Lightweight Reflective Optimization for Multi-Agent RAG (LRO-mRAG)

Akash Pramod Yalla
UMass Amherst
USA
ayalla@umass.edu

Jack Kaefer
UMass Amherst
USA
jkaefer@umass.edu

Khiem Le
UMass Amherst
USA
ktle@umass.edu

Kirat Arora
UMass Amherst
USA
kiratarora@umass.edu

Sanjiv Kumaran Mohanraj
UMass Amherst
USA
smohanraj@umass.edu

Tung Ngo
UMass Amherst
USA
tungngo@umass.edu

Abstract

Multi-agent retrieval-augmented generation (mRAG) systems, pioneered by Salemi (2025) [3], utilize role-specialized agents (e.g., Planner, Searcher) coordinated by a central hub to address knowledge-intensive tasks like multi-hop question answering. However, static prompts restrict adaptability to dataset-specific patterns, leading to error propagation and suboptimal collaboration. Inspired by Multi-Agent System Search (MASS) by Zhou (2025) [6], we propose Lightweight Reflective Optimization for mRAG (LRO-mRAG), a novel approach that iteratively optimizes agent prompts through a reflective loop—identifying failures, evolving prompts locally, retrieving documents based on simulated prompts and distilling global rules—offering a scalable solution for resource-limited settings.

Keywords

multi-agent systems, retrieval-augmented generation, prompt optimization, low-compute AI, multi-hop QA

ACM Reference Format:
Akash Pramod Yalla, Jack Kaefer, Khiem Le, Kirat Arora, Sanjiv Kumaran Mohanraj, and Tung Ngo. 2025. Lightweight Reflective Optimization for Multi-Agent RAG (LRO-mRAG). In *Proceedings of ACM Conference (ACM Conference 2025).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/XXXXXXX.XXXXXXX

1 Problem Statement

In multi-agent retrieval-augmented generation (mRAG) systems, as introduced in the foundational work [3], static prompts $p_i \in P = \{p_1, \ldots, p_n\}$ for agents $A = \{A_1, \ldots, A_n\}$ (e.g., Planner, Searcher) limit adaptability to dataset-specific patterns, such as multi-hop reasoning dependencies, resulting in suboptimal collaboration efficiency and error propagation during iterative answer refinement. Inspired by the MASS framework [6], which automates MAS design through staged optimizations interleaving local prompt refinements Δp_i, we propose to iteratively optimize agent prompts to maximize a performance function $J(P) = w_1 \cdot \text{Correctness}(a, q, C) + w_2 \cdot \text{Faithfulness}(a, C)$, where a is the generated answer, q is the query, C is the corpus, and weights w_1, w_2 (e.g., 0.6, 0.4) reflect metric priorities. This optimization leverages a reflective loop: (1) identify failure points $F = \{f_1, \ldots, f_m\}$ from baseline runs, (2) evolve prompts locally via simulated variants $P' = \{p'_1, \ldots, p'_n\}$ scored by proxy metrics, (3) retrieve nearer documents closest to the new simulated prompt variants, and (4) distill global rules R into the Coordinator's prompt for proactive test-time adaptation. The goal is to outperform mRAG's self-training approach on multi-hop question-answering tasks by enhancing agent collaboration efficiency, formalized as $\arg\max_{P'} J(P')$. Our novel research question is: Can dynamic local refinement of agent prompts, driven by dataset-specific feedback, enable proactive adaptation in mRAG-like systems to reduce reactive errors in multi-hop reasoning?

2 Motivation

This problem matters because multi-agent RAG promises collaborative reasoning for knowledge-intensive tasks, yet rigid prompts cause reactive Coordinator decisions that propagate errors in applications like automated legal research (chaining precedents across cases) or customer support (synthesizing user history with product docs), where incomplete local agent roles (e.g., vague decomposition) lead to low-fidelity outputs; our focus on dynamic prompts and document retrieval motivates a shift to self-evolving systems that distill dataset insights into tailored instructions, fostering robust, low-compute MAS for real-world deployment.

mRAG's self-training approach is computationally intensive due to its reliance on extensive high-reward trajectory optimization, whereas we aim to achieve superior results with reduced compute using our lightweight reflective optimization strategy.

3 Related Work

Multi-agent retrieval-augmented generation (mRAG) frameworks have advanced collaborative retrieval and reasoning, with Salemi (2025) introducing mRAG as a foundational system with role-specialized agents (e.g., Planner for decomposition, Searcher for iterative retrieval) coordinated via a central hub, optimized through self-training on high-reward trajectories to enhance relevance without full gradient propagation [3].

ACM Conference 2025, New York, NY, USA

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/10.1145/XXXXXXX.XXXXXXX

Multi-Agent System Search (MASS) by Zhou (2025) automates MAS design by interleaving prompt and topology optimizations across stages—from local agent refinements to global alignment demonstrating how staged evolution boosts collaborative efficiency in multi-agent systems [6].

Phanse et al. (2025) introduce MSRS, a benchmark for evaluating multi-source RAG with datasets like MSRS-Story, testing retrieval from large corpora. It provides baselines for single-agent RAG, highlighting the need for multi-agent collaboration in sparse, multi-document settings. [2]

Chain of Agents by Zhang (2024) proposes sequential multi-agent collaboration for long-context tasks with worker-manager hierarchies to facilitate multi-hop reasoning across chunks, offering insights into prompt evolution for agent communication [4].

MetaAgent by Zhang (2025) automates multi-agent construction via finite state machines, generating role-specific prompts from task descriptions to enable adaptive workflows, relating to our dynamic evolution for mRAG's Coordinator [5].

MAPGD by Han (2025) introduces multi-agent prompt gradient descent for collaborative optimization, where agents backpropagate feedback to refine shared prompts, extending mRAG's self-training with gradient-free proxies for low-compute settings [1].

Our project builds primarily on mRAG's self-training scaffold, incorporating elements from MASS for exploratory local tuning and insights from MSRS, CoA, MetaAgent and MAPGD to enable prompt dynamism and improve the retrieval.

4 Proposed Approach

We propose **LRO-mRAG**, a framework designed to enhance multi-agent collaboration in resource-constrained environments. Our approach introduces dynamism through two key mechanisms: iterative prompt refinement via reflection and document retrieval based on simulated prompt variants. Figure **??** illustrates the high-level interaction between the reflective loop and the retrieval process.

4.1 Iterative Prompt Refinement

To address "reactive" errors common in static prompting—where agents misinterpret the optimal order of operations or fail to utilize tools effectively—we implemented a reflective optimization loop based on In-Context Learning (ICL). This method treats agent prompts not as fixed instructions, but as evolving contexts that improve through experience without requiring computationally expensive parameter updates.

- **Experience Summarization:** We utilize an external LLM agent, distinct from the coordination loop, to analyze the logs of previous "experiences" (execution trajectories). The model identifies specific failure modes (e.g., calling a validator before a searcher) and success patterns.
- **Advice Injection:** These insights are distilled into structured "Advice Blocks" that categorize strategies into "Typical Positive Result Paths" and "Typical Negative Result Paths."
- **Dynamic Context:** During inference, this advice is injected into the Coordinator's system prompt. For example, if previous runs failed due to premature validation, the advice explicitly warns the Coordinator to prioritize planning and search agents, effectively "refining" the prompt iteratively to steer the agent policy toward higher-reward trajectories.

4.2 Document Retrieval via Simulated Prompt Variants

Standard RAG systems often fail when a single user query does not capture the semantic density required for retrieval. To mitigate this, we developed a **Complex Searcher** agent capable of piecewise search and query simulation.

- **Simulated Query Generation:** Rather than relying on a single pass, the Searcher agent is authorized to simulate multiple variants of the search prompt. If an initial retrieval yields documents deemed irrelevant by the agent, the system triggers a re-ranking loop where the Searcher rephrases the original query to generate new, simulated search prompts that approach the information need from different semantic angles.
- **Piecewise Search:** For multi-hop questions, the Complex Searcher decomposes the global query into sub-questions. It then executes independent BM25 retrievals for these sub-components, simulating the prompts a human researcher might use for each distinct aspect of the problem.
- **Accumulation:** The documents retrieved from these simulated prompt variations are accumulated into a broader context window, increasing the likelihood that the answering agents receive the necessary evidence to construct a faithful response.

5 Experiments

5.1 Dataset

We utilized the Multi-Source Retrieval-Augmented Generation (MSRS) dataset developed by the Yale NLP Lab [2], specifically focusing on the *Story* corpus. This dataset is designed to evaluate systems on multi-hop reasoning tasks where answers must be synthesized from disparate narrative segments.

- **Statistics:** The corpus consists of 1,130 documents, where each document represents a segment of a larger narrative (e.g., chapters or scenes). The associated Question-Answer (QA) set contains 650 pairs.
- **Configuration:** To adhere to computational constraints while maintaining statistical significance, we utilized a randomized subset of 325 QA pairs (50% of the total training set) for our evaluation pipeline.
- **Justification:** We selected MSRS over larger benchmarks (e.g., HotpotQA) because it specifically targets the "multi-source" retrieval challenge essential to mRAG systems. It provides a dense, narrative-driven retrieval environment that rigorously tests an agent's ability to decompose complex queries, making it an ideal testbed for our proposed Complex Searcher and Reflective Optimization modules.

5.2 Evaluation

To assess the performance of LRO-mRAG, we employed a composite evaluation strategy focusing on both the retrieval accuracy and the semantic quality of the generated answers.

- **Faithfulness:** This metric measures the reliability of the retrieval process. It is calculated as a binary score (0 or 1) for each query. A generation is considered faithful if the document ID cited in the final response matches the Gold Document ID associated with the ground truth in the MSRS dataset. This ensures the system is not "hallucinating" the source of its information.
- **Correctness (Token Overlap):** Due to the absence of the proprietary "nugget-based" evaluator used in the original mRAG study, we utilized **Token F1 Score** as a robust proxy for correctness. High token overlap indicates that the model not only retrieved the correct document (Faithfulness) but also extracted and synthesized the specific information required to answer the user's query accurately.

6 Experimental Setup

6.1 Environment and Configuration

All experiments were conducted on Google Colab Pro instances equipped with a single NVIDIA A100 GPU (80GB VRAM). To accommodate the memory constraints of this environment while increasing model capability from our preliminary milestones, we utilized the **Qwen2.5-7B-Instruct** model.

The model was served using `vLLM` to enable OpenAI-compatible API calls for agent interactions. For retrieval, we utilized the BM25 Retriever indexing the MSRS Story corpus.

6.2 Hyperparameters and Preprocessing

We utilized a randomized subset of 325 Question-Answer pairs (50% of the MSRS Story training set) to balance statistical significance with the runtime limitations of the Colab environment. Key configurations include:

- **Max Attempts:** The Coordinator was limited to a maximum of 3 turn-taking steps to prevent infinite loops.
- **Search Verification:** For Experiment 2 (Piecewise Search), the `MAX_VERIFICATION_SAME_QUERY` parameter was set to allow up to 2 re-phrasing attempts per sub-question. **Document Chunking:** To optimize retrieval granularity for the sparse index, we preprocessed the MSRS corpus using a sliding window strategy with a chunk size of 512 words and an overlap of 80 words.
- **Inference Configuration:** The Qwen2.5-7B-Instruct model was served via vLLM using 4-bit GPTQ quantization. item **Generation Parameters:** Agent trajectories were sampled with a temperature of 0.7 to encourage diverse tool usage during the reasoning process.

6.3 Codebase and Implementation

Our implementation is built upon the open-source mRAG framework [3]. However, significant modifications were made to adapt it to low-resource settings. Specifically, we authored the code for:

(1) The **Complex Searcher** agent, including the logic for query decomposition and iterative accumulation.
(2) The **Reflection Agent** pipeline, including the logic for extracting trajectory logs, summarizing them into "Advice Blocks," and injecting them into the Coordinator's context window.

The complete code for our experiments, including the Complex Searcher and Reflective Optimization implementations, is hosted on Google Drive and can be accessed at: https://drive.google.com/drive/folders/1CyatMHyA9w-8PjYdAEVkE7mmDc8rpUcO

7 Baseline Models

To rigorously evaluate the contributions of our proposed LRO-mRAG framework, we establish two distinct baselines that isolate the effects of model capacity and agent autonomy.

7.1 Baseline 1: Rigid-Path mRAG (Efficiency)

This baseline represents the minimal viable configuration for consumer-grade hardware, established during our preliminary milestone.

- **Model: Qwen2.5-3B-Instruct** (4-bit quantized).
- **Policy:** A hard-coded, linear workflow: *Searcher* → *Answerer* → *Finisher*.
- **Constraint:** The Coordinator is stripped of decision-making agency; it strictly follows the pre-defined sequence regardless of retrieval quality.
- **Purpose:** This tests the lower bound of performance where both model parameter count and agent functional freedom are minimized.

7.2 Baseline 2: Standard Multi-Agent RAG (Performance)

To ensure our proposed methods (Piecewise Search and Reflective Optimization) provide value beyond simply scaling up the model, we utilize a stronger baseline representing a standard, unoptimized multi-agent setup.

- **Model: Qwen2.5-7B-Instruct** (4-bit quantized).
- **Policy:** "Functional Freedom." The Coordinator is provided with the full suite of tools—*Planner, Searcher, Reasoner, Validator, Summarizer, Answerer, Finisher*—and is free to invoke them in any order based on its internal reasoning.
- **Constraint:** While the agent has tool freedom, it operates "zero-shot" without the benefit of our proposed *Complex Searcher* decomposition or *Reflective Advice* injection.
- **Purpose:** This serves as the primary control, allowing us to measure whether performance gains in Experiments 2 and 3 are due to our architectural innovations rather than just the superior reasoning capabilities of the 7B parameter model.

8 Results

We evaluated four distinct configurations to assess the impact of model capacity, search granularity, and reflective optimization on retrieval performance. Table 1 summarizes the key metrics across these setups.

The results indicate a clear hierarchy of performance strategies. Scaling from the 3B parameter model (Baseline 1) to the 7B model with functional freedom (Baseline 2) yielded a substantial 17.6% improvement in Faithfulness (0.51 → 0.60). However, increasing search complexity (Exp 2) caused a regression below the 3B baseline. Conversely, the Reflective Optimization loop (Exp 3) outperformed

Table 1: Performance comparison on the MSRS Story Subset (325 pairs). The Standard 7B setup serves as the strong baseline. LRO-mRAG (Reflective) achieves the highest reliability.

Configuration	Faithfulness	Token F1
Baseline 1: Qwen-3B (Rigid Path)	0.51	0.92
Baseline 2: Qwen-7B (Func. Freedom)	0.60	0.94
Exp 2: Piecewise Search	0.49	0.89
Exp 3: Reflective Optimization	**0.66**	**0.95**

the strong 7B baseline by an additional 10%, achieving a peak Faithfulness of 0.66.

9 Analysis

9.1 Research Question Resolution

Our primary research question asked whether dynamic local refinement of prompts could enable proactive adaptation to reduce reasoning errors. The results from Experiment 3 affirmatively answer this. By injecting "Advice Blocks" derived from past trajectories, the **Reflective Optimization** method successfully steered the Coordinator agent away from suboptimal behaviors without requiring weight updates or denser retrieval models.

9.2 Benefit of Agent Autonomy (7B Baseline)

The improvement between Baseline 1 and Baseline 2 confirms that rigid workflows stifle the capabilities of competent LLMs. Logs revealed that the 7B Coordinator frequently invoked the *Reasoner* agent to clarify ambiguous terms before engaging the *Searcher*. This "planning latency"—which was impossible in the rigid 3B baseline—resulted in more precise initial queries, reducing the need for error-prone corrective loops.

9.3 Failure Case: The "Information Explosion"

Experiment 2 (Piecewise Search) represents a significant negative result. We hypothesized that decomposing questions would increase recall. While it likely did, it drastically reduced precision.

- **Noise Injection:** Decomposing a narrative question into sub-components (e.g., "What did Alice do?" + "Where did Bob go?") caused the BM25 retriever to flood the context window with tangentially related segments. The *Answerer* agent struggled to discern the "signal" amidst this noise, leading to hallucinations that lowered the Faithfulness score to 0.49.
- **Resource Ballooning:** Allowing iterative re-phrasing caused VRAM usage to spike, resulting in longer inference times (avg. 58s vs 36s) for worse outcomes. This finding suggests that for sparse retrieval (BM25), *context precision* is more valuable than *context volume*.

9.4 Success of Reflection

The success of Experiment 3 (Faithfulness 0.66) highlights the efficacy of *meta-cognition* over brute-force search. The "Advice Blocks" effectively acted as soft guardrails. For instance, advice explicitly warned against "calling the Validator before the Searcher"—a common hallucination loop in Baseline 2. By correcting these structural errors via prompt injection, the system allocated its limited context window more efficiently, verifying our hypothesis that lightweight, reflective optimizations can outperform complex architectural changes.

9.5 Limitations

Our study faces two primary limitations. First, we relied on BM25 (sparse retrieval); a dense embedding model might have better handled the nuanced sub-queries in Experiment 2. Second, building on the "Information Explosion" failure, we identify a structural deficit: the **Absence of Intermediate Filtering**. While our Complex Searcher increased document recall, the pipeline lacked a mechanism to verify relevance *before* ingestion. The retrieved segments were fed directly to the Answerer, forcing the LLM to filter noise "in-context," which degraded performance. A robust mRAG system requires a dedicated lightweight Re-Ranker (e.g., a Cross-Encoder) to sanitize the context window when utilizing aggressive query decomposition strategies.

10 Miscellaneous

10.1 Engineering Challenges & Optimization

A significant portion of our effort was dedicated to optimizing the inference pipeline for the Google Colab T4 environment. Running a 7B parameter model alongside a retrieval index and a multi-agent loop required aggressive memory management. We implemented custom garbage collection hooks in Python to clear VRAM between agent turns and fine-tuned the `vLLM` GPU utilization parameters (capped at 0.9) to prevent CUDA Out-Of-Memory (OOM) errors during the high-context windows of Experiment 2.

11 Member Contributions

- **Akash Yalla:** Architected the Coordinator's state machine and led the implementation of the Reflective Optimization (Exp 3), while also contributing to the prompt engineering strategy used in the Functional Freedom ablation (Exp 1).
- **Jack Kaefer:** Designed the Complex Searcher architecture and integrated the BM25 pipeline for Piecewise Search (Exp 2), while also fine-tuning the retrieval parameters used across the Baseline comparisons.
- **Khiem Le:** Engineered the MSRS data ingestion pipeline and implemented the automated Faithfulness and Token F1 metrics used to evaluate the outputs of all three experimental configurations.
- **Kirat Arora:** Optimized the vLLM quantization and memory management, enabling the high-context inference required to run the computationally intensive Piecewise Search (Exp 2) and Reflective Optimization (Exp 3) workloads.
- **Sanjiv Mohanraj:** Executed the Baseline and Functional Freedom (Exp 1) benchmarks and conducted the failure mode analysis that directly informed the design of the Reflective Advice logic.
- **Tung Ngo:** Orchestrated the architectural integration of the unified framework and coordinated the final execution of

the full experimental suite, synthesizing the comparative results between Exp 2 and Exp 3.

References

[1] Yichen Han. 2025. MAPGD: Multi-Agent Prompt Gradient Descent for Collaborative Prompt Optimization. *arXiv preprint arXiv:2509.11361* (2025). https://arxiv.org/abs/2509.11361

[2] Rohan Phanse, Yijie Zhou, Kejian Shi, Wencai Zhang, Yixin Liu, Yilun Zhao, and Arman Cohan. 2025. MSRS: Evaluating Multi-Source Retrieval-Augmented Generation. (2025). arXiv:2508.20867 [cs.CL] https://arxiv.org/abs/2508.20867

[3] Alireza Salemi. 2025. CIIR@LiveRAG 2025: Optimizing Multi-Agent Retrieval Augmented Generation through Self-Training. *arXiv preprint arXiv:2506.10844* (2025). https://arxiv.org/abs/2506.10844

[4] Yusen Zhang. 2024. Chain of Agents: Large Language Models Collaborating on Long-Context Tasks. *arXiv preprint arXiv:2406.02818* (2024). https://arxiv.org/abs/2406.02818

[5] Yaolun Zhang. 2025. MetaAgent: Automatically Constructing Multi-Agent Systems Based on Finite State Machines. *arXiv preprint arXiv:2507.22606* (2025). https://arxiv.org/abs/2507.22606

[6] Han Zhou. 2025. Multi-Agent Design: Optimizing Agents with Better Prompts and Topologies. *arXiv preprint arXiv:2502.02533* (2025). https://arxiv.org/abs/2502.02533