# Agentic Aggregation for Parallel Scaling of Long-Horizon Agentic Tasks

**Yoonsang Lee Howard Yen Xi Ye Danqi Chen**
Princeton Language and Intelligence, Princeton University
{yoonsang, hyen, danqic}@cs.princeton.edu xi.ye@princeton.edu

## Abstract

We study parallel test-time scaling for long-horizon agentic tasks such as agentic search and deep research, where multiple rollouts are generated in parallel and aggregated into a final response. While such scaling has proven effective for chain-of-thought reasoning, agentic tasks pose unique challenges: trajectories are long, multi-turn, and tool-augmented, and outputs are often open-ended. Aggregating only final answers discards rich information from trajectories, while concatenating all trajectories exceeds the model's context window. To address this, we propose **AggAgent**, an *aggregation agent that treats parallel trajectories as an environment*. We equip it with lightweight tools to inspect candidate solutions and search across trajectories, enabling it to navigate and synthesize information on demand. Across six benchmarks and three model families (GLM-4.7, Qwen3.5, MiniMax-M2.5), AggAgent outperforms all existing aggregation methods—by up to 5.3% absolute on average and 10.3% on two deep research tasks—while adding minimal overhead, as the aggregation cost remains bounded by a single agentic rollout. Our findings establish agentic aggregation as an effective and cost-efficient approach to parallel test-time scaling.[1]

## 1 Introduction

Scaling test-time compute has emerged as a promising avenue for enhancing the performance of large language models (LLMs) (Wei et al., 2022; Wang et al., 2023; Brown et al., 2024; Welleck et al., 2024; Snell et al., 2025; Muennighoff et al., 2025). This success has largely been demonstrated in standard chain-of-thought (CoT) tasks such as mathematical reasoning and coding (Qi et al., 2025; Zhao et al., 2025). However, *long-horizon agentic tasks*

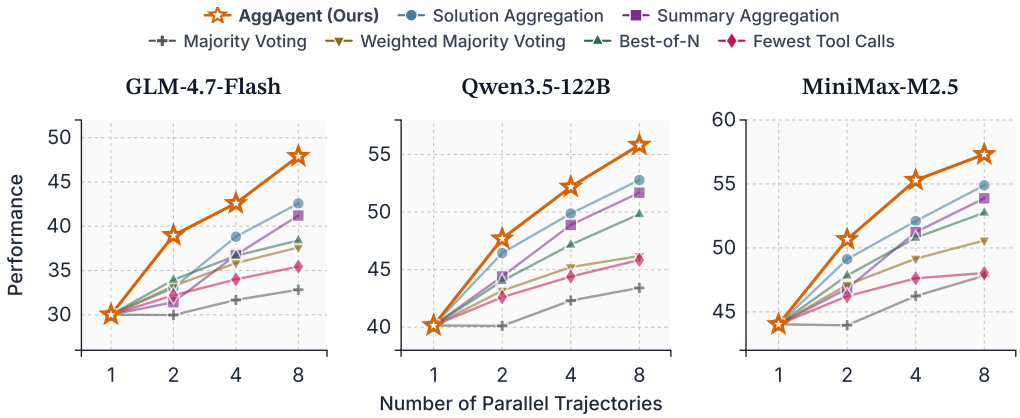


Figure 1: **AggAgent consistently outperforms existing aggregation methods.** We measure the average performance across six long-horizon agentic benchmarks (Section 4) against the number of parallel trajectories. The same model as the rollout agent serves as the aggregator.

[1]Our code is available at https://github.com/princeton-pli/AggAgent.

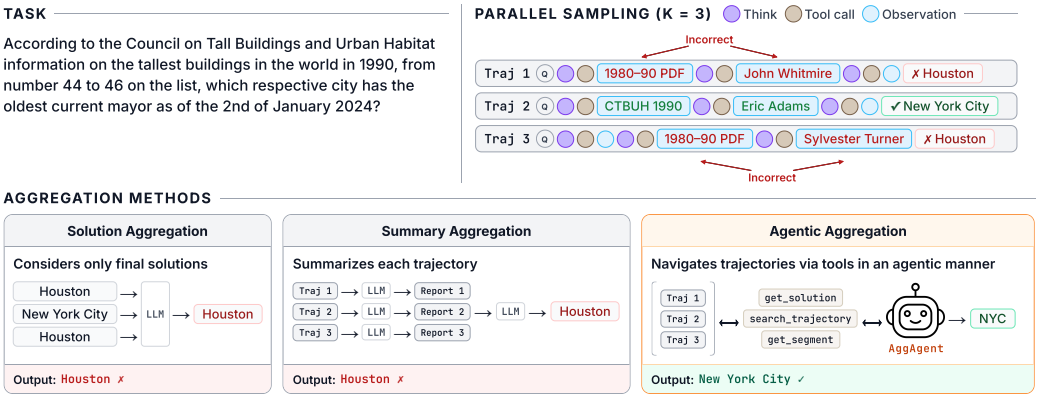


Figure 2: **Overview of aggregation methods for parallel scaling. (Top)** An agent produces $K = 3$ independent rollouts on a long-horizon task. **(Bottom)** *Solution Aggregation* feeds only final solutions to an LLM, discarding intermediate reasoning. *Summary Aggregation* compresses each trajectory into a lossy summary. **AggAgent** (ours) navigates trajectories via tools in an agentic manner, enabling full-fidelity cross-trajectory reasoning at low cost.

(e.g., deep research, software engineering, web navigation) present a fundamentally different challenge: trajectories are multi-turn, spanning hundreds of steps with interleaved tool calls and observations (Yao et al., 2022b).

In this work, we study *parallel scaling* for long-horizon agentic tasks, an approach proven effective for CoT reasoning tasks yet largely unexplored for agentic search and deep research. Parallel scaling enables simultaneous generation of multiple independent trajectories, offering a natural computational advantage (Zhao et al., 2025). Furthermore, parallel scaling offers substantial improvements over single-agent performance (Wei et al., 2025; Li et al., 2026): for instance, GLM-4.7-Flash (Zeng et al., 2025) improves from 27% to 59% on BrowseComp (Wei et al., 2025) and 25% to 51% on HLE (Phan et al., 2025) when scaling from Pass@1 to Pass@8, confirming that correct solutions frequently exist within the parallel rollouts.

The central question then becomes how to effectively *aggregate* these trajectories. While voting and solution aggregation methods have proven effective for mathematical reasoning and coding (Wang et al., 2023; Fu et al., 2026; Zhao et al., 2025), long-horizon agentic tasks pose distinct challenges: evidence is sparse and distributed across multi-turn trajectories, demanding reasoning far beyond shallow heuristics and final solutions (Li et al., 2025a). Furthermore, individual trajectories may each capture only partial progress toward a complex task, requiring cross-trajectory synthesis to assemble a complete solution (Chang et al., 2026). However, existing approaches fall short (Table 1): aggregating only final solutions discards the rich information in trajectories, summarizing trajectories is expensive and lossy, and concatenating all trajectories into a single context is infeasible, as each trajectory spans hundreds of thousands of tokens.

To address these challenges, we propose **AggAgent**, which frames aggregation itself as an agentic task, treating the set of trajectories as an environment to interact with (Figure 2). AggAgent interacts with trajectories on demand through three lightweight tools: `get_solution` (retrieve final solutions of one or all trajectories), `search_trajectory` (keyword search within a trajectory), and `get_segment` (read a specific range of steps). Since these tools operate entirely over an array of completed trajectories—rather than invoking external services like web search or code execution—they incur none of the API cost or latency of such external tools. This agentic aggregation allows for cross-trajectory reasoning and synthesis without the cost of loading all trajectories into context, preserving full fidelity while keeping the aggregation cost bounded by a single agentic rollout.

We evaluate AggAgent on six agentic search and deep research benchmarks across three model families (GLM-4.7, Qwen3.5, MiniMax-M2.5). AggAgent consistently outperforms all

| Criterion | MV | WMV | BoN | FewTool | SolAgg | SummAgg | AggAgent (Ours) |
|---|---|---|---|---|---|---|---|
| **Task-agnostic** | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Non-heuristic** | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ |
| **Trajectory Info.** | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ |
| **Full Fidelity** | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |
| **Aggregation Cost** | Zero | Zero | Zero | Zero | Low | High | Low |

Table 1: **Comparison of aggregation methods**. **Task-agnostic**: not restricted to single short answers, applicable to multi-answer and long-form tasks. **Non-heuristic**: does not rely on shallow signals such as frequency or confidence. **Trajectory Info.**: reasons over full trajectories beyond final solutions. **Full Fidelity**: accesses trajectories without compression or information loss. **Aggregation Cost**: additional inference cost beyond the rollouts.

baselines (Figure 1), improving performance by up to 5.3 points on average and 10.3 points on deep research tasks. Furthermore, AggAgent is Pareto-optimal in cost and performance (Figure 3), adding only a 5.7% overhead over the cost of running 8 parallel agents—compared to 41% for Summary Aggregation—by selectively reading portions of trajectories rather than loading them entirely into context. Notably, AggAgent even surpasses Pass@8 (Section 6.1), demonstrating that effective aggregation can synthesize correct solutions beyond what any single rollout achieves. To summarize, our results show that agentic aggregation is a scalable and effective paradigm for test-time scaling in long-horizon tasks. We demonstrate that AggAgent is effective with off-the-shelf LLMs, and our framework further opens a promising direction for training aggregator agents.

## 2 Problem Formulation

We study long-horizon tasks such as agentic search and deep research. Given a problem $q$, the goal is to produce an output $y$, which may be either a short answer, a set of answers, or a long-form report. An agent $A$ interacts with an external environment to solve $q$ by generating a trajectory $T = (q, r_1, a_1, o_1, \ldots, r_m, a_m, o_m, y)$, where each step consists of internal thinking $r_j$, tool call $a_j$, and resulting observation $o_j$. In parallel scaling, we run the same agent independently for $K$ times, yielding $\mathcal{T} = \{T_1, T_2, \ldots, T_K\}$. This setting is natural in production deployments of frontier systems (Anthropic, 2025; Cursor, 2025; Kimi Team et al., 2026), as the latency is determined by the slowest of the $K$ rollouts.

The core problem centers around how to aggregate these $K$ trajectories and synthesize a better solution $y$. We formalize aggregation as a function $f : (q, \mathcal{T}) \mapsto \hat{y}$, where $\hat{y}$ is the aggregated final solution. For agentic search, $\hat{y}$ is evaluated against the ground-truth answer $y^*$, while for deep research, it is evaluated against problem-specific rubrics.

**Existing methods** We adopt the following aggregation methods from prior works:

- **Majority Voting** (MV) (Wang et al., 2023): selects the most frequent solution. However, voting methods do not work for multi-answer or long-form generation tasks.
- **Best-of-N** (BoN) (Cobbe et al., 2021; Uesato et al., 2022): selects the solution with the highest self-reported confidence. While reward model scores are commonly used as the weight for CoT reasoning tasks, recent works on agentic tasks have adopted the model's self-reported confidence as the weight (Wei et al., 2025; Zhu et al., 2026).
- **Weighted Majority Voting** (WMV) (Li et al., 2023; Wu et al., 2024): selects the solution with the highest total weight, where each solution is weighted by self-reported confidence.
- **Fewest Tool Calls** (FewTool) (Liu et al., 2025; Lu et al., 2025): selects the solution from the trajectory that required the fewest tool calls.
- **Solution Aggregation** (SolAgg) (Qi et al., 2025; Zhao et al., 2025; Qiao et al., 2025): concatenates all $K$ solutions and prompts an LLM to synthesize a final solution.
- **Summary Aggregation** (SummAgg) (Li et al., 2025a; Hu et al., 2026): compresses each trajectory into a summary report, concatenates all reports, and prompts an LLM to generate a final solution.

We classify these baselines into two groups. *Heuristic methods* (MV, WMV, BoN, FewTool) rely on shallow signals such as answer frequency, confidence scores, or trajectory length, without inspecting trajectory content. *LLM-based methods* (SolAgg, SummAgg) leverage LLM reasoning for solution synthesis but remain limited: Solution Aggregation aggregates only final solutions, discarding all intermediate evidence, while Summary Aggregation compresses each trajectory into a summary report before aggregation, incurring irreversible information loss and requiring $K$ additional LLM calls. Table 1 summarizes these limitations.

## 3 Our Approach: AggAgent

To address the aforementioned limitations, we propose AggAgent, where the aggregator interacts with $\mathcal{T}$ directly in an agentic manner. The initial user message provides the problem $q$ and trajectory metadata, which consists of the number of steps, total token count, and tool usage statistics for each trajectory $T_i$. The trajectories themselves are not pre-loaded into the context but reside in the environment and are retrieved on-demand, keeping the aggregation cost bounded by a single context window independent of $K$. This design enables cross-trajectory reasoning at full fidelity and low cost (Table 1).

**Tool design** We provide AggAgent with four tools as follows[2]:

- **`get_solution(traj_id)`**: Retrieves the final solution from each trajectory's last step. Unless traj_id is specified, returns all $K$ solutions by default.
- **`search_trajectory(traj_id, query, role, k)`**: Searches for keywords within a single trajectory. Returns the top matching $k$ steps ranked by ROUGE-L score (Lin, 2004).
- **`get_segment(traj_id, start_step, end_step)`**: Reads the full content of a contiguous range of steps from a single trajectory. Returns raw thinking and tool observations for the specified window.
- **`finish()`**: Submits the final solution along with a reason for the aggregation.

**Workflow** The tool design reflects a natural coarse-to-fine investigative workflow. Rather than reading every trajectory in full, the agent first surveys the metadata and solutions loaded by `get_solution` to identify consensus and disagreements across the rollouts, and to pinpoint trajectories worth closer inspection. It then selectively dives into individual trajectories via `search_trajectory` and `get_segment` to verify key claims against thinking blocks and tool observations. After sufficient cross-trajectory validation, it calls `finish` to submit the final synthesized solution. We provide the full prompts in Appendix B.

**Cost analysis** Aggregation cost is measured in additional LLM calls beyond the $K$ rollouts. Heuristic methods require zero additional calls, while Solution Aggregation requires a single LLM call to synthesize the $K$ solution candidates. Summary Aggregation is the most expensive, as it requires $K$ separate LLM calls before aggregation, each of which can be as long as the maximum context window. In contrast, AggAgent's total context is bounded by a single context window of the model, keeping aggregation cost independent of $K$ and comparable to Solution Aggregation. Moreover, as AggAgent's tools operate entirely over the in-memory trajectory array, they incur none of the latency or cost of external tools like web search and code execution.

## 4 Experimental Setup

**Tasks** We evaluate on six long-horizon agentic tasks broadly in two categories: 1) **Agentic search**: **BrowseComp** (Wei et al., 2025) consists of challenging factual questions demanding exhaustive multi-step web browsing; **BrowseComp-Plus** (Chen et al., 2025) enables controlled evaluation on BrowseComp questions by replacing web search with a local knowledge base; **HLE** (Phan et al., 2025) covers expert-level questions across diverse academic disciplines, emphasizing rigorous reasoning; **DeepSearchQA** (Gupta et al., 2026)

[2]We provide the full tool descriptions in Figure 18 in the appendix.

targets multi-answer queries where completeness across all valid answers is required. 2) **Deep research**: **Healthbench-Hard** (Arora et al., 2025) requires generating comprehensive long-form responses to challenging medical queries; **ResearchRubrics** (Sharma et al., 2025) poses open-ended research tasks evaluated against detailed, multi-criterion rubrics.

Due to high costs of long-horizon rollouts, we randomly sample subsets for evaluation[3]. For HLE, we use 155 search-focused questions (Li et al., 2025a). For ResearchRubrics, we use the full 101 problems. For Healthbench-Hard, we randomly sample 100 instances. For all other tasks, we randomly sample 150 questions following Sun et al. (2025); Zhang et al. (2025a).

**Models** We employ three model families of varying sizes for base trajectory rollouts: GLM-4.7-Flash (30B) (Zeng et al., 2025), Qwen3.5-122B-A10B (122B) (Qwen Team, 2026), and MiniMax-M2.5 (229B) (MiniMax, 2026). We adopt Tongyi DeepResearch (Tongyi DeepResearch Team et al., 2025) as the agent scaffold, use native function calling, allow at most 128K context length with 100 tool calls, and sample 8 independent trajectories. For LLM-based aggregation, we use the same model as the rollout agent.

**Rollout agent tools** For BrowseComp-Plus, we follow the official implementation and provide `search` and `get_document` tools using Qwen3-Embedding-8B (Zhang et al., 2025b) as the retriever. For all other tasks, we provide `search` and `visit` tools following prior work (Yen et al., 2026; Li et al., 2025b; Wu et al., 2025; Gao et al., 2026), where `search` uses Serper API[4] for Google search and `visit` uses crawl4ai[5] to scrape a specific web page.

**Cost and latency** We report total cost and latency per query. Cost includes rollout, tool calls, and aggregation, while latency includes rollout and aggregation.

**Evaluation** We employ LLM-as-a-judge, following the official evaluation setting. Specifically, we use Qwen3.5-397B-A17B for ResearchRubrics and GPT-4.1 (Achiam et al., 2023) for all other datasets. We perform bootstrapped sampling for calculating Metric@$K \in \{1, 2, 4, 8\}$. For BrowseComp, BrowseComp-Plus, and HLE, the predicted answer $\hat{y}$ is evaluated against the gold answer $y^*$. For DeepSearchQA, the predicted answer set is considered correct if it exactly matches the gold answer set. For Healthbench-Hard and ResearchRubrics, the long-form response is evaluated across multiple problem-specific rubrics, and the scores are averaged. The rubrics contain negative scores, hence the total score could also be negative.

We provide further experimental details in Appendix A.

## 5 Results

### 5.1 Main Results

We present the main results on $K = 8$ in Table 2, and present the full results ($K \in \{1, 2, 4, 8\}$) in Appendix C.1. AggAgent consistently outperforms all baselines across all three models, improving over Pass@1 by 13.3–17.9 points and over the strongest baseline (SolAgg) by 2.4–5.3 points on average.

Among the heuristic baselines, Majority Voting performs the worst, while confidence-based methods (WMV, BoN) yield substantial improvements, consistent with prior observations (Wei et al., 2025). However, the gains are less pronounced on DeepSearchQA, Healthbench-Hard, and ResearchRubrics, where the model is poorly calibrated (see Appendix C.4 for details). Notably, voting methods are not applicable to multi-answer or long-form tasks, so their performance gains are limited to single-answer tasks only.[6] Fewest Tool Calls has been popular on BrowseComp due to its simplicity (Liu et al., 2025; Lu et al., 2025), but our results show that it also does not generalize well beyond such tasks.

[3] We report standard deviation in Appendix C.1.

[4] https://serper.dev

[5] https://github.com/unclecode/crawl4ai

[6] We use Pass@1 instead to calculate the average score.

| | Task | Pass@1 | Heuristic Aggregation@8 | | | | LLM-based Aggregation@8 | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | MV | WMV | BoN | FewTool | SolAgg | SummAgg | AggAgent |
| GLM-4.7-Flash | BrowseComp | 27.42 | 32.67 | 50.67 | 51.33 | 41.33 | 53.33 | 55.33 | **56.00** |
| | BrowseComp-Plus | 49.08 | 59.33 | 68.67 | 68.67 | 62.00 | 70.67 | 70.67 | **71.33** |
| | HLE | 25.00 | 26.45 | 27.74 | 30.97 | 27.10 | 32.90 | 34.84 | **37.42** |
| | DeepSearchQA | 32.42 | – | – | 35.33 | 33.33 | 46.00 | 47.33 | **49.33** |
| | Healthbench-Hard | 8.67 | – | – | 9.91 | 8.90 | 15.72 | 7.35 | **27.99** |
| | ResearchRubrics | 37.47 | – | – | 37.70 | 35.21 | 36.84 | 31.72 | **45.31** |
| | **Average** | 30.01 | 32.84 | 37.61 | 38.99 | 34.65 | 42.58 | 41.21 | **47.90** |
| Qwen3.5-122B | BrowseComp | 40.25 | 47.33 | 60.00 | 58.67 | 56.00 | 61.33 | 62.00 | **62.67** |
| | BrowseComp-Plus | 58.67 | 66.67 | 69.33 | 74.00 | 70.67 | 73.33 | 73.33 | **74.67** |
| | HLE | 39.35 | 43.87 | 45.16 | 53.55 | 41.29 | 50.97 | 50.32 | **54.19** |
| | DeepSearchQA | 49.25 | – | – | 57.33 | 54.67 | 62.67 | 64.00 | **66.00** |
| | Healthbench-Hard | 12.87 | – | – | 13.01 | 12.83 | 26.30 | 23.00 | **28.06** |
| | ResearchRubrics | 40.50 | – | – | 42.37 | 39.58 | 42.10 | 37.47 | **49.36** |
| | **Average** | 40.15 | 43.42 | 46.19 | 49.82 | 45.84 | 52.78 | 51.69 | **55.83** |
| MiniMax-M2.5 | BrowseComp | 50.17 | 60.67 | 66.67 | 70.67 | 67.33 | 70.67 | 70.00 | **71.33** |
| | BrowseComp-Plus | 64.17 | 68.67 | 75.33 | 76.00 | 74.67 | 76.67 | 76.67 | **77.33** |
| | HLE | 45.73 | 53.55 | 57.42 | 58.06 | 46.45 | 54.19 | 58.06 | **60.00** |
| | DeepSearchQA | 54.42 | – | – | 64.00 | 56.00 | 62.00 | 61.33 | **65.33** |
| | Healthbench-Hard | 9.67 | – | – | 8.79 | 5.34 | 21.84 | 16.92 | **24.46** |
| | ResearchRubrics | 39.97 | – | – | 39.00 | 38.44 | 44.00 | 40.29 | **45.42** |
| | **Average** | 44.02 | 47.83 | 50.58 | 52.75 | 48.04 | 54.90 | 53.88 | **57.31** |

Table 2: **AggAgent performs the best across all settings.** We evaluate on six long-horizon tasks using three model families, each generating $K$=8 independent rollouts. For LLM-based aggregation methods, the same model is used for both rollout and aggregation. Best metric per row is bolded. – denotes that voting-based methods are not applicable, and we use Pass@1 instead to calculate the average score.

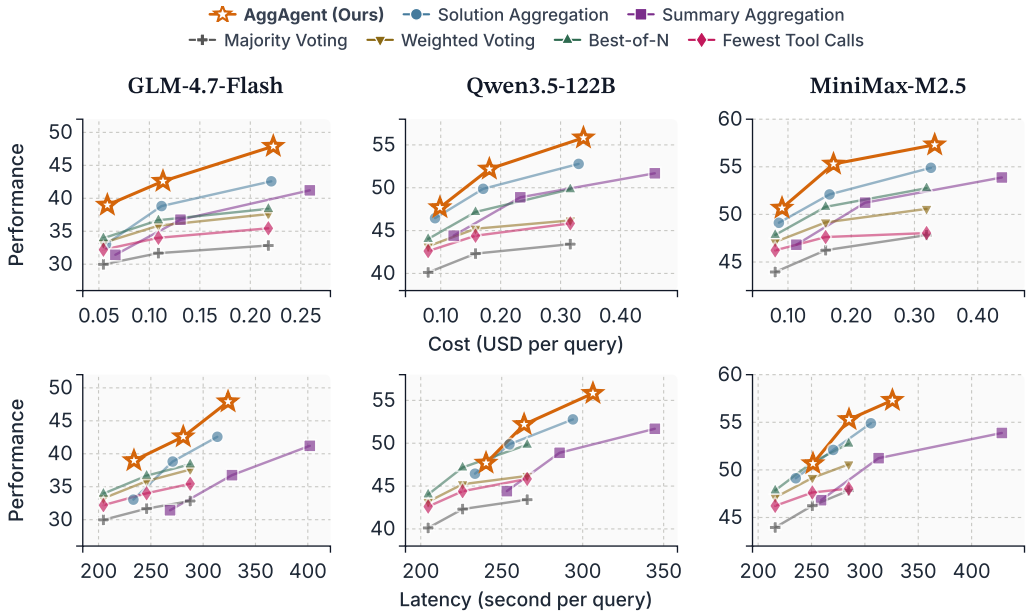


Figure 3: **AggAgent achieves a Pareto-optimal performance–efficiency tradeoff**. We compare aggregation methods at $K \in \{2, 4, 8\}$ parallel samples, averaging performance across six benchmarks, with the same model serving as both rollout agent and aggregator. For each model, the top chart plots average cost (USD per query) vs. performance and the bottom chart plots average latency (seconds per query) vs. performance.

Meanwhile, LLM-based methods consistently outperform heuristic baselines. Summary Aggregation outperforms all other baselines on agentic search tasks, as access to full trajectory evidence aids answer verification. However, it falls significantly behind Solution Aggrega-

tion on deep research tasks, as trajectory compression harms the detail and coherence of long-form responses. While no single baseline is universally optimal, AggAgent achieves strong performance by enabling fine-grained reasoning over trajectories without sacrificing detail or coherence.

### 5.2 Cost and Latency vs. Performance

We compare cost and latency versus performance for different numbers of samples $K \in \{2, 4, 8\}$ in Figure 3 (full breakdown in Appendix C.2). As $K$ increases, all methods become more expensive and slower due to additional rollouts. AggAgent and Solution Aggregation introduce only minimal overhead beyond heuristic baselines—at $K$=8, their overhead over the rollout cost is 5.7% and 3.7%, respectively—yet substantially outperform them. Interestingly, their overhead does not increase proportionally to $K$, suggesting that having more samples can sometimes make aggregation easier by providing stronger supporting evidence, partially offsetting the extra computation spent on harder instances. In contrast, Summary Aggregation incurs 41% overhead at $K$=8 because it must compress each trajectory before aggregation. Overall, AggAgent achieves Pareto-optimal performance and efficiency across benchmarks, showing that agentic aggregation scales efficiently with test-time compute without incurring substantial additional cost.

## 6 Analysis and Discussion

### 6.1 Stronger Models for Aggregation

Could aggregation benefit from employing a stronger model? We investigate this by replacing the aggregator with MiniMax-M2.5 while keeping GLM-4.7-Flash as the base rollout agent. As shown in Figure 4 (full results in Table 15 in the appendix), using a stronger aggregator generally improves performance, with MiniMax-based AggAgent achieving the highest average score and even surpassing Pass@8 on BrowseComp-Plus. This points to *asymmetric model allocation*—a stronger model for aggregation, weaker models for parallel rollouts—as a practical strategy for designing multi-agent systems, consistent with prior work employing stronger models as orchestrators over multiple weaker subagents (Qiao et al., 2025; Zhang et al., 2025a; Akay et al., 2026). In our setting, the aggregator plays the orchestrating role, reasoning across parallel trajectories to produce a final solution.

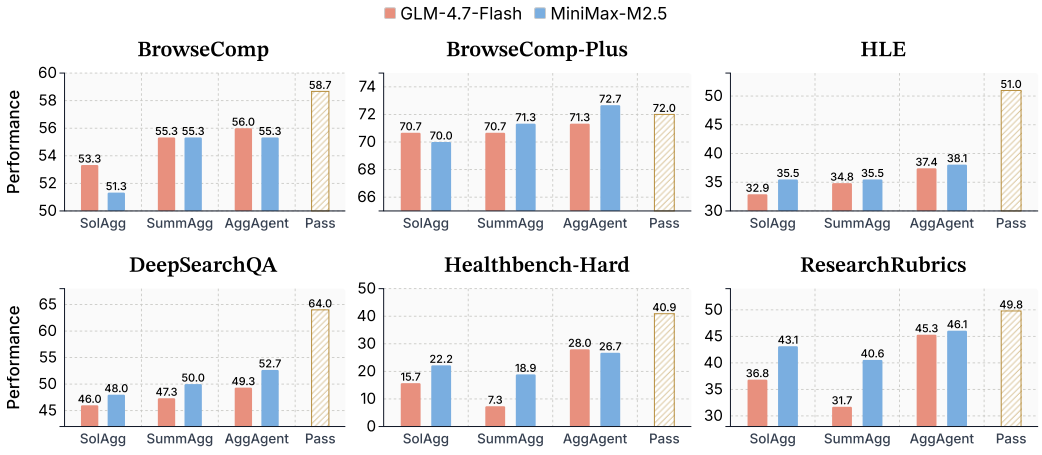


Figure 4: **Employing a stronger aggregator improves performance on LLM-based aggregation methods**. In all cases, GLM-4.7-Flash serves as the rollout agent; blue bars replace the aggregator with the stronger MiniMax-M2.5, while red bars use GLM-4.7-Flash for both roles. Yellow hatched bars denote Pass@$K$. All methods are evaluated at $K$=8.

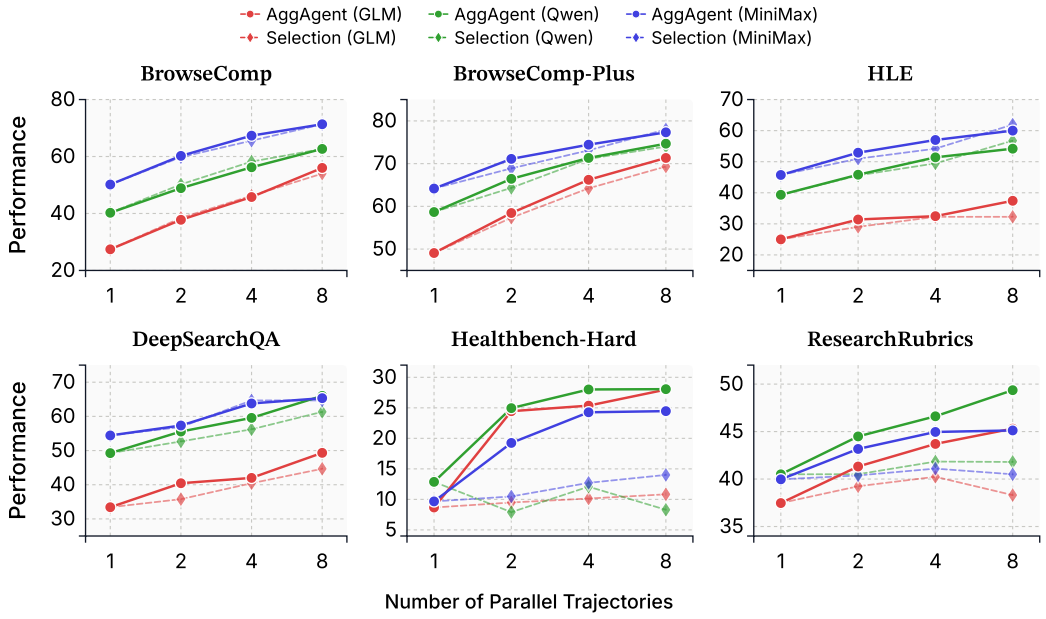


Figure 5: **Ablation of solution synthesis vs. best-trajectory selection**. AggAgent synthesizes a new solution from the collected trajectories; the selection variant selects the single best trajectory directly. Color indicates model (red: GLM-4.7-Flash, green: Qwen3.5-122B, blue: MiniMax-M2.5); line style and marker indicate method (solid + circle: AggAgent, dashed + diamond: selection variant).

### 6.2 Output Design Ablation: Synthesis vs. Selection

We ablate the output design of AggAgent by comparing it against a selection variant, which selects the single best trajectory and adopts its solution directly instead of synthesizing a new solution. As shown in Figure 5, AggAgent performs better overall, though the selection variant remains competitive in some settings. Notably, the selection variant performs poorly on deep research benchmarks, where AggAgent outperforms it by a large margin. We attribute this to a mismatch between selection and the nature of open-ended research tasks: quality is distributed across trajectories, so no single trajectory dominates, yet selection forces an all-or-nothing commitment. Synthesis mitigates this by making local, compositional judgments about which parts of each trajectory to incorporate, rather than committing to a global winner. This distinction is less pronounced for agentic search, where trajectories tend to be clearly correct or clearly wrong, making selection more straightforward. These results suggest that synthesis is often the preferable output design, consistent with our findings in Section 6.1 and Appendix D.4 that AggAgent can compose a correct solution from individually incorrect trajectories. We leave a more thorough ablation of output design to future work.

### 6.3 Tool Design Ablations

**Search strategy variants** We ablate the retrieval backend used by the `search_trajectory` tool by comparing ROUGE-L with BM25 and Qwen3-Embedding-8B. As shown in Table 3(a), performance differences are small, and no retrieval strategy consistently dominates across model families. This suggests that AggAgent's gains are not driven by a particular search backend. We hypothesize that dense embedding retrieval is not uniformly advantageous for trajectory search because aggregation often requires locating exact strings, localized evidence, or tool-returned facts rather than broad semantic matches. The retrieval backend acts as a design tradeoff among retrieval quality, latency, and implementation cost, rather than as a determinant of AggAgent's effectiveness.

**(a) Search strategy variants**

| | ROUGE-L | BM25 | Qwen3-Emb. |
|---|---|---|---|
| GLM-4.7-Flash | **47.90** | 47.37 | 45.41 |
| Qwen3.5-122B | **55.83** | 54.92 | 55.21 |
| MiniMax-M2.5 | 57.31 | 57.89 | **57.91** |

**(b) Disabling thinking traces**

| | Full | No Think. |
|---|---|---|
| GLM-4.7-Flash | 47.90 | 45.13 $_{(-2.77)}$ |
| Qwen3.5-122B | 55.83 | 54.58 $_{(-1.25)}$ |
| MiniMax-M2.5 | 57.31 | 57.28 $_{(-0.03)}$ |

Table 3: **Tool design ablations.** Scores are averaged across six benchmarks at $K$=8 and the same model is used for both rollout generation and aggregation. Table 3(a) compares retrieval backends for search_trajectory: ROUGE-L, BM25, and Qwen3-Embedding-8B; bold indicates the best retrieval backend in each row. Table 3(b) compares the default full-trajectory setting (Full) with a variant that restricts access to internal thinking traces (No Think.); red subscripts show the score change relative to Full.

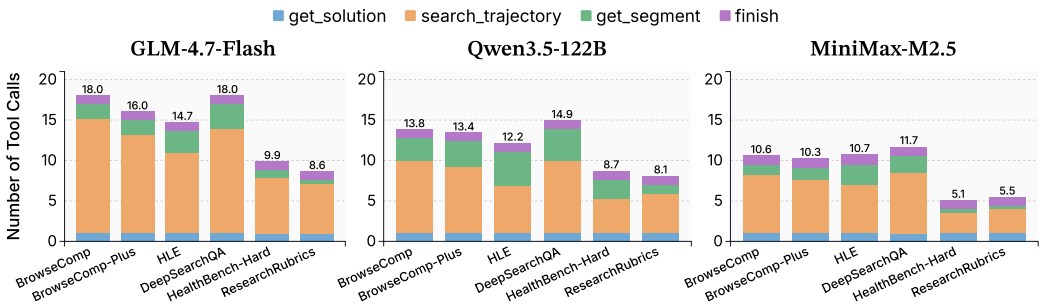


Figure 6: **Average number of tool calls per query by AggAgent.** Numbers above each bar indicate the average total tool calls per query. search_trajectory dominates tool usage, while get_solution and finish are each called approximately once per query. get_segment is used more selectively, reflecting a coarse-to-fine strategy where AggAgent commits to full-content reads only when keyword-level search is insufficient.

**Disabling thinking traces** The default AggAgent environment exposes full trajectories, including internal thinking traces, as our rollout agents generate complete execution logs. However, our method is also compatible with systems that do not return thinking traces. To isolate this factor, we evaluate a variant in which AggAgent cannot access internal reasoning text. As shown in Table 3(b), removing thinking traces causes only a small drop in average performance, and the resulting variant still outperforms all the baselines in Table 2. This indicates that AggAgent's gains primarily come from access to verifiable trajectory evidence rather than dependence on internal reasoning traces. Stronger models remain more robust, suggesting that they can aggregate effectively from tool observations alone.

### 6.4 Tool Usage Patterns

In this section, we analyze the tool usage behaviour of AggAgent. As depicted in Figure 6, `search_trajectory` accounts for the majority of tool calls, while `get_solution` and `finish` are each invoked approximately once per query, consistent with the intended workflow: solutions are surveyed once at the start and the agent terminates after the final synthesis. Meanwhile, `get_segment` is used less often than `search_trajectory`, suggesting that AggAgent reserves full-content reads for cases where keyword-level search alone is insufficient, diving deep only when actually needed. We also observe that stronger models tend to require fewer tool calls overall, suggesting more capable models reach a final solution more efficiently. Deep research tasks similarly exhibit fewer tool calls, potentially due to less need for resolving precise factual conflicts across trajectories.

## 7 Related Work

**Long-horizon agents** LLM agents have emerged as a powerful paradigm, extending language models beyond pure parametric generation to interactive problem solving in external environments. ReAct (Yao et al., 2022b) has been the most prominent workflow, interleaving reasoning, action selection, and environment observations. This has enabled a wide range of applications, including agentic RAG (Jin et al., 2025b; Li et al., 2025c; Jin et al., 2025a), software engineering (Jimenez et al., 2024; Yang et al., 2024; Wang et al., 2025), web navigation (Yao et al., 2022a; Chae et al., 2025; Gandhi & Neubig, 2026), and deep research systems (OpenAI, 2025; Anthropic, 2025; Roucher et al., 2025). Recent work has sought to improve agentic capabilities through advances in RL algorithms (Shao et al., 2024; 2025; Ritter et al., 2026), agentic mid-training (Su et al., 2025), and environment scaling (Fang et al., 2025). Our work is complementary to these efforts: we treat these agents as rollout models and study how to aggregate parallel trajectories into a single high-quality solution.

**Test-time scaling** Early work on test-time scaling for CoT tasks (Kojima et al., 2022; Wei et al., 2022) such as math reasoning and coding showed clear gains from search-based strategies (Yao et al., 2023), extended thinking budgets (Muennighoff et al., 2025; Guo et al., 2025; Snell et al., 2025), majority voting (Wang et al., 2023; Brown et al., 2024), and learning-based aggregation (Qi et al., 2025; Zhao et al., 2025).

However, test-time scaling for long-horizon agents is more challenging, as trajectories are lengthy, heterogeneous, and difficult to compare using simple surface-level signals (e.g., majority or confidence). Parallel scaling with LLM-based aggregation mitigates this problem by generating multiple independent trajectories and reasoning over them to produce a final solution. WebResearcher (Qiao et al., 2025) concatenates final answers and prompts an LLM to synthesize a solution, while ParallelMuse (Li et al., 2025a) compresses each trajectory into a summary report before aggregation, incurring high cost and information loss. Concurrent with our work, KARL (Chang et al., 2026) feeds final answers back to the rollout agent with the same tools enabled, but it disregards full trajectories, and reusing rollout tools can be costly in practice. In contrast, AggAgent enables the aggregator to interact with trajectories in an agentic manner, supporting cross-trajectory reasoning without lossy compression or excessive cost. While AggAgent is training-free and effective with off-the-shelf LLMs, we leave fine-tuning the aggregator as a promising direction for future work.

Orthogonal to our focus on parallel scaling, several works have studied sequential scaling to extend the effective horizon of a single agent through context management (Wu et al., 2025; Yen et al., 2026; Tang et al., 2025; Zeng et al., 2026), context folding (Sun et al., 2025; Ye et al., 2026), or iterative refinement (Zhu et al., 2026; Xiao et al., 2026). These two directions could be combined in practice, for example, by using context management within each rollout and then applying AggAgent across the resulting trajectories (Appendix D.2).

## 8 Conclusion

We introduce AggAgent, an aggregation agent for parallel scaling of long-horizon agentic tasks. By treating parallel trajectories as an interactive environment and navigating them via lightweight in-memory tools, AggAgent enables cross-trajectory reasoning at full fidelity, while avoiding both the information loss of compression and the prohibitive cost of loading all trajectories into context. Across six benchmarks and three models, AggAgent consistently outperforms all baselines, achieving Pareto-optimal performance and efficiency. Our findings establish agentic aggregation as a principled and cost-efficient paradigm for parallel test-time scaling.

## Acknowledgments

We thank Jeffrey Cheng, Simon Park, and all Princeton NLP members for their helpful discussions and feedback. This work is gratefully supported by an NSF CAREER award (IIS-2239290).

## References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Yagiz Can Akay, Muhammed Yusuf Kartal, Esra Alparslan, Faruk Ortakoyluoglu, and Arda Akpinar. SPD-RAG: Sub-agent per document retrieval-augmented generation. *arXiv preprint arXiv:2603.08329*, 2026.

Anthropic. How we built our multi-agent research system, June 2025. URL https://www.anthropic.com/engineering/multi-agent-research-system.

Rahul K Arora, Jason Wei, Rebecca Soskin Hicks, Preston Bowman, Joaquin Quiñonero-Candela, Foivos Tsimpourlas, Michael Sharman, Meghan Shah, Andrea Vallone, Alex Beutel, et al. HealthBench: Evaluating large language models towards improved human health. *arXiv preprint arXiv:2505.08775*, 2025.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Hyungjoo Chae, Sunghwan Kim, Junhee Cho, Seungone Kim, Seungjun Moon, Gyeom Hwangbo, Dongha Lim, Minjin Kim, Yeonjun Hwang, Minju Gwak, Dongwook Choi, Minseok Kang, Gwanhoon Im, ByeongUng Cho, Hyojun Kim, Jun Hee Han, Taeyoon Kwon, Minju Kim, Beong woo Kwak, Dongjin Kang, and Jinyoung Yeo. Web-Shepherd: Advancing PRMs for reinforcing web agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=G2kMroO9UV.

Jonathan D Chang, Andrew Drozdov, Shubham Toshniwal, Owen Oertell, Alexander Trott, Jacob Portes, Abhay Gupta, Pallavi Koppol, Ashutosh Baheti, Sean Kulinski, et al. KARL: Knowledge agents via reinforcement learning. *arXiv preprint arXiv:2603.05218*, 2026.

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, et al. BrowseComp-Plus: A more fair and transparent evaluation benchmark of deep-research agent. *arXiv preprint arXiv:2508.06600*, 2025.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems, 2021. *URL https://arxiv. org/abs/2110.14168*, 9, 2021.

Cursor. Introducing cursor 2.0 and composer, October 2025. URL https://cursor.com/blog/2-0.

Runnan Fang, Shihao Cai, Baixuan Li, Jialong Wu, Guangyu Li, Wenbiao Yin, Xinyu Wang, Xiaobin Wang, Liangcai Su, Zhen Zhang, et al. Towards general agentic intelligence via environment scaling. *arXiv preprint arXiv:2509.13311*, 2025.

Yichao Fu, Xuewei Wang, Hao Zhang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=8LqHs0KIM7.

Apurva Gandhi and Graham Neubig. Go-Browse: Training web agents with structured exploration. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=IpzRWE52yw.

Jiaxuan Gao, Wei Fu, Minyang Xie, Shusheng Xu, Chuyi He, Zhiyu Mei, Banghua Zhu, and Yi Wu. Unlocking long-horizon agentic search with large-scale end-to-end RL. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=MfPDdPUGKi.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

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

Chen Hu, Haikuo Du, Heng Wang, Lin Lin, Mingrui Chen, Peng Liu, Ruihang Miao, Tianchi Yue, Wang You, Wei Ji, et al. Step-DeepResearch technical report. *arXiv preprint arXiv:2512.20491*, 2025.

Jingcheng Hu, Yinmin Zhang, Shijie Shang, Xiaobo Yang, Yue Peng, Zhewei Huang, Hebin Zhou, Xin Wu, Jie Cheng, Fanqi Wan, et al. PaCoRe: Learning to scale test-time compute with parallel coordinated reasoning. *arXiv preprint arXiv:2601.05593*, 2026.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Bowen Jin, Jinsung Yoon, Priyanka Kargupta, Sercan O Arik, and Jiawei Han. An empirical study on reinforcement learning for reasoning-search interleaved LLM agents. *arXiv preprint arXiv:2505.15117*, 2025a.

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 *Second Conference on Language Modeling*, 2025b. URL https://openreview.net/forum?id=Rwhi91ideu.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, SH Cai, Yuan Cao, Y Charles, HS Che, Cheng Chen, Guanduo Chen, et al. Kimi K2.5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

Baixuan Li, Dingchu Zhang, Jialong Wu, Wenbiao Yin, Zhengwei Tao, Yida Zhao, Liwen Zhang, Haiyang Shen, Runnan Fang, Pengjun Xie, et al. ParallelMuse: Agentic parallel thinking for deep information seeking. *arXiv preprint arXiv:2510.24698*, 2025a.

Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baixuan Li, Zhengwei Tao, Xinyu Wang, et al. WebSailor: Navigating super-human reasoning for web agent. *arXiv preprint arXiv:2507.02592*, 2025b.

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. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 5420–5438, 2025c.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. Making language models better reasoners with step-aware verifier. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5315–5333, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.291. URL https://aclanthology.org/2023.acl-long.291/.

Zhuofeng Li, Dongfu Jiang, Xueguang Ma, Haoxiang Zhang, Ping Nie, Yuyu Zhang, Kai Zou, Jianwen Xie, Yu Zhang, and Wenhu Chen. OpenResearcher: A Fully Open Pipeline for Long-Horizon Deep Research Trajectory Synthesis. *arXiv preprint arXiv:2603.20278*, 2026.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013/.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. DeepSeek-V3.2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025.

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.

MiniMax. MiniMax-M2.5, February 2026. URL https://huggingface.co/MiniMaxAI/MiniMax-M2.5.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20286–20332, 2025.

OpenAI. Introducing deep research, February 2025. URL https://openai.com/index/introducing-deep-research/.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, et al. Humanity's last exam. *arXiv preprint arXiv:2501.14249*, 2025.

Jianing Qi, Xi Ye, Hao Tang, Zhigang Zhu, and Eunsol Choi. Learning to reason across parallel samples for LLM reasoning. *arXiv preprint arXiv:2506.09014*, 2025.

Zile Qiao, Guoxin Chen, Xuanzhong Chen, Donglei Yu, Wenbiao Yin, Xinyu Wang, Zhen Zhang, Baixuan Li, Huifeng Yin, Kuan Li, et al. WebResearcher: Unleashing unbounded reasoning capability in long-horizon agents. *arXiv preprint arXiv:2509.13309*, 2025.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Daniel Ritter, Owen Oertell, Bradley Guo, Jonathan Chang, Kianté Brantley, and Wen Sun. LLMs can learn to reason via off-policy RL. *arXiv preprint arXiv:2602.19362*, 2026.

Aymeric Roucher, Albert Villanova del Moral, Merve Noyan, Thomas Wolf, and Clémentine Fourrier. Open-source deepresearch - freeing our search agents, February 2025. URL https://huggingface.co/blog/open-deep-research.

Rulin Shao, Akari Asai, Shannon Zejiang Shen, Hamish Ivison, Varsha Kishore, Jingming Zhuo, Xinran Zhao, Molly Park, Samuel G Finlayson, David Sontag, et al. Dr. Tulu: Reinforcement learning with evolving rubrics for deep research. *arXiv preprint arXiv:2511.19399*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Manasi Sharma, Chen Bo Calvin Zhang, Chaithanya Bandi, Clinton Wang, Ankit Aich, Huy Nghiem, Tahseen Rabbani, Ye Htet, Brian Jang, Sumana Basu, et al. ResearchRubrics: A benchmark of prompts and rubrics for evaluating deep research agents. *arXiv preprint arXiv:2511.07685*, 2025.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=4FWAwZtd2n.

Liangcai Su, Zhen Zhang, Guangyu Li, Zhuo Chen, Chenxi Wang, Maojia Song, Xinyu Wang, Kuan Li, Jialong Wu, Xuanzhong Chen, et al. Scaling agents via continual pre-training. *arXiv preprint arXiv:2509.13310*, 2025.

Weiwei Sun, Miao Lu, Zhan Ling, Kang Liu, Xuesong Yao, Yiming Yang, and Jiecao Chen. Scaling long-horizon llm agent via context-folding. *arXiv preprint arXiv:2510.11967*, 2025.

Qiaoyu Tang, Hao Xiang, Le Yu, Bowen Yu, Yaojie Lu, Xianpei Han, Le Sun, WenJuan Zhang, Pengbo Wang, Shixuan Liu, et al. Beyond turn limits: Training deep search agents with dynamic context window. *arXiv preprint arXiv:2510.08276*, 2025.

Tongyi DeepResearch Team, Baixuan Li, Bo Zhang, Dingchu Zhang, Fei Huang, Guangyu Li, Guoxin Chen, Huifeng Yin, Jialong Wu, Jingren Zhou, et al. Tongyi deepresearch technical report. *arXiv preprint arXiv:2510.24701*, 2025.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. Openhands: An open platform for AI software developers as generalist agents. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=OJd3ayDDoF.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

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.

Sean Welleck, Amanda Bertsch, Matthew Finlayson, Hailey Schoelkopf, Alex Xie, Graham Neubig, Ilia Kulikov, and Zaid Harchaoui. From decoding to meta-generation: Inference-time algorithms for large language models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=eskQMcIbMS. Survey Certification.

Xixi Wu, Kuan Li, Yida Zhao, Liwen Zhang, Litu Ou, Huifeng Yin, Zhongwang Zhang, Xinmiao Yu, Dingchu Zhang, Yong Jiang, et al. ReSum: Unlocking long-horizon search intelligence via context summarization. *arXiv preprint arXiv:2509.13313*, 2025.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Scaling inference computation: Compute-optimal inference for problem-solving with language models. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*, 2024. URL https://openreview.net/forum?id=j7DZWSc8qu.

Teng Xiao, Yige Yuan, Hamish Ivison, Huaisheng Zhu, Faeze Brahman, Nathan Lambert, Pradeep Dasigi, Noah A. Smith, and Hannaneh Hajishirzi. Meta-reinforcement learning with self-reflection for agentic search. *arXiv preprint arXiv:2603.11327*, 2026.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022b.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

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

Howard Yen, Yoonsang Lee, Ashwin Paranjape, Mengzhou Xia, Thejas Venkatesh, Jack Hessel, Danqi Chen, and Yuhao Zhang. Lost in the maze: Overcoming context limitations in long-horizon agentic search. In *Third Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=k4FBtSjHa4.

Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, et al. GLM-4.5: Agentic, reasoning, and coding (ARC) foundation models. *arXiv preprint arXiv:2508.06471*, 2025.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chengxing Xie, Cunxiang Wang, et al. GLM-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

Alex L Zhang, Tim Kraska, and Omar Khattab. Recursive language models. *arXiv preprint arXiv:2512.24601*, 2025a.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, et al. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025b.

Wenting Zhao, Pranjal Aggarwal, Swarnadeep Saha, Asli Celikyilmaz, Jason Weston, and Ilia Kulikov. The majority is not always right: RL training for solution aggregation. *arXiv preprint arXiv:2509.06870*, 2025.

Jialiang Zhu, Gongrui Zhang, Xiaolong Ma, Lin Xu, Miaosen Zhang, Ruiqi Yang, Song Wang, Kai Qiu, Zhirong Wu, Qi Dai, et al. RE-TRAC: REcursive TRAjectory compression for deep search agents. *arXiv preprint arXiv:2602.02486*, 2026.

# A Experimental Details

## A.1 Implementation

We serve all models via vLLM (Kwon et al., 2023) and use the Tongyi DeepResearch scaffold (Tongyi DeepResearch Team et al., 2025) for the agentic workflow. We enable interleaved reasoning and native function calling, with temperature 1.0, top-$p$ 0.95, a 128K context window, 10K max output tokens, and at most 100 tool calls per rollout. Upon exhausting the context limit or tool call budget, the model is given one final opportunity to submit an answer without tool access. All experiments are conducted on 4×H100 GPUs.

## A.2 Rollout Agent Tools

**BrowseComp-Plus** The local corpus contains 100,195 documents. `search` uses Qwen3-Embedding-8B to retrieve the top-5 documents by score, with each snippet truncated to 512 tokens. `get_document` returns the full text of a document by docid, truncated to 4,096 tokens following Su et al. (2025).

- **`search(query)`**: Perform a search on a knowledge source. Returns top-5 hits with docid, score, and snippet. The snippet contains the document's contents (may be truncated based on token limits).
- **`get_document(docid)`**: Retrieve a full document by its docid.

**All other datasets** We adopt the tool implementation from SLIM (Yen et al., 2026). `search` queries Google Search via the Serper API (`https://serper.dev`), returning the top-10 results with snippets of at most 150 characters. `visit` scrapes a URL via crawl4ai (`https://github.com/unclecode/crawl4ai`) and extracts the snippet most relevant to a goal, ranked by ROUGE-L score (Lin, 2004).

- **`search(query)`**: Performs a web search: supply a string 'query'; the tool retrieves the top 10 results for the query.
- **`visit(url, goal)`**: Visit a webpage and return the relevant content based on the goal.

## A.3 Cost Calculation

Let $g_{in}$ and $g_{out}$ denote the per-token input and output costs of a language model, and let $g_s$ and $g_v$ denote the per-call costs of `search` and `visit`, respectively. The total cost per question is $c = c_r + c_t + c_a$.

**Rollout cost** ($c_r$): Following Yen et al. (2026), cached input tokens are excluded from the token count, as caching is standard in long-trajectory settings with shared context. The total rollout cost is $c_r = \sum_i \left( g_{in} \cdot \Delta t_{in}^{(i)} + g_{out} \cdot t_{out}^{(i)} \right)$, where $\Delta t_{in}^{(i)} = t_{in}^{(i)} - t_{in}^{(i-1)}$.

**Tool call cost** ($c_t$): For BrowseComp-Plus, which uses a local corpus, $c_t = 0$. For all other tasks, $c_t = g_s \cdot n_s + g_v \cdot n_v$, where $n_s$ and $n_v$ are the total numbers of `search` and `visit` calls.

**Aggregation cost** ($c_a$): Only LLM-based aggregation methods incur additional aggregation cost. For Solution Aggregation, $c_a$ is a single LLM call: $g_{in} \cdot t_{in} + g_{out} \cdot t_{out}$. For Summary Aggregation, this is summed over $K + 1$ calls. For AggAgent, $c_a$ is computed identically to $c_r$, as it forms an agentic trajectory.

Cost is averaged over all instances for each benchmark. Pricing rates are given in Table 4. Prices are sourced from the respective provider websites: GLM-4.7-Flash (`https://docs.z.ai/guides/overview/pricing`), Qwen3.5-122B (`https://openrouter.ai/qwen/qwen3.5-122b-a10b`), MiniMax-M2.5 (`https://platform.minimax.io/docs/guides/pricing-paygo`), and tool calls (`https://www.firecrawl.dev/pricing`).

| | Input | Output |
|---|---|---|
| *Language Models (per 1M tokens)* | | |
| GLM-4.7-Flash | \$0.07 | \$0.40 |
| Qwen-3.5-122B | \$0.26 | \$2.08 |
| MiniMax-M2.5 | \$0.30 | \$1.20 |
| *Tool Calls (per 1K queries)* | | |
| `search` | \$0.50 | |
| `visit` | \$0.83 | |

Table 4: Pricing used for cost calculation.

### A.4 Latency Measurement

We measure latency by processing one problem at a time, using 2×H100 for GLM-4.7-Flash and 4×H100 for Qwen3.5-122B and MiniMax-M2.5.

**Rollout latency** ($\ell_r$): Wall-clock time from the first step of the trajectory until termination.

**Aggregation latency** ($\ell_a$): For Solution Aggregation, the time of the single LLM call. For Summary Aggregation, we spawn $K$ summarization calls in parallel, wait for all to complete, then issue the final aggregation call; $\ell_a$ covers the full span. For AggAgent, $\ell_a$ is measured the same way as $\ell_r$.

For latency, we sample 30 instances and report the median for efficiency, as latency must be measured sequentially. All other experiments run with batched generation.

### A.5 Evaluation

We evaluate Metric@$K \in \{1, 2, 4, 8\}$ over $N = 8$ independent trajectories via bootstrapped sampling (Equation 1), where $Q$ is the dataset, $S$ is a bootstrap sample of size $K$, and $\mathcal{M}$ is the scoring function between the predicted solution $\hat{y}$ and ground truth $y^*$. For LLM-based aggregation, evaluating all $\binom{N}{K}$ combinations is prohibitively expensive, so we cap the number of combinations to 3.

$$\text{Metric@}K = \frac{1}{|Q|} \sum_{q \in Q} \left( \frac{1}{\binom{N}{K}} \sum_{\substack{S \subseteq \{1,\ldots,N\} \\ |S|=K}} \mathcal{M}(\hat{y}, y^*) \right) \tag{1}$$

### A.6 Rollout Trajectory Statistics

We provide the statistics of base rollout trajectories in Table 5.

## B Prompts

**Rollout agent** We use the system prompt from Tongyi DeepResearch (Tongyi DeepResearch Team et al., 2025) (Figure 12), with an additional instruction from Dr. Tulu (Shao et al., 2025) appended for deep research tasks (Figure 13). For agentic search tasks, we format the user message following BrowseComp (Wei et al., 2025) (Figure 14).

**LLM-as-a-Judge** For BrowseComp, BrowseComp-Plus, and HLE, we find that the original evaluation prompt occasionally produces false judgments, hence we instead use the prompt from Zhu et al. (2026). For ResearchRubrics, we find that the original prompt is unreliable for negative rubrics — consistent with Hu et al. (2025) — so we append a clarifying section to the original prompt (Figure 15). For DeepSearchQA and Healthbench-Hard, we use the official evaluation prompts.

| | Task | # Inst. | # Turns | Tokens (K) | Output (K) | # Tool Calls | Uniq. URLs |
|---|---|---|---|---|---|---|---|
| GLM-4.7-Flash | BrowseComp | 150 | $71.4_{0.9}$ | $64.8_{1.0}$ | $0.64_{0.05}$ | $70.4_{0.9}$ (59.0/11.4) | $324.6_{9.4}$ |
| | BrowseComp-Plus | 150 | $26.0_{0.8}$ | $72.0_{1.8}$ | $0.53_{0.01}$ | $25.1_{0.8}$ (23.1/2.0) | $50.1_{0.7}$ |
| | HLE | 155 | $34.4_{1.7}$ | $40.9_{2.2}$ | $1.39_{0.12}$ | $31.9_{1.5}$ (22.8/9.1) | $86.8_{5.1}$ |
| | DeepSearchQA | 150 | $53.4_{2.4}$ | $48.3_{1.5}$ | $0.71_{0.09}$ | $52.6_{2.4}$ (37.9/14.7) | $126.8_{3.9}$ |
| | Healthbench-Hard | 100 | $17.8_{1.6}$ | $27.0_{2.5}$ | $1.74_{0.07}$ | $16.9_{1.6}$ (7.9/9.0) | $52.5_{6.1}$ |
| | ResearchRubrics | 101 | $34.0_{1.8}$ | $48.2_{2.1}$ | $4.23_{0.11}$ | $33.1_{1.8}$ (15.9/17.2) | $121.4_{6.7}$ |
| Qwen3.5-122B | BrowseComp | 150 | $59.4_{1.0}$ | $61.7_{1.8}$ | $1.36_{0.19}$ | $58.7_{1.0}$ (52.2/6.5) | $349.2_{9.4}$ |
| | BrowseComp-Plus | 150 | $21.2_{0.3}$ | $60.2_{1.0}$ | $1.23_{0.11}$ | $20.4_{0.4}$ (18.8/1.7) | $38.7_{0.6}$ |
| | HLE | 155 | $26.5_{0.8}$ | $35.6_{1.0}$ | $1.86_{0.10}$ | $25.2_{0.8}$ (18.2/7.0) | $84.2_{2.8}$ |
| | DeepSearchQA | 150 | $40.4_{1.4}$ | $45.9_{1.6}$ | $1.04_{0.04}$ | $39.4_{1.4}$ (26.9/12.6) | $116.5_{3.9}$ |
| | Healthbench-Hard | 100 | $7.4_{0.2}$ | $12.1_{0.3}$ | $1.71_{0.03}$ | $6.4_{0.2}$ (2.3/4.1) | $20.2_{1.0}$ |
| | ResearchRubrics | 101 | $15.1_{0.0}$ | $24.0_{0.0}$ | $4.66_{0.00}$ | $14.1_{0.0}$ (7.4/6.6) | $65.6_{0.0}$ |
| MiniMax-M2.5 | BrowseComp | 150 | $54.1_{1.2}$ | $50.1_{1.4}$ | $1.41_{0.10}$ | $53.0_{1.2}$ (42.1/10.9) | $189.5_{5.8}$ |
| | BrowseComp-Plus | 150 | $23.7_{0.5}$ | $64.8_{1.3}$ | $1.11_{0.12}$ | $22.8_{0.5}$ (21.4/1.3) | $48.6_{1.2}$ |
| | HLE | 155 | $28.4_{1.0}$ | $42.3_{1.7}$ | $2.08_{0.19}$ | $26.8_{1.1}$ (17.9/8.9) | $79.4_{5.3}$ |
| | DeepSearchQA | 150 | $44.0_{0.7}$ | $49.1_{1.1}$ | $1.33_{0.06}$ | $42.9_{0.7}$ (26.6/16.3) | $122.0_{2.3}$ |
| | Healthbench-Hard | 100 | $8.7_{0.4}$ | $13.2_{0.6}$ | $1.68_{0.08}$ | $7.6_{0.4}$ (3.4/4.3) | $27.6_{1.6}$ |
| | ResearchRubrics | 101 | $20.9_{0.6}$ | $27.7_{0.5}$ | $3.89_{0.07}$ | $19.9_{0.6}$ (12.2/7.7) | $101.8_{5.1}$ |

Table 5: Trajectory statistics of base rollouts (8 per problem) across three models and six benchmarks. **# Inst.**: number of evaluation instances. **# Turns**: number of agent turns per trajectory. **Tokens (K)**: total input+output tokens in thousands. **Output (K)**: output tokens in thousands. **# Tool Calls**: total tool calls, broken down into search and visit calls (shown in parentheses; get_document is used instead of visit for BrowseComp-Plus). **Uniq. URLs**: number of unique URLs visited. All values are averaged over problems and rollouts, and subscripts denote standard deviation across rollout sets.

**LLM-based aggregation** For Solution Aggregation and Summary Aggregation, we use the prompt from Li et al. (2025a). For AggAgent, we present the prompts in Figure 16 (agentic search) and Figure 17 (deep research), and the full tool descriptions in Figure 18. We vary the solution field description of the `finish` tool (Figure 19) in the following cases: 1) Deep research tasks: the model is instructed to generate a long-form report. 2) Qwen3.5-122B: instructing the model to output in XML format sometimes causes tool parsing errors, so we use plain text format instead. This applies only to agentic search tasks.

# C Results

## C.1 Main Results

We present the full results for Section 5.1 in Tables 9, 10, and 11. AggAgent achieves competitive performance not only at $K = 8$ but also at $K = 2$ and $K = 4$.

## C.2 Cost and Latency vs. Performance

We present the full results for Section 5.2 in Figures 8, 9, and 10. The per-benchmark results are consistent with the averaged figures in the main paper, with AggAgent achieving optimal performance relative to cost and latency.

We also present the cost and latency breakdown in Tables 12, 13, and 14. Across all settings, AggAgent consistently incurs aggregation cost much lower than Summary Aggregation and comparable to Solution Aggregation.

## C.3 Stronger Models for Aggregation

We present the full results for Section 6.1 in Table 15. While a stronger aggregator (MiniMax) generally improves performance across all LLM-based aggregation methods, AggAgent remains the most effective.

### C.4 Confidence Calibration

While prior work has demonstrated the effectiveness of confidence-based aggregation strategies such as Weighted Majority Voting and Best-of-N (Wei et al., 2025; Li et al., 2025a), we find that these methods do not consistently generalize across benchmarks. As illustrated in Figure 11, all three models show well-calibrated confidence on BrowseComp-Plus and BrowseComp (large $C - W$ gap), but are less calibrated on HLE and DeepSearchQA. Moreover, on Healthbench-Hard and ResearchRubrics, confidence is essentially uncorrelated with quality ($r \approx 0$). We hypothesize that this miscalibration explains why confidence-based methods plateau or even degrade as $K$ increases on those tasks. In contrast, LLM-based aggregation, and especially AggAgent, scales reliably with $K$ across all benchmarks.

## D Analysis and Discussion

### D.1 Trajectory-RAG Baseline

AggAgent retrieves evidence from trajectory logs on demand, which raises a natural question: how much of its benefit comes from adaptive, agentic retrieval rather than retrieval over trajectories itself? To study this, we compare against a non-agentic retrieval-augmented generation (Lewis et al., 2020) pipeline that has access to the same trajectory logs but performs retrieval only once before synthesis.

Trajectory-RAG baseline indexes trajectory chunks, retrieves relevant evidence, and feeds the retrieved chunks to a standard LLM aggregator without iterative tool use. Conceptually, this baseline can be viewed as SolAgg augmented with retrieved trajectory evidence. Unlike AggAgent, it does not interactively inspect candidate answers, follow up on disagreements, or issue additional retrieval actions based on intermediate hypotheses.

We evaluate three retrieval backends: ROUGE-L, BM25, and Qwen3-Embedding-8B. In preliminary experiments, we found that filling the full context window with retrieved chunks performs better than using a fixed $k$ such as 25 or 50, improving performance by roughly 2 points. We also include the final solution candidates, which provide an additional 1–2 point gain.

| | **SolAgg** | **SummAgg** | **RAG (ROUGE-L)** | **RAG (BM25)** | **RAG (Qwen3)** | **AggAgent** |
|---|---|---|---|---|---|---|
| GLM-4.7-Flash | 42.58 | 41.21 | 44.10 | 44.45 | 43.66 | **47.90** |
| Qwen3.5-122B | 52.78 | 51.69 | 51.27 | 51.91 | 52.07 | **55.83** |
| MiniMax-M2.5 | 54.90 | 53.88 | 56.32 | 55.70 | 55.24 | **57.31** |

Table 6: **Trajectory-RAG baseline.** Scores are averaged across six benchmarks at $K$=8, with the same model used for both rollout generation and aggregation. RAG variants index trajectory chunks, retrieve evidence with ROUGE-L, BM25, or Qwen3-Embedding-8B, and feed the retrieved chunks plus final solution candidates to a non-agentic LLM aggregator.

As shown in Table 6, trajectory-RAG is competitive, confirming the value of retrieving evidence from trajectories. However, AggAgent consistently performs better across model families, suggesting that one-shot retrieval is not sufficient. AggAgent benefits from adaptively inspecting candidate answers, following up on disagreements, and iteratively verifying evidence before synthesis.

### D.2 Combining Sequential and Parallel Scaling

Our main experiments study aggregation over independently generated rollouts without additional context management. We further test whether AggAgent remains effective when the base rollout agent is strengthened with a simple sequential scaling technique. Specifically, we evaluate Qwen3.5-122B on BrowseComp with a discard-all policy at 80% of the context window following Liu et al. (2025). In the original rollouts, around 15%

of trajectories exceeded the context limit, and another 15% exceeded the tool-call budget. Applying context management improves Pass@1 from 40.25 to 46.50.

We then apply aggregation on top of these stronger rollouts. For SummAgg, trajectories exceeding the context window are handled with cascading summarization over chunks, so the full trajectory is compressed rather than simply truncated.

| K | Pass | Heuristic Aggregation | | | | LLM-based Aggregation | | |
|---|---|---|---|---|---|---|---|---|
| | | MV | WMV | BoN | FewTool | SolAgg | SummAgg | AggAgent |
| 1 | 46.50 | - | - | - | - | - | - | - |
| 2 | 56.07 | 46.31 | 53.45 | 53.43 | 53.43 | 55.11 | **56.00** | **56.00** |
| 4 | 63.90 | 51.85 | 58.40 | 58.60 | 58.60 | 59.78 | 60.44 | **61.33** |
| 8 | 70.67 | 54.00 | 58.67 | 62.67 | 55.33 | 65.33 | 66.67 | **68.00** |

Table 7: **Combining sequential and parallel scaling.** Results are on BrowseComp using Qwen3.5-122B with a simple context management policy for base rollouts. Best aggregation result per row is bolded.

As shown in Table 7, AggAgent demonstrates consistent gains over a wide range of Pass@1 values. Therefore, context management and agentic aggregation are complementary. Improving the base rollout raises absolute performance, while AggAgent provides additional gains over the aggregation baselines.

### D.3 Failure Mode Analysis

In this section, we analyze cases where Pass@$K$ succeeds (i.e., at least one trajectory produces a correct answer) but AggAgent fails. We identify four failure modes: (1) **Retrieval failure**, where the trajectory segment containing decisive evidence is not retrieved; (2) **Evidence interpretation failure**, where correct information is retrieved but the aggregator misinterprets or ignores it; (3) **Conflict resolution failure**, where the aggregator is misled by plausible but incorrect trajectories; and (4) **Synthesis failure**, which occurs especially on deep research tasks when correct partial evidence is not integrated into a coherent final response.

We manually annotate failure cases from MiniMax-M2.5 and report the breakdown in Table 8. The dominant source of failure is evidence interpretation, suggesting that improving the aggregator's ability to verify and reason over retrieved evidence is an important direction for closing the gap to Pass@$K$.

| Dataset | $N$ | Retrieval | Evidence Interpretation | Conflict Resolution | Synthesis |
|---|---|---|---|---|---|
| BrowseComp | 3 | 0.0% | 66.7% | 33.3% | 0.0% |
| BrowseComp-Plus | 2 | 0.0% | 0.0% | 0.0% | 100.0% |
| HLE | 19 | 10.5% | 68.4% | 10.5% | 10.5% |
| DeepSearchQA | 45 | 11.1% | 57.7% | 11.5% | 19.2% |

Table 8: **Failure mode analysis.** We manually annotate MiniMax-M2.5 cases where Pass@8 succeeds but AggAgent fails. $N$ denotes the number of annotated failure cases.

### D.4 Qualitative Analysis

We qualitatively analyze cases where AggAgent succeeds in producing correct answers. AggAgent demonstrates four key behaviours (Figure 7): (1) **Minority answer identification**: identifying the correct answer from multiple trajectories, even when it is held by only a minority, (2) **Disagreement resolution**: resolving inconsistencies and conflicts across trajectories to arrive at a coherent answer, (3) **Cross-trajectory synthesis**: constructing a correct solution by reasoning over trajectories that are all incorrect, and (4) **Heuristic interpretation**: leveraging heuristic signals such as majority vote and confidence scores to guide aggregation, while not relying on them entirely. Together, these behaviours highlight AggAgent's capacity for nuanced, evidence-driven reasoning across parallel trajectories beyond simple voting or selection.

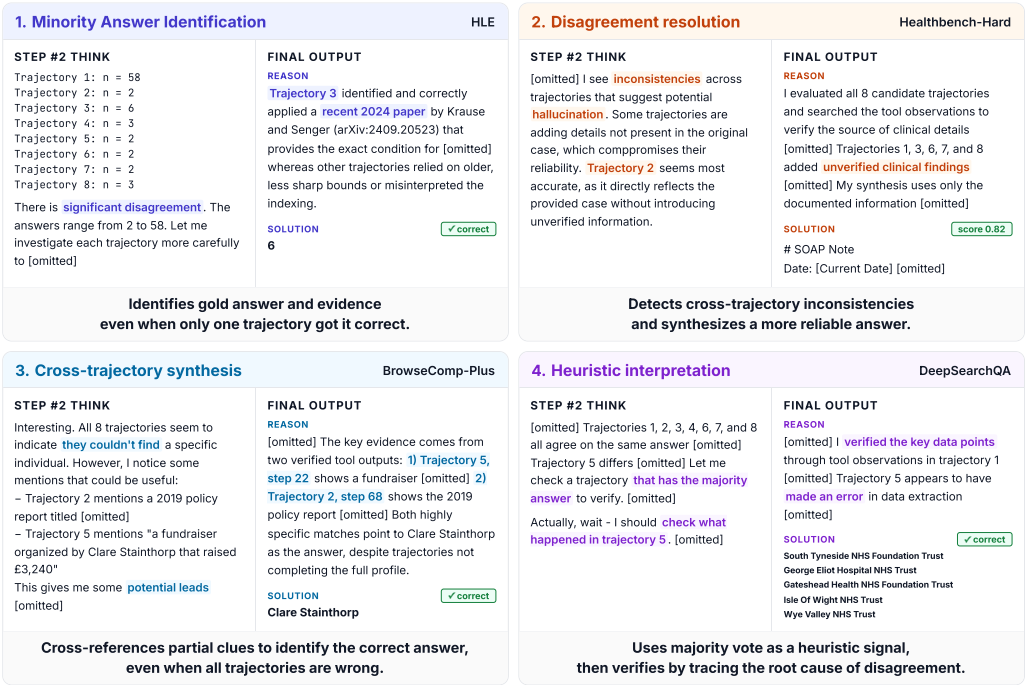


Figure 7: Qualitative examples illustrating four key behaviours of AggAgent.

| Task | K | Pass | Heuristic Aggregation | | | | LLM-based Aggregation | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | MV | WMV | BoN | FewTool | SolAgg | SummAgg | AggAgent |
| BrowseComp | 1 | $27.42_{2.17}$ | - | - | - | - | - | - | - |
| | 2 | $37.86_{2.63}$ | $27.26_{2.36}$ | $35.79_{3.03}$ | $35.60_{3.04}$ | $33.24_{2.47}$ | $35.78_{1.13}$ | $34.89_{2.45}$ | $\mathbf{37.78}_{1.37}$ |
| | 4 | $49.21_{2.04}$ | $30.83_{1.82}$ | $43.87_{2.41}$ | $44.06_{2.64}$ | $38.84_{1.93}$ | $45.33_{0.0}$ | $\mathbf{46.22}_{0.83}$ | $45.78_{1.13}$ |
| | 8 | 58.67 | 32.67 | 50.67 | 51.33 | 41.33 | 53.33 | 55.33 | **56.00** |
| BrowseComp-Plus | 1 | $49.08_{3.09}$ | - | - | - | - | - | - | - |
| | 2 | $59.64_{1.76}$ | $48.71_{3.03}$ | $57.02_{2.33}$ | $57.05_{2.08}$ | $54.31_{2.41}$ | $54.67_{3.40}$ | $58.00_{1.44}$ | $\mathbf{58.44}_{1.37}$ |
| | 4 | $67.36_{1.54}$ | $54.10_{1.98}$ | $63.02_{2.05}$ | $62.81_{1.92}$ | $58.86_{1.81}$ | $63.33_{1.89}$ | $63.33_{0.54}$ | $\mathbf{66.22}_{0.83}$ |
| | 8 | 72.00 | 59.33 | 68.67 | 68.67 | 62.00 | 70.67 | 70.67 | **71.33** |
| HLE | 1 | $25.00_{2.36}$ | - | - | - | - | - | - | - |
| | 2 | $34.01_{2.38}$ | $25.32_{2.15}$ | $28.09_{2.39}$ | $28.29_{2.57}$ | $26.04_{2.35}$ | $29.68_{2.30}$ | $30.75_{0.80}$ | $\mathbf{31.40}_{\mathbf{2.49}}$ |
| | 4 | $42.88_{1.99}$ | $26.58_{1.68}$ | $29.62_{1.49}$ | $30.08_{1.78}$ | $26.53_{2.08}$ | $30.97_{1.05}$ | $31.83_{1.61}$ | $\mathbf{32.47}_{\mathbf{0.61}}$ |
| | 8 | 50.97 | 26.45 | 27.74 | 30.97 | 27.10 | 32.90 | 34.84 | **37.42** |
| DeepSearchQA | 1 | $32.42_{4.36}$ | - | - | - | - | - | - | - |
| | 2 | $43.55_{4.01}$ | - | - | $36.02_{3.41}$ | $33.26_{3.90}$ | $33.78_{2.27}$ | $38.44_{2.06}$ | $\mathbf{40.44}_{1.57}$ |
| | 4 | $54.26_{2.90}$ | - | - | $38.06_{2.34}$ | $33.89_{2.20}$ | $42.44_{1.26}$ | $\mathbf{43.56}_{1.75}$ | $42.00_{1.44}$ |
| | 8 | 64.00 | - | - | 35.33 | 33.33 | 46.00 | 47.33 | **49.33** |
| Healthbench-Hard | 1 | $8.67_{1.98}$ | - | - | - | - | - | - | - |
| | 2 | $21.34_{1.65}$ | - | - | $9.80_{2.15}$ | $9.32_{1.72}$ | $10.50_{2.10}$ | $-1.21_{0.46}$ | $\mathbf{24.45}_{0.52}$ |
| | 4 | $31.96_{1.37}$ | - | - | $9.65_{1.93}$ | $8.48_{1.44}$ | $13.70_{2.18}$ | $4.76_{1.39}$ | $\mathbf{25.36}_{\mathbf{1.59}}$ |
| | 8 | 40.91 | - | - | 9.91 | 8.90 | 15.72 | 7.35 | **27.99** |
| ResearchRubrics | 1 | $37.47_{0.97}$ | - | - | - | - | - | - | - |
| | 2 | $42.71_{0.53}$ | - | - | $37.46_{0.93}$ | $37.41_{0.78}$ | $33.84_{0.45}$ | $27.69_{0.84}$ | $\mathbf{41.32}_{0.72}$ |
| | 4 | $46.57_{0.30}$ | - | - | $37.68_{0.78}$ | $36.55_{0.74}$ | $37.15_{0.46}$ | $30.83_{0.37}$ | $\mathbf{43.70}_{0.58}$ |
| | 8 | 49.79 | - | - | 37.70 | 35.21 | 36.84 | 31.72 | **45.31** |
| **Average (GLM-4.7-Flash)** | 1 | 30.01 | - | - | - | - | - | - | - |
| | 2 | 39.85 | 29.98 | 33.24 | 34.04 | 32.26 | 33.04 | 31.43 | **38.97** |
| | 4 | 48.71 | 31.68 | 35.85 | 37.06 | 33.86 | 38.82 | 36.76 | **42.59** |
| | 8 | 56.06 | 32.84 | 37.61 | 38.99 | 34.65 | 42.58 | 41.21 | **47.90** |

Table 9: Performance comparison across aggregation strategies with GLM-4.7-Flash with varying $K \in \{2, 4, 8\}$. The same model is used for both rollout and aggregation. **Bold** indicates the best-performing method per row (excluding Pass). Subscripts report standard deviation across runs. '-' denotes not applicable.

| Task | K | Pass | Heuristic Aggregation | | | | LLM-based Aggregation | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | MV | WMV | BoN | FewTool | SolAgg | SummAgg | AggAgent |
| BrowseComp | 1 | $40.25_{3.26}$ | - | - | - | - | - | - | - |
| | 2 | $50.38_{2.66}$ | $40.74_{2.95}$ | $47.79_{2.82}$ | $48.05_{2.98}$ | $46.21_{2.82}$ | $48.44_{1.00}$ | $\mathbf{49.33}_{2.88}$ | $48.89_{1.13}$ |
| | 4 | $59.53_{1.84}$ | $45.54_{2.13}$ | $54.30_{2.62}$ | $54.83_{2.38}$ | $51.42_{2.22}$ | $55.78_{0.83}$ | $56.00_{0.94}$ | $\mathbf{56.22}_{1.60}$ |
| | 8 | 67.33 | 47.33 | 60.00 | 58.67 | 56.00 | 61.33 | 62.00 | **62.67** |
| BrowseComp-Plus | 1 | $58.67_{2.13}$ | - | - | - | - | - | - | - |
| | 2 | $67.36_{2.07}$ | $58.14_{2.21}$ | $64.93_{2.05}$ | $64.98_{2.06}$ | $64.36_{1.80}$ | $66.00_{1.09}$ | $65.56_{1.13}$ | $\mathbf{66.44}_{1.13}$ |
| | 4 | $73.86_{1.43}$ | $63.44_{1.91}$ | $69.10_{1.37}$ | $69.78_{1.69}$ | $68.54_{1.44}$ | $69.51_{1.13}$ | $\mathbf{71.78}_{0.83}$ | $71.33_{1.89}$ |
| | 8 | 79.33 | 66.67 | 69.33 | 74.00 | 70.67 | 73.33 | 73.33 | **74.67** |
| HLE | 1 | $39.35_{2.39}$ | - | - | - | - | - | - | - |
| | 2 | $50.12_{2.32}$ | $39.19_{2.54}$ | $43.71_{3.21}$ | $43.71_{2.67}$ | $39.26_{2.22}$ | $45.16_{0.53}$ | $45.59_{2.60}$ | $\mathbf{45.81}_{0.53}$ |
| | 4 | $59.87_{1.82}$ | $42.29_{1.96}$ | $45.28_{2.28}$ | $47.43_{2.70}$ | $39.30_{1.79}$ | $49.25_{1.61}$ | $47.96_{0.30}$ | $\mathbf{51.40}_{0.80}$ |
| | 8 | 67.10 | 43.87 | 45.16 | 53.55 | 41.29 | 50.97 | 50.32 | **54.19** |
| DeepSearchQA | 1 | $49.25_{2.27}$ | - | - | - | - | - | - | - |
| | 2 | $60.76_{2.08}$ | - | - | $53.33_{2.23}$ | $52.17_{2.55}$ | $52.67_{2.88}$ | $52.89_{2.06}$ | $\mathbf{55.56}_{1.75}$ |
| | 4 | $70.02_{1.62}$ | - | - | $56.09_{2.21}$ | $53.67_{2.40}$ | $56.00_{1.44}$ | $58.67_{1.09}$ | $\mathbf{59.56}_{1.13}$ |
| | 8 | 76.67 | - | - | 57.33 | 54.67 | 62.67 | 64.00 | **66.00** |
| Healthbench-Hard | 1 | $12.87_{5.89}$ | - | - | - | - | - | - | - |
| | 2 | $24.52_{3.88}$ | - | - | $13.34_{4.26}$ | $13.62_{3.73}$ | $\mathbf{25.23}_{0.30}$ | $18.84_{1.13}$ | $24.95_{0.39}$ |
| | 4 | $33.45_{2.25}$ | - | - | $14.00_{2.76}$ | $13.70_{2.25}$ | $27.30_{1.91}$ | $22.60_{1.33}$ | $\mathbf{28.01}_{1.06}$ |
| | 8 | 40.74 | - | - | 13.01 | 12.83 | 26.30 | 23.00 | **28.06** |
| ResearchRubrics | 1 | $40.50_{0.91}$ | - | - | - | - | - | - | - |
| | 2 | $44.77_{0.60}$ | - | - | $40.71_{0.67}$ | $40.02_{0.65}$ | $41.17_{0.42}$ | $34.25_{0.53}$ | $\mathbf{44.49}_{0.52}$ |
| | 4 | $48.11_{0.37}$ | - | - | $40.88_{0.62}$ | $39.77_{0.55}$ | $41.41_{0.59}$ | $36.29_{0.35}$ | $\mathbf{46.61}_{0.42}$ |
| | 8 | 50.97 | - | - | 42.37 | 39.58 | 42.10 | 37.47 | **49.36** |
| **Average (Qwen3.5-122B)** | 1 | 40.15 | - | - | - | - | - | - | - |
| | 2 | 49.65 | 40.12 | 43.18 | 44.02 | 42.61 | 46.45 | 44.41 | **47.69** |
| | 4 | 57.47 | 42.32 | 45.22 | 47.17 | 44.40 | 49.88 | 48.88 | **52.19** |
| | 8 | 63.69 | 43.42 | 46.19 | 49.82 | 45.84 | 52.78 | 51.69 | **55.83** |

Table 10: Performance comparison across aggregation strategies with Qwen3.5-122B. The table is formatted the same as Table 9.

| Task | K | Pass | Heuristic Aggregation | | | | LLM-based Aggregation | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | MV | WMV | BoN | FewTool | SolAgg | SummAgg | AggAgent |
| BrowseComp | 1 | $50.17_{3.35}$ | - | - | - | - | - | - | - |
| | 2 | $60.86_{2.32}$ | $50.40_{2.15}$ | $59.40_{2.36}$ | $59.36_{2.34}$ | $57.38_{2.47}$ | $60.00_{1.63}$ | $59.78_{1.13}$ | $\mathbf{60.22}_{1.13}$ |
| | 4 | $68.64_{1.69}$ | $56.58_{2.19}$ | $64.73_{1.88}$ | $66.06_{2.01}$ | $62.63_{1.64}$ | $64.44_{0.63}$ | $66.67_{0.54}$ | $\mathbf{67.33}_{0.0}$ |
| | 8 | 73.33 | 60.67 | 66.67 | 70.67 | 67.33 | 70.67 | 70.00 | **71.33** |
| BrowseComp-Plus | 1 | $64.17_{1.28}$ | - | - | - | - | - | - | - |
| | 2 | $71.12_{1.50}$ | $64.10_{2.02}$ | $69.68_{1.37}$ | $70.02_{1.33}$ | $68.64_{1.69}$ | $70.22_{0.31}$ | $70.44_{0.31}$ | $\mathbf{71.11}_{0.31}$ |
| | 4 | $75.90_{0.94}$ | $67.72_{1.73}$ | $73.57_{1.13}$ | $73.84_{1.11}$ | $71.90_{1.11}$ | $73.56_{0.31}$ | $73.56_{2.45}$ | $\mathbf{74.44}_{0.63}$ |
| | 8 | 78.67 | 68.67 | 75.33 | 76.00 | 74.67 | 76.67 | 76.67 | **77.33** |
| HLE | 1 | $45.73_{1.75}$ | - | - | - | - | - | - | - |
| | 2 | $56.61_{1.57}$ | $45.14_{1.83}$ | $49.52_{1.94}$ | $49.75_{2.11}$ | $46.77_{2.57}$ | $47.53_{2.38}$ | $49.68_{1.90}$ | $\mathbf{52.90}_{0.91}$ |
| | 4 | $65.28_{1.28}$ | $49.09_{1.70}$ | $52.59_{1.45}$ | $52.76_{1.77}$ | $47.48_{2.26}$ | $50.97_{2.11}$ | $56.34_{0.80}$ | $\mathbf{56.99}_{1.22}$ |
| | 8 | 72.26 | 53.55 | 57.42 | 58.06 | 46.45 | 54.19 | 58.06 | **60.00** |
| DeepSearchQA | 1 | $54.42_{1.66}$ | - | - | - | - | - | - | - |
| | 2 | $65.40_{2.21}$ | - | - | $\mathbf{58.02}_{2.26}$ | $55.90_{1.97}$ | $57.78_{1.57}$ | $57.11_{1.57}$ | $57.33_{2.18}$ |
| | 4 | $75.04_{1.86}$ | - | - | $61.46_{2.23}$ | $56.50_{1.47}$ | $59.11_{1.91}$ | $61.33_{0.54}$ | $\mathbf{63.78}_{1.13}$ |
| | 8 | 82.67 | - | - | 64.00 | 56.00 | 62.00 | 61.33 | **65.33** |
| Healthbench-Hard | 1 | $9.67_{3.84}$ | - | - | - | - | - | - | - |
| | 2 | $21.09_{2.91}$ | - | - | $9.74_{2.38}$ | $9.02_{2.67}$ | $17.30_{1.28}$ | $9.74_{0.91}$ | $\mathbf{19.25}_{1.53}$ |
| | 4 | $30.62_{1.93}$ | - | - | $10.58_{2.06}$ | $8.04_{2.45}$ | $21.25_{1.20}$ | $12.37_{1.32}$ | $\mathbf{24.27}_{0.52}$ |
| | 8 | 39.02 | - | - | 8.79 | 5.34 | 21.84 | 16.92 | **24.46** |
| ResearchRubrics | 1 | $39.97_{0.71}$ | - | - | - | - | - | - | - |
| | 2 | $43.86_{0.55}$ | - | - | $40.16_{0.55}$ | $39.63_{0.85}$ | $41.88_{0.57}$ | $34.15_{0.55}$ | $\mathbf{43.17}_{0.65}$ |
| | 4 | $47.06_{0.38}$ | - | - | $40.05_{0.58}$ | $39.19_{0.67}$ | $43.26_{0.24}$ | $37.12_{0.42}$ | $\mathbf{44.96}_{0.53}$ |
| | 8 | 49.74 | - | - | 39.00 | 38.44 | 44.00 | 40.29 | **45.42** |
| **Average (MiniMax-M2.5)** | 1 | 44.02 | - | - | - | - | - | - | - |
| | 2 | 53.16 | 43.95 | 47.11 | 47.84 | 46.22 | 49.12 | 46.82 | **50.66** |
| | 4 | 60.42 | 46.24 | 49.16 | 50.79 | 47.62 | 52.10 | 51.23 | **55.30** |
| | 8 | 65.95 | 47.83 | 50.58 | 52.75 | 48.04 | 54.90 | 53.88 | **57.31** |

Table 11: Performance comparison across aggregation strategies with MiniMax-M2.5. The table is formatted the same as Table 9.

| Task | K | Rollout Cost ($c_r$) | Tool Call Cost ($c_t$) | Aggregation Cost ($c_a$) | | | Rollout Latency ($t_r$) | Aggregation Latency ($t_a$) | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | SolAgg | SummAgg | AggAgent | | SolAgg | SummAgg | AggAgent |
| BrowseComp | 1 | .0086 | .0390 | - | - | - | 185.10 | - | - | - |
| | 2 | .0171 | .0780 | .0025 | .0143 | .0044 | 211.82 | 33.97 | 77.72 | 22.67 |
| | 4 | .0343 | .1560 | .0021 | .0261 | .0048 | 233.74 | 21.38 | 107.27 | 38.51 |
| | 8 | .0685 | .3119 | .0019 | .0506 | .0047 | 261.41 | 18.69 | 139.42 | 31.65 |
| BrowseComp-Plus | 1 | .0067 | 0 | - | - | - | 457.47 | - | - | - |
| | 2 | .0134 | 0 | .0026 | .0142 | .0037 | 536.31 | 42.54 | 84.65 | 25.30 |
| | 4 | .0268 | 0 | .0021 | .0260 | .0038 | 584.04 | 21.19 | 83.42 | 31.56 |
| | 8 | .0536 | 0 | .0020 | .0501 | .0038 | 620.15 | 15.31 | 128.79 | 29.88 |
| HLE | 1 | .0079 | .0190 | - | - | - | 91.77 | - | - | - |
| | 2 | .0158 | .0379 | .0037 | .0118 | .0046 | 125.04 | 29.41 | 54.56 | 33.55 |
| | 4 | .0316 | .0758 | .0038 | .0202 | .0053 | 178.32 | 31.07 | 86.67 | 34.19 |
| | 8 | .0632 | .1517 | .0041 | .0368 | .0055 | 250.64 | 35.24 | 115.87 | 40.32 |
| DeepSearchQA | 1 | .0068 | .0311 | - | - | - | 105.60 | - | - | - |
| | 2 | .0135 | .0623 | .0018 | .0113 | .0043 | 159.65 | 14.96 | 55.49 | 30.48 |
| | 4 | .0271 | .1245 | .0021 | .0210 | .0049 | 225.01 | 18.22 | 69.63 | 33.60 |
| | 8 | .0542 | .2490 | .0024 | .0401 | .0052 | 261.79 | 21.73 | 105.53 | 36.98 |
| Healthbench-Hard | 1 | .0037 | .0114 | - | - | - | 41.20 | - | - | - |
| | 2 | .0075 | .0228 | .0019 | .0081 | .0028 | 52.22 | 22.26 | 46.47 | 23.12 |
| | 4 | .0150 | .0456 | .0023 | .0146 | .0035 | 67.15 | 22.46 | 58.39 | 25.51 |
| | 8 | .0299 | .0912 | .0028 | .0276 | .0039 | 88.27 | 26.84 | 81.48 | 27.32 |
| ResearchRubrics | 1 | .0070 | .0222 | - | - | - | 112.06 | - | - | - |
| | 2 | .0141 | .0444 | .0028 | .0119 | .0044 | 142.95 | 28.61 | 63.13 | 40.03 |
| | 4 | .0282 | .0889 | .0034 | .0219 | .0053 | 187.43 | 35.08 | 84.07 | 46.80 |
| | 8 | .0563 | .1777 | .0045 | .0412 | .0062 | 243.40 | 37.83 | 114.70 | 50.91 |
| **Average (GLM-4.7-Flash)** | 1 | .0068 | .0204 | - | - | - | 165.53 | - | - | - |
| | 2 | .0136 | .0409 | .0025 | .0119 | .0040 | 204.67 | 28.63 | 63.67 | 29.19 |
| | 4 | .0272 | .0818 | .0026 | .0216 | .0046 | 245.95 | 24.90 | 81.58 | 35.03 |
| | 8 | .0543 | .1636 | .0029 | .0411 | .0049 | 287.61 | 25.94 | 114.30 | 36.18 |

Table 12: Cost and latency breakdown with GLM-4.7-Flash. All costs are in USD per query, and all latencies are in seconds per query. '-' denotes not applicable at $K = 1$. BrowseComp-Plus incurs no tool cost.

| Task | K | Rollout Cost ($c_r$) | Tool Call Cost ($c_t$) | Aggregation Cost ($c_a$) | | | Rollout Latency ($t_r$) | Aggregation Latency ($t_a$) | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | SolAgg | SummAgg | AggAgent | | SolAgg | SummAgg | AggAgent |
| BrowseComp | 1 | .0409 | .0315 | - | - | - | 182.79 | - | - | - |
| | 2 | .0817 | .0629 | .0102 | .0565 | .0213 | 224.26 | 21.96 | 53.60 | 39.07 |
| | 4 | .1634 | .1258 | .0103 | .1026 | .0309 | 242.48 | 21.31 | 69.14 | 41.04 |
| | 8 | .3268 | .2517 | .0103 | .1952 | .0209 | 305.38 | 20.93 | 91.82 | 36.25 |
| BrowseComp-Plus | 1 | .0265 | 0 | - | - | - | 564.68 | - | - | - |
| | 2 | .0530 | 0 | .0098 | .0544 | .0186 | 690.97 | 25.55 | 58.80 | 36.38 |
| | 4 | .1060 | 0 | .0099 | .0977 | .0193 | 743.69 | 21.32 | 62.35 | 32.60 |
| | 8 | .2120 | 0 | .0097 | .1893 | .0182 | 831.50 | 19.99 | 99.29 | 41.32 |
| HLE | 1 | .0318 | .0149 | - | - | - | 74.53 | - | - | - |
| | 2 | .0637 | .0298 | .0128 | .0418 | .0173 | 99.80 | 36.98 | 45.68 | 25.04 |
| | 4 | .1274 | .0597 | .0131 | .0705 | .0197 | 125.14 | 24.30 | 60.16 | 31.51 |
| | 8 | .2548 | .1194 | .0127 | .1307 | .0203 | 152.97 | 29.85 | 74.72 | 38.28 |
| DeepSearchQA | 1 | .0277 | .0239 | - | - | - | 89.32 | - | - | - |
| | 2 | .0553 | .0477 | .0088 | .0437 | .0208 | 110.47 | 16.71 | 48.91 | 39.57 |
| | 4 | .1106 | .0954 | .0107 | .0776 | .0244 | 129.99 | 22.32 | 54.54 | 38.96 |
| | 8 | .2212 | .1909 | .0116 | .1494 | .0247 | 173.43 | 20.50 | 65.65 | 40.28 |
| Healthbench-Hard | 1 | .0080 | .0045 | - | - | - | 22.03 | - | - | - |
| | 2 | .0161 | .0091 | .0092 | .0215 | .0141 | 26.67 | 24.29 | 32.69 | 26.68 |
| | 4 | .0321 | .0181 | .0107 | .0376 | .0166 | 29.56 | 26.96 | 40.83 | 28.86 |
| | 8 | .0642 | .0363 | .0130 | .0680 | .0175 | 35.09 | 30.70 | 53.77 | 32.38 |
| ResearchRubrics | 1 | .0182 | .0093 | - | - | - | 66.92 | - | - | - |
| | 2 | .0364 | .0186 | .0183 | .0377 | .0260 | 75.28 | 47.98 | 51.80 | 47.75 |
| | 4 | .0728 | .0371 | .0213 | .0632 | .0276 | 84.19 | 57.39 | 72.33 | 54.95 |
| | 8 | .1457 | .0743 | .0261 | .1134 | .0301 | 96.06 | 47.39 | 87.53 | 55.08 |
| **Average (Qwen3.5-122B)** | 1 | .0255 | .0140 | - | - | - | 166.71 | - | - | - |
| | 2 | .0510 | .0280 | .0115 | .0426 | .0197 | 204.58 | 28.91 | 48.58 | 35.75 |
| | 4 | .1020 | .0560 | .0127 | .0749 | .0231 | 225.84 | 28.93 | 59.89 | 37.99 |
| | 8 | .2041 | .1121 | .0139 | .1410 | .0219 | 265.74 | 28.23 | 78.80 | 40.60 |

Table 13: Cost and latency breakdown with Qwen3.5-122B. The table is formatted the same as Table 12.

| Task | K | Rollout Cost ($c_r$) | Tool Call Cost ($c_t$) | Aggregation Cost ($c_a$) | | | Rollout Latency ($t_r$) | Aggregation Latency ($t_a$) | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | SolAgg | SummAgg | AggAgent | | SolAgg | SummAgg | AggAgent |
| BrowseComp | 1 | .0321 | .0301 | - | - | - | 209.34 | - | - | - |
| | 2 | .0642 | .0602 | .0055 | .0389 | .0122 | 259.73 | 267.45 | 300.34 | 294.52 |
| | 4 | .1285 | .1204 | .0049 | .0715 | .0120 | 292.90 | 300.42 | 347.88 | 319.73 |
| | 8 | .2569 | .2409 | .0049 | .1357 | .0118 | 314.26 | 323.33 | 480.36 | 352.57 |
| BrowseComp-Plus | 1 | .0298 | 0 | - | - | - | 392.76 | - | - | - |
| | 2 | .0596 | 0 | .0052 | .0459 | .0114 | 519.48 | 528.28 | 567.93 | 550.86 |
| | 4 | .1191 | 0 | .0037 | .0876 | .0111 | 632.33 | 639.21 | 708.68 | 663.04 |
| | 8 | .2382 | 0 | .0025 | .1651 | .0111 | 716.32 | 723.99 | 968.27 | 744.93 |
| HLE | 1 | .0358 | .0164 | - | - | - | 165.06 | - | - | - |
| | 2 | .0716 | .0327 | .0088 | .0385 | .0116 | 194.42 | 217.01 | 239.92 | 227.34 |
| | 4 | .1433 | .0654 | .0087 | .0689 | .0144 | 228.92 | 244.59 | 304.89 | 259.10 |
| | 8 | .2865 | .1308 | .0082 | .1294 | .0126 | 264.72 | 280.52 | 404.20 | 310.09 |
| DeepSearchQA | 1 | .0282 | .0268 | - | - | - | 149.91 | - | - | - |
| | 2 | .0564 | .0536 | .0039 | .0378 | .0141 | 179.42 | 189.97 | 219.72 | 210.96 |
| | 4 | .1129 | .1072 | .0039 | .0721 | .0147 | 193.99 | 207.52 | 247.52 | 232.83 |
| | 8 | .2257 | .2144 | .0047 | .1412 | .0148 | 227.26 | 236.98 | 350.82 | 261.86 |
| Healthbench-Hard | 1 | .0074 | .0052 | - | - | - | 28.86 | - | - | - |
| | 2 | .0148 | .0105 | .0035 | .0143 | .0057 | 33.13 | 49.61 | 64.43 | 58.19 |
| | 4 | .0295 | .0209 | .0043 | .0267 | .0070 | 34.08 | 52.90 | 75.92 | 58.94 |
| | 8 | .0591 | .0419 | .0058 | .0507 | .0084 | 40.55 | 65.48 | 95.72 | 72.02 |
| ResearchRubrics | 1 | .0151 | .0125 | - | - | - | 93.25 | - | - | - |
| | 2 | .0302 | .0250 | .0086 | .0257 | .0125 | 109.61 | 159.09 | 161.16 | 166.29 |
| | 4 | .0603 | .0499 | .0109 | .0480 | .0149 | 122.48 | 177.29 | 192.37 | 177.12 |
| | 8 | .1207 | .0998 | .0151 | .0919 | .0183 | 146.17 | 203.61 | 270.58 | 213.30 |
| **Average (MiniMax-M2.5)** | 1 | .0237 | .0152 | - | - | - | 173.20 | - | - | - |
| | 2 | .0474 | .0303 | .0059 | .0335 | .0113 | 215.97 | 235.24 | 258.92 | 251.36 |
| | 4 | .0949 | .0606 | .0061 | .0625 | .0123 | 250.78 | 270.32 | 312.88 | 285.13 |
| | 8 | .1898 | .1213 | .0069 | .1190 | .0128 | 284.88 | 305.65 | 428.33 | 325.80 |

Table 14: Cost and latency breakdown with MiniMax-M2.5. The table is formatted the same as Table 12.

| Task | K | Pass | GLM-4.7-Flash | | | MiniMax-M2.5 | | |
|---|---|---|---|---|---|---|---|---|
| | | | SolAgg | SummAgg | AggAgent | SolAgg | SummAgg | AggAgent |
| BrowseComp | 1 | 27.42 | - | - | - | - | - | - |
| | 2 | 37.86 | 35.78 | 34.89 | 37.78 | 36.22 | 37.11 | **39.11** |
| | 4 | 49.21 | 45.33 | 46.22 | 45.78 | 46.22 | 44.44 | **46.44** |
| | 8 | 58.67 | 53.33 | 55.33 | **56.00** | 51.33 | 55.33 | 55.33 |
| BrowseComp-Plus | 1 | 49.08 | - | - | - | - | - | - |
| | 2 | 59.64 | 54.67 | 58.00 | 58.44 | 58.22 | 58.67 | **59.33** |
| | 4 | 67.36 | 63.33 | 63.33 | 66.22 | 64.89 | 65.33 | **68.44** |
| | 8 | 72.00 | 70.67 | 70.67 | 71.33 | 70.00 | 71.33 | **72.67** |
| HLE | 1 | 25.00 | - | - | - | - | - | - |
| | 2 | 34.01 | 29.68 | 30.75 | **31.40** | 27.74 | 30.54 | 29.46 |
| | 4 | 42.88 | 30.97 | 31.83 | 32.47 | 32.26 | 30.97 | **33.76** |
| | 8 | 50.97 | 32.90 | 34.84 | 37.42 | 35.48 | 35.48 | **38.06** |
| DeepSearchQA | 1 | 32.42 | - | - | - | - | - | - |
| | 2 | 43.55 | 33.78 | 38.44 | 40.44 | 36.22 | 35.56 | **42.89** |
| | 4 | 54.26 | 42.44 | 43.56 | 42.00 | 41.11 | 41.78 | **43.78** |
| | 8 | 64.00 | 46.00 | 47.33 | 49.33 | 48.00 | 50.00 | **52.67** |
| Healthbench-Hard | 1 | 8.67 | - | - | - | - | - | - |
| | 2 | 21.34 | 10.50 | −1.21 | **24.45** | 18.06 | 10.25 | 22.81 |
| | 4 | 31.96 | 13.70 | 4.76 | **25.36** | 22.53 | 16.85 | 24.22 |
| | 8 | 40.91 | 15.72 | 7.35 | **27.99** | 22.18 | 18.90 | 26.74 |
| ResearchRubrics | 1 | 37.47 | - | - | - | - | - | - |
| | 2 | 42.71 | 33.84 | 27.69 | 41.32 | 40.43 | 33.90 | **43.42** |
| | 4 | 46.57 | 37.15 | 30.83 | 43.70 | 43.02 | **44.76** | 44.69 |
| | 8 | 49.79 | 36.84 | 31.72 | 45.31 | 43.14 | 40.56 | **46.07** |
| **Average** | 1 | 30.01 | - | - | - | - | - | - |
| | 2 | 39.85 | 33.04 | 31.43 | 38.97 | 36.15 | 34.34 | **39.50** |
| | 4 | 48.71 | 38.82 | 36.76 | 42.59 | 41.67 | 40.69 | **43.56** |
| | 8 | 56.06 | 42.58 | 41.21 | 47.90 | 45.02 | 45.27 | **48.59** |

Table 15: Performance of LLM-based aggregation strategies on GLM-4.7-Flash as the base rollout model, comparing GLM (same model) and MiniMax (stronger model) as the LLM aggregator, across varying $K$. **Bold** indicates the best-performing method per row (excluding Pass). Results show that using a stronger aggregator leads to better performance. '-' denotes not applicable at $K=1$.

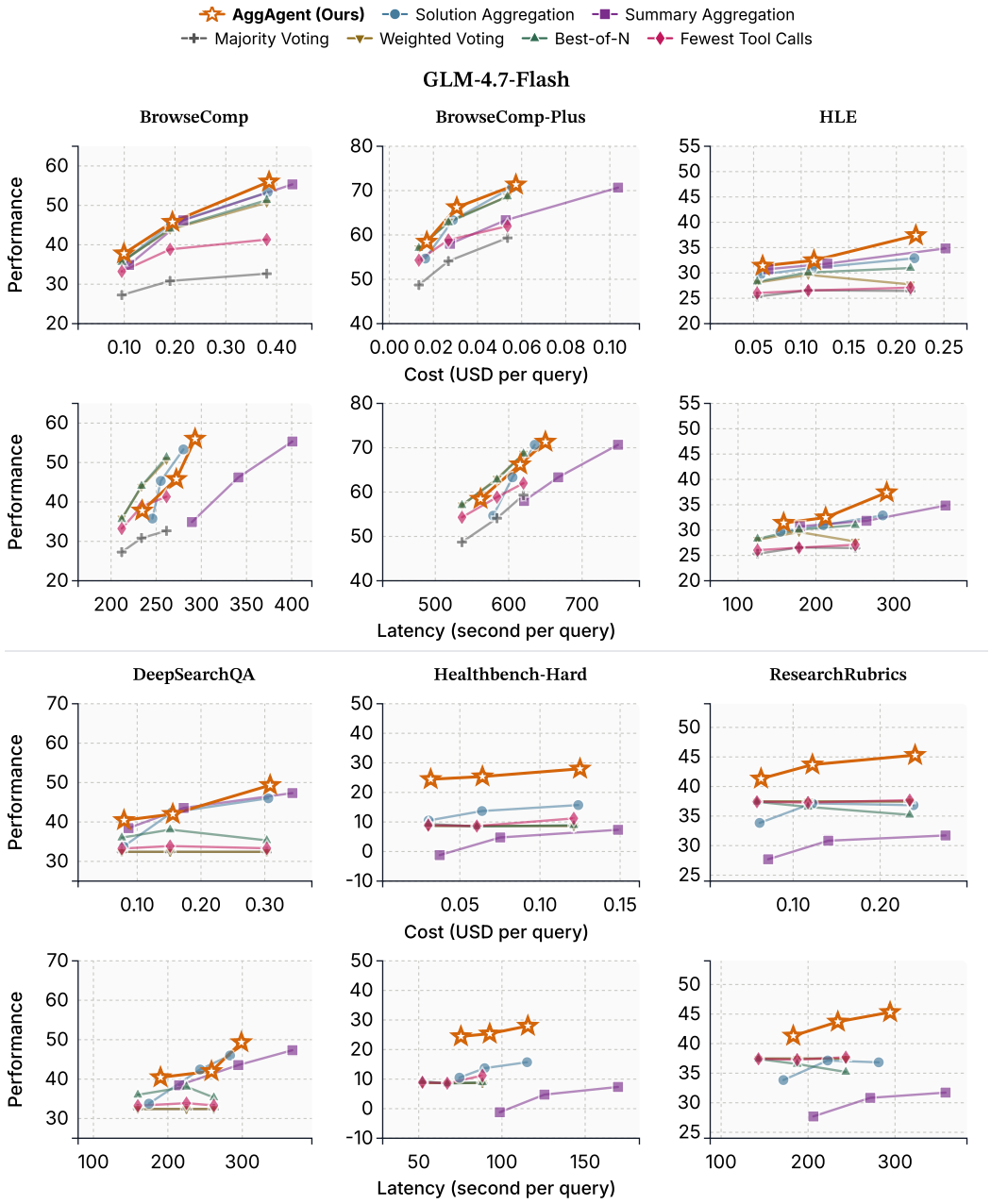


Figure 8: Performance-efficiency tradeoff of aggregation methods across six benchmarks at varying numbers of parallel samples $K = \{2, 4, 8\}$ using GLM-4.7-Flash as both the rollout agent and the aggregator. For each group of three benchmarks, the first row shows performance vs. cost (USD per query) and the second row shows performance vs. latency (seconds per query). AggAgent consistently achieves higher performance at comparable or lower cost and latency than existing aggregation strategies.

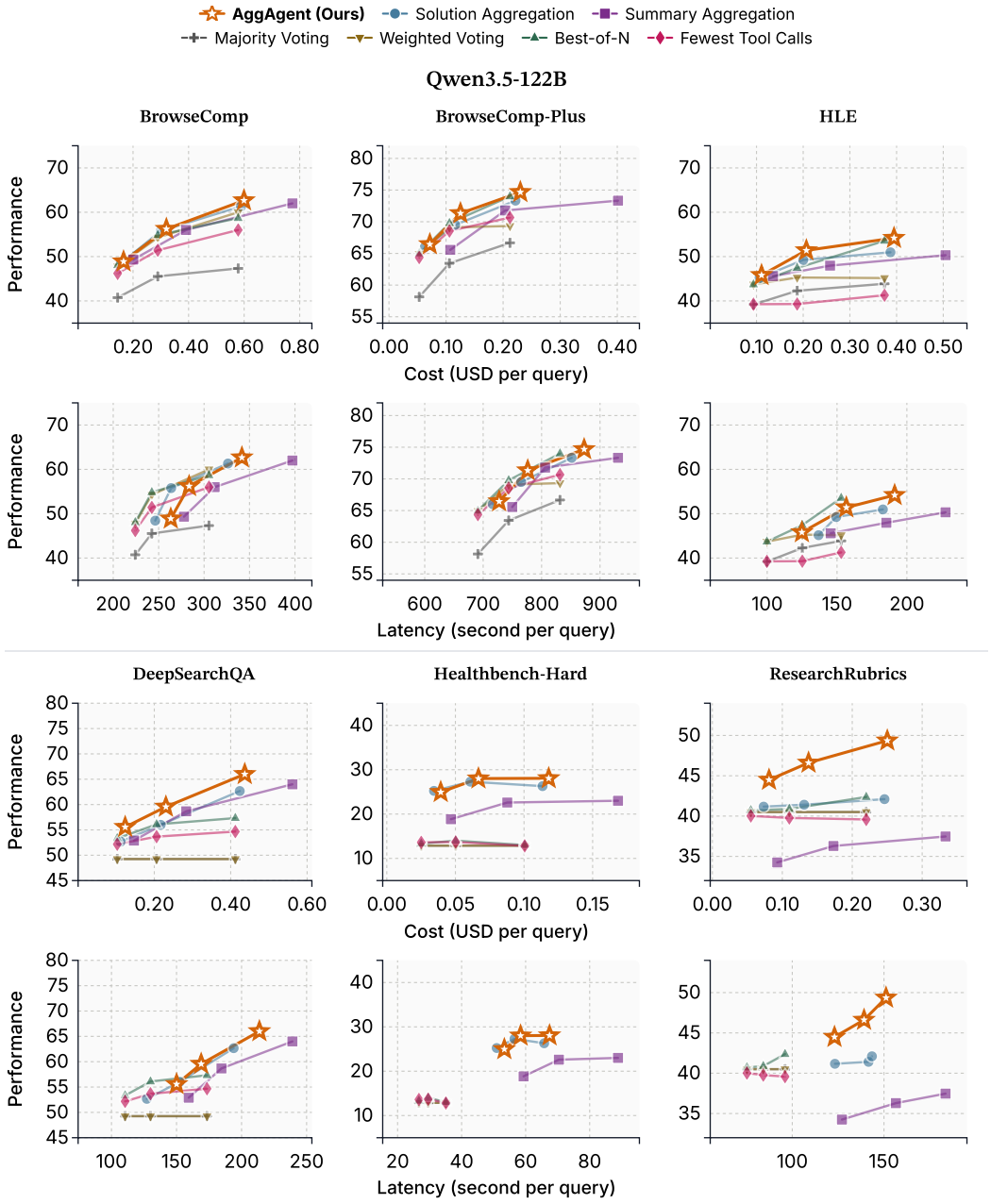


Figure 9: Performance-efficiency trade-off of aggregation methods using Qwen3.5-122B as both the rollout agent and the aggregator. The figure is formatted the same as Figure 8.

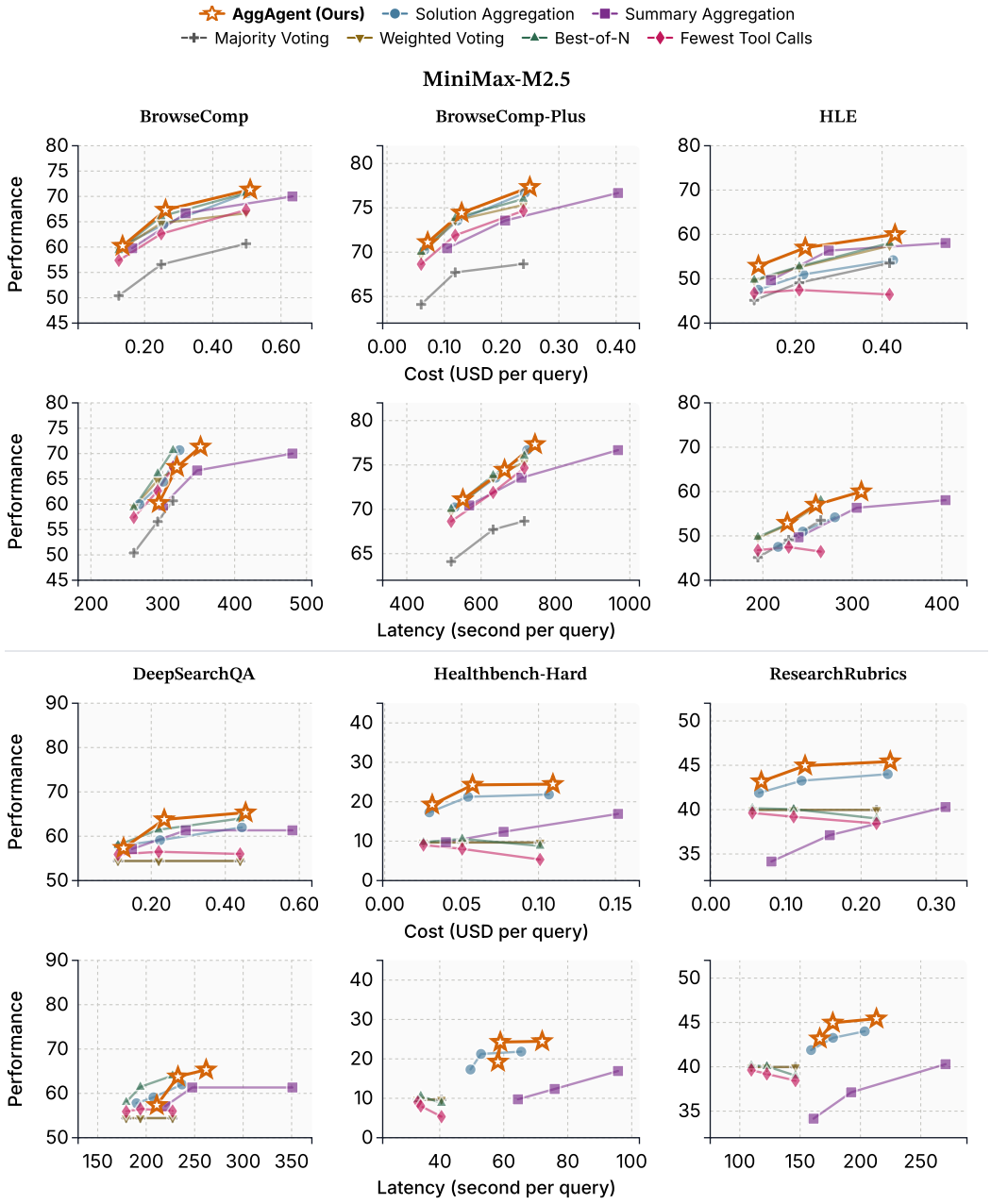


Figure 10: Performance-efficiency trade-off of aggregation methods using MiniMax-M2.5 as both the rollout agent and the aggregator. The figure is formatted the same as Figure 8.

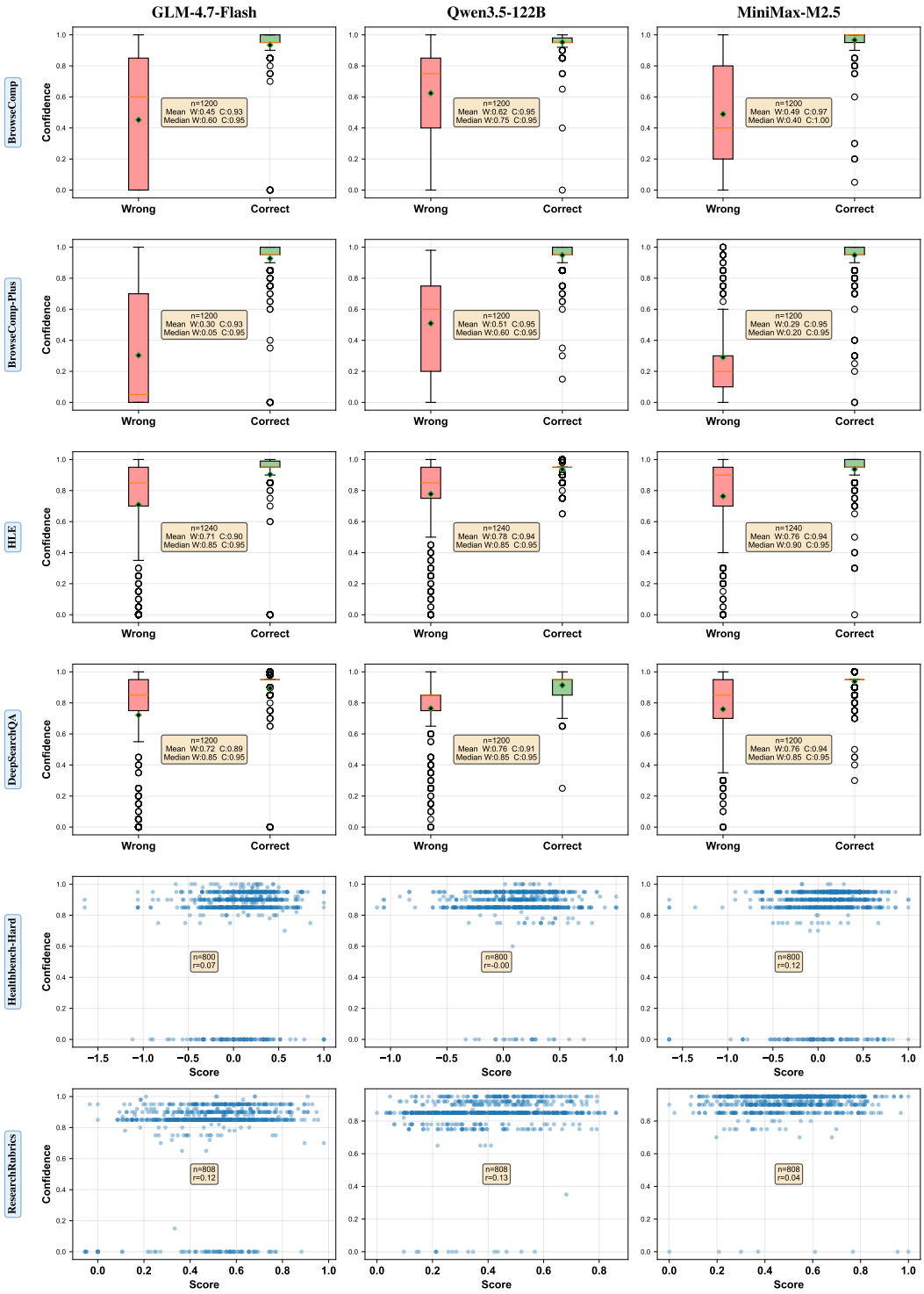


Figure 11: Confidence calibration across all six benchmarks. Each row is a benchmark and each column is a model. For binary-correctness (agentic search) benchmarks, box plots show self-reported confidence by outcome (**W**rong vs. **C**orrect). For continuous-score (deep research) benchmarks, scatter plots show confidence vs. rubric score with Pearson $r$. Models are well-calibrated on BrowseComp-Plus and BrowseComp, moderately calibrated on HLE and DeepSearchQA, but poorly calibrated on Healthbench-Hard and ResearchRubrics ($r \approx 0$).

**Rollout agent system prompt for agentic search tasks**

You are a deep research assistant. Your core function is to conduct thorough, multi-source investigations into any topic. You must handle both broad, open-domain inquiries and queries within specialized academic fields. For every request, synthesize information from credible, diverse sources to deliver a comprehensive, accurate, and objective response. After you have gathered sufficient information, provide the definitive response and complete the task.

Figure 12: Rollout agent system prompt for agentic search tasks, from Tongyi DeepResearch Team et al. (2025).

**Additional rollout agent system prompt for deep research tasks**

For the given question, please write a comprehensive, evidence-backed answer to scientific questions. You should ground every nontrivial claim in retrieved snippets. Cite using <cite url="...">...</cite> drawn only from returned snippets. Please prefer authoritative sources (peer-reviewed papers, reputable benchmarks/docs) and prioritize recent work for fast-moving areas. You should acknowledge uncertainty and conflicts; if evidence is thin or sources disagree, state it and explain what additional evidence would resolve it. It's important to structure with clear markdown headers and a coherent flow. In each section, write 2-5 sentence paragraphs with clear topic sentences and transitions; use lists sparingly only when they improve clarity. Ideally, you should synthesize rather than enumerate content: it's helpful to group findings across papers, explain relationships, and build a coherent narrative that answers the question, supported by citations. Most importantly, DO NOT invent snippets or citations and never fabricate content. At the end of your response, on a new line, include your confidence score in the following format: Confidence: {{your confidence score between 0% and 100% for your answer}}

Figure 13: Additional rollout agent system prompt for deep research tasks, from Shao et al. (2025).

**Rollout agent user message for agentic search tasks**

Your response should be in the following format:
Explanation: {{your explanation for your final answer}}
Exact Answer: {{your succinct, final answer}}
Confidence: {{your confidence score between 0% and 100% for your answer}}

Figure 14: Rollout agent user message for agentic search tasks, from Wei et al. (2025).

**Additional instruction for evaluating ResearchRubrics**

## Handling Negative-Weight Criteria
Some criteria describe **undesirable behaviors** and carry a **negative weight**. The Weight field in the rubric indicates this. For these criteria:
- **Satisfied (Score: 1.0)**: The undesirable behavior IS present in the document. This will reduce the overall score.
- **Not Satisfied (Score: 0.0)**: The undesirable behavior is ABSENT from the document. This is the desired outcome and does not penalize the score.
Your verdict must reflect whether the described behavior is literally present or absent in the document — not whether the document is generally good or bad.

Figure 15: Additional instruction for evaluating ResearchRubrics.

AggAgent system prompt for agentic search tasks

You are an aggregation agent. You are provided with a task and a set of candidate trajectories from independent agents that attempted to solve it. Your goal is to synthesize the most accurate, complete solution by drawing on the best reasoning and evidence across trajectories. You do NOT have access to the ground truth solution.
—
**RESPONSIBILITIES**
1. Evaluate tool results and reasoning quality across all candidate trajectories.
2. Identify the most reliable final solution based on verifiable tool observations, logical consistency, and correct tool application.
3. If no single trajectory is fully reliable, synthesize a corrected solution using only verified components from across trajectories.
4. Deliver your synthesized solution in the required format and provide justification.
—
**REQUIRED PROCEDURE**
You must follow these steps before calling 'finish'.
1. **Survey the landscape** — Read the TRAJECTORY METADATA in the user message. Identify which trajectories are worth inspecting based on step counts and patterns.
2. **Retrieve full solutions** — Call 'get_solution' (no arguments) to get the final content from every trajectory's last step, or pass a trajectory_id to retrieve one specific trajectory.
3. **Verify with tool observations** — Do not rely solely on final solutions or a trajectory's own reasoning. For key claims or divergences, go back and inspect what the tools actually returned: - Use **search_trajectory**(trajectory_id, query) to locate steps where a specific term or claim appears. Use role='tool' to restrict to actual tool responses when verifying whether a fact was directly observed — this avoids misleading matches on agent reasoning. - Use **get_segment**(trajectory_id, start_step, end_step) to read a contiguous range of steps (max 5). After finding a relevant step via search, read it in full along with surrounding steps to see the raw tool output and surrounding context.
4. **Cross-check** — Confirm: (a) tool observations in the log match what the agent claims, (b) reasoning is not circular, (c) arithmetic and logic are correct.
—
**OPERATIONAL GUIDELINES**
- **Tool results are ground truth; agent reasoning is not.** Within each trajectory, what a tool *returned* is an objective observation. What the agent *concluded* from it is an interpretation that may be wrong. When in conflict, trust the tool output over the agent's written reasoning about it.
- **Count evidence, not trajectories.** A single trajectory with a clear, unambiguous tool observation supporting answer X is stronger evidence than many trajectories that *reasoned* their way to Y without grounding in tool outputs. Majority agreement alone is not sufficient — check what the tools actually showed.
- **Identify Divergence:** Focus on steps where agents disagree. Determine which agent's *observation from the environment* was correct, not which agent sounded more confident.
- **Evidence Grounding:** Ensure tool observations directly support conclusions. If the log shows an error or empty result, the agent cannot validly claim success from that step.
- **Quality over Confidence:** Prefer trajectories with validated, step-by-step reasoning over those that only state a confident conclusion.
—
**COMMON PITFALLS**
- **Hallucinated Observations:** The agent claims a tool returned X, but the log shows Y (or nothing).
- **Silent Failures:** The agent receives an error but continues as if it succeeded.
- **Circular Logic:** The agent assumes the answer before deriving it from data.
- **Arithmetic/Logical Errors:** The data is correct, but the calculation or inference is flawed.
- **Majority Bias:** Do not treat numerical agreement among trajectories as strong evidence. Many trajectories reaching the same conclusion via similar reasoning is weaker than one trajectory with a concrete, verifiable tool result.
—
**SOLUTION FORMAT (finish tool)**
The 'solution' argument must be a single string with exactly two XML sections: <explanation>...</explanation><answer>...</answer>.
- **CORRECT:** Self-contained. A reader who never saw trajectories understands how the answer was derived. No mentions of "trajectory 1", "get_solution", or "agent". Example:

<explanation>We need the 2020 population. The census table shows state X with 1.2M and state Y with 0.8M; the question asks for the sum. 1.2 + 0.8 = 2.0 million.</explanation><answer>2.0 million</answer>
- **WRONG:** "Trajectory 2 had the right answer so I chose it." or "According to get_solution, the answer is 42." Do NOT reference trajectory IDs or tools in the solution.
—
**TERMINATION**
Call 'finish' only after verifying key reasoning against actual tool outputs. Do not finish after only reading metadata or only get_solution; verify at least one critical claim with get_segment or search_trajectory when trajectories disagree.

Figure 16: AggAgent system prompt for agentic search tasks.

**AggAgent system prompt for deep research tasks**

You are an aggregation agent. You are provided with a task and a set of candidate trajectories from independent agents that attempted to solve it. Your goal is to synthesize the most accurate, complete solution by drawing on the best reasoning and evidence across trajectories. You do NOT have access to the ground truth.
—
**RESPONSIBILITIES**
1. Evaluate tool results and reasoning quality across all candidate trajectories.
2. Identify the most reliable final solution based on verifiable tool observations, logical consistency, and correct tool application.
3. If no single trajectory is fully reliable, synthesize a corrected solution using only verified components from across trajectories.
4. Deliver your synthesized solution in the required format and provide justification.
—
**REQUIRED PROCEDURE**
You must follow these steps before calling 'finish'.
1. **Survey the landscape** — Read the TRAJECTORY METADATA in the user message. Identify which trajectories are worth inspecting based on step counts and patterns.
2. **Retrieve full solutions** — Call 'get_solution' (no arguments) to get the final content from every trajectory's last step, or pass a trajectory_id to retrieve one specific trajectory.
3. **Verify with tool observations** — Do not rely solely on final solutions or a trajectory's own reasoning. For key claims or divergences, go back and inspect what the tools actually returned: - Use **search_trajectory**(trajectory_id, query) to locate steps where a specific term or claim appears. Use role='tool' to restrict to tool responses when verifying whether a fact was directly observed. - Use **get_segment**(trajectory_id, start_step, end_step) to read a contiguous range of steps (max 5). After finding a relevant step via search, read it in full along with surrounding steps to see the raw tool output and surrounding context.
4. **Cross-check** — Confirm: (a) tool observations in the log match what the agent claims, (b) reasoning is not circular, (c) arithmetic and logic are correct.
5. **Synthesize** — Write a unified response that: - Covers every important aspect addressed by any candidate - Takes the highest-quality treatment of each aspect (not just the most common) - Resolves contradictions by preferring more specific, better-supported, or more precise content - Reads as a single coherent response, not a patchwork
—
**QUALITY CRITERIA**
- **Completeness:** The synthesized response must be at least as comprehensive as the best individual candidate, and more comprehensive where candidates complement each other.
- **Accuracy:** Prefer specific, precise content over vague generalizations. When candidates conflict, do not average — choose the more defensible position.
- **Coherence:** The final response must flow naturally. Integrate content rather than concatenating sections.
- **Self-contained:** Do not mention trajectories, agents, candidates, or aggregation anywhere in the response.
- **Citations:** Ground every nontrivial claim in retrieved snippets. Cite using <cite url="...">...</cite> drawn only from returned snippets; never fabricate URLs or content.

—
**COMMON PITFALLS**
- **Cherry-picking the best-sounding candidate:** Length or fluency is not quality. A shorter candidate may cover a critical aspect better.
- **Ignoring minority candidates:** A single candidate covering an important aspect well outweighs many candidates that omit it.
- **Concatenation instead of synthesis:** Stitching sections together without integrating them produces an incoherent response. Rewrite to unify.
- **Contradiction averaging:** If candidates disagree, do not hedge — reason about which is more accurate and commit to it.
- **Omitting details:** If a candidate covers a subtopic with more depth, preserve that depth in the synthesis.
—
**TERMINATION**
Call 'finish' with 'solution_report' (your complete synthesized response) and 'reason' (a concise account of how you combined the candidates and resolved any conflicts) after you have read and compared all candidates.

Figure 17: AggAgent system prompt for deep research tasks.

AggAgent tool descriptions

```
{
  "type": "function",
  "function": {
    "name": "get_solution",
    "description": "Retrieves the final content from trajectories'
        last step. Returns a list of {trajectory_id, content}
        entries.",
    "parameters": {
      "type": "object",
      "properties": {
        "trajectory_id": {
          "type": "integer",
          "description": "Trajectory index. Omit to retrieve all
              trajectories."
        }
      },
      "required": [],
      "additionalProperties": false
    }
  },
  "strict": true
},
{
  "type": "function",
  "function": {
    "name": "search_trajectory",
    "description": "Searches for keywords or phrases within a
        single trajectory. Returns top matching steps ranked by
        relevance score.",
    "parameters": {
      "type": "object",
      "properties": {
        "trajectory_id": {"type": "integer", "description": "
            Trajectory index to search within."},
        "query": {"type": "string", "description": "Search term or
            phrase."},
        "role": {"type": "string", "enum": ["tool", "assistant"], "
            description": "Optional. Filter to 'tool' steps (actual
```

```
            environment observations) or 'assistant' steps only.
            Omit to search all steps."},
        "k": {"type": "integer", "description": "Max matches to
            return (default 5, max 10).", "default": 5}
      },
      "required": ["trajectory_id", "query"],
      "additionalProperties": false
    }
  },
  "strict": true
},
{
  "type": "function",
  "function": {
    "name": "get_segment",
    "description": "Reads the full content of a contiguous range of
         steps from a trajectory (max 5). Use after
        search_trajectory to inspect a step in full with its
        surrounding context.",
    "parameters": {
      "type": "object",
      "properties": {
        "trajectory_id": {"type": "integer", "description": "
            Trajectory index."},
        "start_step": {"type": "integer", "description": "Start
            step (inclusive)."},
        "end_step": {"type": "integer", "description": "End step (
            inclusive); end_step - start_step <= 4."}
      },
      "required": ["trajectory_id", "start_step", "end_step"],
      "additionalProperties": false
    }
  },
  "strict": true
},
{
  "type": "function",
  "function": {
    "name": "finish",
    "description": "Submits the final synthesized solution.",
    "parameters": {
      "type": "object",
      "properties": {
        "solution": {"type": "string", "description": "A
            comprehensive, standalone solution as a single string
            with two XML sections: <explanation>detailed reasoning
            leading to the answer</explanation><answer>the exact
            answer</answer>. The explanation must be self-contained
            and make sense without any reference to trajectories or
            aggregation."},
        "reason": {"type": "string", "description": "Meta-reasoning
             explaining your decision: how you evaluated the
            trajectories, what evidence you relied on, and how you
            resolved any conflicts or inconsistencies."}
      },
    "required": ["solution", "reason"]
    }
  },
  "strict": true
}
```

Figure 18: AggAgent tool descriptions.

**Variants of AggAgent finish tool**

```
Deep Research Tasks:
"solution": {"type": "string", "description": "The synthesized long
    -form response. Write it as a complete, standalone report -- do
    not reference trajectories, agents, or aggregation. Cite using <
    cite url=\"...\">...</cite> if necessary."}

Qwen3.5-122B-A10B on Agentic Search Tasks:
"solution": {"type": "string", "description": "A comprehensive,
    standalone solution as a single string in the following format:\
    nExplanation: {detailed reasoning leading to the answer}\nExact
    Answer: {the exact answer}\nThe explanation must be self-
    contained and make sense without any reference to trajectories
    or aggregation."}
```

Figure 19: Variants of AggAgent finish tool.