FROM DELEGATION TO COORDINATION: LEARNING EFFICIENT MULTI-AGENT DEEP RESEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-agent deep-research systems use delegation to parallelize search and keep long tool traces out of the main agent's context. Yet delegation helps only when task allocation and subtask execution are coordinated: the main agent must issue useful subtasks, while subagents must return reliable evidence. Existing training either optimizes only the main agent or broadcasts the same team advantage to all agents, which can reward misleading subagent reports and penalize useful ones. We introduce **LEAD-RL** (**L**earning **E**ffective **A**gent **D**elegation with Reinforcement Learning), a joint main-agent–subagent training framework with contribution-aware credit routing. A training-time rubric evaluates the research utility of each subagent report and gates its inherited team advantage, while final-answer correctness remains the only team reward; no token, step, or speedup reward is used. Across five benchmarks and two model families, LEAD-RL improves answer quality. On BrowseComp, it improves accuracy by 2.49 percentage points and reduces critical-path cost by 24.5% over joint training with shared team credit. During validation, accuracy increases by 6.78 points at approximately constant critical-path cost. These results show that contribution-aware credit assignment improves the quality–efficiency trade-off of multi-agent deep research.

1 INTRODUCTION

Deep-research agents answer complex questions through iterative search, tool use, verification, and cross-source synthesis (Tongyi DeepResearch Team, 2025; MiroMind Team, 2026). Their interaction histories accumulate search results, intermediate hypotheses, and unsuccessful attempts alongside useful evidence. Context-management methods compress or selectively retain this growing state (Wu et al., 2025; Ye et al., 2026; Lu et al., 2026). Delegation offers a complementary mechanism: subagents investigate self-contained subtasks in isolated contexts and return condensed subagent reports. Following prior swarm systems (Kimi Team, 2026; Ning et al., 2026), we study delegation as *active context management*: the policy controls where research is performed and what information returns to the main agent's global context.

Separate contexts do not guarantee effective delegation. The main agent must scope useful subtasks and integrate their results; subagents must search effectively and report reliable evidence. Poor coordination can create redundant research, misleading subagent reports, or a long subagent branch that delays the investigation. The learning problem is therefore to improve task allocation and execution together, not simply to use more agents.

Existing approaches learn delegation from supervised trajectories (Ning et al., 2026), train a main agent against frozen subagents (Kimi Team, 2026), or jointly optimize multiple roles (Mo et al., 2025; Xu et al., 2026). Joint main-agent–subagent training enables co-adaptation but leaves a credit-assignment ambiguity: successful rollouts can contain misleading subagent reports, while unsuccessful rollouts can contain useful evidence that the main agent fails to integrate. Broadcasting the same outcome advantage to every subagent can thus reinforce harmful behavior or penalize productive research.

Our key observation is that delegation provides a natural granularity for credit assignment: the subagent report. Each report corresponds to a specific delegated brief and contains its own evidence and conclusion, allowing its research utility to be assessed separately from the final team outcome.

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.

Building on this structure, we introduce **LEAD-RL** (**L**earning **E**ffective **A**gent **D**elegation with Reinforcement Learning), which jointly updates a shared policy for the main agent and subagents from complete swarm trajectories. A training-time rubric classifies each subagent report as helpful, neutral, or harmful and gates its inheritance of the team advantage. Helpful reports retain only positive team advantages, harmful reports retain only negative ones, and neutral reports retain the original advantage. The main agent continues to receive the team advantage directly, so its delegation decisions remain supervised by final task outcomes. The rubric therefore filters misassigned credit rather than adding a process reward: final-answer correctness remains the sole team objective, without explicit token, step, or speedup terms.

We assess efficiency through critical-path cost, which compares execution depth across systems under a common harness, and idealized speedup, which measures parallelization of each trajectory's own workload rather than wall-clock acceleration. Both are interpreted alongside answer quality and total tokens, since parallelism alone need not reflect useful research.

Across five benchmarks and two model families, our trained models improve answer quality and compete with substantially larger systems. Controlled BrowseComp comparisons isolate the RL stage: LEAD-RL improves accuracy by 2.49 percentage points and reduces critical-path cost by 24.5% relative to joint main-agent–subagent training with shared team credit. Validation accuracy also improves at approximately constant critical-path cost. These results suggest that distributing outcome credit more selectively can shape both research quality and computational behavior without explicit cost rewards.

Our main contributions are:

- We develop a framework for delegation as *active context management*, coupling task allocation, isolated subagent research, and the integration of condensed subagent reports.
- We propose *LEAD-RL*, combining joint main-agent–subagent training with report-level, rubric-based credit routing under an outcome-only team reward.
- We evaluate swarm efficiency through complementary absolute critical-path cost and relative idealized speedup, alongside answer quality and token consumption.
- We demonstrate gains across five benchmarks and two model families, with controlled RL comparisons and validation dynamics revealing a better accuracy–efficiency trade-off.

2 Related Work

2.1 Deep Research and Context Management

Deep-research agents are trained to acquire evidence through retrieval and open-web interaction. Search-R1 and DeepResearcher establish this setting (Jin et al., 2025; Zheng et al., 2025b), while WebSailor, Tongyi DeepResearch, REDSearcher, MiroThinker, and OpenSeeker extend its training data and interaction strategies (Li et al., 2025; Tongyi DeepResearch Team, 2025; Chu et al., 2026; MiroMind Team, 2026; Du et al., 2026). This line primarily improves how a research agent searches for evidence.

A complementary line addresses how evidence is retained once the interaction history becomes long. ReSum summarizes state (Wu et al., 2025), AgentFold consolidates trajectories (Ye et al., 2026), and LongSeeker provides context-editing operations (Lu et al., 2026). These methods transform a single history, whereas Kimi K2.5 and SearchSwarm distribute research across isolated subagent contexts (Kimi Team, 2026; Ning et al., 2026). We study contribution-aware joint RL for this distributed form of active context management.

2.2 Multi-Agent Delegation and Credit Assignment

A main agent's task assignments determine what its subagents investigate, while their reports shape its subsequent decisions. SearchSwarm learns delegation from supervised trajectories (Ning et al., 2026). Kimi K2.5 trains the main agent with frozen subagents (Kimi Team, 2026), whereas MATPO and WideSeek-R1 jointly optimize both roles (Mo et al., 2025; Xu et al., 2026). Joint training also requires deciding how the team outcome should guide each agent's updates. In cooperative reinforce-

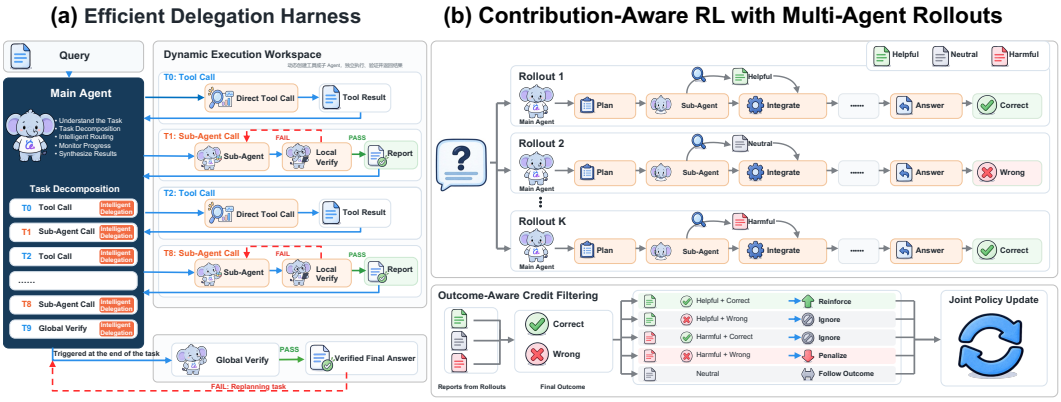

Figure 1: Overview of LEAD-RL. **(a)** The efficient delegation harness combines dynamic task routing with local and global verification. **(b)** Contribution-aware RL filters team-level outcome credit using subagent report judgments and jointly updates the main agent and subagents.

ment learning, COMA uses counterfactual baselines and QMIX factorizes team values (Foerster et al., 2018; Rashid et al., 2020). Lin et al. use LLM-generated rewards to provide agent-specific feedback (Lin et al., 2025). For collaborating language agents, CCPO studies counterfactual and self-evaluated contributions (Li et al., 2026). Our focus is on conflicts between the team outcome and the utility of individual subagent reports: a successful rollout may contain misleading reports, and an unsuccessful one may contain useful evidence. Rather than estimating a report's causal effect on the final answer, LEAD-RL judges its evidential utility and uses that judgment to gate the inherited team advantage. Helpful reports do not inherit negative advantages, harmful reports do not inherit positive advantages, and neutral reports retain the team signal. Final-answer correctness remains the scalar team reward.

3 Method

Overview. Figure 1 summarizes LEAD-RL. The harness supports direct tool use, parallel subagents, and evidence verification, allowing the policy to distribute research across contexts. Training jointly updates both roles from complete swarm rollouts, routing team advantages according to subagent report contributions. Critical-path cost and idealized speedup characterize the learned behavior but are not reward terms; supervision remains anchored to final task outcomes.

3.1 Problem Formulation and Efficiency Measures

Deep research with parallel delegation. We consider a long-horizon research task in which an agent answers a question q through multi-turn reasoning and tool use. Following the main-distributes, sub-executes paradigm (Ning et al., 2026), the system consists of a main agent and a dynamically instantiated set of subagents. All roles are generated by a shared policy π_θ under role-specific instructions.

The main agent maintains the global research state, uses tools directly, and invokes `call_sub_agent`(B_g) with briefs $B_g = \{b_{g,1}, \ldots, b_{g,n_g}\}$ at event g to dispatch n_g subagents in parallel. Subagent i executes a trajectory $H^S_{g,i}$ in an isolated context and returns a subagent report $r_{g,i}$ to the main agent. Section 3.2 describes the brief, subagent report, and verification protocol governing this interaction.

A complete rollout is $H = (H^M, \{H^S_{g,i}\}_{g,i}, y)$, where H^M is the trajectory of the main agent and y its final answer, scored by the task evaluator $R_{\text{perf}}(q, y)$.

Critical steps. Let $\mathcal{G}(H)$ denote the set of delegation events in trajectory H. For each event $g \in \mathcal{G}(H)$, let n_g be the number of subagents dispatched in parallel, and let $N^S_{g,i}(H)$ denote the number of non-report tool-call steps taken by its i-th subagent. Let $N_M(H)$ denote the total number

of tool-call steps taken by the main agent, where each invocation of `call_sub_agent` is counted as one step taken by the main agent.

If all agent actions were executed sequentially, the trajectory would require

$$C_{\text{seq}}(H) = N_M(H) + \sum_{g \in \mathcal{G}(H)} \sum_{i=1}^{n_g} N_{g,i}^S(H) \tag{1}$$

tool-call steps. We refer to this quantity as the *sequential-equivalent cost*, which measures the total amount of work performed across the swarm.

Subagents belonging to the same delegation event execute concurrently. The main agent resumes after their subagent reports return, so the effective cost of an event is determined by its longest subagent branch. Following the critical-path abstraction used for parallel agent systems (Kimi Team, 2026), we define

$$C_{\text{crit}}(H) = N_M(H) + \sum_{g \in \mathcal{G}(H)} \max_{1 \le i \le n_g} N_{g,i}^S(H). \tag{2}$$

Thus, delegation efficiency depends on the longest subagent branch: launching more subagents does not shorten an unbalanced critical path.

Parallel speedup. We quantify the benefit obtained from parallel delegation using the idealized speedup ratio

$$S(H) = \frac{C_{\text{seq}}(H)}{C_{\text{crit}}(H)}. \tag{3}$$

The ratio compares sequential and parallel execution of the same workload, with $S(H) = 1$ when no parallelism is used. It measures step-level parallelism, not wall-clock acceleration: heterogeneous tool latency, communication overhead, and resource contention are abstracted away.

Efficiency as an evaluation target. Critical-path cost supports cross-system comparison, while speedup measures parallelization of a system's own workload. Since redundant subagents can inflate speedup, we interpret it alongside answer quality, critical-path cost, and total tokens. These measures evaluate whether outcome-only learning improves the accuracy–efficiency trade-off; they are not reward terms requiring hand-designed cost coefficients.

3.2 Efficient Delegation Harness

Figure 1(a) illustrates the interaction protocol used to collect swarm rollouts and execute the learned policy. At each step, the main agent chooses between direct tool use and parallel subagent calls. A local verifier filters each report before it returns, and a global verifier checks the synthesized answer before submission. The harness defines these interfaces; the shared policy controls routing, search, and synthesis.

Dynamic task routing. The main agent routes each information need to direct search or delegation based on the accumulated evidence, batching independent needs into parallel calls. Each brief contains a bounded objective, relevant evidence, and unresolved constraints, so the subagent does not require the full main-agent history. The policy generates this routing rather than following fixed stages.

Isolated execution and evidence return. Each subagent searches in an isolated context and produces a report with findings, citations, and remaining uncertainty. Its raw trajectory stays isolated; only the verified report returns, whereas direct tool observations enter the main context. Delegation thus controls which retrieval histories are condensed into the global research state.

Local and global verification. A local verifier checks each candidate report for brief coverage and evidential support. A passing report returns to the main agent; a failed check sends feedback for additional research or revision. At answer time, a global verifier checks the synthesized response for consequential claims, citations, and task constraints. A passing response is released; a failed check

returns control for further research. Both verifiers are role-conditioned invocations of the shared policy, use no reference answer at inference, and are distinct from the training-time contribution rubric in Figure 1(b).

3.3 Contribution-Aware Joint Main-Agent–Subagent Training

Efficient delegation requires the main agent's task allocation and the subagents' search behavior to improve together. Kimi K2.5, for example, optimizes the main agent while holding subagents fixed (Kimi Team, 2026). Besides preventing co-adaptation between the main agent and subagents, this can create a role-policy mismatch when a shared updated checkpoint is used for both roles at inference. We instead optimize both roles from the same swarm rollouts and route subagent credit according to the contribution of each returned subagent report.

Joint main-agent–subagent training. The main agent and subagents share a policy π_θ, conditioned on their respective role contexts. For each question q, we sample K complete rollouts $H_j = (H_j^M, \{H_{j,i}^S\}_{i=1}^{n_j}, y_j)$, $j = 1, \ldots, K$, where H_j^M contains the main agent's reasoning, delegation, and synthesis, and $H_{j,i}^S$ records a subagent's search and reporting. Both roles are sampled from the same evolving policy and included in the same policy update. The policy governing task allocation therefore adapts alongside the policy responsible for subagent search and reporting.

The roles shape each other's training data: the main agent controls subtask scope and requested evidence, while subagents determine the subagent reports returned for synthesis. Updating both together avoids training the main agent against permanently frozen subagents and then deploying an updated shared checkpoint in both roles.

Role-specific instructions and contexts induce distinct conditional behaviors despite parameter sharing. Each complete interaction of the main agent or a subagent forms one logical training trajectory, regardless of its number of model–tool turns. Tool observations provide context but receive no language-model loss, preserving credit at the subagent report level.

Each rollout receives a team reward $R_j = R_{\text{perf}}(q, y_j)$ based only on its final answer. There is no bonus for delegation, rubric labels, shorter trajectories, or higher speedup. For the K rollouts of the same question, we compute the group-relative advantage

$$A_j^{\text{team}} = \frac{R_j - \text{mean}_k\, R_k}{\text{std}_k\, R_k + \epsilon}. \tag{4}$$

The main agent receives $A_j^M = A_j^{\text{team}}$, supervising its delegation decisions through the overall research outcome. Subagents, however, should not indiscriminately inherit the same signal: a useful subagent report may appear in an unsuccessful rollout, while a misleading subagent report may be corrected by a successful main agent. We therefore assign subagent credit according to its estimated contribution.

Dynamic subagent contribution monitoring. Separate from the final-answer judge that supplies the team reward, a contribution judge assesses each subagent report. It examines the question q, reference answer $y^\star$, delegated brief $b_{j,i}$, returned subagent report $r_{j,i}$, and supporting search evidence $E_{j,i}$:

$$c_{j,i} = \mathcal{J}\left(q, y^\star, b_{j,i}, r_{j,i}, E_{j,i}\right) \in \{+1, 0, -1\}. \tag{5}$$

A *helpful* contribution ($+1$) provides accurate, evidence-supported information that advances the research, such as resolving a key fact, eliminating a plausible alternative, or clarifying an ambiguity. A *harmful* contribution (-1) introduces materially misleading conclusions or unsupported claims. Inconclusive or uncertain contributions are assigned *neutral* credit (0).

The evidence consists only of tool outputs captured from that subagent's execution, not additional searches performed by the judge. Neither the main agent's final response nor its correctness verdict is included in the contribution-judging input. The assessment therefore concerns the subagent report's potential research utility, rather than whether the team ultimately succeeded.

The rubric assesses the relevance and evidential quality of each subagent's output, rather than merely whether it completed its assignment or whether the final team answer happened to be correct. A subagent report may therefore be helpful without stating the reference answer—for example, by

evidence-groundedly eliminating a plausible candidate—and an apparently relevant subagent report remains neutral when its support or downstream utility cannot be established. The judge returns a label, a confidence score, the decisive claim, and a short justification. We use a conservative decision policy: helpful or harmful judgments with confidence below 0.8 are mapped to neutral, while materially misleading claims are required for a harmful label. This reduces the risk that rubric noise suppresses valid subagent behavior.

Assessment is repeated for newly sampled subagent reports throughout training, rather than being assigned once to a fixed data set. The contribution signal thus tracks changes in both the briefs produced by the main agent and the subagent reports produced by subagents. The judge is used only for credit assignment during training; neither it nor the reference answer is available to the agents at inference time. The complete rubric is provided in Appendix A.

Contribution-aware credit routing. We route the team advantage through a direction-consistent gate:

$$g(c, A) = \begin{cases} \max(A, 0), & c = +1, \\ A, & c = 0, \\ \min(A, 0), & c = -1. \end{cases} \tag{6}$$

Helpful subagents are protected from negative team advantages, while harmful subagents cannot receive positive team advantages. Neutral subagents retain outcome supervision, preserving learning where individual utility is uncertain. The rubric therefore intervenes only when the direction of subagent credit is sufficiently clear.

The resulting subagent advantage is

$$A^S_{j,i} = \frac{w(t)}{\max(1, |\mathcal{I}_j|)} g(c_{j,i}, A^{\text{team}}_j), \tag{7}$$

where $\mathcal{I}_j$ contains trainable subagents with nonzero gated advantages, and $w(t)$ controls the relative weight of subagent learning and is annealed during training. Normalization prevents rollouts with more subagents from receiving disproportionately large aggregate updates.

Let $\ell_{\text{GSPO}}(h, A)$ denote the clipped policy loss of Group Sequence Policy Optimization (GSPO) (Zheng et al., 2025a) for trajectory h with advantage A. We jointly optimize

$$\mathcal{L}_{\text{joint}} = \mathbb{E}\left[\frac{1}{K}\sum_{j=1}^{K}\left(\ell_{\text{GSPO}}(H^M_j, A^{\text{team}}_j) + \sum_{i=1}^{n_j}\ell_{\text{GSPO}}(H^S_{j,i}, A^S_{j,i})\right)\right] + \beta\mathcal{L}_{\text{KL}}. \tag{8}$$

The KL term anchors both roles to the same reference policy. Normalized subagent advantages prevent additional branches from automatically increasing the aggregate subagent update. The objective rewards neither delegation frequency nor a prescribed swarm topology.

Appendix A summarizes the rollout, judging, credit-routing, and joint-update procedure.

Why outcome-only learning can improve efficiency. LEAD-RL corrects two supervision errors: penalizing helpful subagents for team failures and reinforcing harmful subagents for team successes. More reliable subagent reports may reduce compensatory searches and support more selective delegation, while normalized subagent credit avoids scaling updates with subagent count. This is a hypothesis about learned behavior, not a guarantee of minimum cost. We examine it through accuracy–efficiency trajectories and comparisons with main-only and shared-team-credit RL, without prescribing a trade-off between correctness and execution cost.

4 Experiments

4.1 Experimental Setup

Benchmarks. We evaluate LEAD-RL on BrowseComp (Wei et al., 2025) and BrowseComp-ZH (Zhou et al., 2025) for English and Chinese factual search, GAIA (Mialon et al., 2024) for general

assistant tasks, Humanity's Last Exam (HLE) (Center for AI Safety et al., 2026) for expert questions, and DeepSearchQA (Gupta et al., 2026) for set-valued research. We use the full BrowseComp benchmark and tool-augmented HLE. DeepSeek V4 Pro judges our runs and validation checkpoints; external scores retain their source evaluation protocols.

Evaluation metrics. We report final-answer accuracy for BrowseComp, BrowseComp-ZH, GAIA, and HLE, and mean per-question answer-set F_1 for DeepSearchQA; all are percentages. Efficiency metrics are total main agent and subagent tokens per question (***Tokens/Q***), mean critical-path cost (C_{crit}/Q; Equation 2), and idealized speedup (Equation 3). We also report rubric-assessed helpful and harmful subagent report rates.

Baselines. Table 1 compares closed-source systems, large open-source models, and lightweight research agents; Appendix B lists their references. External scores come from technical reports or model cards and may use different scaffolds, so they provide context rather than a controlled training comparison.

Within the same swarm harness, we evaluate Qwen3.8-Max (Qwen Team, 2026b), DeepSeek V4 Pro and DeepSeek V4 Flash (DeepSeek, 2026), Ling-3-Flash (inclusionAI, 2026), and Qwen3.6-35B-A3B (Qwen Team, 2026a). Matched base-versus-trained and swarm-on/off runs isolate training and delegation effects.

Training variants. We compare four training variants: *SFT initialization*; *Main-only RL*, which keeps the subagents fixed; *joint RL with shared team credit*, which broadcasts the team advantage to both roles; and *LEAD-RL*, which gates subagent advantages using rubric-based report labels. All three RL variants start from the same SFT checkpoint and are trained on the same data. We select checkpoints using the same validation set.

Implementation details. We use sample-level GSPO with $K = 4$ rollouts per question. DeepSeek V4 Pro provides final-answer and contribution judgments using separate temperature-0 prompts whose outputs never enter agent contexts. Correctness is the sole team reward; rubric labels only route subagent credit. Appendix A gives the judging protocol and hyperparameters. Controlled comparisons share the question split, decoding configuration, harness, and execution budget.

4.2 Performance across Benchmarks and Model Families

Table 1 shows gains across both model families and all five benchmarks, spanning factual search, general tool use, and set-valued research. This breadth suggests improvements beyond a single answer format or architecture. These comparisons assess the complete pipeline; Section 4.4 isolates RL relative to SFT.

The Qwen-based model also surpasses the reported MiroThinker-1.7-mini results across all five benchmarks without additional context-management strategies beyond swarm delegation. This comparison with a similarly sized, context-managed model supports distributing research across subagent contexts as an effective approach to long-horizon tasks. External evaluation protocols differ; the matched comparisons below examine learned delegation more directly.

4.3 Accuracy–Efficiency Trade-offs under Swarm Execution

Table 2 compares the effect of swarm execution before and after training. Enabling the swarm changes the base Ling model's accuracy from 59.0% to 58.0%, whereas LEAD-RL improves from 66.5% to 73.5%. With the swarm enabled, training raises accuracy from 58.0% to 73.5%. Token consumption increases slightly from 330.7k to 336.9k tokens per question, while critical-path cost decreases from 251.0 to 244.22 steps per question. The trained model therefore gains more from delegation without a substantial increase in token consumption.

This improvement makes a substantially smaller model competitive with larger systems in the same harness. LEAD-RL (124B-A5.1B) outperforms DeepSeek V4 Flash (284B-A13B) while using fewer tokens and critical steps, despite having less than half its total and active parameters. It also reaches 73.5% accuracy, within 1.8 percentage points of the 1.6T-A49B DeepSeek V4 Pro, with

Table 1: Main results across five deep-research benchmarks. DeepSearchQA reports mean answer-set F_1; all other columns report accuracy. * denotes context management, † denotes runs executed in our swarm harness, and blue rows denote LEAD-RL.

Model	Size	BrowseComp	BrowseComp-ZH	GAIA	HLE (w/ Tools)	DeepSearchQA (F_1)
Closed-source models						
GPT-5	–	54.9	65.0	76.4	35.2	79.0
Claude-4.5-Sonnet	–	24.1	42.4	66.0	32.0	27.9
Open-source models						
GLM-4.7	355B-A32B	67.5*	66.6*	–	42.8	–
DeepSeek V3.2	671B-A37B	67.6*	65.0*	75.1	40.8	60.9
LongCat-Flash-Thinking-2601	560B-A27B	73.1*	77.7*	–	–	–
MiniMax-M2	230B-A10B	44.0	–	75.7	–	–
Step-3.5-Flash	196B-A11B	69.0*	66.9	84.5	–	–
Open-source lightweight models						
RedSearcher	30B-A3B	57.4*	58.2*	80.1	34.3	–
LongSeeker	30B-A3B	61.5*	62.5*	77.7*	–	–
MiroThinker-1.5-mini	30B-A3B	56.1*	66.8*	72.0*	31.0*	–
MiroThinker-1.7-mini	30B-A3B	67.9*	72.3*	80.3*	36.4*	67.9*
Models evaluated in our swarm harness						
Qwen3.8-Max†	2.4T-A95B	72.3	73.4	92.1	–	–
DeepSeek V4 Pro†	1.6T-A49B	75.3	74.7	97.0	62.2	93.5
DeepSeek V4 Flash†	284B-A13B	71.2	73.4	92.2	58.8	91.9
Ling-3-Flash†	124B-A5.1B	58.0	71.3	73.1	30.8	80.2
Qwen3.6-35B-A3B†	35B-A3B	58.3	70.9	81.6	41.7	83.2
Baseline: Ling-3-Flash						
LEAD-RL (Ling-3-Flash)†	124B-A5.1B	**73.5**	**76.3**	**94.2**	**56.2**	**86.3**
Baseline: Qwen3.6-35B-A3B						
LEAD-RL (Qwen3.6-35B-A3B)†	35B-A3B	**70.0**	**72.9**	**88.4**	**54.3**	**85.2**

Table 2: Effect of swarm execution on BrowseComp. Accuracy columns are matched runs under the same budget; efficiency is reported for swarm-enabled runs.

		Accuracy (%) ↑		Swarm efficiency		
Model	**Size**	**w/o Swarm**	**w/ Swarm**	**Tokens/Q (k) ↓**	**C_{crit}/Q ↓**	**Speedup ↑**
Qwen3.8-Max	2.4T-A95B	63.2	72.3	277.3	159.09	1.67x
DeepSeek V4 Pro	1.6T-A49B	67.4	75.3	350.1	266.13	1.39x
DeepSeek V4 Flash	284B-A13B	65.2	71.2	401.7	293.80	1.44x
Ling-3-Flash	124B-A5.1B	59.0	58.0	330.7	251.0	1.69x
LEAD-RL (Ours)	124B-A5.1B	66.5	73.5	336.9	244.22	1.71x

lower measured execution costs. These comparisons show that training effective delegation can narrow the performance gap to much larger models while retaining a compact parameter footprint.

Qwen3.8-Max uses fewer tokens and has a shorter critical path than LEAD-RL. LEAD-RL achieves 73.5% accuracy on BrowseComp, 1.2 percentage points higher than Qwen3.8-Max, with substantially fewer total and active parameters.

4.4 Effect of Contribution-Aware Joint Main-Agent–Subagent Training

Table 3 compares four training variants on BrowseComp. Main-only RL and joint RL with shared team credit both yield lower accuracy than the SFT checkpoint. LEAD-RL achieves 73.49% accuracy, exceeding joint RL with shared team credit by 2.49 percentage points. It also reduces critical-path cost from 323.30 to 244.22 steps per question, a decrease of 24.5%.

LEAD-RL also produces more rubric-assessed helpful subagent reports and fewer harmful ones than shared team credit. Relative to SFT, it improves accuracy while reducing critical-path cost by 39.5%. More reliable reports may reduce corrective search, although we do not measure that mechanism directly. The gains therefore extend beyond recovering performance lost under other RL variants and arise without an explicit cost penalty.

Table 3: Training variants on BrowseComp. Main-only RL freezes subagents, while joint RL with shared credit updates both roles using the team advantage. LEAD-RL gates subagent advantages using report utility labels. Helpful and Harmful denote judge-assessed report rates. C_{crit}/Q measures critical-path steps per question; Speedup measures idealized step-level parallelism.

		Report labels (%)		Efficiency	
Training variant	**Acc. (%) ↑**	**Helpful ↑**	**Harmful ↓**	C_{crit}/Q **↓**	**Speedup**
SFT initialization	72.50	27.11	12.89	403.37	1.35×
Main-only RL	70.39	27.42	11.21	421.76	1.42×
Joint RL (shared credit)	71.00	21.15	13.84	323.30	1.39×
LEAD-RL (ours)	**73.49**	**31.52**	**9.87**	**244.22**	**1.71×**

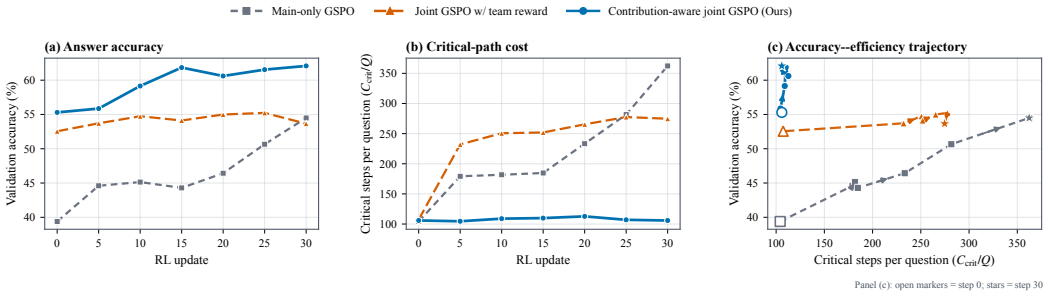

Figure 2: Validation dynamics across RL checkpoints on a fixed validation set. Open markers and stars denote the first and last recorded checkpoints; arrows indicate training direction.

4.5 Learning Dynamics without Explicit Cost Rewards

Figure 2 compares checkpoints on a fixed validation set; step 0 follows the first RL update, so comparisons are within-run over the common recorded horizon.

Main-only and shared-team-credit training develop substantially longer critical paths; the latter gains little accuracy despite this growth. LEAD-RL instead gains 6.78 accuracy points at approximately constant critical-path cost, producing the upward trajectory in panel (c), even though cost is absent from the reward. The favorable trade-off thus develops during training rather than appearing only at the final checkpoint. Appendix C reports the raw on-policy measurements; later checkpoints remain stable within the observed horizon.

Rubric-guided filtering offers one explanation: blocking positive updates to helpful subagent reports and negative updates to helpful ones reduces conflicting supervision. This may promote more useful delegated research and fewer unnecessary follow-up searches, but the mechanism remains a hypothesis rather than a directly measured effect.

5 Conclusion

We presented LEAD-RL, which learns delegation as active context management through joint main-agent–subagent training and rubric-based credit routing. Final-answer correctness remains the sole team reward. Across five benchmarks and two model families, the trained models improve answer quality and compete with substantially larger systems. Controlled RL comparisons and validation dynamics show improved accuracy–efficiency trade-offs without explicit cost penalties. These findings suggest that correcting how subagents inherit outcome credit can improve both research quality and the use of computation in agent swarms.

REFERENCES

Anthropic. Introducing claude sonnet 4.5, 2025. https://www.anthropic.com/news/claude-sonnet-4-5.

Center for AI Safety, Scale AI, and HLE Contributors Consortium. A benchmark of expert-level academic questions to assess ai capabilities. *Nature*, 649:1139–1146, 2026. doi: 10.1038/s41586-025-09962-4.

Zheng Chu, Xiao Wang, Jack Hong, Huiming Fan, Yuqi Huang, Yue Yang, Guohai Xu, Chenxiao Zhao, Cheng Xiang, Shengchao Hu, Dongdong Kuang, Ming Liu, Bing Qin, and Xing Yu. RED-Searcher: A scalable and cost-efficient framework for long-horizon search agents. *arXiv preprint arXiv:2602.14234*, 2026. https://arxiv.org/abs/2602.14234.

DeepSeek. Deepseek v4 preview release, 2026. https://api-docs.deepseek.com/news/news260424.

DeepSeek-AI. Deepseek-v3.2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025.

Yuwen Du, Rui Ye, Shuo Tang, Xinyu Zhu, Yijun Lu, Yuzhu Cai, and Siheng Chen. OpenSeeker: Democratizing frontier search agents by fully open-sourcing training data. In *Conference on Language Modeling*, 2026. https://arxiv.org/abs/2603.15594.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. https://ojs.aaai.org/index.php/AAAI/article/view/11794.

GLM Team. Glm-4.7: Advancing the coding capability, 2025. https://z.ai/blog/glm-4.7.

Nikita Gupta, Riju Chatterjee, Lukas Haas, Connie Tao, Andrew Wang, Chang Liu, Hidekazu Oiwa, Elena Gribovskaya, Jan Ackermann, John Blitzer, Sasha Goldshtein, and Dipanjan Das. Deepsearchqa: Bridging the comprehensiveness gap for deep research agents. *arXiv preprint arXiv:2601.20975*, 2026.

Ailin Huang, Ang Li, Aobo Kong, Bin Wang, Binxing Jiao, Bo Dong, Bojun Wang, Boyu Chen, Brian Li, Buyun Ma, et al. Step 3.5 flash: Open frontier-level intelligence with 11b active parameters. *arXiv preprint arXiv:2602.10604*, 2026.

inclusionAI. Ling-3.0-flash. Hugging Face, 2026. https://huggingface.co/inclusionAI/Ling-3.0-flash.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O. Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Conference on Language Modeling*, 2025. https://openreview.net/forum?id=Rwhi91ideu.

Kimi Team. Kimi K2.5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026. https://arxiv.org/abs/2602.02276.

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*, 2025. https://arxiv.org/abs/2507.02592.

Zhongyi Li, Wan Tian, Jinju Chen, Huiming Zhang, Yang Liu, Yikun Ban, and Fuzhen Zhuang. Counterfactual credit policy optimization for multi-agent collaboration. *arXiv preprint arXiv:2603.21563*, 2026. https://arxiv.org/abs/2603.21563.

Muhan Lin, Shuyang Shi, Yue Guo, Vaishnav Tadiparthi, Behdad Chalaki, Ehsan Moradi Pari, Simon Stepputtis, Woojun Kim, Joseph Campbell, and Katia Sycara. Speaking the language of teamwork: LLM-guided credit assignment in multi-agent reinforcement learning. *arXiv preprint arXiv:2502.03723*, 2025. https://arxiv.org/abs/2502.03723.

Yijun Lu, Rui Ye, Yuwen Du, Jiajun Wang, Songhua Liu, and Siheng Chen. LongSeeker: Elastic context orchestration for long-horizon search agents. *arXiv preprint arXiv:2605.05191*, 2026. https://arxiv.org/abs/2605.05191.

Meituan LongCat Team. Longcat-flash-thinking-2601 technical report. *arXiv preprint arXiv:2601.16725*, 2026.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: A benchmark for general ai assistants. In *International Conference on Learning Representations*, 2024. https://proceedings.iclr.cc/paper_files/paper/2024/hash/25ae35b5b1738d80f1f03a8713e405ec-Abstract-Conference.html.

MiniMax-AI. Meet minimax-m2, 2025. https://github.com/MiniMax-AI/MiniMax-M2.

MiroMind Team. MiroThinker-1.7 & H1: Towards heavy-duty research agents via verification. *arXiv preprint arXiv:2603.15726*, 2026. https://arxiv.org/abs/2603.15726.

Zhanfeng Mo, Xingxuan Li, Yuntao Chen, and Lidong Bing. Multi-agent tool-integrated policy optimization. *arXiv preprint arXiv:2510.04678*, 2025. https://arxiv.org/abs/2510.04678.

Pu Ning, Quan Chen, Kun Tao, Xinyu Tang, Tianshu Wang, Qianggang Cao, Xinyu Kong, Zujie Wen, Zhiqiang Zhang, and Jun Zhou. SearchSwarm: Towards delegation intelligence in agentic LLMs for long-horizon deep research. *arXiv preprint arXiv:2606.09730*, 2026. https://arxiv.org/abs/2606.09730v1.

OpenAI. Introducing gpt-5, 2025. https://openai.com/index/introducing-gpt-5/.

Qwen Team. Qwen3.6-35B-A3B. Hugging Face, 2026a. https://huggingface.co/Qwen/Qwen3.6-35B-A3B.

Qwen Team. Qwen3.8-2.4T-A95B. Hugging Face, 2026b. https://huggingface.co/Qwen/Qwen3.8-2.4T-A95B.

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. https://jmlr.org/papers/v21/20-081.html.

Tongyi DeepResearch Team. Tongyi DeepResearch technical report. *arXiv preprint arXiv:2510.24701*, 2025. https://arxiv.org/abs/2510.24701.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. Browsecomp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*, 2025.

Xixi Wu, Kuan Li, Yida Zhao, Liwen Zhang, Litu Ou, Huifeng Yin, Zhongwang Zhang, Xinmiao Yu, Dingchu Zhang, Yong Jiang, Pengjun Xie, Fei Huang, Minhao Cheng, Shuai Wang, Hong Cheng, and Jingren Zhou. ReSum: Unlocking long-horizon search intelligence via context summarization. *arXiv preprint arXiv:2509.13313*, 2025. https://arxiv.org/abs/2509.13313.

Zelai Xu, Zhexuan Xu, Ruize Zhang, Chunyang Zhu, Shi Yu, Weilin Liu, Quanlu Zhang, Wenbo Ding, Chao Yu, and Yu Wang. WideSeek-R1: Exploring width scaling for broad information seeking via multi-agent reinforcement learning. *arXiv preprint arXiv:2602.04634*, 2026. https://arxiv.org/abs/2602.04634.

Rui Ye, Zhongwang Zhang, Kuan Li, Huifeng Yin, Zhengwei Tao, Yida Zhao, Liangcai Su, Liwen Zhang, Wenbiao Yin, Zile Qiao, Xinyu Wang, Pengjun Xie, Fei Huang, Jingren Zhou, Siheng Chen, and Yong Jiang. AgentFold: Long-horizon web agents with proactive context folding. In *International Conference on Learning Representations*, 2026. https://openreview.net/forum?id=IuZoTgsUws.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025a. https://arxiv.org/abs/2507.18071.

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. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 414–431, 2025b. doi: 10.18653/v1/2025.emnlp-main.22. https://aclanthology.org/2025.emnlp-main.22/.

Peilin Zhou, Bruce Leon, Xiang Ying, Can Zhang, Yifan Shao, Qichen Ye, Dading Chong, Zhiling Jin, Chenxuan Xie, Meng Cao, et al. Browsecomp-zh: Benchmarking web browsing ability of large language models in chinese. *arXiv preprint arXiv:2504.19314*, 2025.

A CONTRIBUTION RUBRIC AND JOINT MAIN-AGENT–SUBAGENT TRAINING

Two distinct judging tasks. We use DeepSeek V4 Pro as an external judge for two training signals, with temperature 0 and a maximum completion budget of 16,384 tokens per request. The final-answer judge receives the original question, reference answer, and main agent response. Following the BrowseComp judging protocol, it extracts the final answer and checks semantic equivalence to the reference, allowing a small numerical tolerance where appropriate. Its parsed `correct: yes/no` verdict determines the binary team reward. The contribution judge instead evaluates individual subagent reports; its labels only route the resulting team advantage and never alter that reward.

Contribution-judge inputs. Each item contains the original question, reference answer, delegated brief, subagent report, and evidence captured from that subagent's tool calls. No main agent final response or team-correctness label is included. Evidence entries retain the tool name, arguments, and returned content. We prioritize visited-page content over search results, preserving the original order within each category. The judge does not browse for additional evidence or infer unobserved tool calls. The question, reference answer, brief, subagent report, and evidence use character budgets of 4,000, 2,000, 4,000, 6,000, and 8,000, respectively; truncated fields are explicitly marked. Evidence may be empty, in which case useful claims still require support in the supplied inputs.

Contribution rubric. The rubric evaluates potential research utility rather than downstream use by the main agent. Supported elimination of a plausible candidate can be helpful even without stating the final answer; an unsupported answer guess is neutral, even if it matches the reference. Harmful labels require a materially misleading assertion, not merely a failed or incomplete search. The complete judge instruction is reproduced below.

Contribution-rubric judge prompt

You are a strict but conservative grader of one SearchSwarm subagent's research contribution to solving a question.

For every item you receive:

- `original_question`
- `reference_answer`
- `delegated_task`
- `report`
- `evidence`: excerpts actually returned by the subagent's tools; it may be empty

Classify exactly one label:

- **helpful:** The subagent report supplies accurate, non-trivial information that materially helps derive or verify the reference answer, OR it correctly rules out a plausible competing hypothesis or resolves an ambiguity in a way that materially narrows the search. The useful claim must be supported by the supplied subagent report/evidence, not merely guessed.
- **harmful:** The subagent report makes a confident, materially misleading claim that pushes the investigation toward a wrong answer or away from the reference answer. This includes confidently endorsing a wrong candidate, presenting weak extrapolation as established fact, contradicting decisive supplied evidence, or fabricating evidence.
- **neutral:** Everything else, including relevant but generic work, duplicated work, an unresolved candidate list, an unsupported guess, a cautious failed search, weakly supported speculation, or information whose contribution cannot be established from the supplied inputs.

Rules:

1. Judge potential research utility, not whether the main agent used the subagent report and not whether the main agent's final response was correct.
2. The subagent report need not literally contain the reference answer. A well-supported elimination of a plausible wrong hypothesis can be helpful.
3. Do not infer missing searches, visits, citations, or facts. Prefer neutral when support or effect is unclear.
4. Use harmful only for a material misleading assertion, not merely for an incomplete or unsuccessful investigation.
5. The reference answer anchors the final target, but intermediate claims may still be useful if they form a valid path toward it.
6. Be robust to spelling variants, translations, and aliases; judge semantics, not string equality.

7. A subagent report that merely states the reference answer without supporting evidence is neutral.

Return JSON only:

```
{"items":[{"id":"...", "label":"helpful|neutral|harmful",
"confidence":0.0, "decisive_claim":"...", "reason":"..."}]}
```

Return exactly one result for every input id and no additional fields.

Structured output and conservative filtering. Requests batch up to two subagent reports, each identified by an item ID. The judge returns a JSON record containing the label, confidence, decisive claim, and justification for every input ID. We validate the ID mapping and label/confidence ranges. We treat the judge-reported confidence as a filtering signal rather than a calibrated probability: helpful or harmful predictions below 0.8 are mapped to neutral. Confidence and textual justifications are not added to the reward or used as continuous advantage weights; only the effective three-way label controls credit routing.

Incomplete trajectories and judge failures. Subagents that terminate because of model-side limits or fail to produce a subagent report receive neutral credit without an LLM contribution judgment. Subagent trajectories affected by infrastructure failures are excluded from contribution-based subagent updates. This differs from temporary unavailability of the auxiliary contribution judge: after bounded retries, completed subagents fall back to neutral while the already computed main agent reward is preserved. Neutral subagents with trainable tokens retain ordinary team-advantage supervision; neutral does not mean that their trajectories are discarded. Contribution judgments and reference answers are used for training-side supervision, not returned to the agents, and the contribution judge adds no inference-time computation.

Joint main-agent–subagent training procedure. For each question, we sample $K = 4$ complete swarm rollouts from the current shared policy. Final-answer rewards are normalized within the group to produce A_j^{team}. The trajectory of the main agent inherits this advantage directly. For each subagent, the rubric label is used by the gate in Equation 6: helpful subagent reports retain only positive team advantages, harmful subagent reports retain only negative advantages, and neutral subagent reports retain the original advantage. The main agent and retained subagent trajectories are then optimized together using the joint objective in Equation 8. Thus, the rubric neither changes the task reward nor rewards additional delegation; it only routes the existing outcome signal across subagent trajectories.

Training hyperparameters. Each update contains 32 questions with $K = 4$ complete rollouts per question, yielding approximately 128 main agent trajectories and their associated subagent trajectories. Both roles share model parameters and use role-specific contexts. Each main agent rollout may instantiate up to 16 subagents. We use sample-level GSPO with a KL penalty coefficient of 0.001 against the reference policy. The relative subagent weight $w(t)$ is held at 0.4 for the first 10% of training, annealed to 0.2 by 70%, and held constant thereafter.

B Additional Baseline Details

The closed-source baselines in Table 1 include GPT-5 (OpenAI, 2025) and Claude-4.5-Sonnet (Anthropic, 2025). Large open-source baselines include GLM-4.7 (GLM Team, 2025), DeepSeek V3.2 (DeepSeek-AI, 2025), LongCat-Flash-Thinking-2601 (Meituan LongCat Team, 2026), MiniMax-M2 (MiniMax-AI, 2025), and Step-3.5-Flash (Huang et al., 2026). At the 30B-A3B scale, we compare against RedSearcher (Chu et al., 2026), LongSeeker (Lu et al., 2026), and MiroThinker-1.5/1.7-mini (MiroMind Team, 2026).

C Raw On-Policy Training Dynamics

Figure 3 complements the fixed-validation analysis in Section 4.5 with measurements from the on-policy training batches. The main-only, shared-team-credit, and LEAD-RL runs contain 35, 34, and 60 recorded updates, respectively. Each thin mark is the mean of one sampled update batch; the

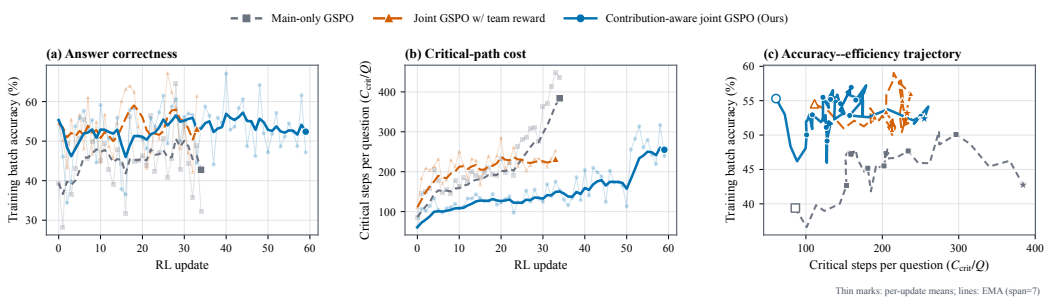

Figure 3: Raw on-policy training dynamics. Thin marks show per-update batch means and dark curves span-7 EMAs; open markers and stars denote the initial and final updates.

dark curves use an exponential moving average (EMA) with span seven. Because both questions and trajectories vary across updates, these measurements expose the behavior optimized during training but are not estimates of generalization performance.

Outcome signal. Panel (a) is intentionally noisy because every update evaluates a newly sampled set of questions and trajectories. Its EMA remains largely in the 50–55% range for the two joint-training variants, while main-only GSPO rises from a lower initialization. The absence of a monotonic training curve is therefore not evidence of optimization failure: group-relative GSPO uses within-question reward differences, whereas the plotted batch mean also reflects variation in batch difficulty. Fixed-set validation in Figure 2 provides the controlled measure of performance improvement.

Growth of the critical path. Panel (b) reveals a much clearer difference between the objectives. At update 30, the EMA critical-path costs of main-only GSPO, joint GSPO with shared team reward, and LEAD-RL are 296.45, 226.18, and 132.23 steps per question, respectively. LEAD-RL is therefore 55.4% below main-only training and 41.5% below shared-team-credit training at the same update, while their EMA batch correctness values are 50.09%, 55.06%, and 55.19%. Thus, the baselines obtain their training progress by rapidly expanding the effective research path, whereas contribution-aware joint training remains substantially less search-intensive.

Accuracy–efficiency trajectory. Panel (c) combines the two signals. During the shared training horizon, LEAD-RL occupies the upper-left region more consistently: it preserves batch correctness while avoiding the pronounced rightward drift of the baselines. In the longer LEAD-RL run, critical-path cost begins to rise more rapidly after approximately 50 updates without a sustained increase in batch correctness. This late-stage behavior illustrates why monitoring efficiency throughout RL is necessary and why checkpoint selection should consider the joint accuracy–efficiency frontier rather than defaulting to the final optimization step.

D Case Study: Delegation as Active Context Management

We examine one successful LEAD-RL trajectory to illustrate the learned structure of delegation. The example requires matching several independent clues against historical cricket scorecards. The main agent first identifies five plausible matches, assigns one candidate to each of five subagents in a single parallel dispatch, and then verifies and synthesizes the returned evidence. For readability, we omit raw search results and subagent scratchpads and show only the information that crosses the delegation boundary. This qualitative example is not used in the aggregate evaluation.

User question.

In a cricket match played between 1980 and 1990 (exclusive), a player was dismissed in one of the rarest ways, with no credit given to the bowler. In the same match, three players were dismissed for a duck, and three players were run out. The team that won the toss later won the game, and the player of the match was given to the player from the winning team who scored the highest runs in that match. How many runs were given by extras in total in that match?

Flow.

1. **Candidate formation.** The main agent interprets the no-bowler-credit clue as an unusual dismissal and narrows the period to five candidate international matches involving either *handled the ball* or *obstructing the field*.
2. **Parallel verification.** It dispatches all five candidates concurrently. Every subagent receives the same checklist: recover the scorecard, count ducks and run-outs, verify the toss and result, verify the player of the match, and total the extras.
3. **Central synthesis.** The subagent reports eliminate four candidates. The main agent independently revisits the surviving scorecard and the unusual-dismissal record, then adds the extras from the two innings.

Parallel candidate decomposition. The main agent keeps the global constraint set in its own context but externalizes the expensive scorecard retrieval. Its single dispatch creates five bounded and mutually exclusive research branches:

Main agent dispatch (abridged)

Candidate branches: Mohsin Khan, Pakistan–Australia Test (1982); Desmond Haynes, West Indies–India Test (1983); Mohinder Amarnath, India–Australia ODI (1986); Rameez Raja, Pakistan–England ODI (1987); and Mohinder Amarnath, India–Sri Lanka ODI (1989).

Shared verification checklist: retrieve the full scorecard; count ducks; count run-outs; identify the toss and match winners; identify the player of the match; and compute total extras across all innings.

This decomposition avoids giving a subagent the original open-ended riddle. Each subagent instead receives a named match and a fixed evidence schema, which makes the returned subagent reports directly comparable and cheap for the main agent to integrate.

Evidence returned across the delegation boundary. The subagent assigned to the 1987 Pakistan–England match returns the following compact findings from its retrieved scorecard and dismissal record. The box reproduces what crossed the delegation boundary as a qualitative trajectory excerpt, rather than as independently annotated benchmark evidence.

Subagent report: Pakistan vs. England, Karachi, 20 November 1987

- Rameez Raja was dismissed for *obstructing the field*, an unusual dismissal for which no bowler receives credit.
- The three ducks were Phil DeFreitas, Abdul Qadir, and Asif Mujtaba; the three run-outs were Mike Gatting, Shoaib Mohammad, and Manzoor Elahi.
- England won both the toss and the match. Graham Gooch was player of the match after scoring 142, the highest score of the match.
- The England innings contained 20 extras and the Pakistan innings contained 14 extras.

The trajectory's parallel subagent reports summarize candidate elimination as follows. These values are report contents rather than independently annotated ground truth:

Candidate match	Ducks	Run-outs	Toss winner won	PoM condition	Extras	Decision
Pakistan–Australia, 1982	5	3	No	No	65	Reject
West Indies–India, 1983	3	2	Draw	–	87	Reject
India–Australia, 1986	1	5	Yes	No	21	Reject
Pakistan–England, 1987	**3**	**3**	**Yes**	**Yes**	**34**	**Retain**
India–Sri Lanka, 1989	1	3	Yes	Yes	25	Reject

Main agent synthesis. After receiving the subagent reports, the main agent independently rechecks the surviving scorecard for the three ducks, three run-outs, toss, result, and player-of-the-match fields, and separately verifies the dismissal rule. It then performs the remaining computation, $20 + 14 = 34$.

Final answer

Only Pakistan vs. England, 2nd ODI at Karachi on 20 November 1987, satisfies all constraints. England's innings contained 20 extras and Pakistan's contained 14, giving a combined total of **34 runs in extras**.

This trace illustrates delegation as active context management rather than indiscriminate search expansion. The main agent preserves the compact global hypothesis set and final decision, while subagents absorb five independent evidence-retrieval traces and return only constraint-aligned summaries. Because the branches are independent, their scorecard searches can be executed in parallel; because they share a reporting schema, the main agent can eliminate candidates without importing their raw browsing histories into its context.