Left**) and the maximum context length scaling (**Right**) on model performance (averaged Accuracy and Recall of three datasets). All methods show consistent performance improvements as the effective inference computation scales.

Method	Average	
	F1	Acc
Vanilla RAG	47.0	43.2
IterDRAG	49.8	45.9
AirRAG		
+ n_{all}=1	62.9	58.8
+ n_{all}=3	63.4	62.1
+ n_{a_1,a_4}=3, n_{a_2,a_3,a_5}=1	63.2	62.0
+ n_{a_1,a_4}=3, n_{a_2,a_3,a_5}=1, q_{div}=1.0	**65.1**	**63.9**

Table 2: Performance comparison with different computationally optimal strategies on the HotpotQA, MuSiQue and 2Wiki datasets. n_{a_i} denotes the number of output sequences of the action a_i in a single extension. q_{div} indicates that setting top-p to 1.0 and temperature to 1.0 for query-related actions, i.e. SAY and QT, increases the diversity of reasoning. The default sampling parameters top-p, top-k and temperature are set to 0.8, 50 and 0.7 respectively. Rational sampling strategies further improve performance across multiple datasets.

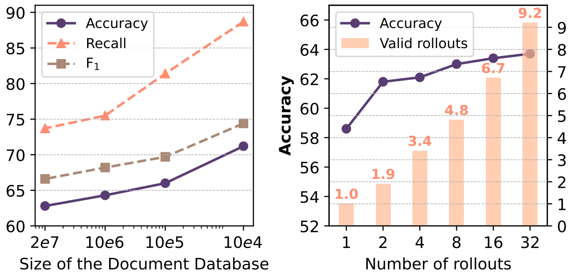

Figure 4: **Left**: Performance comparison under different size of document database. A streamlined database can maintain a better performance. **Right**: Performance comparison in increasing the number of valid rollouts. Sampling a higher number of diverse reasoning paths consistently improves accuracy.

ing samples requires high computational costs and high-quality raw data. In the practical application scenario, we can choose the appropriate method while balancing efficiency and effectiveness.

4.5 Qualitative Analysis (RQ5)

To make it easier to understand why our proposed AirRAG works, we present a qualitative analysis in MuSiQue. Existing iterative methods are often trapped in a single solution space when confronted with complex tasks. As illustrated in Figure 13, these iterative methods exhibit a key limitation that insufficient or ambiguous retrieval context can lead to repetitive follow-up queries until it reaches the predefined maximum depth of iterations. This inefficient iteration results in high computational cost and incorrect answer. In contrast, our proposed AirRAG designs efficient reasoning actions to achieve autonomous planning and reasoning. As shown in Figure 14, the SAY action decomposes the original query into a more rational sequence of sub-queries, and then the combination of RA and QT ensures the accuracy of the intermediate reasoning step. We eventually leverage the efficient reasoning trajectory to obtain the correct answer.

5 Conclusions

In this paper, we propose AirRAG, a novel RAG approach to fully leverage the planning and reasoning capabilities of LLMs. AirRAG designs an efficient action space for the controllable reasoning generation. We also introduce Monte Carlo Tree Search to expand the solution space. Meanwhile, by employing the tree-based search and self-consistency verification, we explore potential reasoning paths and achieve comprehensive inference computation scaling. In addition, computationally optimal strategies are used to apply more computation to key actions, leading to further performance improvements. Experimental results on diverse QA datasets demonstrate the significant superiority of AirRAG over other methods designed for complex deep search scenarios.

Limitations

Although our model achieves competitive performance in various RAG tasks, there are some methods and limitations that can be improved. The current optimal computation allocation strategy is derived from sufficient experiments. We can consider designing an automated policy model to implement the trade-off between computational cost and performance. Despite great efforts in the inference scaling of RAG, the experimental analysis may be limited due to the massive computational cost of tree-based search approaches. We will explore more complex reasoning tasks to verify the robustness and effectiveness of our approach. In addition, the large search space also brings more noise information, so we will further investigate the reward model or strategy to explore a better reasoning path.

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 *The Twelfth International Conference on Learning Representations*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michał Podstawski, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of Thoughts: Solving Elaborate Problems with Large Language Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690.

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. 2024. Alphamath almost zero: Process supervision without process. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. 2025. Research: Learning to reason with search for llms via reinforcement learning. *Preprint*, arXiv:2503.19470.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. 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.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, and et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

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*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Paul I Jaffe, Russell A Poldrack, Robert J Schafer, and Patrick G Bissett. 2023. Modelling human behaviour in cognitive tasks with latent dynamical systems. *Nature Human Behaviour*, 7(6):986–1000.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. 2024. Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7036–7050, Mexico City, Mexico. Association for Computational Linguistics.

Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992, Singapore. Association for Computational Linguistics.

Jiajie Jin, Yutao Zhu, Xinyu Yang, Chenghao Zhang, and Zhicheng Dou. 2024. Flashrag: A modular toolkit for efficient retrieval-augmented generation research. *CoRR*, abs/2405.13576.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. In *Proceedings of the 40th International Conference* *on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 15696–15707. PMLR.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*, pages 282–293, Berlin, Heidelberg. Springer Berlin Heidelberg.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025. Search-o1: Agentic search-enhanced large reasoning models. *Preprint*, arXiv:2501.05366.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting in retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5303–5315, Singapore. Association for Computational Linguistics.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822, Toronto, Canada. Association for Computational Linguistics.

Zhenting Qi, Mingyuan Ma, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. 2024. Mutual reasoning makes smaller llms stronger problem-solvers. *Preprint*, arXiv:2408.06195.

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.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 31210–31227. PMLR.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *Preprint*, arXiv:2408.03314.

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

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. 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)*, pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Multilingual e5 text embeddings: A technical report. *arXiv preprint arXiv:2402.05672*.

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

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. 2024. Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

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*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Tian Yu, Shaolei Zhang, and Yang Feng. 2024. Auto-rag: Autonomous retrieval-augmented generation for large language models. *Preprint*, arXiv:2411.19443.

Zhenrui Yue, Honglei Zhuang, Aijun Bai, Kai Hui, Rolf Jagerman, Hansi Zeng, Zhen Qin, Dong Wang, Xuanhui Wang, and Michael Bendersky. 2024. Inference scaling for long-context retrieval augmented generation. *Preprint*, arXiv:2410.04343.

Di Zhang, Jianbo Wu, Jingdi Lei, Tong Che, Jiatong Li, Tong Xie, Xiaoshui Huang, Shufei Zhang, Marco Pavone, Yuqiang Li, Wanli Ouyang, and Dongzhan Zhou. 2024. Llama-berry: Pairwise optimization for o1-like olympiad-level mathematical reasoning. *Preprint*, arXiv:2410.02884.

Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H. Chi, Quoc V Le, and Denny Zhou. 2024. Take a step back: Evoking reasoning via abstraction in large language models. In *The Twelfth International Conference on Learning Representations*.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. 2025. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. *Preprint*, arXiv:2504.03160.

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

A Implementation Details

For evaluation, we randomly select 1,000 samples from the whole validation sets of each dataset as our final test set, with a fixed random seed 0. To better understand the complexity of multi-hop reasoning in these datasets, we analyze the hop distribution of the HotpotQA, MuSiQue, and 2WikiMultiHopQA test sets in Figure 6. The statistics show that there is a high proportion of complex reasoning queries with 3 hops or more (aboout 30%, 50%, 25%). HotpotQA lacks explicit hop annotations, so we instead count the number of supporting facts. MuSiQue has a significantly higher proportion of 3-hop and 4-hop queries compared to the other datasets, indicating great reasoning complexity. This observation is further corroborated by our experimental results in Table 1 and Figure 7. The performance of our approach on MuSiQue is much lower than those of the other two datasets.

In the retrieval process, we employ the *multilingual-e5-base* (Wang et al., 2024) as the retriever and use the widely used Wikipedia dump from December 2018 as the retrieval corpus (Karpukhin et al., 2020) which comprises over 21 million passages. For generation, the default sampling parameters top-p, top-k and temperature are set to 0.8, 50 and 0.7 respectively. Evaluation metrics include Exact Match (EM), F1 score (F1), and Accuracy (Acc), where accuracy indicates whether the ground truth is a substring of the final generated answer. For reward model training, we sample 8,000 question-answer pairs from each dataset and generate more than 156,000 reasoning paths using our proposed AirRAG (rollouts=32, n=4, q_{div}=1.0). In inference scaling experiments, we sample maximum computation budgets $L_{\max}$ (e.g., 8k, 16k, 32k, 64k and 128k tokens). The $L_{\max}$ (maximum effective context length) denotes the maximum number of input tokens across all rollouts following (Yue et al., 2024). The predetermined maximum tree depth d is set to 10, specifically indicating that the SAY and SA actions are executed once, while the RA-QT or QT-RA actions have a maximum of 4 iterations.

B Additional Experiment Results

We evaluate the performance of AirRAG on various complex QA datasets. Table 3 compares its accuracy and F1 with strong baselines under the given inference computation budget, which is implemented based on *Qwen2.5-14B-Instruct* and one million document database. The optimal performance exhibits consistent gains as $L_{\max}$ expands, which is termed as the *inference scaling laws* for RAG (Yue et al., 2024). We integrate the remaining methods for a given maximum computational budget into our approach, dubbed as *AirRAG-Blender*. The best results are obtained by using only the SA action to refine the final answer from all candidates, as shown in Table 3. This also demonstrates the flexibility of our approach architecture. In addition, to verify the robustness and generalization of AirRAG, Table 4 shows the performance on more diverse LLMs and datasets. For a fair comparison, we utilize the widely used Wikipedia dump from December 2018 (Karpukhin et al., 2020) as the retrieval corpus. We observe consistent improvements over vanilla RAG and existing iterative methods (more than 10% on average). The significant boost over IterDRAG and Auto-RAG suggests that AirRAG explores more effective reasoning paths through the human-like thinking paradigm and tree-based search. Furthermore, we present detailed inference scaling results for each dataset individually, as shown in Figure 7 and Figure 8.

C Inference Efficiency Analysis

Given the inherently large search space of tree-based search, we design computational optimization strategies for different actions to avoid inefficient and redundant expansions, as shown in Table 2. Furthermore, in Section 3.2.2, we propose pruning strategies for state nodes and reasoning paths. These optimizations significantly reduce inefficient LLM inference and repetitive path exploration. In practical applications, we can select appropriate configuration parameters such as rollout, n, and $L_{\max}$ based on computational resource budgets and time constraints, ensuring effectiveness while achieving inference efficiency comparable to current mainstream iterative RAG approaches. We analyze the average inference efficiency per sample on the HotpotQA dataset in Table 5.

D Prompt Examples

Given a user input query, our proposed AirRAG, as shown in Figure 2, first attempts the *direct answer* (DA) action without prompts and performs *system analysis* (SAY) using the prompt in Figure 9. Subsequently, AirRAG performs *retrieval and answer* (RA) with the prompt in Figure 11, or *query transformation* (QT) to generate refined queries for

$L_{\max}$	Method	HotpotQA		MuSiQue		2Wiki		**Average**	
		F1	Acc	F1	Acc	F1	Acc	**F1**	**Acc**
8k	ZeroShot QA	42.5	41.3	13.5	12.1	48.2	47.3	34.7	33.6
	Vanilla RAG	70.3	65.4	23.0	17.7	55.8	53.4	49.7	45.5
	IterDRAG*	74.3	69.1	26.7	19.4	60.5	57.6	53.8	48.7
	AirRAG-Lite	**80.6**	**75.4**	35.4	28.9	75.3	73.1	63.8	59.1
	AirRAG	79.6	75.2	**41.0**	**35.0**	**76.0**	**74.2**	**65.6**	**61.5**
	AirRAG-Blender	81.1	79.8	41.6	36.4	82.2	81.7	68.3	66.0
32k	Vanilla RAG	77.1	72.0	29.0	22.9	60.9	58.1	55.7	51.0
	IterDRAG*	77.7	71.6	30.8	22.3	63.0	60.2	57.1	51.4
	AirRAG-Lite	82.4	76.9	36.7	30.1	78.8	76.8	66.0	61.3
	AirRAG	**82.5**	**77.4**	**43.2**	**36.3**	**80.4**	**78.9**	**68.7**	**64.2**
	AirRAG-Blender	82.9	80.6	43.3	37.6	83.4	83.0	69.9	67.1
128k	IterDRAG*	76.8	71.0	31.7	24.8	65.5	62.4	58.0	52.7
	AirRAG-Lite	82.5	77.1	35.7	30.4	78.3	76.0	65.5	62.2
	AirRAG	**83.3**	**78.0**	**43.5**	**36.5**	**82.3**	**80.5**	**69.7**	**65.0**
	AirRAG-Blender	83.7	81.4	43.9	38.5	84.4	84.2	70.6	68.0

Table 3: Overall evaluation results under different computational resource budgets, where *Qwen2.5-14B-Instruct* is used as the generator LLM. * indicates the results reproduced by us. $L_{\max}$ denotes the maximum number of input tokens across all rollouts. The best results for each $L_{\max}$ are in **bold**. The number of both rollouts and output sequences is set to 1 for our proposed AirRAG methods.

Method	NQ	TriviaQA	PopQA	WebQA	HotpotQA	2Wiki
	EM	EM	F1	EM	F1	F1
Vanilla RAG	35.1	58.8	36.7	15.7	35.3	21.0
Self-RAG	36.4	38.2	32.7	21.9	29.6	25.1
Iter-RetGen	36.8	60.1	37.9	18.2	38.3	21.6
Auto-RAG	37.9	60.9	47.8	25.1	44.9	48.9
AirRAG	**53.6**	**63.2**	**51.8**	**52.6**	**67.6**	**66.3**

Table 4: Performance comparison on six benchmarks, where *Llama3-8B-Instruct* is used as the generator LLM. Partial experimental results are quoted from Jin et al. (2024) and Yu et al. (2024). The best results are in **bold**. The number of both rollouts and output sequences is set to 1. The number of documents for a single retrieval is set to 5. Our proposed AirRAG significantly outperform the others.

$L_{\max}$	Method	database_size	retrieval_time	retrieval_number	e2e
8k	Vanilla RAG	100w	0.900	1.00	3.828
	IterDRAG	100w	1.833	2.04	6.703
	AirRAG-Lite	100w	2.205	2.25	8.482
	AirRAG	100w	3.453	3.84	12.752

Table 5: Performance analysis of inference efficiency. $L_{\max}$ denotes the maximum number of input tokens across all rollouts. The retrieved database contains approximately one million documents. e2e and retrieval_time denote the average total time for a single question-answering process and the time spent on retrieval respectively, measured in seconds. Other inference configurations remain consistent with those in Table 3.

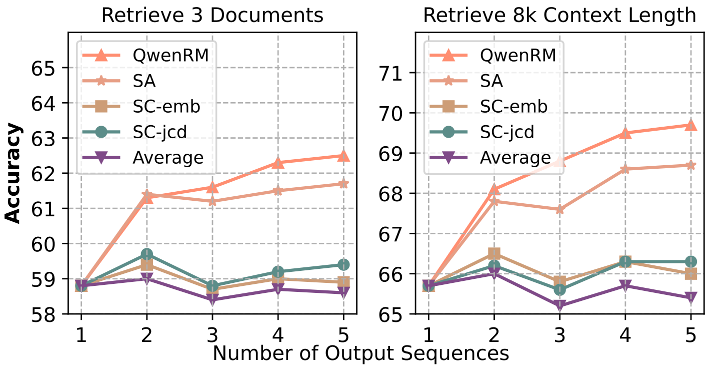

Figure 5: Performance comparison of different verification methods. "QwenRM" is short for reward model trained on the Qwen model. "SA" is the reasoning action of summary and answer. "SC-emb/jcd" are two self-consistency verification methods based on text embeddings and jaccard similarity. "Average" is the average score over all candidate rollouts. The single retrieval process is set to retrieve three documents or fixed 8k context.

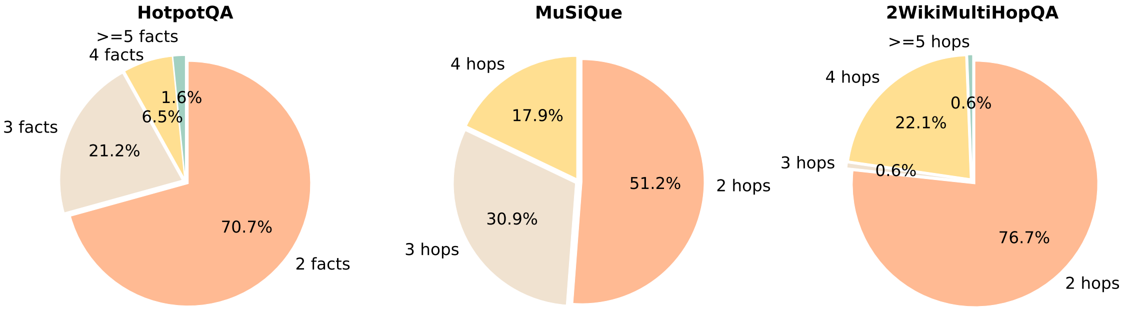

Figure 6: Overview of the distribution of query complexity over three multi-hop QA datasets.

better retrieval and answer. This process of RA-QT or QT-RA can continuously iterate until no new sub-queries arise or the maximum iteration depth is reached. Finally, the *summary answer* (SA) in Figure 11 utilizes all the information and conclusions from intermediate steps to refine the final answer.

E Case Study

We select a sample from the complex multi-hop dataset MuSiQue to analyse in detail, as shown in Figures 13 and 14.

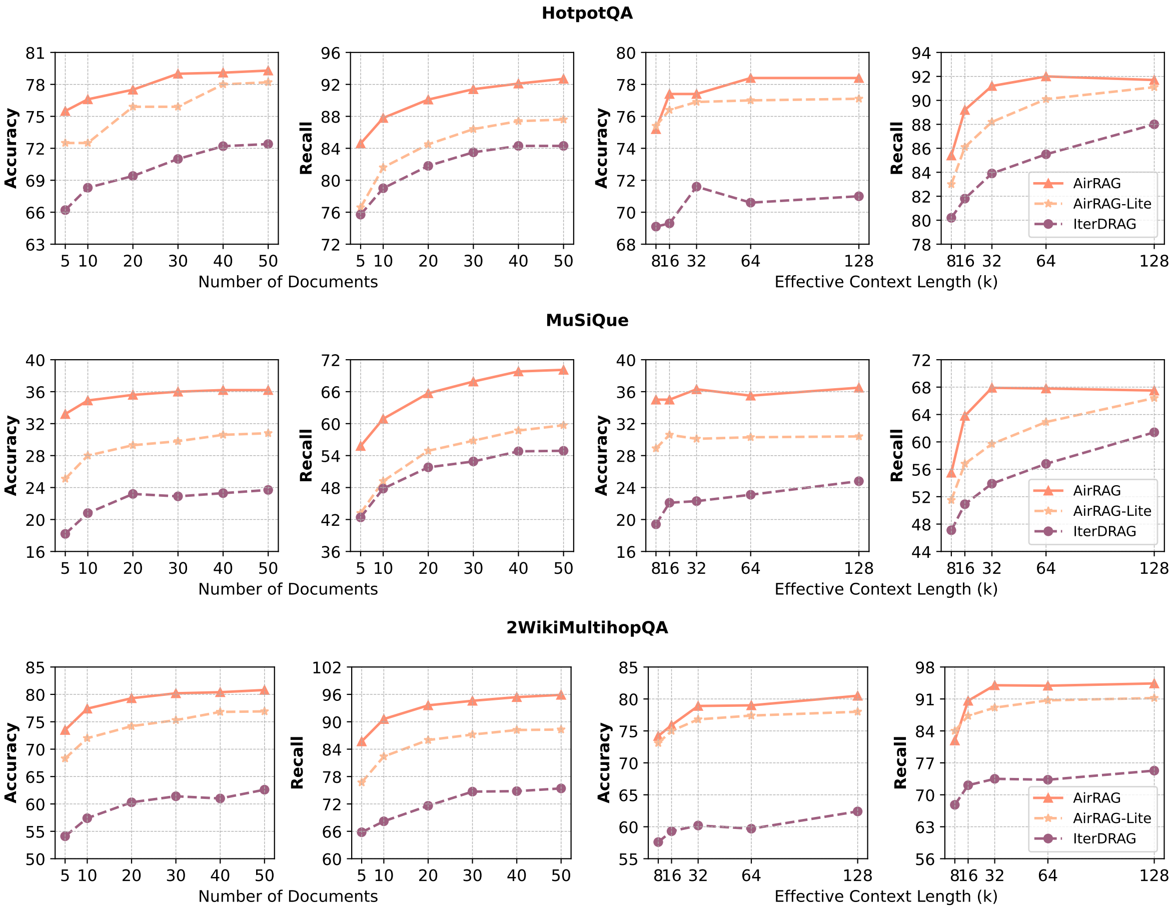

Figure 7: Impact of the retrieved document number scaling and the maximum context length scaling over three datasets.

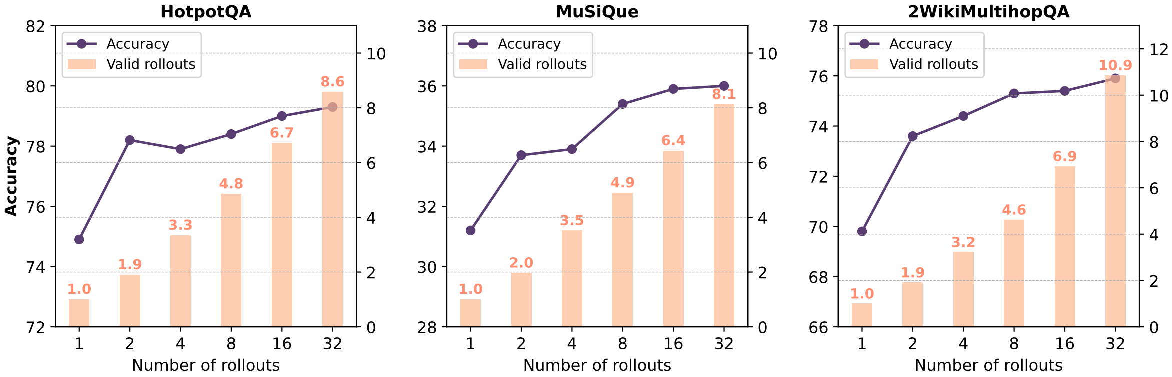

Figure 8: Performance comparison on the number of different effective rollouts over three datasets. Sampling more diverse reasoning paths consistently improves accuracy.

Example prompt for SAY reasoning action

Given the user query, you may rephrase it for better clarity, summarize it at a higher level, or decompose it into multiple sub-queries to facilitate more effective information retrieval and response generation. If no modification is necessary, return "None". Otherwise, list sub-queries, each on a new line.
<Here are some examples.>
Query: {question}
Output:

Figure 9: Example prompt for SAY reasoning action.

Example prompt for QT reasoning action

Given the context provided, please determine whether rephrasing, summarization, or decomposition into sub-queries is necessary to enhance the accuracy and efficiency of information retrieval and response generation. If no modification is required, return "None". Subsequent queries should be listed individually.
<Here are some examples.>
Main Query: {question}
History: {history}
This Query: {this_question}

Figure 10: Example prompt for QT reasoning action.

Example Prompt for RA actions

You are an expert in question answering. I am going to give you some contexts with may or may not be relevant to the question. Answer the question according to the contexts.
{contexts}
Question: {question}

Figure 11: Example Prompt for RA actions.

Example prompt for SA reasoning action

You are an expert in question answering. Given the context, sub-queries and responses, output a correct and concise answer to User Query.
<Here are some examples.>
User Query: {question}
{history}
Contexts: {contexts}
Final Answer:

Figure 12: Example prompt for SA reasoning action.

The iterative methods often traps in a solution space

Context: <Retrieved documents>
Question: What is the mouth of watercourse for the body of water where Bartram's Covered Bridge is located?
Follow up: Which body of water is Bartram's Covered Bridge located on?
Follow up: | Intermediate answer: | So the final answer is:

Output: Follow up: Which body of water is Bartram's Covered Bridge located on?

(Note: The context does not specify which body of water Bartram's Covered Bridge is located on, but it does mention that the bridge carries Goshen Road over Crum Creek. Therefore, we **need to confirm that Crum Creek is the body of water in question before proceeding to find the mouth of the watercourse.**)

...
··· (It enters an endless cycle of iteration until it reaches predefined maximum depth.)

Gold Answer: Delaware River
Process Annotation: Q1: Bartram's Covered Bridge » crosses
A1: Crum Creek
Q2: Q1 » mouth of the watercourse
A2: Delaware River

The iterative self-exploration often traps in a solution space with low-quality reasoning steps even after many attempts.

Figure 13: Incorrect reasoning path of iterative RAG methods. Ambiguity or missing information in the intermediate steps leads in duplicate sub-query.

The simple and effective action space of AirRAG leads to the correct reasoning trajectory

Question: What is the mouth of watercourse for the body of water where Bartram's Covered Bridge is located?

Action1: SAY
Input: What is the mouth of watercourse for the body of water where Bartram's Covered Bridge is located?
Output: **Where is Bartram's Covered Bridge located?**
What is the mouth of the watercourse for the body of water near Bartram's Covered Bridge?

Action2: RA
Input: <Retrieved documents> Where is Bartram's Covered Bridge located?
Output: Bartram's Covered Bridge is located on the border between Delaware County and Chester County, Pennsylvania, carrying Goshen Road over Crum Creek.

Action3: QT
Input: What is the mouth of the watercourse for the body of water near Bartram's Covered Bridge?
Output: **What is the mouth of Crum Creek?**

Action4: RA
Input: <Retrieved documents> What is the mouth of Crum Creek?
Output: Based on the provided contexts, there are two different locations referred to as Crum Creek, and they flow into different bodies of water:
1. The Crum Creek in Delaware County and Chester County, Pennsylvania, flows into the Delaware River in Eddystone, Pennsylvania.
2. The Crum Creek (also called Cathatachua Creek) in East Creek, New York, flows into the Mohawk River.
Since the question does not specify which Crum Creek is being referred to, both answers are possible. However, if we consider the first context as the primary reference, the mouth of Crum Creek is the Delaware River in Eddystone, Pennsylvania.

Action5: SA
Input: <Retrieved documents> <Sub-queries with answers>
Output: The mouth of the watercourse for the body of water where Bartram's Covered Bridge is located, **Crum Creek**, is the **Delaware River** in Eddystone, Pennsylvania.

Gold Answer: Delaware River.

AirRAG obtains the correct answer by performing five reasoning actions efficiently.

Figure 14: Effective reasoning rollout of AirRAG. The fundamental action space and tree-based search provides greater compatibility and flexibility, which avoids being trapped in a single solution space by a low-quality intermediate step for chain-like reasoning paradigm.