BEYOND MONOLITHIC ARCHITECTURES: MULTI-AGENT SEARCH AND KNOWLEDGE OPTIMIZATION FOR AGENTIC SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic search has emerged as a promising paradigm for complex information seeking by enabling Large Language Models (LLMs) to interleave reasoning with tool use. However, prevailing systems largely rely on monolithic agents, which suffer from structural bottlenecks such as unconstrained reasoning outputs that inflate trajectories, sparse outcome-level rewards that hinder credit assignment, and stochastic search noise that destabilize learning. To address these challenges, we propose **M-ASK** (Multi-Agent Search and Knowledge), a framework for *multi-agent joint optimization* in agentic search. M-ASK decomposes the search process into two complementary groups of trainable agents: Search Behavior Agents, which govern planning and search actions, and Knowledge Management Agents, which refine, compress, and maintain the evolving internal context. Crucially, this role decomposition is not introduced merely for modular orchestration; instead, M-ASK jointly optimizes these heterogeneous agents within a unified training framework, so that search behavior and knowledge management are learned as a coordinated multi-agent policy. To make such coordination effective, M-ASK employs turn-level rewards that provide fine-grained supervision for intermediate search decisions and knowledge updates. Experiments on the evaluated QA benchmarks show that M-ASK achieves the strongest average performance among the compared baselines and exhibits substantially improved training stability in our experiments.[1]

1 INTRODUCTION

The rapid evolution of Large Language Models (LLMs) has fundamentally reshaped information retrieval, driving a paradigm shift from passive keyword matching to *Agentic Search* Shi et al. (2025a). Unlike traditional retrieval systems, agentic search systems function as autonomous decision-makers capable of iterative planning, external tool querying, and information synthesis to address complex, multi-hop user needs. Recent systems such as Search-R1 Jin et al. (2025), R1-Searcher Song et al. (2025a), and DeepResearcher Zheng et al. (2025b) integrate reasoning directly into LLM-based multi-round search workflows. By executing multi-step search within a single response via end-to-end optimization, these approaches significantly enhance performance on complex QA tasks—capabilities that are difficult for static search paradigms to provide Ma et al. (2023); Ke et al. (2024).

However, training robust agents to navigate these dynamic search environments presents significant challenges. Prevailing approaches predominantly adopt a *monolithic* architecture that executes multi-round search within a single, continuous response. In this paradigm, the LLM shoulders the heavy burden of both trajectory planning and information processing throughout generation. We argue that this monolithic design is structurally vulnerable to three intertwined obstacles: (i) **unconstrained output length**, where agents generate verbose reasoning chains that extend search horizons without necessarily increasing information density; (ii) **sparse rewards**, where feedback is typically delayed until task completion, hindering effective credit assignment; and (iii) **search noise**, where external tools such as search engines introduce noise and irrelevant data into the context.

[1] An anonymized code package is available at `https://anonymous.4open.science/r/M-ASK-7713/`.

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

These challenges are coupled rather than isolated. Extended trajectories and sparse outcome-only feedback create a severe *multi-turn credit assignment* problem, while noisy retrieved context further obscures the learning signal. Consequently, monolithic agents Jin et al. (2025) frequently suffer from suboptimal trajectories and states, together with high training instability Deng et al. (2025).

To overcome the limitations of monolithic agentic search, we propose **M-ASK** (**M**ulti-**A**gent **S**earch and **K**nowledge), a framework that reformulates agentic search as a problem of **multi-agent joint optimization** rather than single-agent reasoning. Instead of relying on one monolithic policy to simultaneously plan trajectories, process noisy search results, and construct intermediate context, M-ASK decomposes the search process into two complementary groups of *jointly optimized* agents:

1. **Search Behavior Agents** (including the Planning, Search, and Answer Agents), which are responsible for trajectory planning, interacting with the search engine, and generating answers;
2. **Knowledge Management Agents** (including the Summary and Update Agents), which refine retrieved information, prune noisy observations, and maintain a concise internal knowledge state throughout the search process.

Crucially, this role decomposition is not introduced merely for modular orchestration at inference time. Instead, M-ASK treats these heterogeneous roles as a **cooperative multi-agent system** and **jointly optimizes** them within a unified training framework. To make such collaboration effective, M-ASK augments outcome rewards with **turn-specific dense rewards**, providing immediate, role-aligned supervision for intermediate search decisions and knowledge-state updates. This joint optimization enables Search Behavior Agents and Knowledge Management Agents to **co-adapt**: the latter actively stabilize the evolving state space, while the former learn to plan and search more accurately on top of the refined knowledge state. As a result, M-ASK mitigates the coupled impact of retrieval noise, multi-turn interactions, and sparse supervision in monolithic agentic search.

Our contributions are summarized as follows:

- We identify the compound impact of output verbosity, sparse rewards, and tool noise in monolithic agentic search, and analyze how these factors are associated with training instability (Section 4.3).
- We introduce M-ASK, a framework for **multi-agent joint optimization** in agentic search, which decouples search behavior from knowledge management and trains these roles collaboratively under dense, turn-specific rewards.
- Across seven QA benchmarks, M-ASK reaches an overall average F1 of **50.09**, improves HotpotQA by **+5.82** over the best baseline, and remains stable in all 10 runs at 1,000 steps (Search-R1 collapses in 9/10 runs).

2 Related Work

From Iterative RAG to Agentic Search. The paradigm of Retrieval-Augmented Generation (RAG) has evolved significantly from static "retrieve-then-read" pipelines Ma et al. (2023); Shi et al. (2023); Ke et al. (2024); Shi et al. (2024; 2025b) to dynamic systems. Early iterative approaches, such as IRCoT Trivedi et al. (2023), Self-RAG Asai et al. (2023), and EXSEARCH Shi et al. (2025c) (trained with generalized expectation-maximization, GEM), introduced feedback loops where LLMs actively decide when to retrieve. Recent advancements have formalized this into **Agentic Search**, where models plan multi-step trajectories to solve open-ended problems using external tools. Notably, emerging *Deep Research* agents, such as DeepResearcher Zheng et al. (2025b), Search-o1 Li et al. (2025b), Search-R1 Jin et al. (2025), and the R1-Searcher series Song et al. (2025a;b), integrate retrieval directly into the reasoning chains of LLMs to tackle multi-turn QA tasks.

Context Management and Optimization To overcome static context limitations, recent works explore dynamic optimization via uncertainty-based filtering Jimenez Gutierrez et al. (2024); Ji et al. (2025) or explicit memory agents Yu et al. (2025); Yan et al. (2025). More aggressively, DeepNote Wang et al. (2024) and MemSearcher Yuan et al. (2025) treat context as an evolving state. However, DeepNote does not jointly optimize search and management end-to-end, while MemSearcher's monolithic design can lead to coarse credit assignment when rewards are assigned only at the trajectory level. M-ASK addresses these pitfalls by *decoupling* search and knowledge

management into two specialized agents. Through multi-agent reinforcement learning with turn-specific dense rewards, our approach achieves precise credit assignment.

Multi-Agent Systems for Information Retrieval To overcome the limitations of monolithic agents, multi-agent systems (MAS) commonly decompose complex tasks into sub-tasks and assign them to agents with specialized roles. General frameworks, such as MetaGPT Hong et al. (2024) and AutoGen Wu et al. (2024), exemplify this paradigm by enabling structured role specialization and coordination among multiple agents. Within the information retrieval domain, MindSearch Chen et al. (2024) utilizes a graph-based planner alongside parallel web searchers for query decomposition, while MMOA-RAG Chen et al. (2025a) allocates specialized agents for query rewriting, document selection, and answer generation. Similarly, MAO-ARAG Chen et al. (2025b) optimizes a planner to balance retrieval effectiveness with efficiency, whereas JADE Chen et al. (2026) jointly optimizes planner and executor roles under a shared backbone. Despite their potential, many search-centric MAS emphasize prompt engineering or standard supervised fine-tuning (SFT), with less emphasis on optimizing *interaction dynamics*. Although MMOA-RAG incorporates MAPPO Yu et al. (2022), it optimizes a shared global reward, e.g., the F1 score of the final answer. We argue that such sparse global feedback is suboptimal for multi-turn collaboration, as it exacerbates the *credit assignment problem*, failing to accurately evaluate intermediate steps. M-ASK addresses this limitation by introducing distinct turn-level rewards for Search and Knowledge agents, ensuring that each role is optimized for its specific contribution to the collective goal.

3 METHOD

We propose **M-ASK**, a multi-agent framework for jointly optimizing search planning and knowledge management in agentic search. In this section, we first formulate the problem as a *Sequential Decentralized Partially Observable Markov Decision Process* Bernstein et al. (2002); Oliehoek et al. (2016). Second, we detail the specifications of the specialized agents. Finally, we present the framework's execution flow and the joint optimization process utilizing a turn-level reward mechanism.

Design rationale. The framework follows a direct failure-to-mechanism chain. Unconstrained generation is addressed by assigning short, function-specific actions to separate roles; search planning therefore does not have to retain and rewrite every retrieved document. Retrieval noise is addressed by making the Knowledge Management group an explicit state-transition module: the Summary Agent filters evidence, while the Update Agent either overwrites an unreliable trajectory item or appends a distinct logical hop. Sparse outcome feedback is addressed by evaluating the intermediate answer after each turn and sharing its marginal improvement with the Search, Summary, and Update agents. These choices yield a single coordination loop—search, filter, update, and evaluate—in which the state representation, agent roles, and reward granularity are aligned with the three failure modes identified in the Introduction.

3.1 PROBLEM FORMULATION

We formulate the multi-hop QA task as a *Sequential Decentralized Partially Observable Markov Decision Process*. Unlike standard formulations where agents act simultaneously, our framework, M-ASK, operates in a turn-based manner. Specifically, at each search turn t, only one designated agent, denoted as π_{active}, is activated. This agent receives a partial observation of the environment and executes an action to advance the search process. Although agents are activated sequentially, they are not optimized as isolated modules; rather, they form a cooperative multi-agent system whose roles are jointly learned to solve the overall search task. To enable such coordination across search turns, agents communicate indirectly by reading from and writing to a shared, structured knowledge state.

Structured Knowledge State Agents communicate via a shared, structured *Knowledge State* $\mathcal{K}$. We define $\mathcal{K}_t$ as the knowledge state maintained at the t-th search turn:

$$\mathcal{K}_t = \begin{Bmatrix} \text{“question”} : q, \\ \text{“thinking_trajectory”} : \mathcal{T}_t, \\ \text{“predicted_answer”} : a_t \end{Bmatrix} \tag{1}$$

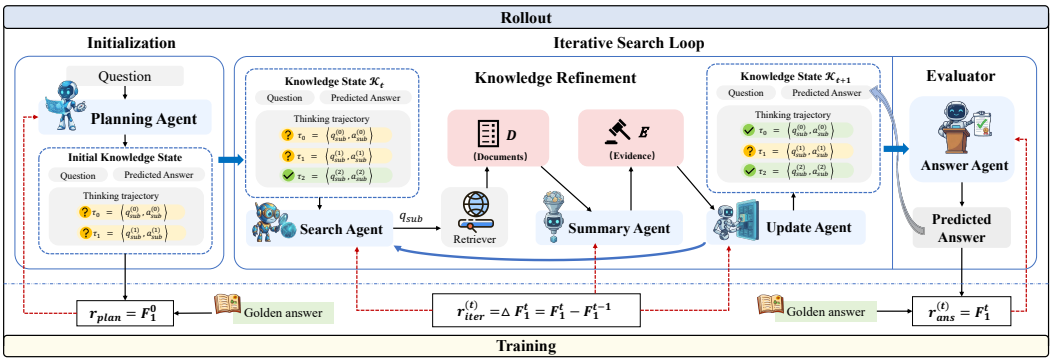


Figure 1: Overview of the M-ASK framework. (1) **Rollout**: The Planning Agent initializes the state $\mathcal{K}_0$, followed by an iterative loop where Search and Knowledge Management Agents refine the trajectory. Crucially, the Answer Agent updates the prediction after each turn. (2) **Training**: A hybrid reward mechanism assigns absolute scores (F_1^0 and F_1^t) to the Planning and Answer Agents, respectively, while the collaborative agents (Search, Summary, Update) share the marginal improvement (ΔF_1^t) to incentivize step-wise refinement.

Let $\mathcal{T}_t = [\tau_1, \ldots, \tau_m]$, where each $\tau_j = \langle q_{\text{sub}}^{(j)}, a_{\text{sub}}^{(j)} \rangle$ stores a sub-query and its evidence. The state is the collaboration interface: Planning initializes $(\mathcal{T}_0, a_0)$, while later turns refine $\mathcal{T}_t$ and update a_t. A concrete state example is given in Appendix H.

Figure 1 summarizes the initialization, iterative refinement, and joint optimization flow; complete pseudocode appears in Appendix I. M-ASK jointly optimizes all roles through the shared state and reward signals.

3.2 M-ASK Architecture

M-ASK employs a team of specialized agents, categorized by their functional roles. Together, these roles form a cooperative multi-agent system in which search behavior and knowledge management are jointly optimized through shared state transitions and turn-level supervision. We denote the set of agent roles as $\mathcal{A} = \{A_{\text{plan}}, A_{\text{search}}, A_{\text{sum}}, A_{\text{upd}}, A_{\text{ans}}\}$. For each agent role $i \in \mathcal{A}$, we use I_i to denote its role-specific instruction, and the corresponding role-conditioned policy is implemented by the shared LLM as $\pi_\theta(\cdot \mid \cdot, I_i)$.

3.2.1 Agent Roles and Inference Workflow

The Search Behavior Agents consist of a Planning Agent, which initializes $\mathcal{K}_0$, a Search Agent, which emits either a sub-query or `<end>`, and an Answer Agent, which synthesizes a_t from the current trajectory. The Knowledge Management Agents comprise a Summary Agent, which filters retrieved documents into evidence E, and an Update Agent, which applies either an in-place refinement or an append operation:

$$op, \mathcal{K}_{t+1} \leftarrow \pi_{\text{upd}}(\mathcal{K}_t, q'_{\text{sub}}, E).$$

Thus, each turn follows the compact sequence Search $\rightarrow$ Summary $\rightarrow$ Update, with the Answer Agent generating intermediate answers during training so that their F1 scores evaluate state changes. The loop stops when Search emits `<end>` or the turn budget is reached, after which the final answer is generated. Table 7 in Appendix G records the full role specifications; a worked correction example is given in Appendix J.

One-turn state transition. Given $\mathcal{K}_t$, the Search Agent emits a sub-query q'_{sub}. The Summary Agent converts retrieved documents into evidence E_t, and the Update Agent applies either `<Update>` τ_j `</Update>` to correct an existing step or `<Add>` τ_{new} `</Add>` to add a new hop, producing $\mathcal{K}_{t+1}$. The Answer Agent then generates a_{t+1}; its F1 score evaluates the resulting state change, and the shared signal $\Delta F_1^t = F_1(a_{t+1}, y) - F_1(a_t, y)$ assigns credit to the complete search–filter–update unit. This transition makes the causal link explicit: evidence filtering changes the state, the state changes the answer, and the answer change supervises the actions that produced it.

3.3 M-ASK TRAINING VIA TURN-LEVEL REWARDS

We employ PPO with parameter sharing Schulman et al. (2017). Although agents act in different roles and at different turns, they are trained as a cooperative multi-agent system with a shared policy and critic rather than as isolated modules. To align the SBA and KMA groups despite their different functional roles, we design a hybrid reward mechanism that differentiates between *state utilization* and *state refinement*.

State-Based Reward (Absolute F1) For agents responsible for generating solution outputs (A_{plan} and A_{ans}), the objective is to maximize the absolute quality of the current state.

$$r_{\text{plan}} = \mathrm{F}_1(a_0, y), \quad r_{\text{ans}}^{(t)} = \mathrm{F}_1(a_t, y) \tag{2}$$

Here, a_t is the answer synthesized from $\mathcal{K}_t$ and y is the ground truth answer. Note that during training, A_{ans} acts as an evaluator at every search turn t, providing an anchor signal for the joint optimization of the evolving search state.

Transition-Based Reward (Shared Incremental Gain) Crucially, the iterative phase requires tight collaboration between the Search Behavior Agent (A_{search}) and the Knowledge Management Agents ($A_{\text{sum}}, A_{\text{upd}}$). Although they belong to different functional groups, their actions are co-dependent: effective search requires precise state updates, and useful updates depend on accurate retrieval.

To enforce this **local cooperation**, we assign a **shared incremental reward** to all agents active in the loop. It is important to distinguish the execution frequency of the Answer Agent between phases. During **inference**, the Answer Agent is triggered only *once* after the search terminates to generate the final output. However, during **training**, it assumes an additional role as an *intermediate evaluator*. It is invoked at every turn t to synthesize a temporary answer a_t based on the current state $\mathcal{K}_t$. This mechanism allows us to utilize the answer score $r_{\text{ans}}^{(t)} = \mathrm{F}_1(a_t, y)$ (defined in Eq. 2) to measure the immediate answer quality. The iteration reward is then defined as the marginal improvement over the previous turn:

$$r_{\text{iter}}^{(t)} = r_{\text{ans}}^{(t)} - r_{\text{ans}}^{(t-1)} = \mathrm{F}_1(a_t, y) - \mathrm{F}_1(a_{t-1}, y) \tag{3}$$

By sharing the identical $r_{\text{iter}}^{(t)}$ signal, the Search Agent (A_{search}), Summary Agent (A_{sum}), and Update Agent (A_{upd}) are jointly incentivized to maximize the marginal information gain of each turn. This mechanism binds them into a cooperative sub-team, where the Search Agent learns to fetch necessary information and the Knowledge Agents learn to distill it efficiently. In this sense, the shared incremental reward does not merely provide denser supervision, but also defines a collaborative optimization unit over search, filtering, and state refinement. If A_{search} outputs `<end>`, it receives a reward of 0, ensuring the team only terminates the collaboration when further search yields no positive gain.

Optimization Objective (Parameter Sharing). To enhance sample efficiency and enable knowledge transfer across different reasoning phases, we employ a **parameter-sharing** strategy. All functional agents share a unified LLM policy π_θ and are distinguished by their role-specific instructions $\{I_i\}_{i \in \mathcal{A}}$.

As described in Algorithm 1, token-level experiences from all agents and all search turns are collected into a unified rollout buffer $\mathcal{B}$. Each sample is indexed by a search turn t, an agent role $i \in \mathcal{A}$, and a token position k within the trajectory generated by agent i at turn t. Let $u_{t,i,k}$ denote the action token generated by agent i at turn t under observation $o_{t,i,k}$. The shared policy is optimized by minimizing:

$$\mathcal{L}_{\text{actor}}(\theta) = -\hat{\mathbb{E}}_{(t,i,k)\sim\mathcal{B}}\left[\min\left(\rho_{t,i,k}\hat{A}_{t,i,k}, \text{clip}(\rho_{t,i,k}, 1-\epsilon, 1+\epsilon)\hat{A}_{t,i,k}\right)\right], \tag{4}$$

where $\rho_{t,i,k} = \frac{\pi_\theta(u_{t,i,k}|o_{t,i,k},I_i)}{\pi_{\theta_{\text{old}}}(u_{t,i,k}|o_{t,i,k},I_i)}$ is the PPO probability ratio. The shared critic V_ϕ is trained with the unified value loss:

$$\mathcal{L}_{\text{critic}}(\phi) = \hat{\mathbb{E}}_{(t,i,k)\sim\mathcal{B}}\left[\|V_\phi(o_{t,i,k}, I_i) - R_{t,i,k}\|^2\right], \tag{5}$$

where $R_{t,i,k}$ is the cumulative discounted return for agent i from token position k in search turn t, computed from the role-specific turn-level rewards.

4 EXPERIMENTS

To validate the efficacy of M-ASK, we conduct comprehensive experiments to answer the following research questions: **RQ1**: Does M-ASK outperform existing monolithic and multi-agent frameworks? **RQ2**: How does M-ASK compare to the monolithic baseline (Search-R1) in terms of training stability? **RQ3**: Is it beneficial to jointly model and optimize Knowledge Management and Search Behavior? **RQ4**: Do turn-specific dense rewards provide better credit assignment than global outcome-based rewards?

These questions mirror the proposed mechanism rather than treating the framework as a black-box architecture. RQ1 tests the end-to-end value of the coordinated loop, RQ2 tests whether role separation stabilizes optimization, RQ3 isolates the contribution of explicit state filtering and refinement, and RQ4 isolates the contribution of turn-level supervision. The main results, stability curves, and ablations are therefore read together: accuracy measures the usefulness of the final state, collapse and response-length analyses measure optimization behavior, and component ablations test whether each proposed component supports the mechanism.

Together, these results address the four research questions quantitatively. M-ASK reaches 50.09 average F1 and improves over the strongest baseline by 3.30 points overall, including a 5.82-point gain on HotpotQA. The stability study records 0/10 collapses for M-ASK versus 9/10 for Search-R1 at 1,000 steps. Removing Knowledge Management lowers the multi-hop average by 2.86 points, while replacing turn-level rewards with a global reward lowers it by 15.23 points. These measurements provide evidence for the respective contributions of role decoupling, knowledge-state filtering, and local credit assignment.

4.1 EXPERIMENTAL SETUP

Datasets We evaluate our framework on a diverse set of open-domain QA benchmarks categorized by reasoning complexity. For **Single-hop QA**, we use Natural Questions (NQ) Kwiatkowski et al. (2019), PopQA Mallen et al. (2023), and AmbigQA Min et al. (2020) to test factual retrieval accuracy. For **Multi-hop QA**, to evaluate complex reasoning and trajectory planning, we employ HotpotQA Yang et al. (2018), 2WikiMultiHopQA Ho et al. (2020), Musique Trivedi et al. (2022), and Bamboogle Press et al. (2022).

Baselines We evaluate M-ASK against two categories of methods: **Static Modular Search** and **Agentic Search**. Detailed implementations and settings for baselines are provided in Appendix D.

Implementation Details Our implementation is developed based on the official `verl` repository[2]. For the experiments, we use **Qwen2.5-Instruct** Team (2024) as the backbone LLM. We utilize the English Wikipedia as our retrieval corpus, indexed via **E5** Wang et al. (2022) for dense retrieval. We report the standard **F1 Score** as the evaluation metric.

4.2 MAIN RESULTS ANALYSIS (RQ1)

To answer **RQ1**, we evaluate M-ASK against competitive baselines across seven datasets. As shown in Table 1, M-ASK achieves the highest average F1 score (**50.09**), outperforming both monolithic and multi-agent frameworks. Appendix K reports retrieval-call, token, throughput, and training-cost measurements.

Dominance on Complex QA and Generalization. M-ASK achieves its strongest relative gain on the complex reasoning task **HotpotQA**, where it surpasses the best baseline by a substantial margin of **+5.82**. This supports the value of coordinated role optimization for the multi-hop QA tasks evaluated here. Furthermore, M-ASK also performs well on the additional evaluated multi-hop datasets (e.g., **2Wiki** and **Musique**), exceeding the compared agentic baselines on these datasets. These results indicate that the coordination strategy remains useful across the evaluated QA datasets, while broader open-web generalization remains future work. A notable exception is **Bamboogle**, where M-ASK trails MAO-ARAG, partly due to the small test set size (only 125 queries) and MAO-ARAG's use of

[2] `https://github.com/volcengine/verl`

Table 1: Main comparison of F1 scores on single-hop and multi-hop QA benchmarks with **Qwen2.5-7B-Instruct** as the backbone. We report group-wise and overall averages. The best results are **bolded** and the second-best results are underlined.

Method	Single-hop QA				Multi-hop QA					Avg
	NQ	PopQA	AmbigQA	Avg	HotpotQA	2Wiki	Musique	Bam.	Avg	All
Static Modular Search										
LLM w/o RAG	17.53	14.93	23.53	18.66	17.76	22.58	8.58	17.14	16.52	17.44
Vanilla RAG	40.60	42.74	56.20	46.51	28.33	25.91	25.20	25.28	26.18	34.89
RRR Ma et al. (2023)	54.60	**50.46**	65.41	56.82	46.21	41.52	18.27	36.59	35.65	44.72
BGM Ke et al. (2024)	54.21	49.51	65.97	56.56	46.85	37.79	17.55	37.38	34.89	44.18
MMOA-RAG Chen et al. (2025a)	55.44	50.21	68.02	57.89	49.21	41.66	17.26	37.20	36.33	45.57
Agentic Search										
Adaptive RAG Jeong et al. (2024)	36.52	35.59	45.32	39.14	42.38	39.62	25.48	34.85	35.58	37.11
MAO-ARAG Chen et al. (2025b)	36.82	41.85	47.03	41.90	46.65	43.96	22.38	**49.84**	40.85	41.30
DeepNote Wang et al. (2024)	54.03	49.80	67.57	57.13	52.49	36.22	22.17	45.22	39.03	46.79
Search-o1 Li et al. (2025b)	32.43	33.55	43.43	36.47	44.86	39.69	20.11	37.56	35.56	35.95
Search-R1 Jin et al. (2025)	52.57	46.98	65.25	54.93	46.87	39.03	17.97	38.69	35.64	43.91
M-ASK (Ours)	**57.40**	50.10	**68.33**	**58.61**	**58.31**	**46.12**	**26.20**	44.18	**43.70**	**50.09**

Table 2: Performance across Qwen2.5-Instruct model scales. M-ASK outperforms Search-R1 on NQ and HotpotQA in our measurements.

Dataset	Method	7B	3B	1.5B	0.5B
NQ	Search-R1	52.57	42.35	41.39	36.64
	M-ASK (Ours)	**57.40**	**56.94**	**56.21**	**48.41**
HotpotQA	Search-R1	46.87	40.96	39.22	19.32
	M-ASK (Ours)	**58.31**	**54.75**	**53.73**	**50.79**

frozen answer generators that may exploit parametric knowledge for factoid questions. Nevertheless, M-ASK remains competitive, trailing DeepNote by only $\sim$1.0 point while outperforming other baselines.

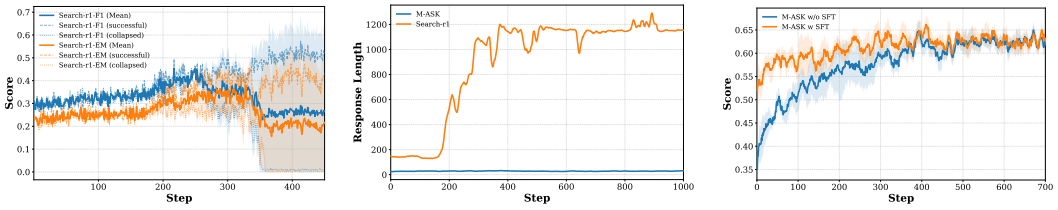
Comparison with Monolithic, Modular, and Recent Agentic Frameworks. Table 1 shows a clear hierarchy among framework designs: the monolithic Search-R1 underperforms M-ASK, while DeepNote improves through modular knowledge management but remains 3.30 points behind M-ASK. Additional agentic-search and process-reward comparisons are reported in Appendix E. Under the same in-domain protocol, M-ASK improves over R1-Searcher by 3.60 and 8.51 F1 points on NQ and HotpotQA, respectively, and over EviNote-RAG by 1.10 and 2.61 points.

Robustness Across Model Scales. Table 2 reports an additional scale analysis with Qwen2.5-Instruct backbones.

M-ASK outperforms Search-R1 at every evaluated scale on both NQ and HotpotQA. The advantage is already visible at 7B (+4.83 F1 on NQ and +11.44 on HotpotQA) and widens as the backbone becomes smaller, reaching +11.77 and +31.47 at 0.5B. At the smallest scale, M-ASK retains 50.79 F1 on HotpotQA, whereas Search-R1 drops to 19.32. Because both methods use the same backbone at each scale and are evaluated under the same setup, this pattern is consistent with the multi-agent design helping allocate limited model capacity across specialized search and knowledge-management roles, while turn-level rewards provide localized supervision; we interpret it as evidence of scale robustness rather than causal proof of any single component.

4.3 Training Stability and Convergence (RQ2)

To investigate the training dynamics and stability of our framework compared to monolithic approaches, we visualize the training curves on the HotpotQA dataset in Figure 2. Furthermore, to rigorously quantify robustness, we conducted 10 independent training runs for both methods and recorded the rate of "model collapse"—defined as the performance score dropping to near zero and failing to recover. These statistics are summarized in Table 3.



(a) Search-R1 training dynamics (b) Average response length (c) M-ASK training dynamics

Figure 2: Training curves on HotpotQA. In subfigures (a) and (c), solid lines denote the mean across multiple runs and shaded regions indicate variance. Search-R1 shows substantial instability and frequent mode collapse, while M-ASK converges much more stably. Subfigure (b) compares the average response length during training, showing that M-ASK remains concise whereas Search-R1 suffers from severe response bloating.

Table 3: Training stability analysis comparing the rate of model collapse (performance degrading to ≈ 0) at different training stages across 10 independent runs.

Method	**Collapse Rate (over 10 runs)**		
	@ 200 Steps	**@ 500 Steps**	**@ 1000 Steps**
Search-R1	1/10 (10%)	7/10 (70%)	9/10 (90%)
M-ASK (Ours)	**0/10 (0%)**	**0/10 (0%)**	**0/10 (0%)**

Catastrophic Collapse in Monolithic Agents. As shown in Table 3 and Figure 2(a), Search-R1 exhibits severe training instability, with the collapse rate increasing from 10% at 200 steps to **90%** at 1000 steps, consistent with Deng et al. (2025). We attribute this to the interaction of multi-turn interactions, sparse rewards, and retrieval noise: noisy evidence accumulates in the monolithic context, while outcome-only rewards provide weak credit assignment signals, often pushing the policy into degenerate states.

Analysis of Response Length and Mechanism. Figure 2(b) further shows that Search-R1 rapidly suffers from context bloating, whereas M-ASK maintains short and stable outputs. This reflects three stabilizing factors: role decoupling reduces generation horizons, turn-level rewards provide localized credit assignment, and Knowledge Management Agents filter retrieval noise before it accumulates. Together, these factors are consistent with M-ASK avoiding collapse and converging stably even without supervised warm-starting, as shown in Figure 2(c).

Joint Interpretation of Stability Signals. Collapse rate and response length provide complementary stability signals. M-ASK combines concise per-turn generations with higher final-task scores and zero collapses in 10 runs, making its short outputs more consistent with controlled state transitions than premature termination. In contrast, Search-R1's longer responses coincide with a rapidly rising collapse rate, suggesting that extra tokens are not reliably converted into useful evidence when search, reasoning, and context maintenance remain coupled. Together, these results support the role of coordinated decomposition and local supervision in stabilizing training.

4.4 Ablation Studies (RQ3 & RQ4)

To validate the contribution of individual components in M-ASK, we conduct ablation studies to observe the impact across different task complexities. The results are summarized in Table 4.

4.4.1 Impact of Collaborative Knowledge Management (RQ3)

To answer whether explicitly modeling knowledge management is beneficial, we evaluate the variant **w/o KMA**. In this setting, we remove the *Summary* and *Update* agents; the *Search Behavior Agent* interacts directly with the raw search engine output and appends full documents to the context. This setup simulates the unconstrained information flow of a standard monolithic interaction. However, unlike the fully monolithic *Search-R1*, this variant retains the underlying multi-agent paradigm and

Table 4: Ablation study. **Avg.** columns show the aggregated performance on Single-hop (3 datasets) and Multi-hop (4 datasets) benchmarks. Δ represents the performance drop (gray) or gain (blue) compared to the full model.

Method	Single-hop Datasets								Multi-hop Datasets									
	NQ		PopQA		AmbigQA		**Avg.**		HotpotQA		2Wiki		Musique		Bam.		**Avg.**	
	F1	Δ	F1	Δ	F1	Δ	**F1**	Δ	F1	Δ	F1	Δ	F1	Δ	F1	Δ	**F1**	Δ
M-ASK (Ours)	57.40	–	50.10	–	68.33	–	**58.61**	–	58.31	–	46.12	–	26.20	–	44.18	–	**43.70**	–
Ablation Studies																		
w/o KMA	54.93	-2.47	48.36	-1.74	64.79	-3.54	**56.03**	**-2.58**	55.06	-3.25	46.58	+0.46	19.39	-6.81	42.33	-1.85	**40.84**	**-2.86**
w/o T-L Reward	54.95	-2.45	47.12	-2.98	62.04	-6.29	**54.70**	**-3.91**	42.57	-15.74	33.27	-12.85	11.95	-14.25	26.10	-18.08	**28.47**	**-15.23**

independent optimization methods, thereby strictly isolating the impact of the missing knowledge management module.

Analysis. The results in Table 4 show a consistent performance degradation across both single-hop ($\Delta - 2.58\%$) and multi-hop ($\Delta - 2.86\%$) benchmarks. This is consistent with the interpretation that, without the active filtering provided by the KMA group, the search agent is more exposed to noise in the retrieval results, which can contribute to hallucinations or distracted reasoning. We note a slight anomaly on 2WikiMultiHopQA ($+0.46\%$), where the unfiltered model performs marginally better. However, this is outweighed by the severe drops in more complex datasets like Musique (-6.81%) and HotpotQA (-3.25%). Overall, explicit knowledge management is beneficial for stabilizing the reasoning trajectory against retrieval noise in this setting.

4.4.2 Efficacy of Turn-Specific Dense Rewards (RQ4)

To assess the necessity of our reward shaping mechanism, we evaluate **w/o T-L Reward**. In this variant, we disable the turn-level incremental rewards (Eq. 2 and Eq. 3) and instead optimize all agents using only the final outcome reward (Global F1) at the end of the episode, similar to MMOA-RAG Chen et al. (2025a).

Analysis. This ablation highlights the importance of fine-grained credit assignment in multi-turn search. On single-hop datasets, the performance drop is moderate (avg. $\Delta - 3.91\%$), as the short trajectories make final-answer supervision relatively sufficient. However, on multi-hop datasets, performance degrades substantially (avg. $\Delta - 15.23\%$), with HotpotQA dropping by more than 15 points. This contrast indicates that global outcome rewards are inadequate for multi-turn optimization, where different agents and turns contribute unequally to the final answer.

By assigning each turn an incremental ΔF1 reward, M-ASK provides denser, more localized supervision, substantially improving optimization in complex multi-hop settings. Together with the expanded comparison in Appendix E (Table 6), these results suggest that the gain comes from combining *turn-level rewards* with *explicit multi-agent role decomposition*, so each reward aligns with a meaningful collaborative decision unit.

5 Conclusion

In this paper, we proposed M-ASK, a multi-agent framework that addresses the structural instability of monolithic agentic search by aligning three elements: a structured knowledge state, specialized roles for search and state management, and turn-level supervision.

Across the evaluated seven benchmarks, M-ASK achieves the strongest average performance among the compared baselines, especially on complex multi-hop tasks, and shows improved training stability in our repeated-run study. The results and ablations support the central mechanism-level claim: role decomposition controls generation and retrieval noise, explicit state management preserves useful evidence, and shared marginal rewards provide local credit for the complete search–filter–update unit. Future work will extend this design to broader tool-use scenarios beyond information retrieval.

REFERENCES

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*, 2023.

Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4): 819–840, 2002.

Yiqun Chen, Hangyu Mao, Jiaxin Mao, Shiguang Wu, Tianle Zhang, Bin Zhang, Wei Yang, and Hongxing Chang. Ptde: Personalized training with distilled execution for multi-agent reinforcement learning. *arXiv preprint arXiv:2210.08872*, 2022.

Yiqun Chen, Lingyong Yan, Weiwei Sun, Xinyu Ma, Yi Zhang, Shuaiqiang Wang, Dawei Yin, Yiming Yang, and Jiaxin Mao. Improving retrieval-augmented generation through multi-agent reinforcement learning. *arXiv preprint arXiv:2501.15228*, 2025a.

Yiqun Chen, Erhan Zhang, Lingyong Yan, Shuaiqiang Wang, Jizhou Huang, Dawei Yin, and Jiaxin Mao. Mao-arag: Multi-agent orchestration for adaptive retrieval-augmented generation. *arXiv preprint arXiv:2508.01005*, 2025b.

Yiqun Chen, Erhan Zhang, Tianyi Hu, Shijie Wang, Zixuan Yang, Meizhi Zhong, Xiaochi Wei, Yan Gao, Yi Wu, Yao Hu, et al. Jade: Bridging the strategic-operational gap in dynamic agentic rag. *arXiv preprint arXiv:2601.21916*, 2026.

Zehui Chen, Kuikun Liu, Qiuchen Wang, Jiangning Liu, Wenwei Zhang, Kai Chen, and Feng Zhao. Mindsearch: Mimicking human minds elicits deep ai searcher. *arXiv preprint arXiv:2407.20183*, 2024.

Yuqin Dai, Guoqing Wang, Yuan Wang, Kairan Dou, Kaichen Zhou, Zhanwei Zhang, Shuo Yang, Fei Tang, Jun Yin, Pengyu Zeng, et al. Evinote-rag: Enhancing rag models via answer-supportive evidence notes. *arXiv preprint arXiv:2509.00877*, 2025.

Wenlong Deng, Yushu Li, Boying Gong, Yi Ren, Christos Thrampoulidis, and Xiaoxiao Li. On grpo collapse in search-r1: The lazy likelihood-displacement death spiral. *arXiv preprint arXiv:2512.04220*, 2025.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*, 2020.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*, 2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. *arXiv preprint arXiv:2403.14403*, 2024.

Yuelyu Ji, Rui Meng, Zhuochun Li, and Daqing He. Memory-aware and uncertainty-guided retrieval for multi-hop question answering. *arXiv preprint arXiv:2503.23095*, 2025.

Bernal Jimenez Gutierrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in Neural Information Processing Systems*, 37:59532–59569, 2024.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Zixuan Ke, Weize Kong, Cheng Li, Mingyang Zhang, Qiaozhu Mei, and Michael Bendersky. Bridging the preference gap between retrievers and llms. *arXiv preprint arXiv:2401.06954*, 2024.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019.

Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baixuan Li, Zhengwei Tao, Xinyu Wang, Weizhou Shen, Junkai Zhang, Dingchu Zhang, Xixi Wu, Yong Jiang, Ming Yan, Pengjun Xie, Fei Huang, and Jingren Zhou. Websailor: Navigating super-human reasoning for web agent. *arXiv preprint arXiv:2507.02592*, 2025a.

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

Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

Rui Lu, Zhenyu Hou, Zihan Wang, Hanchen Zhang, Xiao Liu, Yujiang Li, Shi Feng, Jie Tang, and Yuxiao Dong. Deepdive: Advancing deep search agents with knowledge graphs and multi-turn rl. *arXiv preprint arXiv:2509.10446*, 2025.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. Query rewriting for retrieval-augmented large language models. *arXiv preprint arXiv:2305.14283*, 2023.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 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)*, pp. 9802–9822. Association for Computational Linguistics, 2023. URL https://aclanthology.org/2023.acl-long.546/.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. Ambigqa: Answering ambiguous open-domain questions. *arXiv preprint arXiv:2004.10645*, 2020.

Frans A Oliehoek, Christopher Amato, et al. *A concise introduction to decentralized POMDPs*, volume 1. Springer, 2016.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*, 2022.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51, 2020.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhengliang Shi, Shen Gao, Zhen Zhang, Xiuying Chen, Zhumin Chen, Pengjie Ren, and Zhaochun Ren. Towards a unified framework for reference retrieval and related work generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5785–5799, 2023.

Zhengliang Shi, Shuo Zhang, Weiwei Sun, Shen Gao, Pengjie Ren, Zhumin Chen, and Zhaochun Ren. Generate-then-ground in retrieval-augmented generation for multi-hop question answering. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7339–7353, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.397. URL https://aclanthology.org/2024.acl-long.397/.

Zhengliang Shi, Yiqun Chen, Haitao Li, Weiwei Sun, Shiyu Ni, Yougang Lyu, Run-Ze Fan, Bowen Jin, Yixuan Weng, Minjun Zhu, Qiujie Xie, Xinyu Guo, Qu Yang, Jiayi Wu, Jujia Zhao, Xiaqiang Tang, Xinbei Ma, Cunxiang Wang, Jiaxin Mao, Qingyao Ai, Jen-Tse Huang, Wenxuan Wang, Yue Zhang, Yiming Yang, Zhaopeng Tu, and Zhaochun Ren. Deep research: A systematic survey, 2025a. URL https://arxiv.org/abs/2512.02038.

Zhengliang Shi, Lingyong Yan, Weiwei Sun, Yue Feng, Pengjie Ren, Xinyu Ma, Shuaiqiang Wang, Dawei Yin, Maarten de Rijke, and Zhaochun Ren. Direct retrieval-augmented optimization: Synergizing knowledge selection and language models. *arXiv preprint arXiv:2505.03075*, 2025b.

Zhengliang Shi, Lingyong Yan, Dawei Yin, Suzan Verberne, Maarten de Rijke, and Zhaochun Ren. Iterative self-incentivization empowers large language models as agentic searchers. *arXiv preprint arXiv:2505.20128*, 2025c.

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

Huatong Song, Jinhao Jiang, Wenqing Tian, Zhipeng Chen, Yuhuan Wu, Jiahao Zhao, Yingqian Min, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-searcher++: Incentivizing the dynamic knowledge acquisition of llms via reinforcement learning. *arXiv preprint arXiv:2505.17005*, 2025b.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

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

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 10014–10037, 2023.

Yi Wan, Jiuqi Wang, Liam Li, Jinsong Liu, Ruihao Zhu, and Zheqing Zhu. Rethinking the design of reinforcement learning-based deep research agents. *arXiv preprint arXiv:2510.15862*, 2026.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

Ruobing Wang, Qingfei Zhao, Yukun Yan, Daren Zha, Yuxuan Chen, Shi Yu, Zhenghao Liu, Yixuan Wang, Shuo Wang, Xu Han, et al. Deepnote: Note-centric deep retrieval-augmented generation. *arXiv preprint arXiv:2410.08821*, 2024.

Jialong Wu, Baixuan Li, Runnan Fang, Wenbiao Yin, Zhenglin Wang, Zhengwei Tao, Ding-Chu Zhang, Zekun Xi, Robert Tang, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. Webdancer: Towards autonomous information seeking agency. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversation. In *Conference on Language Modeling (COLM)*, 2024.

Yutao Xie, Nathaniel Thomas, Nicklas Hansen, Yang Fu, Li Erran Li, and Xiaolong Wang. Tips: Turn-level information-potential reward shaping for search-augmented llms. In *The Fourteenth International Conference on Learning Representations*, 2026.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Hinrich Schütze, Volker Tresp, and Yunpu Ma. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *arXiv preprint arXiv:2508.19828*, 2025.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624, 2022.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, et al. Memagent: Reshaping long-context llm with multi-conv rl-based memory agent. *arXiv preprint arXiv:2507.02259*, 2025.

Qianhao Yuan, Jie Lou, Zichao Li, Jiawei Chen, Yaojie Lu, Hongyu Lin, Le Sun, Debing Zhang, and Xianpei Han. Memsearcher: Training llms to reason, search and manage memory via end-to-end reinforcement learning. *arXiv preprint arXiv:2511.02805*, 2025.

Wenlin Zhang, Xiangyang Li, Kuicai Dong, Yichao Wang, Pengyue Jia, Xiaopeng Li, Yingyi Zhang, Derong Xu, Zhaocheng Du, Huifeng Guo, et al. Process vs. outcome reward: Which is better for agentic rag reinforcement learning. *arXiv preprint arXiv:2505.14069*, 2025.

Xuhui Zheng, Kang An, Ziliang Wang, Yuhang Wang, and Yichao Wu. Stepsearch: Igniting llms search ability via step-wise proximal policy optimization. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 21805–21830, 2025a.

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

Yuhang Zhou, Kai Zheng, Qiguang Chen, Mengkang Hu, Qingfeng Sun, Can Xu, and Jingjing Chen. Offseeker: Online reinforcement learning is not all you need for deep research agents. *arXiv preprint arXiv:2601.18467*, 2026.

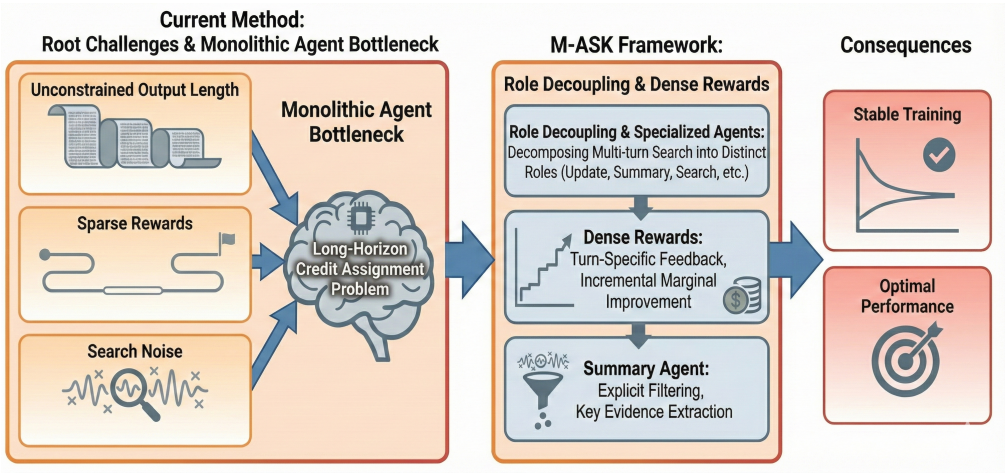


Figure 3: Challenges of current monolithic methods and the M-ASK solution. Existing agents face delayed credit assignment across multi-turn trajectories because output length is unconstrained, rewards are sparse, and search results are noisy. M-ASK addresses these bottlenecks through role decoupling and turn-level dense rewards.

A Limitations

Our evaluation focuses on retrieval-augmented QA over an English Wikipedia corpus, with answer-level F1 providing a verifiable training signal. This scope does not establish performance on open-ended web research, interactive browsing, or domains without gold answers. The general turn-level reward is $r_t = S(\mathcal{K}_t) - S(\mathcal{K}_{t-1})$; without reference answers, S would need a rubric, reward model, or task-specific evaluator, whose reliability remains open. M-ASK also requires coordinated rollout and state passing across role-conditioned calls, increasing engineering complexity even though measured costs are comparable to or lower than the monolithic baseline. Finally, our ablations isolate knowledge management and reward design; a complete factorial study of every role remains future work.

AI use statement

Generative AI tools were used primarily to revise and polish the manuscript's language, including grammar, clarity, and style. All AI-assisted changes were manually reviewed by the authors.

Reproducibility statement

The main text specifies the task, agent roles, reward equations, datasets, backbone, retrieval setup, and metric. Appendix D gives baseline settings, Appendix F records role prompts, Appendix I gives the joint-training procedure, and Appendix K reports compute and serving measurements. An anonymized code package is available at `https://anonymous.4open.science/r/M-ASK-7713`.

B Challenges of Monolithic Agentic Search

C Rationale for Parameter Sharing Strategy

In the M-ASK framework, we employ a parameter-sharing strategy where a single Large Language Model (LLM) backbone π_θ serves as the underlying policy for all functional agents (i.e., $A_{\text{plan}}, A_{\text{search}}, A_{\text{sum}}, A_{\text{upd}}, A_{\text{ans}}$), distinguished solely by role-specific system instructions. We adopt this design based on three critical considerations:

1. Theoretical Foundation in MARL Parameter sharing is a well-established and effective paradigm in Multi-Agent Reinforcement Learning (MARL) Rashid et al. (2020); Lowe et al. (2017);

Yu et al. (2022); Chen et al. (2022). In cooperative settings, sharing parameters allows agents to learn a unified representation of the state space, which often leads to faster convergence and improved training stability compared to maintaining independent policies. By optimizing a single set of parameters on the aggregated experiences of all roles, the model can efficiently generalize across the diverse phases of the search and reasoning process.

2. Computational and Storage Efficiency Unlike traditional RL agents based on small Multi-Layer Perceptrons (MLPs), agents in our framework are initialized with LLMs containing billions of parameters. Maintaining independent policy networks for five distinct agents would result in a linear increase in memory consumption ($\mathcal{O}(N)$), rendering the training process computationally prohibitive and difficult to deploy. Parameter sharing reduces the storage requirement to $\mathcal{O}(1)$, significantly lowering the barrier for training and inference. This efficiency is paramount for complex RAG systems like M-ASK, allowing us to allocate resources toward longer context windows rather than redundant model weights.

3. Inherent Multi-Task Capability of LLMs LLMs inherently possess strong multi-task capabilities, enabling them to perform distinct tasks based on contextual instructions (prompts) without modifying their internal weights. In M-ASK, different agents (e.g., the *Search Agent* deciding on queries vs. the *Summary Agent* extracting evidence) share fundamental reasoning competencies, such as reading comprehension and logical deduction. Parameter sharing leverages this synergy: skills learned in the *summarization* task can implicitly enhance the *answer generation* capability via shared representations. By conditioning the shared π_θ on role-specific prompts I_{role}, we effectively project the model's general capabilities into specific functional subspaces, achieving role specialization without architectural redundancy.

4. Empirical Comparison with Alternative Specialization Strategies Parameter sharing in M-ASK is primarily adopted as a practical design choice for efficient training and serving, rather than as a strict prerequisite for effective role specialization. The core objective of our framework is not merely to assign different functional roles to different agents, but to optimize these roles *jointly* within a coordinated multi-agent search process. Under this view, the main source of improvement should come from collaborative optimization over complementary roles, rather than from a particular parameter partitioning strategy.

To examine whether stronger structural specialization is necessary, we compare three representative variants with increasing degrees of parameter isolation across roles: (1) our default **full-parameter shared** strategy, where all roles are implemented by a single trainable LLM; (2) a **separate LoRA adapters** variant, which preserves a shared backbone while assigning each role an independent lightweight adaptation module Hu et al. (2022); and (3) a **fully separate** variant, in which each agent is assigned its own independently updated model parameters. Beyond effectiveness, these variants also differ in *relative serving complexity*, i.e., the number of parameter sets that must be loaded, switched, or maintained at inference time. We report this cost qualitatively as **Low/Medium/High**: **Low** corresponds to serving a single shared model for all roles; **Medium** corresponds to a shared backbone with multiple role-specific adapters, which introduces additional adapter management and switching overhead; and **High** corresponds to maintaining fully separate parameters for all agent roles, resulting in substantially higher memory and deployment overhead.

Table 5: Comparison of different role specialization strategies in M-ASK. All variants achieve highly similar performance, while the fully shared strategy yields the lowest relative serving complexity. This indicates that the effectiveness of M-ASK does not hinge on a particular parameter-sharing strategy.

Method	Role-Specific Parameters	Serving Complexity	NQ	HotpotQA
Full-Parameter + Shared (Ours)	No	Low	57.40	58.31
Separate LoRA Adapters	Yes	Medium	57.22	58.19
Fully Separate Parameters	Yes	High	57.44	58.40

As shown in Table 5, all three strategies achieve highly similar performance on both NQ and HotpotQA. The separate LoRA variant does not provide noticeable gains over the default shared

design, while the fully separate variant yields only marginal improvements ($+0.04$ on NQ and $+0.09$ on HotpotQA). These small differences suggest that explicit parameter isolation is not a major factor behind the gains of M-ASK.

More importantly, the effectiveness differences should be interpreted together with serving complexity. The fully shared strategy requires only a single model to be served across all roles, making it the simplest and most efficient option in deployment. In contrast, the LoRA-based variant introduces additional complexity from managing multiple role-specific adapters, and the fully separate strategy further increases deployment burden by requiring distinct model parameters for every agent role. Therefore, although the fully separate variant is marginally stronger, the gain is too small to justify its substantially higher serving complexity.

Overall, these results support two conclusions. First, the effectiveness of M-ASK is robust to different specialization strategies: neither stronger parameter isolation nor shared parameterization fundamentally changes the overall outcome. This indicates that the main benefit of M-ASK comes from its *role-decoupled workflow design and joint optimization objective*, rather than from a particular parameter-sharing scheme. Second, because the fully shared strategy achieves nearly identical performance while incurring the lowest serving complexity, it provides the best practical trade-off between effectiveness and deployment efficiency.

D Implementation Details of Baselines

We compare M-ASK against two categories of methods:

- **Static Modular Search:** *LLM w/o RAG* (closed-book), *Vanilla RAG* (standard retrieve-then-generate), *RRR* Ma et al. (2023) (RL-based query reformulation), and *BGM* Ke et al. (2024) (RL-based document selection), and *MMOA-RAG* Chen et al. (2025a) (multi-agent RL).
- **Agentic Search:** *Adaptive RAG* Jeong et al. (2024) (adaptive workflow), *MAO-ARAG* Chen et al. (2025b) (planner-executors optimization), *DeepNote* Wang et al. (2024) (knowledge management) and *Search-R1* Jin et al. (2025) (monolithic agent optimized via RL reasoning training).

To ensure a fair comparison, all baselines and our M-ASK method are unified under the same experimental setting. We utilize `Qwen2.5-7B-Instruct` as the backbone model. Specifically, for components within the baselines that do not require training, we employ the pre-trained version of `Qwen2.5-7B-Instruct`. For components requiring training, we fine-tune them based on the `Qwen2.5-7B-Instruct` initialization. Detailed implementation notes for each baseline are provided below:

Standard Baselines

- **LLM w/o RAG:** This represents the closed-book setting where the LLM directly generates answers based on its internal parametric knowledge without accessing external corpora.
- **Vanilla RAG:** A standard retrieve-then-generate pipeline. It retrieves the top-5 relevant documents based on the query and concatenates them with the prompt to generate the answer using the pre-trained LLM.
- **RRR** Ma et al. (2023): This method employs the PPO algorithm to end-to-end train a query rewriter module. In our reproduction, to prevent the performance bottleneck potentially caused by a frozen generator, we also fine-tuned the answer generation module, ensuring it is well-trained alongside the rewriter.
- **BGM** Ke et al. (2024): BGM uses PPO to train a document selection module. Although the original implementation may utilize a frozen generator, we fine-tuned the answer generation module in our reproduction to align with the robust setting of RRR and maximize overall performance.
- **MMOA-RAG** Chen et al. (2025a): This framework utilizes Multi-Agent PPO (MAPPO) Yu et al. (2022) to train three distinct agents—a query rewriter, a document selector, and an answer generator—guided by a shared final reward mechanism. We adopted

Table 6: Expanded comparison on NQ and HotpotQA. M-ASK is compared with agentic-search and process-reward baselines under the same in-domain setting.

Method	NQ	HotpotQA
Agentic Search		
Search-o1 Li et al. (2025b)	32.43	44.86
R1-Searcher Song et al. (2025a;b)	53.80	49.80
Search-R1 Jin et al. (2025)	52.57	46.87
Process Reward		
EviNote-RAG Dai et al. (2025)	56.30	55.70
StepSearch Zheng et al. (2025a)	39.79	50.02
ReasonRAG Zhang et al. (2025)	39.65	41.53
TIPS Xie et al. (2026)	40.38	45.98
M-ASK (Ours)	**57.40**	**58.31**

`Qwen2.5-7B-Instruct` as the backbone model for this multi-agent framework, strictly following the original logic while maintaining consistency with our unified setting.

Agentic Search (Adaptive Workflow)

- **Adaptive RAG** Jeong et al. (2024): This method trains a classifier to dynamically determine which of three distinct retrieval workflows (Directly Answer, Simple RAG, or Iterative RAG) should handle a given query. In our implementation, we fine-tuned `Qwen2.5-7B-Instruct` to serve as this classifier, while utilizing the pre-trained version of `Qwen2.5-7B-Instruct` for executing the subsequent static workflows.
- **MAO-ARAG** Chen et al. (2025b): MAO-ARAG features a hierarchical multi-agent architecture composed of a planner and multiple executors, sharing a modular structure similar to our M-ASK. The original work utilizes PPO with a shared final reward to train specifically the planner agent. Consistent with our unified setting, the planner module is a fine-tuned `Qwen2.5-7B-Instruct`, while the executor modules employ the pre-trained version of the same model.
- **DeepNote** Wang et al. (2024): DeepNote is a modular, note-centric framework that decomposes complex QA into iterative retrieval, note refinement, and answer generation, where evolving notes explicitly track accumulated knowledge and guide subsequent retrieval decisions; it is further optimized with Direct Preference Optimization (DPO) Rafailov et al. (2023) to improve both intermediate note construction and final response quality.
- **Search-R1** Jin et al. (2025): Unlike modular designs, Search-R1 adopts a monolithic architecture where multi-turn search and reasoning processes occur within a single response generation. It is optimized via RL using outcome-based rewards. For this baseline, we obtained the experimental results using the code provided in the author's official open-source repository[3].
- **Search-o1** Li et al. (2025b): Search-o1 is an agentic search-enhanced reasoning framework that dynamically integrates external retrieval into the reasoning process and introduces a Reason-in-Documents module to refine retrieved evidence before injecting it into the reasoning chain.

E Expanded Baseline Comparison

To complement the main seven-dataset comparison, we report expanded comparisons with recent agentic-search and process-reward methods under the same in-domain training and evaluation protocol.

[3] `https://github.com/PeterGriffinJin/Search-R1`

M-ASK achieves the best score on both datasets, exceeding R1-Searcher by +3.60 F1 on NQ and +8.51 on HotpotQA, and EviNote-RAG by +1.10 and +2.61, respectively. The larger HotpotQA margin is consistent with the role of explicit state management in multi-hop search. The result also suggests that process rewards are most effective when their granularity is aligned with the structured Search–Summary–Update transition, rather than attached to an undifferentiated trajectory.

F Prompt for Different Agents

F.1 Prompt for Planning Agent

The following is the full system prompt used for decomposing queries. The `[EXAMPLE_PROMPT]` placeholder represents the few-shot examples injected at runtime.

System Prompt

```
You are a knowledgeable reasoning assistant.  Your task is to
decompose a given question into multiple sub-questions based purely
on your own knowledge (do not use external tools), and answer each
sub-question based on your own knowledge.  If you do not know the
answer, write 'unknown' exactly.

Output all results strictly in the following format for easy
parsing:  <q1>sub-question text</q1>
<a1>answer text</a1>
<q2>sub-question text</q2>
<a2>answer text</a2>

Continue numbering sequentially until done.  At the end, output
your final conclusion answer to the original question in the tag
<predicted_answer>...</predicted_answer>, and inside this tag
follow the style of:  [EXAMPLE_PROMPT]
```

User Input

```
Query:  {query}
```

F.2 Prompt for Search Agent

System Prompt

```
You are a search query generation assistant.
Your input contains:
- The original question
- A thinking_trajectory (list of sub-questions and their
sub-answers)
- A search_history (list of past search queries and their summaries)

Note:
- The search_history represents queries YOU have already searched in
previous turns.
- You MUST avoid generating queries that are identical to or
semantically similar to any query in search_history.
- Only generate a new query if it provides new, distinct information
that has not yet been searched.

Your task:
1.  If any sub-question in thinking_trajectory has an answer
'unknown' or unclear, generate a specific, searchable query to find
```

```
the missing information.
2. If you believe additional sub-questions are needed to answer
the original question, generate a specific searchable query for
that.
3. If no further search is needed, output <end>.

Output only one of the following two formats:
<search>Your query here</search>
<end>

Do not include any extra text, explanations, or other tags.
```

User Input

```
Original question: {question}

Thinking trajectory:
{trajectory_text}

Search history:
{history_text}

Output only one of the following two formats:
<search>Your query here</search>
<end>

Do not include any extra text, explanations, or other tags.
```

F.3 Prompt for Summary Agent

System Prompt

```
You are an evidence extraction assistant.
You will be given:
- A search query
- Several candidate documents related to that query

Your task:
1. From the provided documents, extract the key evidence that
directly answers the query.
2. If no key evidence exists in the documents, output 'No useful
information'.

Output exactly in one of the two formats:
<evidence>your key evidence here</evidence>
<evidence>No useful information</evidence>

Do not include explanations, reasoning steps, or any text outside
the <evidence> tags.
```

User Input

```
Query: {query}
Candidate Documents: {docs_text}
```

F.4 Prompt for Update Agent

System Prompt

```
You are an update detection assistant.
You are given:
- An original question and its thinking_trajectory (sub-questions
and their answers)
- The predicted_answer derived from thinking_trajectory
- A new question and its evidence

Your task:
1. If the new question and evidence can replace one existing
sub-question-answer pair in thinking_trajectory, output
<Update>ti</Update> where ti is the key.
2. If the new question and evidence contain information not
present in thinking_trajectory, output <Add>t(n+1)</Add> where n
is current number of t's.
3. Output strictly one of these formats and no other text.
```

User Input

```
Original question: {orig_question}
Thinking trajectory: {trajectory_text}
New query: {new_query}
Evidence: {new_evidence}
```

F.5 Prompt for Answer Agent

System Prompt

```
You are a final answer generation assistant.
You are given:
- An original question
- A thinking_trajectory (list of sub-questions and their answers)

Your task:
1. Use the provided thinking_trajectory's answers to determine the
final answer to the original question.
2. You must only output in the following strict format as shown in
[EXAMPLE_PROMPT]

IMPORTANT: Output must be:
<predicted_answer>your answer here</predicted_answer>
No extra text, no reasoning.
```

User Input

```
Question: {question}
Thinking trajectory: {trajectory_text}
```

G Detailed Agent Specifications

In this section, we provide comprehensive specifications for the multi-agent architecture employed in M-ASK. Due to space constraints in the main text, we present the granular details of the Search Behavior Agents (SBA) and Knowledge Management Agents (KMA) here. Table 7 formally defines the specific roles, input-output interfaces, and action spaces for each agent, elucidating their operational logic within the iterative retrieval process.

Table 7: Comprehensive specifications of the agents within the M-ASK framework, detailing their roles, functional mechanisms, and input/output formalizations.

Module	Agent	Role	Description & Action Space	Input / Output Formulation
SBA	**Planning**	*Initializer*	Leverages parametric memory to generate an initial reasoning trajectory $\mathcal{T}_0$ and a preliminary answer a_0, encapsulating them into the initial state.	**In:** Query q **Out:** $\mathcal{K}_0 \leftarrow \pi_{\text{plan}}(q)$
	Search	*Navigator*	Evaluates information sufficiency to decide between exploration and termination. 1) Generate sub-query q'_{sub} to expand knowledge. 2) Output `<end>` to terminate the loop.	**In:** Query q, Trajectory $\mathcal{T}_t$ **Out:** $Act \leftarrow \pi_{\text{search}}(\mathcal{K}_t)$, where $Act \in \{q'_{\text{sub}}, \texttt{<end>}\}$
	Answer	*Solver*	Synthesizes the final answer prediction conditioned on the original question and the accumulated reasoning trajectory.	**In:** Query q, Trajectory $\mathcal{T}_t$ **Out:** $a \leftarrow \pi_{\text{ans}}(q, \mathcal{T}_t)$
KMA	**Summary**	*Filter*	Distills pertinent evidence from retrieved documents while actively discarding irrelevant noise to ensure information density.	**In:** Sub-query q'_{sub}, Documents D **Out:** $E \leftarrow \pi_{\text{sum}}(q'_{\text{sub}}, D)$
	Update	*Dynamic Refiner*	Judiciously decides how to integrate evidence to evolve the trajectory: • `<Update>`τ_i`</Update>` (**In-Place Refinement**): Overwrites hallucinations/vague steps. • `<Add>`τ_{new}`</Add>` (**Expansion**): Appends necessary logical hops.	**In:** State $\mathcal{K}_t$, sub-query q'_{sub}, Evidence E **Out:** $op, \mathcal{K}_{t+1} \leftarrow \pi_{\text{upd}}(\mathcal{K}_t, q'_{\text{sub}}, E)$

H Example of Structured Knowledge State

To facilitate a better understanding of the data structure defined in Eq. 1, we provide a concrete example of the Structured Knowledge State $\mathcal{K}_t$.

Consider a multi-hop question q: *"Who is the CEO of the company that developed ChatGPT?"*. Table 8 illustrates the state $\mathcal{K}_t$, where the agent has successfully decomposed the query and retrieved the necessary evidence chain.

Table 8: An example of the Knowledge State $\mathcal{K}_t$. The trajectory $\mathcal{T}_t$ contains a sequence of sub-query and sub-answer pairs that logically support the final answer.

Key	Value (Content)
`"question"` (q)	Who is the CEO of the company that developed ChatGPT?
`"thinking_trajectory"` ($\mathcal{T}_t$)	(τ_1): $\langle q_{sub}^{(1)} \rangle$: Which company developed the chatbot? $\langle a_{sub}^{(1)} \rangle$: The example assumes that the chatbot was developed by OpenAI. (τ_2): $\langle q_{sub}^{(2)} \rangle$: Who is the CEO of OpenAI? $\langle a_{sub}^{(2)} \rangle$: The example uses Sam Altman as the answer.
`"predicted_answer"` (a_t)	Sam Altman

I Joint Training Algorithm Details

In this section, we provide comprehensive pseudocode for the M-ASK joint training process, complementing the methodology described in Section 3.3. Algorithm 1 details the execution flow of the **Parameter-Shared** strategy, where a unified policy π_θ is employed across all functional roles (Planning, Search, Summary, Update, and Answer).

The procedure consists of two distinct phases:

- **Data Collection Phase:** Agents interact sequentially to solve the task. Notably, the Answer Agent serves as an intermediate evaluator at each step t, calculating the marginal performance gain ΔF1 to provide dense, turn-specific supervision.

- **Unified Optimization Phase:** Experiences from all roles are stored in a mixed rollout buffer $\mathcal{B}$. The shared model parameters θ and critic ϕ are then jointly updated via PPO, maximizing the objective function derived from these heterogeneous interaction trajectories.

Algorithm 1: M-ASK Joint Training (Parameter-Shared)

Input: Data $\mathcal{D}$, Unified Model π_θ, Critic V_ϕ, Role Instructions $I = \{I_{\text{plan}}, I_{\text{search}}, \dots\}$, Batch B
Output: Optimized Parameters θ^*, ϕ^*

Initialize θ, ϕ; Initialize unified rollout buffer $\mathcal{B}$;
for *iteration* $i = 1, \dots, M$ **do**
  // 1. Data Collection Phase
  while $|\mathcal{B}| < B$ **do**
    Sample $(q, y) \sim \mathcal{D}$;
    // Phase I: Initialization
    $\mathcal{K}_0 \leftarrow \pi_\theta(q; I_{\text{plan}})$; $a_0 \leftarrow \mathcal{K}_0$.predicted_answer;
    $S_{prev} \leftarrow \text{F1}(a_0, y)$;
    Add transition $(I_{\text{plan}}, q, \mathcal{K}_0, r = S_{prev})$ to $\mathcal{B}$;
    // Phase II: Iterative Collaboration
    $t \leftarrow 0$;
    while $t < T_{max}$ **do**
      // Step 1: SBA Decision
      $Action \leftarrow \pi_\theta(\mathcal{K}_t; I_{\text{search}})$; // Prompt as Searcher
      if $Action ==$ <end> **then**
        Add transition $(I_{\text{search}}, \mathcal{K}_t, \texttt{<end>}, r = 0)$ to $\mathcal{B}$;
        break;
      end
      $q'_{sub} \leftarrow Action$;
      // Step 2: KMA Execution
      $D \leftarrow \text{SearchEngine}(q'_{sub})$;
      $E \leftarrow \pi_\theta(q'_{sub}, D; I_{\text{sum}})$; // as Summarizer
      $op, \mathcal{K}_{next} \leftarrow \pi_\theta(\mathcal{K}_t, q'_{sub}, E; I_{\text{upd}})$; // as Updater
      // Step 3: Evaluation
      $a_{next} \leftarrow \pi_\theta(\mathcal{K}_{next}; I_{\text{ans}})$; // as Answerer
      $S_{curr} \leftarrow \text{F1}(a_{next}, y)$;
      $\Delta\text{F1} \leftarrow S_{curr} - S_{prev}$;
      // Step 4: Store Mixed Experiences
      Add $(I_{\text{search}}, \mathcal{K}_t, q'_{sub}, r = \Delta\text{F1})$ to $\mathcal{B}$;
      Add $(I_{\text{sum}}, q'_{sub}, D, E, r = \Delta\text{F1})$ to $\mathcal{B}$;
      Add $(I_{\text{upd}}, \mathcal{K}_t, E, op, r = \Delta\text{F1})$ to $\mathcal{B}$;
      Add $(I_{\text{ans}}, \mathcal{K}_{next}, a_{next}, r = S_{curr})$ to $\mathcal{B}$;
      $\mathcal{K}_{t+1} \leftarrow \mathcal{K}_{next}$; $S_{prev} \leftarrow S_{curr}$; $t \leftarrow t + 1$;
    end
  end
  // 2. Unified Optimization Phase
  Compute GAE on unified buffer $\mathcal{B}$ using shared Critic V_ϕ;
  for *epoch* $k = 1, \dots, K$ **do**
    Sample mixed mini-batches of multi-agent experiences from the unified rollout buffer $\mathcal{B}$;
    Update the shared policy π_θ and critic V_ϕ via PPO by minimizing $\mathcal{L}_{\text{actor}}(\theta)$ and $\mathcal{L}_{\text{critic}}(\phi)$;
  end
  Clear buffer $\mathcal{B}$;
end

J Detailed Case Study: Structured Knowledge State Evolution

The trajectory presented in Table 9 and Table 10 is a synthetic illustrative example of how M-ASK can address structural limitations of monolithic agents. It highlights three aspects of the proposed design:

1. Correction of Parametric Hallucinations via "Targeting". A pervasive challenge in agentic search is the "cold start" problem, where models must plan without initial observations. In the **Init** phase, the Planning Agent relies on its parametric memory and incorrectly predicts "Guadalajara." Crucially, M-ASK does not treat this initial plan as ground truth but as a *search target*. In **Turn 1**, the Update Agent identifies a conflict between the parametric "Guadalajara" and the retrieved non-parametric evidence ("Puebla"). Instead of attempting to reconcile the two or hedging, the agent prioritizes external evidence, executing an `<Update>` action to overwrite the hallucination. This illustrates one possible error-correction pattern that can limit the propagation of an initial error in the displayed trajectory.

2. Deep Verification of Temporal Constraints. The transition to **Turn 2** highlights the Search Agent's ability to handle complex constraints. The query contains a specific temporal condition ("1943"). While a naive retriever might stop after finding the entity "Puebla," the Search Agent actively verifies the historical context ("significant event... in 1943"). The subsequent `<Add>t2</Add>` action illustrates how a new, distinct piece of background information can be represented as a separate step. In this synthetic example, the "professionalization of the league" supports the answer's validity. By appending it as a new step (t_2), the system constructs a logical evidence chain: *Entity Existence (t_1) + Historical Context (t_2).*

3. Active Prevention of Context Bloating. The example's **Turn 3** illustrates a second state-management pattern. The agent retrieves specific confirmation linking "Puebla" to the "Athletic Club." In a monolithic trajectory that appends each retrieval, this new evidence would increase the accumulated context as search steps continue. In contrast, M-ASK employs **In-Place Refinement**. The Update Agent recognizes that this information is a more precise version of t_1. It triggers `<Update>t1` to refine the query and answer (adding "specifically Puebla") rather than creating a redundant t_3. This mechanism maintains a concise state space ($\mathcal{K}_3$) in the displayed example, illustrating how in-place refinement can limit output growth and preserve a focused context.

K Computational Overhead Analysis

In this section, we provide a detailed breakdown of the computational overhead for M-ASK compared to the monolithic baseline, Search-R1. Table 11 reports the efficiency metrics evaluated on the HotpotQA dataset under identical hardware configurations.

To interpret the results correctly, we define the metrics as follows:

- **Avg. Output Length during Training**: The average number of tokens generated in a *single* LLM invocation per agent.
- **Total Output Tokens per Query (Inference)**: The cumulative sum of tokens generated across the *entire* multi-turn search process for a given query.

Efficiency in Token Generation. The comparison highlights a fundamental structural difference. Search-R1 adopts a monolithic approach, where the entire reasoning and search trajectory is generated in a single, unconstrained stream, resulting in a high volume of tokens ($1100 \sim 1200$) per query. In contrast, M-ASK decomposes the process into granular, atomic steps. Although this involves multi-turn interactions, the **average output per agent invocation** remains short ($31 \sim 32$ tokens). Consequently, the **Total Output Tokens** for M-ASK are significantly lower ($\mathbf{460 \sim 470}$), representing an approximate **60% reduction** in computational load compared to the monolithic baseline.

Throughput Efficiency and Implementation Details. As shown in Table 11, M-ASK currently achieves a higher throughput ($\mathbf{8.35}$ QPS) compared to Search-R1 (5.00 QPS). However, we acknowledge that this gap is influenced by the rollout strategy: M-ASK utilizes an **asynchronous rollout** strategy to manage varying turn lengths, whereas Search-R1 uses a standard synchronous rollout. We believe that optimizing Search-R1 with asynchronous execution would likely increase its QPS. Thus, rather than claiming strict superiority, we conclude that M-ASK maintains a throughput magnitude **comparable** to the monolithic baseline (in the same order of magnitude). This demonstrates that the

Table 9: Full execution log for the query: *"What Mexican city had an Athletic Club whose football team played in the Primera Fuerza in 1943?"* (Part I). The **State** rows explicitly show the three components of $\mathcal{K}_t$. The trajectory steps are indexed as $[\tau_i]$, allowing the Update Agent to target specific steps (e.g., <Update>τ_1</Update>).

Target Question:	What Mexican city had an Athletic Club whose football team played in the Primera Fuerza in 1943?
Reference answer:	**Puebla**

Turn	Role	Full Action Output & Structured Knowledge State ($\mathcal{K}_t$)
Init	**Planning Agent**	<q1> What Mexican city had an Athletic Club whose football team played in the Primera Fuerza in 1943? </q1> <a1> Guadalajara </a1> <predicted_answer> Guadalajara </predicted_answer>
	State $\mathcal{K}_0$ (Initial Structure): "question": What Mexican city had an Athletic Club whose football team played in the Primera Fuerza in 1943? "thinking_trajectory": [$[\tau_1]$ $\langle q_{sub}^{(1)} \rangle$: What Mexican city had an Athletic Club whose football team played in the Primera Fuerza in 1943? $\langle a_{sub}^{(1)} \rangle$: **Guadalajara** *(Parametric Hallucination)*] "predicted_answer": **Guadalajara**	
Turn 1	**Search Agent**	<search> What Mexican city had an Athletic Club whose football team played in the Primera Fuerza in 1943? </search>
	Search Engine	*[External Tool] Retrieved 5 document fragments (∼100 words each) containing keyword matches for "Mexican city", "Athletic Club", "Primera Fuerza", "1943".*
	Summary Agent	<evidence> Puebla, which has a city of Puebla, had an Athletic Club, Puebla A.C., whose football team played in the Primera Fuerza. </evidence>
	Update Agent	<Update>τ_1</Update> (**Corrects Hallucination**)
	Answer Agent	<predicted_answer> Puebla </predicted_answer>
	State $\mathcal{K}_1$ (Corrected Answer): "question": What Mexican city had an Athletic Club whose football team played in the Primera Fuerza in 1943? "thinking_trajectory": [$[\tau_1]$ $\langle q_{sub}^{(1)} \rangle$: What Mexican city had an Athletic Club whose football team played in the Primera Fuerza in 1943? $\langle a_{sub}^{(1)} \rangle$: **Puebla, which has a city of Puebla, had an Athletic Club, Puebla A.C., whose football team played in the Primera Fuerza.**] "predicted_answer": **Puebla** *(Updated via retrieval)*	

Table 10: Full execution log for the query: *"What Mexican city had an Athletic Club whose football team played in the Primera Fuerza in 1943?"* (Part II). The **State** rows explicitly show the three components of $\mathcal{K}_t$. The trajectory steps are indexed as $[\tau_i]$, allowing the Update Agent to target specific steps (e.g., <Update>τ_1</Update>).

Turn	Role	Full Action Output & Structured Knowledge State ($\mathcal{K}_t$)
Turn 2	**Search Agent**	<search> What is a significant event or fact about Puebla A.C.'s football team's participation in the Primera Fuerza in 1943? </search>
	Search Engine	*[External Tool] Retrieved 5 document fragments (∼100 words each) related to "Puebla A.C.", "Primera Fuerza participation", "1943 events".*
	Summary Agent	<evidence> Puebla A.C.'s football team played in the Primera Fuerza prior to the professionalization and development of the Primera División de México in 1943. </evidence>
	Update Agent	<Add>τ_2</Add> (Appends Context)
	Answer Agent	<predicted_answer> Puebla </predicted_answer>
	State $\mathcal{K}_2$ (Expanded Context): "question": What Mexican city had an Athletic Club whose football team played in the Primera Fuerza in 1943? "thinking_trajectory": [$[\tau_1]$ $\langle q_{sub}^{(1)} \rangle$: What Mexican city had an Athletic Club whose football team played in the Primera Fuerza in 1943? $\langle a_{sub}^{(1)} \rangle$: Puebla, which has a city of Puebla, had an Athletic Club, Puebla A.C., whose football team played in the Primera Fuerza. , $[\tau_2]$ $\langle q_{sub}^{(2)} \rangle$: **What is a significant event or fact about Puebla A.C.'s football team's participation in the Primera Fuerza in 1943?** $\langle a_{sub}^{(2)} \rangle$: **Puebla A.C.'s football team played in the Primera Fuerza prior to the professionalization and development of the Primera División de México in 1943.**] "predicted_answer": **Puebla**	
Turn 3	**Search Agent**	<search> What Mexican city, specifically Puebla, had an Athletic Club, Puebla A.C., whose football team played in the Primera Fuerza in 1943? </search>
	Search Engine	*[External Tool] Retrieved 5 document fragments verifying "Puebla", "Puebla A.C.", "Primera Fuerza 1943".*
	Summary Agent	<evidence> Puebla, specifically Puebla, had an Athletic Club, Puebla A.C., whose football team played in the Primera Fuerza in 1943. </evidence>
	Update Agent	<Update>τ_1</Update> (**Refines Existing Step**)
	Answer Agent	<predicted_answer> Puebla </predicted_answer>
	State $\mathcal{K}_3$ (Refined Trajectory): "question": What Mexican city had an Athletic Club whose football team played in the Primera Fuerza in 1943? "thinking_trajectory": [$[\tau_1]$ $\langle q_{sub}^{(1)} \rangle$: What Mexican city, **specifically Puebla**, had an Athletic Club, Puebla A.C., whose football team played in the Primera Fuerza in 1943? $\langle a_{sub}^{(1)} \rangle$: **Puebla, specifically Puebla, had an Athletic Club, Puebla A.C., whose football team played in the Primera Fuerza in 1943.** , $[\tau_2]$ $\langle q_{sub}^{(2)} \rangle$: What is a significant event or fact about Puebla A.C.'s football team's participation in the Primera Fuerza in 1943? $\langle a_{sub}^{(2)} \rangle$: Puebla A.C.'s football team played in the Primera Fuerza prior to the professionalization and development of the Primera División de México in 1943.] "predicted_answer": **Puebla**	
Turn 4	**Search Agent**	<end> (Termination Triggered)

Final prediction: **Puebla** **(✓Matches reference answer)**

Table 11: Comparison of computational overhead between the monolithic Search-R1 and our collaborative M-ASK framework on HotpotQA.

Metric	**Search-R1**	**M-ASK (Ours)**
Avg. Output Length during Training (tokens)	$1100 \sim 1200$	$\mathbf{31 \sim 32}$
Total Output Tokens per Query (Inference)	$1100 \sim 1200$	$\mathbf{460 \sim 470}$
Avg. retrieval calls per query	1.97	1.99
Training hardware	$8 \times$ H800	$8 \times$ H800
Time to convergence	≈ 83 h	≈ 26 h
Training cost	≈ 664 GPU-hours	≈ 208 GPU-hours
Throughput (Queries per Second)	5.00	**8.35**

Table 12: Reference BrowseComp results from web-search agents. Most rows use 7–9B backbones; WebDancer-QwQ-32B is included because it is the source model for the reported 3.8 score. The reported metrics and evaluation settings differ across works, so the entries are contextual rather than a strict ranking.

Method	Backbone	Benchmark / metric	Score
WebDancer-QwQ Wu et al. (2025)	QwQ-32B	BrowseComp-en, pass@1	3.8
WebSailor Li et al. (2025a)	Qwen2.5-7B	BrowseComp-en, pass@1	6.7
DeepDive Lu et al. (2025)	GLM-Z1-9B-0414	BrowseComp, accuracy (%)	6.3
Rethinking RL-Based Deep Research Agents Wan et al. (2026)	Qwen2.5-7B-Instruct	BrowseComp reward $\times 100$, four-run mean	6.2
OffSeeker-8B Zhou et al. (2026)	Qwen3-8B	BrowseComp-en, pass@1 (LLM judge)	12.8

overhead of multi-agent coordination does not create a bottleneck, and the system remains efficient under our measured setup.

L Exploratory 20-Turn Web Search Check

To provide a capability reference for the long-horizon setting, Table 12 lists reported BrowseComp results from web-search agents with similar model scales. Most rows use 7–9B backbones, while WebDancer-QwQ-32B is retained as the source configuration for its reported 3.8 score. These results are background context rather than a controlled ranking: the original works use different training data, browsing environments, interaction budgets, judges, and metric definitions.

The reference rows indicate the rough difficulty level reached by small and medium web-search agents, while preserving the metrics reported by each source. They motivate a same-backbone, same-budget check of whether M-ASK can learn useful long-horizon search behavior.

We therefore run an exploratory 20-turn web-search evaluation using the same Qwen3-8B backbone for all methods. The trainable methods use OpenSeeker-style tasks involving iterative search and page reading, and every method receives the same maximum budget of 20 search turns per query. The goal is a simple validation of long-horizon deep-search effectiveness beyond fixed-corpus Wikipedia QA.

Interpretation. Under this controlled setting, the untrained multi-agent control scores 4.0, below ReAct at 4.4, showing that role decomposition alone does not provide the capability. After joint training, M-ASK outperforms Search-R1 at every checkpoint: 9.5 versus 6.0 at 0.5 epoch, 11.2 versus 7.5 at 1.0 epoch, and 13.7 versus 8.1 at 1.5 epochs. The margin therefore grows from $+3.5$ to $+5.6$ points, while M-ASK improves by 4.2 points over training compared with 2.1 for Search-R1. This controlled trend provides simple evidence that jointly optimizing Search and Knowledge Management Agents with turn-level rewards remains effective for iterative web search and evidence reading.

Table 13: Exploratory web-search scores at different training stages under a 20-turn interaction budget.

Method	Untrained	0.5 epoch	1.0 epoch	1.5 epochs
ReAct	4.4	–	–	–
Untrained multi-agent	4.0	–	–	–
Search-R1	–	6.0	7.5	8.1
M-ASK	–	9.5	11.2	13.7