DAGent: Evaluate-then-Grow Planning for Deep Research Agents

Hanwen Liu[1,2] **Yuanfu Sun**[1,2] **Qiaoyu Tan**[2*]
[1]New York University
[2]New York University Shanghai
{hl6825, ys6310, qt2097}@nyu.edu

Abstract

Deep research tasks require agents to navigate large knowledge spaces, synthesize evidence across many sources, and adapt their plans as intermediate findings emerge. Directed acyclic graph (DAG)-based multi-agent systems are well suited to this setting because they support parallel execution and isolate each sub-task within a focused dependency context. Yet existing DAG-based agents typically instantiate a task-level plan before execution and then patch that plan during execution. This *Plan-then-Patch* strategy is brittle for deep research: the system commits most strongly when its evidence is weakest, and often wastes computation on branches that should not have been planned in the first place. To address this brittleness, we propose **DAGent**, a DAG-based multi-agent framework that introduces **Evaluate-then-Grow** incremental planning. Instead of committing to a full DAG upfront, an Orchestrator grows the task graph one batch at a time, conditioning each subsequent expansion on confidence and uncertainty signals from completed nodes. To support long-horizon evidence use without overloading each sub-task, DAGent maintains a hierarchical context layer that propagates compact QueryDocs by default while preserving full execution traces for on-demand recall. The recorded DAG topology in turn admits structural RL signals that an outcome-only objective cannot define; we instantiate this with **DAGRPO**, a GRPO adaptation that augments the objective with a topology-conditioned credit on Executor rollouts and a structural compliance regularization on Orchestrator plans. Across BrowseComp-Plus, GAIA, and xbench-DeepSearch, DAGent surpasses the strongest training-free baseline by 5.3 / 5.8 / 2.0 points at the Qwen3-235B-A22B scale, with the lead replicating across four open-source backbones from four vendors and extending to GPT-5 at 327K context. At the Qwen3-8B scale, DAGRPO further improves over a same-budget outcome-only GRPO baseline by 3.0 average Pass@1 points. A same-architecture comparison shows that incremental, evidence-conditioned planning reaches higher accuracy at lower per-task token, tool-call, and step footprints than its Plan-then-Patch counterpart, so its accuracy gain comes from more targeted evidence expansion rather than additional computation. Code is available at https://github.com/hanwenliu6825/DAGent.

1 Introduction

The rapid advancement of large language models has given rise to a new class of information-intensive tasks broadly referred to as *deep research*: tasks that require an AI system to autonomously navigate large knowledge spaces, synthesize information across dozens or hundreds of sources, and resolve complex queries through extensive multi-step reasoning. A growing body of work has shown that

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

DAG-based multi-agent systems are an effective architecture for such tasks, decomposing a query into a directed acyclic graph of sub-tasks so that independent branches execute in parallel and each agent operates within a focused context window informed only by its dependencies rather than the full upstream history. The paradigm has been applied to parallel function calling [19], web search [6], multi-hop retrieval [46, 5], and long-horizon task decomposition [22].

Deep research, however, places greater demands on the *planning mechanism* than traditional information search: intermediate findings can redirect subsequent investigation and invalidate assumptions that looked reasonable at the outset, making static or semi-static planning a poor fit. Recent DAG-based deep research systems acknowledge this through execution-time editing. Flash-Searcher [38] produces its complete plan in a single decomposition call and then periodically updates it. FlowSearch [15] initializes its knowledge-flow graph over several pre-execution planner iterations and then lets a separate refiner rewrite the graph through six edit operations. In both systems a task-covering plan is committed before any node has executed, and later edits patch that standing plan. We term this paradigm **Plan-then-Patch**: it makes its largest planning commitment precisely when the system knows least about the problem, and the resulting trajectories mix plan commitment with plan revision, degrading answer accuracy and wasting computation as task complexity grows.

Therefore, we argue that benefiting fully from DAG-based decomposition in deep research requires both the planning paradigm and the reinforcement learning (RL) objective to reflect the fact that the DAG is built incrementally during execution.

To this end, we propose **DAGent**, a DAG-based multi-agent framework that replaces Plan-then-Patch with **Evaluate-then-Grow** incremental planning; Figure 1 contrasts the two paradigms. Instead of committing to a full task decomposition upfront, an Orchestrator grows the task DAG one batch of nodes at a time, using feedback from each completed node to decide what to plan next; the newly added nodes are then executed in parallel, each by a ReAct Executor that runs a standard ReAct loop [54] with domain-specific tools. To support long-horizon evidence use without overloading each sub-task, DAGent maintains a hierarchical context layer on top of this incremental DAG: each node exposes a compact summary for default propagation to downstream nodes, and a complete execution trace from which a downstream ReAct Executor can retrieve targeted information on demand. Figure 2 illustrates the overall architecture.

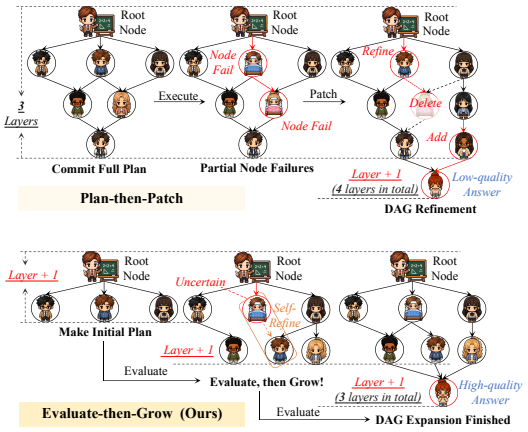


Figure 1: Comparison of Plan-then-Patch (top) and Evaluate-then-Grow (bottom) paradigms.

The recorded DAG topology that Evaluate-then-Grow produces enables RL signals that outcome-only objectives cannot define: rollouts at different DAG positions contribute unequally to the final answer, yet an outcome-only GRPO objective assigns the trajectory-level reward uniformly across all rollouts. We exploit this with **DAGRPO**, a DAG-conditioned RL adaptation that augments a GRPO-style objective [42] with two structural signals: a topology-conditioned credit on Executor rollouts based on whether their outputs feed the final synthesis, and a structural compliance regularization on Orchestrator plans that fail validation. Both signals are computed from the recorded trajectory alone, and the credit is uniquely determined by it because the DAG is append-only: no node is ever deleted or rewired, so the context a node saw when it ran coincides with its ancestry in the final graph.

We evaluate DAGent on BrowseComp-Plus, GAIA, and xbench-DeepSearch, covering local retrieval, web search, and Chinese-language deep research. Across backbones from Qwen3-8B to Qwen3-235B-A22B, DAGent outperforms the strongest training-free baseline on all three benchmarks, by 5.3 / 5.8 / 2.0 points at Qwen3-235B-A22B. At Qwen3-32B, the strongest baseline is the authors' released implementation of FlowSearch, which DAGent leads by 4.0 / 3.8 / 2.0 points. The lead over the strongest training-free baseline replicates across four open-source backbones from four vendors, and, scaled to GPT-5 at 327K context, DAGent surpasses the strongest reported agent system on each benchmark in Figure 4 by 5.7 / 2.5 / 2.0 points. DAGRPO further improves Qwen3-8B DAGent

by 3.0 average points over its outcome-only GRPO counterpart at the same training budget. Our **contributions** are summarized below.

1. **Evaluate-then-Grow planning.** We introduce an incremental DAG planning paradigm that departs from Plan-then-Patch: the task DAG is grown during execution through structured node-status evaluation rather than committed to upfront and reactively edited. To the best of our knowledge, DAGent is the first deep research agent to condition planning decisions on structured per-node feedback that summarizes the sub-task outcome, rationale, and reliability.
2. **DAGRPO.** We present a DAG-conditioned RL adaptation for deep research agents that exploits the incremental DAG through topology-conditioned credit assignment and a structural compliance regularization on Orchestrator plans.
3. **Broad empirical advantage.** We show that the advantage of DAGent is broad: it holds across BrowseComp-Plus, GAIA, and xbench-DeepSearch, replicates across four open-source backbones from four vendors, and persists when scaled to GPT-5 at 327K context. We further show that the structural signals of DAGRPO add gains on top of an outcome-only GRPO baseline at the same training budget, and a sensitivity sweep over the topology-conditioned credit confirms that the gain is not a hyperparameter artifact.

2 Related Work

2.1 Deep Research Agents

Long-horizon deep research stresses both the context window and the planning depth of an LLM, motivating two complementary lines of work that extend the linear reasoning-and-acting paradigm [54]. The first manages context across a long linear trajectory: summary-based methods repeatedly compress the history into a summary state [14, 51, 18, 26], and folding-style methods fold completed segments of the trajectory into summaries [45, 55, 41]. The second decomposes the task into a structured plan: DAG planners decompose a query into a dependency graph of tool calls, sub-queries, or sub-tasks [19, 6, 46, 5, 59, 25], and Plan-then-Patch deep research agents commit a task-covering plan before any node executes, in one decomposition call [38] or over several planner iterations [15], and then edit it during execution; concurrent variants explore node-scoped replanning [22] and hierarchical-outline planning [24]. A growing set of proprietary deep research products [34, 28, 44, 31, 29] and agent frameworks [39, 61, 37] report results on public deep research benchmarks. Across these lines, planning for deep research either operates on linear trajectories with no multi-parent dependencies, or commits the structured plan before any node has executed. DAGent differs by growing the DAG one batch at a time under structured per-node signals: the confidence and uncertainty fields of each completed node drive every subsequent expansion of the graph.

2.2 Structure-Aware RL-Based Optimization for LLM Agents

Group-based policy gradient methods such as GRPO [42] and DAPO [56] are widely used to train LLM agents, with adaptations to summary-based and fold-based long-horizon agents [51, 45]. With an outcome-only reward, these objectives assign the trajectory-level reward uniformly across rollout tokens and ignore intra-trajectory structure. Another line of work injects finer-grained structural signals into the advantage at different granularities: turn-level [48], step-level via anchor-state grouping [11], tree-structured rollouts [16], and tool-use DAGs with a graph-based reward [27]. GAP [50] trains graph-based tool planning with an outcome-only reward, and DeepPlanner [10], designed for deep research planning, shapes GRPO advantages with an entropy-based term without exploiting DAG topology. Concurrent work on multi-agent RL assigns credit by role [13], by counterfactual removal of an agent [23], or over a learned communication topology among a fixed set of agents [2]; Appendix B.3 contrasts these with DAGRPO. A topology-conditioned credit over the answer-feeding chain is uniquely determined by the recorded trajectory only when the graph is append-only. Plan-then-Patch trajectories do not guarantee this property, since a node can be planned, executed, and later deleted or rewired, so its ancestry in the final graph need not match what the synthesis actually read. DAGRPO instantiates this signal on top of Evaluate-then-Grow.

3 Method

Figure 2 illustrates the DAGent architecture. Section 3.1 introduces Evaluate-then-Grow incremental planning and the hierarchical context layer built on it; Section 3.2 introduces DAGRPO, a DAG-conditioned RL adaptation that exploits the DAG topology recorded during execution.

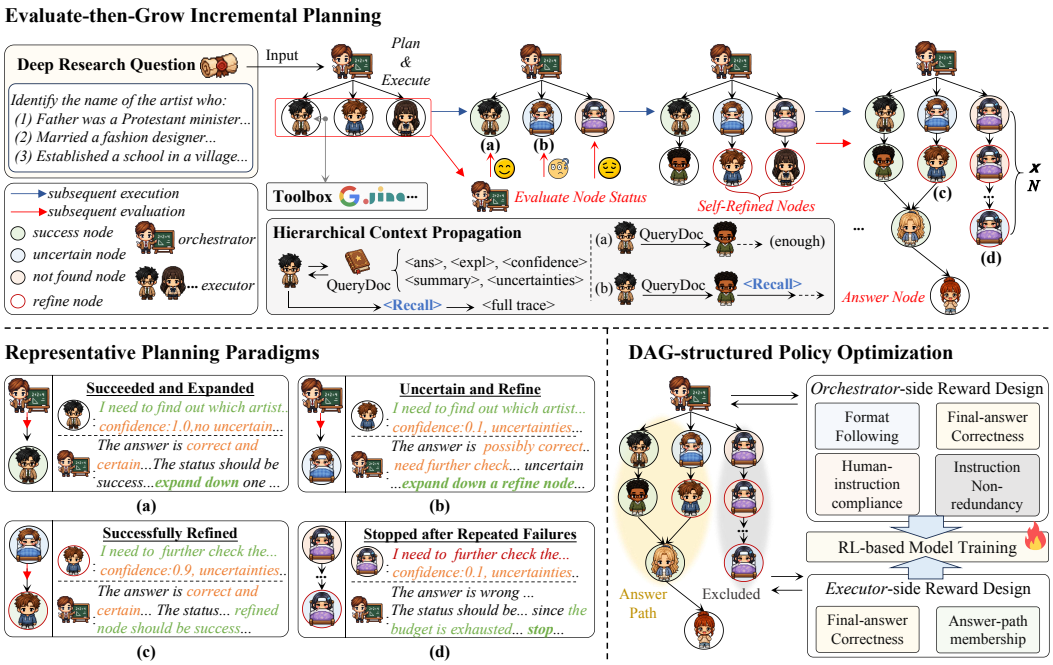


Figure 2: Overview of DAGent. In the top panel, the Orchestrator evaluates each executed batch and grows the DAG until it plans the answer node. Each child node receives the QueryDocs of its parents and can recall details from their full traces on demand. The bottom-left panel shows the planning cases (a–d) marked at nodes of the top panel. The bottom-right panel summarizes DAGRPO (§3.2), in which Executor rollouts outside the answer path (Excluded) receive attenuated credit.

3.1 Evaluate-then-Grow Incremental Planning

Deep research planning is a sequential decision problem under evolving evidence: each completed sub-task may redirect the investigation or invalidate prior assumptions. Unlike Plan-then-Patch agents, which commit to a full DAG before execution and patch it during execution, Evaluate-then-Grow commits one batch at a time, conditioning each subsequent expansion on the structured outputs of completed nodes. We next formalize the DAG state, per-node evidence, the Evaluate-then-Grow loop, and hierarchical context propagation, which together define how DAGent executes a task.

DAG State. A deep research task $\mathcal{T}$ is modeled as a DAG $\mathcal{G} = (V, E)$, where each node $v_i \in V$ is an atomic sub-task and each edge $(v_j, v_i) \in E$ indicates that v_i depends on the output of v_j. The graph is initialized empty and incrementally expanded by the Orchestrator. Each node contains a description δ_i for the sub-task, an execution prompt p_i for the Executor, a dependency set $\mathrm{dep}_i \subset V$ of required upstream nodes, and a status σ_i assigned by the Orchestrator. We also define two special node types: a *refine* node, which re-executes a sub-task with an alternative search strategy, and an *answer* node, which synthesizes the final response from upstream evidence.

Per-Node Evidence. Each node v_i emits a compact structured summary, the *QueryDoc* (D_i):

$$D_i = (\mathrm{ans}_i,\ \mathrm{expl}_i,\ \mathrm{steps}_i,\ \mathrm{conf}_i,\ \mathrm{unc}_i), \tag{1}$$

where ans_i is the answer to the current sub-task, expl_i is a brief explanation of the answer, steps_i summarizes the key execution steps, $\mathrm{conf}_i \in [0, 1]$ denotes the confidence of the Executor, and unc_i

lists uncertainties that may require further verification. The first three fields record the sub-task outcome and rationale, while the last two signal its reliability for Orchestrator evaluation.

The node also keeps the complete execution trace of its Executor as the *InteractionTranscript* (T_i):

$$T_i = \langle (a_1^{(i)}, o_1^{(i)}),\ \ldots,\ (a_n^{(i)}, o_n^{(i)}) \rangle, \tag{2}$$

where $a_t^{(i)}$ is an action (e.g., search or page retrieval) at step t and $o_t^{(i)}$ the corresponding observation.

Evaluate-then-Grow DAG Expansion. At each iteration k, the Orchestrator π_O evaluates the most recently executed batch and produces a planning decision:

$$P_k = \pi_O(\mathcal{T},\ \mathcal{G}_k) = (S_k,\ V_k^{\text{new}}). \tag{3}$$

The status evaluation S_k assigns each node in the latest executed batch a QueryDoc-grounded status: *Success*, indicating confident evidence acquisition; *Uncertain*, indicating evidence that requires further verification or refinement; or *Not Found*, indicating valid execution without relevant evidence. Nodes marked *Uncertain* or *Not Found* spawn refine nodes for further verification or alternative search, with each sub-task capped at three attempts: one initial execution and two refinements. The targeted expansion V_k^{new} then instantiates new nodes with their descriptions, prompts, and dependency edges; nodes within a batch depend only on already executed nodes and execute in parallel. Termination is triggered when the Orchestrator schedules an answer node in V_k^{new}; this node depends on the evidence nodes selected as relevant by the Orchestrator and is required to appear alone in its batch, ensuring access to the intended upstream evidence for final synthesis.

Hierarchical Context Propagation. In long-horizon deep research, the total length of the InteractionTranscripts grows with every executed node, but a downstream node typically needs only the conclusions of its dependencies. Each sub-task is executed by a ReAct Executor π_R equipped with domain-specific tools and RECALLTOOL. Upon completion, π_R produces a QueryDoc and preserves its full interaction history as an InteractionTranscript. By default, a node receives only the QueryDocs of its direct dependencies as upstream context. We call this policy *Selective Propagation*; it bounds the upstream context of a node by its fan-in rather than by the graph depth or the total number of nodes. When the compact QueryDoc is insufficient, e.g., when downstream execution needs the exact wording of a passage or a previously rejected candidate, the Executor can retrieve targeted information from the full transcript of a direct dependency by calling RECALLTOOL:

$$E = \text{Recall}(v_j,\ \textit{goal},\ T_j), \tag{4}$$

where $v_j \in \text{dep}_i$ is a direct dependency, *goal* specifies the information needed by the Executor, and a language model extracts from T_j a focused evidence snippet E conditioned on this goal. RECALLTOOL is a fallback rather than a default: in the Qwen3-32B runs it is invoked in 22.0 / 13.6 / 11.0% of tasks on BrowseComp-Plus / GAIA / xbench-DeepSearch (Appendix D.4).

***Remark*.** Each iteration appends a new batch of nodes with their dependency edges, and no node is ever deleted or rewired, so the recorded DAG is exactly the graph that ran; the structural RL signals of §3.2 are defined on this record.

3.2 DAGRPO: DAG-Conditioned Reinforcement Learning

Every Evaluate-then-Grow training trajectory contains its recorded DAG, which carries information that linear-trajectory methods cannot use: rollouts at different DAG positions contribute unequally to the final answer. Some Executor rollouts lie on the dependency chain feeding the answer node, while others explore alternatives that the synthesis never consumes. An outcome-only GRPO objective assigns the trajectory-level reward $R(\tau) \in \{0, 1\}$ uniformly across all rollouts and ignores this DAG-induced structure. DAG-conditioned Group Relative Policy Optimization (DAGRPO) uses this structure. It keeps the clipped surrogate of GRPO and augments the objective with two structural signals, both applied to the per-rollout reward before group normalization: a *topology-conditioned credit* on Executor rollouts and a *structural compliance regularization* on Orchestrator plans.

Learning Objective. For a task $\mathcal{T}$, we sample G on-policy trajectories from $\pi_{\theta_{\text{old}}}$; each trajectory τ comprises one Orchestrator sub-rollout and the ReAct Executor sub-rollouts it spawns, jointly indexed

by i with response tokens y_i, and is assigned a trajectory-level outcome reward $R(\tau) \in \{0, 1\}$ by an LLM judge that verifies final-answer correctness, as detailed in §4.1. The DAGRPO objective is

$$\mathcal{J}_{\text{DAGRPO}}(\theta) \;=\; \mathbb{E}\left[\frac{1}{|y_\tau|}\sum_{i,\,t}\min\Big(\rho_{i,t}(\theta)\,\hat{A}_i^{(u)},\; \text{clip}\big(\rho_{i,t}(\theta),\,1-\epsilon_\ell,\,1+\epsilon_h\big)\,\hat{A}_i^{(u)}\Big)\right], \tag{5}$$

where $u \in \{\text{orch}, \text{exec}\}$ denotes the role of sub-rollout i, $\rho_{i,t}(\theta)$ is the per-token importance ratio between the current and old policies, $|y_\tau| = \sum_i |y_i|$ is the total response-token count in τ, $\epsilon_h > \epsilon_\ell$ gives the DAPO-style asymmetric clip [56], and the role-separated group-relative advantage is

$$\hat{A}_i^{(u)} \;=\; \frac{\tilde{r}_i^{(u)} - \mu_u}{\sigma_u}, \qquad \tilde{r}_i^{(u)} = \begin{cases} R(\tau) & u = \text{exec},\; v_i \in S(\tau) \\ \alpha\, R(\tau) & u = \text{exec},\; v_i \notin S(\tau) \\ R(\tau) + r_{\text{proc}}(\tau) & u = \text{orch}, \end{cases} \tag{6}$$

where (μ_u, σ_u) are computed within the corresponding role group: the Orchestrator group contains the G Orchestrator sub-rollouts, one per sampled trajectory, while the Executor group contains all retained Executor rollouts spawned by them (Appendix C.4). The coefficient $\alpha \in [0, 1]$ attenuates the credit of off-chain Executor rollouts, $S(\tau)$ denotes the answer-inclusive closure, and $r_{\text{proc}}(\tau) \le 0$ is the structural compliance regularization; both signals are detailed below. We also include a small KL-to-reference loss outside Eq. (5) to stabilize long-horizon agent training. Additional details and the reasons behind these choices are given in Appendix B.

Topology-Conditioned Credit. Let t_{answer} denote the answer node scheduled in the terminating iteration. We define the *answer-inclusive closure* $S(\tau)$ as t_{answer} together with all of its ancestors in $\mathcal{G}$: the set of ReAct Executor rollouts whose outputs directly or transitively fed the final synthesis. Executor rollouts in $S(\tau)$ (on-chain) receive the full $R(\tau)$, and those outside it (off-chain) receive $\alpha R(\tau)$. When τ contains no answer node, we set $\alpha = 1$ as a fallback and recover the outcome-only GRPO behavior. The closure is read off the recorded graph without re-execution, and Appendix B explains why it cannot be exploited by inflating dependency edges during training.

Structural Compliance Regularization. Evaluate-then-Grow imposes structural constraints on Orchestrator decisions, requiring each plan to be syntactically valid, consistent with the incremental execution loop, and non-redundant. The Orchestrator must output parseable JSON with valid dependency references; assign every node in the previous batch an explicit *Success*, *Uncertain*, or *Not Found* status; plan at least one new node per turn; expand each refinable *Uncertain* or *Not Found* node with a refine node within the three-attempt budget; schedule the answer node as a singleton batch so final synthesis can access all upstream evidence; and avoid duplicate sub-task prompts. Let $N(\tau)$ denote the number of Orchestrator turns in τ that violate these constraints. We define $r_{\text{proc}}(\tau) = -\lambda_{\text{proc}} \min(N(\tau), K_{\text{proc}})$, where $\lambda_{\text{proc}} \in (0, 1]$ controls the regularization scale and K_{proc} caps its magnitude. This term is added to the trajectory-level Orchestrator reward to penalize structurally invalid planning decisions.

***Remark*.** Two components of DAGent build on the DAG that Evaluate-then-Grow records: DA-GRPO uses it to define credit-assignment signals that an outcome-only objective cannot express, and the hierarchical context layer bounds the upstream context of each node by its fan-in rather than by the trajectory length. Table 2 measures the incremental loop directly: with all context components kept, replacing it with Plan-then-Patch costs 4.7 overall Pass@1 points, and replacing it with a single upfront plan costs 12.2. Algorithm 1 in Appendix A summarizes the complete DAGent workflow.

4 Experiments

We organize the experiments around four research questions ***(RQs)***. ***RQ1***: How does DAGent compare with deep research baselines on representative benchmarks in both training-free and training-based settings? ***RQ2***: Does the lead replicate across backbone scales, vendors, and a frontier closed-source backbone? ***RQ3***: Which planning, context, and DAGRPO components drive the gains of DAGent? ***RQ4***: How efficient is DAGent compared with the Plan-then-Patch alternative?

4.1 Experimental Setup

Benchmarks. We evaluate on three deep research benchmarks, each with a fixed retrieval backend that every method shares. BrowseComp-Plus [7] pairs the original BrowseComp queries [47] with a verified corpus; retrieval is local dense retrieval over this corpus with Qwen3-Embedding-8B [58], and we adopt the 680/150 train/evaluation split of Sun et al. [45], balanced across easy, medium, and hard difficulty. GAIA [30] is evaluated on its 103-task text-only validation subset with live Google search through Serper [40] and page extraction through Jina [17]; only the GPT-5 comparison in Figure 4 uses the full 165-task validation set, as prior work does, and adds image, document, and audio inspector tools for its multi-modal questions (Appendix D.1). The 2505 release of xbench-DeepSearch [4, 52] contains 100 Chinese-language tasks and uses the same Serper and Jina stack as GAIA. Checkpoints trained on BrowseComp-Plus transfer zero-shot to the other two benchmarks.

Baselines. We compare against five agent families under the same backbone and tool stack: (1) ReAct Agent [54] at 32K and 109K context with a pre-overflow warning as a termination control; (2) Summary Agent [14, 51, 18, 26], which invokes a summary on overflow with up to 10 sessions; (3) Fold Agent [45, 55, 41], representative of folding-style methods; (4) Flash-Searcher [38], a Plan-then-Patch DAG agent reproduced from the authors' release with summary interval 4 to fit the 32K-per-sub-task budget; and (5) FlowSearch [15], a Plan-then-Patch DAG agent evaluated at Qwen3-32B only with the authors' released implementation and its execution-conditioned refiner (the Coordinator) enabled (configuration in Appendix C.3). At Qwen3-8B we additionally report the three trainable baselines under outcome-only GRPO, and DAGent under both outcome-only GRPO with $\alpha = 1.0$ and no structural compliance regularization (denoted `GRPO-DAGent`) and DAGRPO (denoted `DAGRPO-DAGent`); the GRPO variant is the same-budget direct baseline for DAGRPO.

Implementation. Backbones are Qwen3-8B, Qwen3-32B, and Qwen3-235B-A22B [53] for training-free evaluation and Qwen3-8B for RL training, all with thinking disabled and, per sub-task, a 32K-token response budget on top of an 8K-token prompt (Appendix C.2). RL is trained on BrowseComp-Plus with the DAPO asymmetric clip [56], the topology-conditioned credit at $\alpha = 0.5$, and the structural compliance regularization. Inference uses greedy decoding; Pass@1 is scored by an LLM judge validated against three human annotators (Cohen's $\kappa = 0.95$; Appendix C.5), and the same judge provides the outcome reward $R(\tau)$ during RL training.

4.2 Main Results (RQ1, RQ2)

Table 1 reports Pass@1 across three benchmarks for DAGent and the baselines of §4.1 at three Qwen3 backbone scales, in training-free and training-based settings.

Observation 1: ***DAGent leads the strongest training-free baseline at every Qwen3 backbone scale, from 8B to 235B-A22B, on all three benchmarks.*** At Qwen3-32B, DAGent reaches 47.3 / 55.3 / 65.0 on BrowseComp-Plus / GAIA / xbench-DeepSearch, 4.0 / 3.8 / 2.0 points above FlowSearch, the strongest baseline at this scale, and 6.0 / 10.6 / 7.0 points above the strongest of the remaining baselines. The lead persists at 8B (+5.3 / +7.8 / +5.0) and 235B-A22B (+5.3 / +5.8 / +2.0), and on every difficulty split DAGent matches or exceeds every baseline, with two ties at 235B-A22B; relative gains grow with difficulty at 8B and 32B.

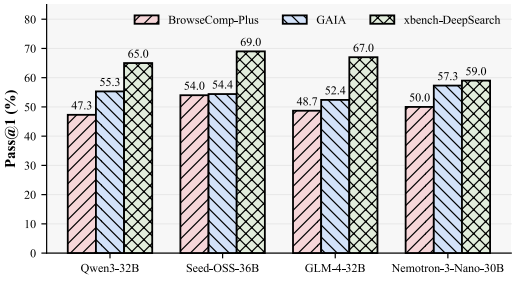


Figure 3: Cross-backbone DAGent Pass@1 (%); per-baseline breakdown in Appendix D.1.

Observation 2: ***DAGRPO further improves Qwen3-8B DAGent over a same-budget outcome-only GRPO baseline by 3.0 average points, isolating the gain to its structural signals.*** DAGRPO improves the BrowseComp-Plus / GAIA / xbench-DeepSearch Pass@1 of DAGent from 40.0 / 46.6 / 60.0 in training-free mode to 49.6 / 53.4 / 65.7 over three seeds, exceeding the GRPO baseline by 3.6 / 2.6 / 2.7 points; per-seed values appear in Appendix E.1. The GRPO and DAGRPO runs differ only in α and the structural compliance regularization, so this margin comes from the structural signals rather

Table 1: Main results on BrowseComp-Plus, GAIA, and xbench-DeepSearch: Pass@1 (%), best result per block in **bold**. Training-free rows use a single greedy inference pass; training-based rows report mean±std (sample standard deviation) over three training seeds on the benchmark averages. FlowSearch is evaluated with the authors' released implementation at Qwen3-32B only (§4.1).

Backbone	Max #Token	Method	BrowseComp-Plus				GAIA				xbench-DS
			Easy	Med.	Hard	Avg.	L1	L2	L3	Avg.	Avg.
Training-free											
Qwen3-8B	32K	ReAct Agent	46.0	14.0	0.0	20.0	38.5	23.1	25.0	29.1	39.0
	109K	ReAct Agent	64.0	18.0	2.0	28.0	46.2	26.9	25.0	34.0	44.0
	32K×N	Summary Agent	76.0	20.0	2.0	32.7	51.3	26.9	16.7	35.0	54.0
	32K×N	Fold Agent	78.0	24.0	2.0	34.7	48.7	28.8	16.7	35.0	52.0
	32K×N	Flash-Searcher	78.0	24.0	2.0	34.7	51.3	32.7	25.0	38.8	55.0
	32K×N	**DAGent (Ours)**	**84.0**	**30.0**	**6.0**	**40.0**	**59.0**	**40.4**	**33.3**	**46.6**	**60.0**
Qwen3-32B	32K	ReAct Agent	50.0	16.0	2.0	22.7	46.2	28.8	8.3	33.0	54.0
	109K	ReAct Agent	66.0	24.0	4.0	31.3	51.3	36.5	16.7	39.8	58.0
	32K×N	Summary Agent	80.0	36.0	0.0	38.7	51.3	28.8	25.0	36.9	55.0
	32K×N	Fold Agent	82.0	40.0	2.0	41.3	56.4	38.5	16.7	42.7	55.0
	32K×N	Flash-Searcher	80.0	34.0	2.0	38.7	56.4	40.4	25.0	44.7	58.0
	32K×N	FlowSearch	84.0	42.0	4.0	43.3	66.7	46.2	25.0	51.5	63.0
	32K×N	**DAGent (Ours)**	**88.0**	**46.0**	**8.0**	**47.3**	**69.2**	**50.0**	**33.3**	**55.3**	**65.0**
Qwen3-235B-A22B	32K	ReAct Agent	84.0	32.0	10.0	42.0	56.4	55.8	33.3	53.4	60.0
	109K	ReAct Agent	90.0	50.0	16.0	52.0	61.5	57.7	**41.7**	57.3	64.0
	32K×N	Summary Agent	88.0	60.0	20.0	56.0	59.0	53.8	25.0	52.4	60.0
	32K×N	Fold Agent	**94.0**	56.0	14.0	54.7	61.5	53.8	25.0	53.4	62.0
	32K×N	Flash-Searcher	90.0	54.0	14.0	52.7	64.1	55.8	25.0	55.3	70.0
	32K×N	**DAGent (Ours)**	**94.0**	**68.0**	**22.0**	**61.3**	**74.4**	**59.6**	**41.7**	**63.1**	**72.0**
Training-based											
Qwen3-8B	32K	GRPO-ReAct Agent	72.0	16.0	4.0	30.7 ± 1.8	41.0	26.9	25.0	32.0 ± 2.9	43.0 ± 2.6
	109K	GRPO-ReAct Agent	76.0	24.0	4.0	34.7 ± 2.4	48.7	30.8	25.0	36.9 ± 1.9	48.0 ± 2.0
	32K×N	GRPO-Summary Agent	80.0	30.0	4.0	38.0 ± 2.0	53.8	30.8	25.0	38.8 ± 1.7	57.0 ± 1.7
	32K×N	GRPO-Fold Agent	84.0	34.0	6.0	41.3 ± 1.3	51.3	32.7	25.0	38.8 ± 2.6	56.0 ± 3.0
Qwen3-8B	32K×N	**GRPO-DAGent**	88.0	40.0	10.0	46.0 ± 1.2	61.5	46.8	**33.3**	50.8 ± 1.1	63.0 ± 1.0
Qwen3-8B	32K×N	**DAGRPO-DAGent**	**90.0**	**48.0**	**10.7**	$\mathbf{49.6 \pm 1.0}$	**64.1**	**50.0**	**33.3**	$\mathbf{53.4 \pm 1.0}$	$\mathbf{65.7 \pm 1.5}$

than the training budget. The same outcome-only GRPO training improves each trainable baseline by 2.9 to 10.7 points, and DAGRPO-DAGent leads the strongest of them by 8.3 / 14.6 / 8.7 points.

Observation 3: The lead replicates across four mid-size open-source backbones from four vendors and persists at frontier scale with GPT-5 at 327K context. Table 9 in Appendix D.1 shows that DAGent maintains the per-backbone top position across Qwen3-32B, Seed-OSS-36B [1], GLM-4-32B [57], and Nemotron-3-Nano-30B [32]; Figure 3 plots its Pass@1 per backbone. DAGent at GPT-5 327K leads the strongest reported baseline by 5.7 / 2.5 / 2.0 points on BrowseComp-Plus / GAIA / xbench-DeepSearch (Figure 4), with GAIA evaluated on its full 165-task validation set with multi-modal tools; the baseline list and number provenance appear in Appendix D.1.

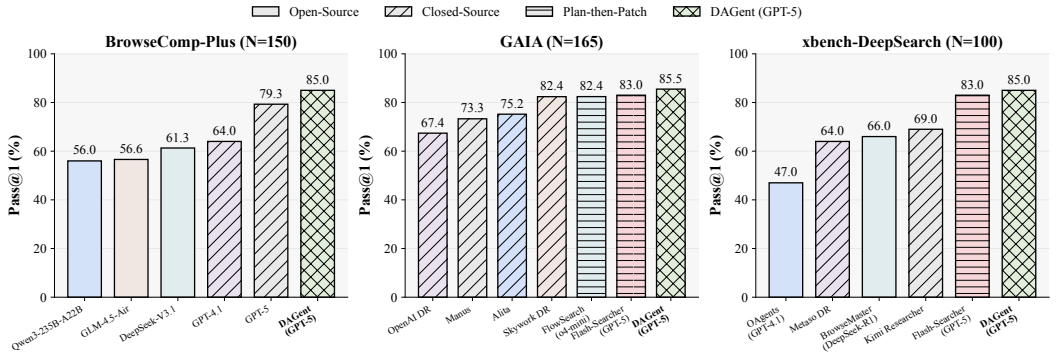


Figure 4: Pass@1 (%) of DAGent with GPT-5 and of state-of-the-art agent systems on BrowseComp-Plus, GAIA, and xbench-DeepSearch; baseline numbers are reported in prior work (Appendix D.1).

Table 2: Component ablation with Qwen3-32B in training-free mode: Pass@1 (%) per benchmark, Overall is the unweighted mean, and Δ is the change from DAGent (Full). All rows use a single greedy inference pass, as in the training-free rows of Table 1.

Method	BrowseComp-Plus		GAIA		xbench-DS		Overall	
	Avg.	Δ	Avg.	Δ	Avg.	Δ	Avg.	Δ
DAGent (Full)	**47.3**	—	**55.3**	—	**65.0**	—	**55.9**	—
Plan-then-Patch variant	42.0	**-5.3**	50.5	**-4.8**	61.0	**-4.0**	51.2	**-4.7**
w/o Evaluate-then-Grow Planning	33.3	**-14.0**	42.7	**-12.6**	55.0	**-10.0**	43.7	**-12.2**
w/o Selective Propagation	34.7	**-12.6**	44.7	**-10.6**	57.0	**-8.0**	45.5	**-10.4**
w/o QueryDoc	39.3	**-8.0**	47.6	**-7.7**	60.0	**-5.0**	49.0	**-6.9**
w/o InteractionTranscript	43.3	**-4.0**	52.4	**-2.9**	63.0	**-2.0**	52.9	**-3.0**

4.3 Ablation Study (RQ3)

Table 2 ablates the planning and context components of training-free DAGent on Qwen3-32B. Table 3 ablates the two DAGRPO signals and sweeps α on Qwen3-8B; its DAGRPO and GRPO-baseline rows reproduce the three-seed means of Table 1, and the remaining rows use a single training seed.

***Observation 4:* Within the same DAGent architecture, Evaluate-then-Grow outperforms Plan-then-Patch, which in turn outperforms a single upfront plan, and removing any of the three context components costs 2.0 to 12.6 points.** Replacing Evaluate-then-Grow with Plan-then-Patch costs 5.3 / 4.8 / 4.0 points on BrowseComp-Plus / GAIA / xbench-DeepSearch; since the variant keeps the refine and add operations of execution-time patching, this gap comes from when the plan is committed, not from whether it can be revised. A single upfront plan without replanning costs another 8.7 / 7.8 / 6.0 points, 14.0 / 12.6 / 10.0 in total. Among the context components, removing Selective Propagation costs the most (12.6 / 10.6 / 8.0), followed by QueryDoc (8.0 / 7.7 / 5.0) and InteractionTranscript (4.0 / 2.9 / 2.0), in the same order on every benchmark; the last row is also the RECALLTOOL ablation, since removing the transcript disables recall (Appendix D.4).

***Observation 5:* Both DAGRPO structural signals improve Pass@1 on all three benchmarks, and every tested $\alpha > 0$ stays above the GRPO baseline, with the best value at $\alpha = 0.5$.** Removing the topology-conditioned credit ($\alpha = 1.0$) costs 2.3 / 1.9 / 1.7 points and removing the structural compliance regularization costs 1.6 / 1.0 / 0.7 points (Table 3), 2.0 and 1.1 points Overall; every per-benchmark difference has the same sign. With the regularization on, $\alpha = 0.5$ is best on every benchmark and every tested $\alpha > 0$ scores above the GRPO baseline on all three, so improving over GRPO does not depend on a tuned α. Setting $\alpha = 0$ gives off-chain rollouts zero reward even in correct trajectories and falls below the GRPO baseline on all three benchmarks, so off-chain rollouts carry learning value and are attenuated rather than removed. Appendix E.2 traces both signals during training: the credit separates on-chain from off-chain Executor rewards, and the regularization lowers the Orchestrator format-warning rate faster than GRPO.

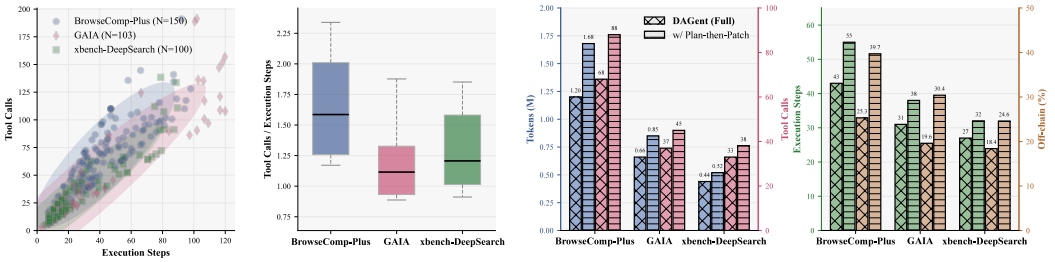
Table 3: DAGRPO component ablation and α sensitivity on Qwen3-8B: Pass@1 (%) per benchmark, Overall is the unweighted mean, and Δ is the change from DAGRPO; the Overall Δ is the mean of the three per-benchmark Δ values and can differ by 0.1 from the difference of the rounded Overall values. The DAGRPO and GRPO-baseline rows are the three-seed means of Table 1; all other rows are single-seed runs with the same training budget.

α	Regularization	Configuration	BrowseComp-Plus		GAIA		xbench-DS		Overall	
			Avg.	Δ	Avg.	Δ	Avg.	Δ	Avg.	Δ
0.00	on	—	44.0	**-5.6**	49.5	**-3.9**	61.0	**-4.7**	51.5	**-4.7**
0.25	on	—	48.7	**-0.9**	52.4	**-1.0**	64.0	**-1.7**	55.0	**-1.2**
0.50	on	**DAGRPO**	**49.6**	—	**53.4**	—	**65.7**	—	**56.2**	—
0.75	on	—	48.0	**-1.6**	52.4	**-1.0**	64.0	**-1.7**	54.8	**-1.4**
1.00	on	w/o topology credit	47.3	**-2.3**	51.5	**-1.9**	64.0	**-1.7**	54.3	**-2.0**
0.50	off	w/o compliance regularization	48.0	**-1.6**	52.4	**-1.0**	65.0	**-0.7**	55.1	**-1.1**
1.00	off	GRPO baseline	46.0	**-3.6**	50.8	**-2.6**	63.0	**-2.7**	53.3	**-3.0**

4.4 Efficiency Analysis (RQ4)

We measure the per-task footprint of DAGent with Qwen3-32B in training-free mode. The 32K×N of Table 1 is a per-sub-task cap shared by all multi-context methods, not a per-task budget; the totals below are what each workflow spends under it. Execution steps count Orchestrator iterations plus the longest Executor trajectory per parallel batch; tool calls count all search, open_page, recall, and plan invocations. Appendix D.3 extends the comparison to Flash-Searcher and FlowSearch and adds external tool calls (search and open_page only) and wall-clock time.

Observation 6: ***DAGent makes more tool calls than execution steps on most tasks, because the nodes of a batch run in parallel and an Executor turn can issue several tool calls; the ratio is highest on BrowseComp-Plus, the benchmark that needs the most tool calls.*** Median tool-call-to-step ratios are 1.6, 1.1, and 1.2 on BrowseComp-Plus, GAIA, and xbench-DeepSearch in Figure 5(a), with the narrowest interquartile range on GAIA. Mean per-task footprints are 42.7 / 31.2 / 26.8 steps, 67.6 / 36.8 / 32.9 tool calls, and 1.20M / 0.66M / 0.44M total input plus output tokens.



(a) (b)

Figure 5: Per-task efficiency of DAGent with Qwen3-32B in training-free mode. (a) Tool calls against execution steps, with 2.2σ confidence ellipses, and the per-task tool-call-to-step ratio. (b) Per-task tokens, tool calls, steps, and off-chain share of DAGent (Full) and the Plan-then-Patch variant.

Observation 7: ***Within the same DAGent architecture, Evaluate-then-Grow is more accurate than Plan-then-Patch on every benchmark while using fewer tokens, tool calls, and execution steps per task.*** Plan-then-Patch commits a larger DAG before observing node-level evidence: the off-chain share of Executor nodes rises from 25.3 / 19.6 / 18.4% to 39.7 / 30.4 / 24.6%, and the variant spends 40 / 29 / 18% more tokens and 28 / 21 / 21% more execution steps per task (Figure 5(b); per-task means in Table 10 of Appendix D.3). It also makes 35 / 27 / 21% more external tool calls and needs 35 / 27 / 23% more wall-clock time per task, while scoring 5.3 / 4.8 / 4.0 Pass@1 points below DAGent (Full) (Table 2). The accuracy gain of Evaluate-then-Grow therefore comes from more targeted evidence expansion rather than from additional computation.

5 Conclusion

We presented DAGent, a DAG-based multi-agent framework for deep research that replaces Plan-then-Patch with Evaluate-then-Grow incremental planning, and DAGRPO, a DAG-conditioned RL adaptation that exploits the recorded topology Evaluate-then-Grow produces. Across BrowseComp-Plus, GAIA, and xbench-DeepSearch, DAGent leads the strongest training-free baseline at every Qwen3 scale, by 5.3 / 5.8 / 2.0 points at Qwen3-235B-A22B, keeps the lead across four open-source backbones from four vendors, and, with GPT-5 at 327K context, surpasses the strongest reported agent systems, including closed-source ones. At Qwen3-32B, DAGent is more accurate than FlowSearch and the same-architecture Plan-then-Patch variant while costing less per task than both. At Qwen3-8B, DAGRPO adds 3.0 average Pass@1 points over a same-budget outcome-only GRPO baseline.

Acknowledgments and Disclosure of Funding

We sincerely thank the anonymous reviewers, the area chair, and the senior area chair for their thoughtful and constructive feedback. This research is supported by the National Natural Science Foundation of China (62402320), the Shanghai Science and Technology Program (24YF2731000), and the NYU Shanghai Boost Fund. The authors declare no competing interests.

References

[1] ByteDance Seed Team. Seed-OSS open-source models. Hugging Face model card, 2025. URL https://huggingface.co/ByteDance-Seed/Seed-OSS-36B-Instruct. Model card for the Seed-OSS-36B-Instruct checkpoint; companion repository at https://github.com/ByteDance-Seed/seed-oss.

[2] Yueyang Cang, Xiaoteng Zhang, Erlu Zhao, Zehua Ji, Yuhang Liu, Yuchen He, Zhiyuan Ning, Yijun Chen, Wenge Que, and Li Shi. Graph-GRPO: Stabilizing multi-agent topology learning via group relative policy optimization. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 20222–20231. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.1010. URL https://aclanthology.org/2026.findings-acl.1010/.

[3] Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. ChatEval: Towards better LLM-based evaluators through multi-agent debate. In *The Twelfth International Conference on Learning Representations*, pages 9079–9093, 2024. URL https://openreview.net/forum?id=FQepisCUWu.

[4] Kaiyuan Chen, Yixin Ren, Yang Liu, Xiaobo Hu, Haotong Tian, Tianbao Xie, Fangfu Liu, Haoye Zhang, Hongzhang Liu, Yuan Gong, Chen Sun, Han Hou, Hui Yang, James Pan, Jianan Lou, Jiayi Mao, Jizheng Liu, Jinpeng Li, Kangyi Liu, Kenkun Liu, Rui Wang, Run Li, Tong Niu, Wenlong Zhang, Wenqi Yan, Xuanzheng Wang, Yuchen Zhang, Yi-Hsin Hung, Yuan Jiang, Zexuan Liu, Zihan Yin, Zijian Ma, and Zhiwen Mo. xbench: Tracking agents productivity scaling with profession-aligned real-world evaluations. arXiv preprint arXiv:2506.13651, 2025. URL https://arxiv.org/abs/2506.13651.

[5] Shengyuan Chen, Chuang Zhou, Zheng Yuan, Qinggang Zhang, Zeyang Cui, Hao Chen, Yilin Xiao, Jiannong Cao, and Xiao Huang. You don't need pre-built graphs for RAG: Retrieval augmented generation with adaptive reasoning structures. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 30270–30278. AAAI Press, 2026. doi: 10.1609/aaai.v40i36.40278. URL https://ojs.aaai.org/index.php/AAAI/article/view/40278.

[6] Zehui Chen, Kuikun Liu, Qiuchen Wang, Jiangning Liu, Wenwei Zhang, Kai Chen, and Feng Zhao. MindSearch: Mimicking human minds elicits deep AI searcher. In *The Thirteenth International Conference on Learning Representations*, pages 90007–90029, 2025. URL https://openreview.net/forum?id=xgtXkyqw1f.

[7] Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Sahel Sharifymoghaddam, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, Yanxi Li, Haoran Hong, Xinyu Shi, Xuye Liu, Hosna Oyarhoseini, Nandan Thakur, Crystina Zhang, Luyu Gao, Wenhu Chen, and Jimmy Lin. BrowseComp-Plus: A fair and disentangled evaluation benchmark for deep search agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 22349–22370. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.1023. URL https://aclanthology.org/2026.acl-long.1023/.

[8] DeepSeek-AI. DeepSeek-V3.1. Hugging Face model card, 2025. URL https://huggingface.co/deepseek-ai/DeepSeek-V3.1.

[9] Mingxuan Du, Benfeng Xu, Chiwei Zhu, Licheng Zhang, Xiaorui Wang, and Zhendong Mao. DeepResearch Bench: A comprehensive benchmark for deep research agents. In *The Fourteenth International Conference on Learning Representations*, pages 42414–42448, 2026. URL https://openreview.net/forum?id=hQ0K2Hhq7H.

[10] Wei Fan, Wenlin Yao, Zheng Li, Feng Yao, Xin Liu, Liang Qiu, Qingyu Yin, Yangqiu Song, and Bing Yin. DeepPlanner: Scaling planning capability for deep research agents via advantage shaping. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 7510–7525. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.370. URL https://aclanthology.org/2026.findings-acl.370/.

[11] Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *Advances in Neural Information Processing Systems*, volume 38, pages 46375–46408. Curran Associates, Inc., 2025. doi: 10.52202/085713-1544. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/420c9f777c0b4f78d515e53cf74d58b2-Abstract-Conference.html.

[12] GLM-4.5 Team, Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, et al. GLM-4.5: Agentic, reasoning, and coding (ARC) foundation models. arXiv preprint arXiv:2508.06471, 2025. URL https://arxiv.org/abs/2508.06471.

[13] Haoyang Hong, Jiajun Yin, Yuan Wang, Jingnan Liu, Zhe Chen, Ailing Yu, Ji Li, Zhiling Ye, Hansong Xiao, Yefei Chen, Hualei Zhou, Yun Yue, Minghui Yang, Chunxiao Guo, Junwei Liu, Peng Wei, and Jinjie Gu. Multi-agent deep research: Training multi-agent systems with M-GRPO. arXiv preprint arXiv:2511.13288, 2025. URL https://arxiv.org/abs/2511.13288.

[14] Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. HiAgent: Hierarchical working memory management for solving long-horizon agent tasks with large language model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32779–32798. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.1575. URL https://aclanthology.org/2025.acl-long.1575/.

[15] Yusong Hu, Runmin Ma, Yue Fan, Jinxin Shi, Zongsheng Cao, Yuhao Zhou, Jiakang Yuan, Shuaiyu Zhang, Shiyang Feng, Xiangchao Yan, Shufei Zhang, Wenlong Zhang, Lei Bai, and Bo Zhang. FlowSearch: Advancing deep research with dynamic structured knowledge flow. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 21212–21245. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.971. URL https://aclanthology.org/2026.acl-long.971/.

[16] Yuxiang Ji, Ziyu Ma, Yong Wang, Guanhua Chen, Xiangxiang Chu, and Liaoni Wu. Tree search for LLM agent reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, pages 87362–87388, 2026. URL https://openreview.net/forum?id=ZpQwAFhU13.

[17] Jina AI. Reader. GitHub repository, 2024. URL https://github.com/jina-ai/reader. Accessed 2 November 2026.

[18] Minki Kang, Wei-Ning Chen, Dongge Han, Huseyin A. Inan, Lukas Wutschitz, Yanzhi Chen, Robert Sim, and Saravan Rajmohan. ACON: Optimizing context compression for long-horizon LLM agents. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 55321–55375. PMLR, 2026. URL https://proceedings.mlr.press/v306/kang26b.html.

[19] Sehoon Kim, Suhong Moon, Ryan Tabrizi, Nicholas Lee, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. An LLM compiler for parallel function calling. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 24370–24391. PMLR, 2024. URL https://proceedings.mlr.press/v235/kim24y.html.

[20] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. 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*, SOSP '23, pages 611–626, New York, NY, USA, 2023. Association for Computing Machinery. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

[21] J. Richard Landis and Gary G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174, 1977. doi: 10.2307/2529310. URL https://www.jstor.org/stable/2529310.

[22] Yunfan Li, Bingbing Xu, Xueyun Tian, Xiucheng Xu, and Huawei Shen. Beyond entangled planning: Task-decoupled planning for long-horizon agents. arXiv preprint arXiv:2601.07577, 2026. URL https://arxiv.org/abs/2601.07577.

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

[24] Zijian Li, Xin Guan, Bo Zhang, Shen Huang, Houquan Zhou, Shaopeng Lai, Ming Yan, Yong Jiang, Pengjun Xie, Fei Huang, Jun Zhang, and Jingren Zhou. WebWeaver: Structuring web-scale evidence with dynamic outlines for open-ended deep research. In *The Fourteenth International Conference on Learning Representations*, pages 4173–4217, 2026. URL https://openreview.net/forum?id=MtNCJjlrKt.

[25] Yexiong Lin, Shanshan Ye, Yu Yao, Zhen Fang, Bo Han, and Tongliang Liu. SquidAgent: Parallelize wisely, coordinate efficiently. In *Advances in Neural Information Processing Systems*, 2026. URL https://arxiv.org/abs/2610.08647.

[26] Miao Lu, Weiwei Sun, Weihua Du, Zhan Ling, Xuesong Yao, Kang Liu, and Jiecao Chen. Beyond the context window: Scaling agentic RL via end-to-end optimized context compression. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 21074–21125. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.966. URL https://aclanthology.org/2026.acl-long.966/.

[27] Yifu Lu, Shengjie Liu, and Li Dong. OrchDAG: Complex tool orchestration in multi-turn interactions with plan DAGs. In *NeurIPS 2025 Workshop on Multi-Turn Interactions in Large Language Models*, 2025. URL https://openreview.net/forum?id=uZE8mTYvHE.

[28] Manus AI. Manus: Hands On AI. Official website, 2025. URL https://manus.im/. The homepage reported GAIA results in its Benchmarks section at launch; archived snapshot of 8 March 2025 at https://web.archive.org/web/20250308015504/https://manus.im/.

[29] Metaso. Xin SOTA! Rang "Shendu Yanjiu" geng shen yidiandian [New SOTA! Making "Deep Research" a little deeper]. WeChat official-account post, 2025. URL https://mp.weixin.qq.com/s/-w48s1XuIEcWg6Rd3ZGRXg. In Chinese. Post on Metaso's official WeChat account (bylined "Metaso AI Search"), 15 July 2025, announcing the free "Deep Research" (Shendu Yanjiu) mode and reporting BrowseComp and xbench-DeepSearch results; product page at https://metaso.cn/.

[30] Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for general AI assistants. In *The Twelfth International Conference on Learning Representations*, pages 9025–9049, 2024. URL https://openreview.net/forum?id=fibxvahvs3.

[31] Moonshot AI. Kimi-Researcher: End-to-end RL training for emerging agentic capabilities. Blog post, 2025. URL https://moonshotai.github.io/Kimi-Researcher/.

[32] NVIDIA. Nemotron 3 Nano: Open, efficient mixture-of-experts hybrid Mamba-Transformer model for agentic reasoning. arXiv preprint arXiv:2512.20848, 2025. URL https://arxiv.org/abs/2512.20848.

[33] OpenAI. GPT-4o mini: advancing cost-efficient intelligence. OpenAI announcement, 2024. URL https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/. Published 18 July 2024.

[34] OpenAI. Introducing deep research. OpenAI announcement, 2025. URL https://openai.com/index/introducing-deep-research/.

[35] OpenAI. Introducing GPT-4.1 in the API. OpenAI announcement, 2025. URL https://openai.com/index/gpt-4-1/.

[36] OpenAI. GPT-5 system card. OpenAI, 2025. URL https://openai.com/index/gpt-5-system-card/. Also available as arXiv:2601.03267, https://arxiv.org/abs/2601.03267.

[37] Xianghe Pang, Shuo Tang, Rui Ye, Yuwen Du, Yaxin Du, and Siheng Chen. BrowseMaster: Towards scalable web browsing via tool-augmented programmatic agent pair. In *NeurIPS 2025 Workshop on Scaling Environments for Agents*, 2025. URL https://openreview.net/forum?id=MYIkHcP10t.

[38] Tianrui Qin, Qianben Chen, Sinuo Wang, Xinyu Yang, Kang Zhu, He Zhu, Dingfeng Shi, Xinxin Liu, Ge Zhang, Jiaheng Liu, Xitong Gao, Yuchen Eleanor Jiang, and Wangchunshu Zhou. Flash-Searcher: Fast and effective web agents via DAG-based parallel execution. In *The Fourteenth International Conference on Learning Representations*, pages 45026–45066, 2026. URL https://openreview.net/forum?id=QuaJ6kJaBm.

[39] Jiahao Qiu, Xuan Qi, Tongcheng Zhang, Xinzhe Juan, Jiacheng Guo, Yifu Lu, Yimin Wang, Zixin Yao, Qihan Ren, Xun Jiang, Xing Zhou, Dongrui Liu, Ling Yang, Yue Wu, Kaixuan Huang, Shilong Liu, Hongru Wang, and Mengdi Wang. Alita: Generalist agent enabling scalable agentic reasoning with minimal predefinition and maximal self-evolution. arXiv preprint arXiv:2505.20286, 2025. URL https://arxiv.org/abs/2505.20286.

[40] Serper. The World's Fastest & Cheapest Google Search API. Official website, 2026. URL https://serper.dev. Accessed 2 November 2026.

[41] Jiaqi Shao, Yufeng Miao, Wei Zhang, and Bing Luo. FoldAct: Efficient and stable context folding for long-horizon search agents. arXiv preprint arXiv:2512.22733, 2025. URL https://arxiv.org/abs/2512.22733.

[42] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. arXiv preprint arXiv:2402.03300, 2024. URL https://arxiv.org/abs/2402.03300.

[43] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys '25, pages 1279–1297, New York, NY, USA, 2025. Association for Computing Machinery. doi: 10.1145/3689031.3696075. URL https://doi.org/10.1145/3689031.3696075.

[44] Skywork AI. Skywork launches Skywork Super Agents globally: The "AI-powered office suite" built on deep research. Press release, PR Newswire, May 2025. URL https://www.newswire.ca/news-releases/skywork-launches-skywork-super-agents-globally-the-ai-powered-office-suite-built-on-deep-research-873710174.html. Reports a GAIA score of 82.42 for the Skywork deep research agent.

[45] Weiwei Sun, Miao Lu, Zhan Ling, Kang Liu, Xuesong Yao, Yiming Yang, and Jiecao Chen. Scaling long-horizon agent via Context Folding. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 116894–116910. PMLR, 2026. URL https://proceedings.mlr.press/v306/sun26x.html.

[46] Prakhar Verma, Sukruta Prakash Midigeshi, Gaurav Sinha, Arno Solin, Nagarajan Natarajan, and Amit Sharma. Plan*RAG: Efficient test-time planning for retrieval augmented generation. In *ICLR 2025 Workshop on Reasoning and Planning for Large Language Models*, 2025. URL https://openreview.net/forum?id=gi9aqlYdBk.

[47] 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. URL https://arxiv.org/abs/2504.12516.

[48] Quan Wei, Siliang Zeng, Chenliang Li, William Brown, Oana Frunza, Wei Deng, Yuriy Nevmyvaka, Yang Katie Zhao, Alfredo Garcia, and Mingyi Hong. Reinforcing multi-turn reasoning in LLM agents via turn-level reward design and credit assignment. In

NeurIPS 2025 Workshop on Multi-Turn Interactions in Large Language Models, 2025. URL https://openreview.net/forum?id=drP7qVUnUt.

[49] Ryan Z.Y. Wong, Jiawei Wang, Junjie Zhao, Li Chen, Yan Gao, Long Zhang, Xuan Zhou, Zuo Wang, Kai Xiang, Ge Zhang, Wenhao Huang, Yang Wang, and Ke Wang. WideSearch: Benchmarking agentic broad info-seeking. In *The Fourteenth International Conference on Learning Representations*, pages 10012–10086, 2026. URL https://openreview.net/forum?id=Q7YUY7zGkZ.

[50] Jiaqi Wu, Qinlao Zhao, Zefeng Chen, Kai Qin, Yifei Zhao, Xueqian Wang, and Yuhang Yao. GAP: Graph-based agent planning with parallel tool use and reinforcement learning. In *NeurIPS 2025 Workshop on Knowledge Graphs & Agentic Systems Interplay (NORA)*, 2025. URL https://openreview.net/forum?id=7bJIVHEvLm.

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

[52] xbench. xbench-DeepSearch. Hugging Face dataset card, 2025. URL https://huggingface.co/datasets/xbench/DeepSearch. Dataset card for the 2505 release of xbench-DeepSearch; a later release is published as xbench/DeepSearch-2510. Companion repository at https://github.com/xbench-ai/xbench-evals.

[53] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. arXiv preprint arXiv:2505.09388, 2025. URL https://arxiv.org/abs/2505.09388.

[54] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

[55] Rui Ye, Zhongwang Zhang, Kuan Li, Huifeng Yin, Zhengwei Tao, Yida Zhao, Liangcai Su, Liwen Zhang, Wenbiao Yin, Zile Qiao, Xinyu Wang, Pengjun Xie, Fei Huang, Jingren Zhou, Siheng Chen, and Yong Jiang. AgentFold: Long-horizon web agents with proactive context folding. In *The Fourteenth International Conference on Learning Representations*, pages 154754–154776, 2026. URL https://openreview.net/forum?id=IuZoTgsUws.

[56] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems*, volume 38, pages 113222–113244. Curran Associates, Inc., 2025. doi: 10.52202/085713-3775. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/a4277440d50f1f15d2cb4c14f7e0c0d2-Abstract-Conference.html.

[57] Z.ai. GLM-4-32B-0414. Hugging Face model card, 2025. URL https://huggingface.co/zai-org/GLM-4-32B-0414. Companion repository at https://github.com/zai-org/GLM-4.

[58] Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 Embedding: Advancing text embedding and reranking through foundation models. arXiv preprint arXiv:2506.05176, 2025. URL https://arxiv.org/abs/2506.05176.

[59] Shu Zhao, Tan Yu, Anbang Xu, Japinder Singh, Aaditya Shukla, and Rama Akkiraju. ParallelSearch: Train your LLMs to decompose query and search sub-queries in parallel with reinforcement learning. arXiv preprint arXiv:2508.09303, 2025. URL https://arxiv.org/abs/2508.09303.

[60] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc., 2023. doi: 10.52202/075280-2020. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html.

[61] He Zhu, Tianrui Qin, King Zhu, Heyuan Huang, Yeyi Guan, Jinxiang Xia, Yi Yao, Hanhao Li, Ningning Wang, Pai Liu, Tianhao Peng, Xin Gui, Xiaowan Li, Yuhui Liu, Yuchen Eleanor Jiang, Jun Wang, Changwang Zhang, Xiangru Tang, Ge Zhang, Jian Yang, Minghao Liu, Xitong Gao, Jiaheng Liu, and Wangchunshu Zhou. OAgents: An empirical study of building effective agents. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 13354–13369. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-emnlp.720. URL https://aclanthology.org/2025.findings-emnlp.720/.

Appendix

Table of Contents

A DAGent Algorithm **17**

B Design Decisions **17**
B.1 Evaluate-then-Grow versus Plan-then-Patch 17
B.2 DAGRPO 18
B.3 Relation to Concurrent Credit-Assignment Methods 19

C Implementation Details **19**
C.1 Benchmarks, Tool Stack, and Judge 19
C.2 DAGent Framework Parameters 19
C.3 Baseline Configurations and Caps 20
C.4 RL Training Hyperparameters 20
C.5 Human Agreement Study on the LLM Judge 21

D Additional Training-Free Results **21**
D.1 Cross-Backbone and Frontier-Scale Details 22
D.2 Plan-then-Patch Variant Details 23
D.3 Extended Efficiency Comparison 23
D.4 Behavioral Statistics of the Evaluate-then-Grow Loop 24

E Additional RL Results **25**
E.1 Per-Seed Training Results 25
E.2 DAGRPO Training Dynamics 26
E.3 Efficiency as a Byproduct of RL Training 26

F Limitations **27**

G Prompts **27**
G.1 Orchestrator System Prompt 27
G.2 ReAct Executor System Prompt 29
G.3 Recall Extraction Prompt 31
G.4 LLM Judge Prompts 32

A DAGent Algorithm

Algorithm 1 summarizes the complete DAGent workflow described in §3.1.

B Design Decisions

This appendix explains the main design decisions of DAGent and DAGRPO and relates DAGRPO to concurrent credit-assignment methods.

B.1 Evaluate-then-Grow versus Plan-then-Patch

The Plan-then-Patch label refers to when a planning decision is made and how much the system knows at that moment, not to how many nodes one call produces. Flash-Searcher [38] produces its entire plan in one decomposition call before execution and then periodically updates it, removing

Algorithm 1 DAGent: Evaluate-then-Grow Workflow

Require: Task $\mathcal{T}$, Orchestrator policy π_O, ReAct Executor policy π_R, maximum iterations K
Ensure: Final answer $\hat{a}$

$\mathcal{G} \leftarrow (\emptyset, \emptyset)$ ▷ Initialize empty DAG
for $k = 1, 2, \ldots, K$ **do**
  $(S_k, V_k^{\text{new}}) \leftarrow \pi_O(\mathcal{T}, \mathcal{G})$ ▷ Evaluate & plan, Eq. (3)
  Update node statuses in $\mathcal{G}$ according to S_k
  Add nodes V_k^{new} and dependency edges to $\mathcal{G}$
  for each $v_i \in V_k^{\text{new}}$ **in parallel do**
    $C_i \leftarrow \bigoplus_{j \in \text{dep}_i} D_j$ ▷ Selective Propagation
    $(D_i, T_i) \leftarrow \pi_R(p_i, C_i)$ ▷ Sub-task execution
    Store D_i, T_i in node v_i
  end for
  if V_k^{new} contains an answer node **then**
    break
  end if
end for
return answer field from $D_{t_{\text{answer}}}$

resolved nodes and inserting new ones. FlowSearch [15] builds its knowledge flow over several planner iterations that all complete before any node runs, and then adjusts the flow with graph transformation operations based on intermediate outcomes. In both systems a plan that covers the whole task exists before execution, and execution-time edits patch that standing plan. In DAGent every node, including the answer node, is created only after the nodes it depends on have executed and been evaluated, and no node is ever deleted or rewired, so the recorded graph is exactly the graph that ran. The first batch is the one planning decision made without completed nodes, so it is planned from the task description alone. Appendix D.4 shows that 61.9% of first-batch nodes are evaluated as *Success* and that the refine nodes required for the weak ones, those evaluated as *Uncertain* or *Not Found*, turn 44.4% of them into *Success*.

B.2 DAGRPO

Role-separated normalization. The Orchestrator and the Executors produce different numbers of rollouts per task and operate on different reward distributions. A single shared (μ, σ) across both roles would let the more numerous Executor rewards dominate the Orchestrator gradient signal, so the group statistics in Eq. (6) are computed within each role.

Multiplicative off-chain attenuation. The off-chain reward $\alpha R(\tau)$ in Eq. (6) is multiplicative so that it gives no topology signal on failed trajectories, where $R(\tau) = 0$. Off-chain rollouts in a failed trajectory may carry useful evidence that the synthesis never consumed, and a constant subtractive penalty would push the policy away from such queries. The α sweep in Table 3 shows that off-chain rollouts carry learning value: $\alpha = 0$, which gives them zero reward even in correct trajectories, falls below the outcome-only GRPO baseline on all three benchmarks, whereas attenuating their reward with $\alpha = 0.5$ gives the best result on every benchmark.

The answer-inclusive closure as a structural proxy. $S(\tau)$ is a reachability signal read off the recorded graph without re-execution. It is a proxy for contribution rather than a causal attribution, and three properties prevent it from being exploited by inflating dependencies during training. First, nothing in the objective grows with the number of ancestors: the Orchestrator reward $R(\tau) + r_{\text{proc}}(\tau)$ does not count them, and Executor advantages are centered on the group mean, so moving more rollouts on-chain only redistributes credit. If every rollout were on-chain, the objective would reduce to the $\alpha = 1.0$ row with regularization on in Table 3, which DAGRPO beats by 2.0 points overall. Second, every extra edge has a cost: the DAG is append-only, so the closure can only be enlarged by adding a dependency edge, and under Selective Propagation each such edge places an irrelevant QueryDoc into the context of the node that consumes it. Third, dependency inflation would drive the off-chain share toward zero, and the trained policies do not: Appendix E.3 shows the off-chain share

decreasing but staying at or above 12.7% under both GRPO and DAGRPO, while the node count per task falls and Pass@1 rises on all three benchmarks.

Trajectory-level structural compliance regularization. The scale $\lambda_{\text{proc}} = 0.3$ keeps the penalty of a correct plan with up to three violating turns below the outcome reward, so such a plan is not ranked below a clean failure, and the cap $K_{\text{proc}} = 5$ bounds the penalty of a heavily violating plan at 1.5. Aggregating r_{proc} at the trajectory level rather than at the token level keeps the Orchestrator gradient tied to $R(\tau)$ and lets the group baseline absorb the across-trajectory mean of frequent violations. Apart from the requirements of parseable JSON, at least one new node per turn, and no duplicate sub-task prompts, the constraints that the penalty enforces (§3.2) control how evidence flows rather than the format of a plan: every executed node must receive an explicit status, every weak node with attempts left must be re-examined, the final synthesis must see all evidence assigned to it, and dependencies may point only to nodes that have already run and been evaluated. Appendix D.4 reports that, in the evaluation logs, tasks with at least one structural violation score 14.7 Pass@1 points lower than tasks with none (43.6 against 58.3).

B.3 Relation to Concurrent Credit-Assignment Methods

Three concurrent methods are closest to DAGRPO. Graph-GRPO [2] optimizes the communication topology over a fixed set of agents by scoring each edge across a sampled group of topologies; in DAGent the graph is the task decomposition itself, and its nodes do not exist until the run creates them. M-GRPO [13] computes group-relative advantages separately for a main agent and its sub-agents, which is the closest analogue of our role-separated normalization, but it credits a sub-agent rollout by the outcome and an LLM rating of its sub-task, whereas our credit depends on whether the rollout feeds the answer node. CCPO [23] estimates the marginal contribution of an agent by removing it counterfactually; in a research DAG the nodes below a removed node have already consumed its QueryDoc, so the counterfactual would require re-executing that subgraph, whereas $S(\tau)$ is read off the recorded graph without re-execution. Append-only growth is what makes this signal well defined: the ancestry of a node in the final graph is the context it saw when it ran, which no longer holds once executed nodes can be deleted or rewired, as in the periodic plan updates of Flash-Searcher.

C Implementation Details

C.1 Benchmarks, Tool Stack, and Judge

Table 4 lists the evaluation set, retrieval backend, and judge prompt of each benchmark. Within a benchmark every method uses the same backend, so differences between methods come from the workflow and not from the retrieval API. GAIA and xbench-DeepSearch use live Google search; BrowseComp-Plus uses local dense retrieval with Qwen3-Embedding-8B over its verified corpus because its protocol requires it. Pass@1 is scored by GPT-4o-mini [33] with GPT-4.1 [35] as a tie-breaker on borderline cases (Appendix C.5); the judge prompts are listed in Appendix G.

Table 4: Benchmarks and tool stack; all methods on a benchmark share its backend and judge.

	BrowseComp-Plus	**GAIA**	**xbench-DeepSearch**
Tasks	150	103 (text-only validation)	100 (2505 release)
Difficulty splits	50 easy / 50 medium / 50 hard	39 L1 / 52 L2 / 12 L3	none
Language	English	English	Chinese
Search	Local dense retrieval	Google search via Serper	Google search via Serper
Page access	Verified corpus	Jina extraction	Jina extraction
Judge prompt	Extended benchmark rubric	Equivalence prompt	Benchmark rubric (Chinese)

C.2 DAGent Framework Parameters

Table 5 lists the parameters of the DAGent framework. The context budget is enforced per agent context: each Executor sub-task runs in its own context, and the Orchestrator runs in one multi-turn context per task; in each context the prompt is capped at 8,192 tokens and the response, including tool outputs or execution results, at 32,768 tokens. This is the 32K×N budget of Table 1, which every

multi-context baseline shares. The Executor timeout applies to a single sub-task; the whole task is additionally bounded by a session timeout of 5,400 s, a bound that no evaluated task reaches.

Table 5: DAGent framework parameters, grouped by scope. "Per context" denotes the Orchestrator conversation of one task or one Executor sub-task; "per task" denotes one benchmark query.

Parameter	Scope	Value
Prompt length	per context	8,192 tokens
Response length, including tool outputs	per context	32,768 tokens
Orchestrator iterations K	per task	30
Session timeout	per task	5,400 s
Executor turns	per sub-task	200
Executor timeout, evaluation / training	per sub-task	600 s / 1,800 s
Attempts, one initial execution plus refines	per sub-task	3
Search results top-k	per search call	10
Decoding at evaluation	–	greedy
Thinking mode	–	disabled

C.3 Baseline Configurations and Caps

Table 6 lists the budget-related settings of the baseline workflows; the DAGent settings are those of Table 5, and the Plan-then-Patch variant shares them and adds a patch budget. Flash-Searcher keeps the 40-step cap of the authors' release; only its summary interval is changed from 8 to 4 so that each summary fits within the 32K per-sub-task budget shared by all multi-context methods. FlowSearch runs the question-answering preset of the released implementation with three changes: the model of every role is set to the evaluation backbone, the Coordinator (the execution-conditioned refiner described by Hu et al. [15]) is enabled because the preset ships with it disabled, and the tool set is restricted to the shared `search` and `open_page` backends; all official budgets are kept.

Table 6: Budget caps of the baseline workflows (the DAGent caps are listed in Table 5).

Workflow	Setting	Cap
ReAct Agent	Context window	32K or 109K tokens
Summary Agent / Fold Agent	Sessions per task	10
Flash-Searcher	Action steps per task	40
	Summary interval	4
FlowSearch	Main iterations per task	5
	Planner iterations / nodes	2 / 7
	Parallel workers	10
	Sub-tasks / tool calls per node	2 / 5
	Correction attempts	3
	Task timeout	2,400 s
Plan-then-Patch variant	Patch turns per task	30
	Session timeout	5,400 s

C.4 RL Training Hyperparameters

Table 7 lists the RL training and DAGRPO hyperparameters. Training uses LoRA on all linear layers with asynchronous vLLM [20] rollouts; unlisted settings inherit the verl [43] defaults. Per trajectory, at most 10 Executor rollouts enter the update: on-chain rollouts first, deepest first, then randomly sampled off-chain rollouts. The GRPO baseline uses the same rule and differs from DAGRPO only in $\alpha = 1.0$ and the absence of the structural compliance regularization.

Table 7: RL training and DAGRPO hyperparameters.

Parameter	Value
Hardware	$2\times$ H200, tensor parallel 2
FSDP offload	parameters and optimizer states
LoRA rank / alpha	64 / 64, all linear layers
Optimizer, learning rate	AdamW, 1×10^{-5}
Tasks per batch, rollouts per task G	32, 8
PPO mini-batch (tasks / trajectories)	16 / 128
Asymmetric clip ϵ_ℓ / ϵ_h	0.20 / 0.28
Entropy coefficient	0.001
KL-to-reference coefficient	0.001
Update steps	21
α, DAGRPO / GRPO baseline	0.5 / 1.0
λ_{proc}, K_{proc}	0.3, 5
Rollout sampling temperature	1.0 (verl default)
Training seeds	42, 123, 777

C.5 Human Agreement Study on the LLM Judge

The Pass@1 numbers in §4 are produced by GPT-4o-mini with GPT-4.1 as a tie-breaker on borderline cases, applied through three benchmark-specific judge prompts: an HLE-derived rubric for BrowseComp-Plus that extends the grader of its protocol [7] with explicit answer-equivalence rules, a short equivalence prompt for GAIA that replaces the original quasi-exact-match scoring [30] for cross-benchmark consistency, and a Chinese-language rubric for xbench-DeepSearch [52]. We validate each judge against three annotators, following Du et al. [9], Wong et al. [49], and the LLM-as-judge meta-evaluation literature [60, 3].

Setup. We sample 150 (question, predicted answer, reference answer) tuples stratified $50/50/50$ across the three benchmarks, drawn proportionally from DAGent and four agent baselines so that correct and incorrect predictions appear in roughly equal share. Three volunteer annotators from our lab, with reading-comprehension and information-retrieval backgrounds, independently provide a forced binary judgment, blinded to the score of the LLM judge.

***Observation 8:* Each per-benchmark Cohen's κ between the human-majority label and the LLM judge falls in the almost-perfect bracket, with the aggregate $\kappa = 0.95$ close to the inter-annotator agreement of Fleiss' $\kappa = 0.97$.** Table 8 reports Fleiss' κ on the three-human ensemble, Cohen's κ between the human-majority label and the LLM judge, and raw percent agreement. All per-benchmark Cohen's κ values fall in the almost-perfect bracket of Landis and Koch [21], and the aggregate raw agreement of 97.3% approaches the 97.8 to 98.3% range that Wong et al. [49] report for four judge models. Cohen's κ is 0.96 on both English benchmarks and 0.92 on xbench-DeepSearch, where Chinese-language tasks admit longer phrasal answers and surface-form variation.

Table 8: Human agreement on the LLM judge across 150 (question, predicted answer, reference answer) tuples. Fleiss' κ is computed on the three-human ensemble; Cohen's κ is between the human-majority label and the LLM judge; raw % is between the same pair.

Subset	N	Fleiss' κ (3H)	Cohen's κ (H vs J)	Raw %
BrowseComp-Plus	50	0.98	0.96	98.0
GAIA	50	0.98	0.96	98.0
xbench-DeepSearch	50	0.96	0.92	96.0
Aggregate	150	0.97	0.95	97.3

D Additional Training-Free Results

This appendix collects the training-free results that supplement §4.2, §4.3, and §4.4.

D.1 Cross-Backbone and Frontier-Scale Details

Table 9 gives the per-baseline breakdown behind Figure 3: on each of the four backbones, DAGent leads the strongest baseline on every benchmark. All rows share the tool stack of the main results, and the Qwen3-32B block reproduces the corresponding rows of Table 1, which additionally includes FlowSearch, evaluated at Qwen3-32B only. This section also lists the baselines of the frontier-scale comparison in Figure 4 and describes its multi-modal extension.

Frontier-scale baselines. For the GPT-5 327K comparison in Figure 4, the BrowseComp-Plus baselines are 100B+ LLMs running ReAct at 327K context, namely Qwen3-235B-A22B, GLM-4.5-Air [12], DeepSeek-V3.1 [8], GPT-4.1 [35], and GPT-5 [36], with numbers taken from Sun et al. [45]; the GAIA and xbench-DeepSearch baselines are deep research systems discussed in §2.1, with numbers from Qin et al. [38] except OpenAI Deep Research and FlowSearch on GAIA, whose numbers are from Hu et al. [15]. On GAIA, DAGent reaches 85.5 and the three runners-up, Flash-Searcher (GPT-5), FlowSearch (o4-mini), and Skywork DR, score 82.4 to 83.0.

Multi-modal extension for the GAIA full validation set. The GPT-5 327K comparison in Figure 4 evaluates GAIA on its full 165-task validation set rather than the 103-task text-only subset used elsewhere, following the protocol of Qin et al. [38]. The 62 multi-modal questions in the full set involve attached image, audio, and document files that the text-only Orchestrator and ReAct Executor cannot process directly. Following the framework of Zhu et al. [61], we extend the tool registry with three inspector tools: an image inspector that base64-encodes attachments and queries a vision model, a document inspector that parses PDF, spreadsheet, and structured-text files for question-conditioned extraction, and an audio inspector that transcribes audio attachments before extraction. Each attached file is pre-processed into a textual description via the appropriate inspector and prepended to the question; the same inspectors remain available as agent-callable tools during execution. All other DAGent components are unchanged, so the full-set evaluation uses the same planning and context-management mechanisms as the text-only experiments.

Table 9: Training-free Pass@1 (%) of DAGent and the baselines on four mid-size open-source backbones from four vendors, with the tool stack of the main results; best result per backbone in **bold**. The Qwen3-32B block reproduces the corresponding rows of Table 1.

Backbone	Vendor	Method	BrowseComp-Plus				GAIA				xbench-DS
			Easy	Med.	Hard	Avg.	L1	L2	L3	Avg.	Avg.
Qwen3-32B	Alibaba	ReAct Agent (32K)	50.0	16.0	2.0	22.7	46.2	28.8	8.3	33.0	54.0
		ReAct Agent (109K)	66.0	24.0	4.0	31.3	51.3	36.5	16.7	39.8	58.0
		Summary Agent	80.0	36.0	0.0	38.7	51.3	28.8	25.0	36.9	55.0
		Fold Agent	82.0	40.0	2.0	41.3	56.4	38.5	16.7	42.7	55.0
		Flash-Searcher	80.0	34.0	2.0	38.7	56.4	40.4	25.0	44.7	58.0
		DAGent (Ours)	**88.0**	**46.0**	**8.0**	**47.3**	**69.2**	**50.0**	**33.3**	**55.3**	**65.0**
Seed-OSS-36B	ByteDance	ReAct Agent (32K)	64.0	18.0	0.0	27.3	43.6	40.4	8.3	37.9	58.0
		ReAct Agent (109K)	88.0	44.0	2.0	44.7	43.6	48.1	16.7	42.7	62.0
		Summary Agent	82.0	40.0	2.0	41.3	48.7	51.9	**25.0**	47.6	59.0
		Fold Agent	88.0	36.0	8.0	44.0	43.6	46.2	16.7	41.7	60.0
		Flash-Searcher	88.0	44.0	4.0	45.3	51.3	51.9	**25.0**	48.5	67.0
		DAGent (Ours)	**94.0**	**58.0**	**10.0**	**54.0**	**64.1**	**53.8**	**25.0**	**54.4**	**69.0**
GLM-4-32B	Zhipu AI	ReAct Agent (32K)	52.0	16.0	2.0	23.3	41.0	25.0	25.0	31.1	56.0
		ReAct Agent (109K)	68.0	24.0	4.0	32.0	48.7	32.7	25.0	37.9	60.0
		Summary Agent	82.0	36.0	2.0	40.0	43.6	30.8	25.0	35.0	57.0
		Fold Agent	84.0	40.0	4.0	42.7	51.3	36.5	25.0	40.8	58.0
		Flash-Searcher	82.0	34.0	2.0	39.3	53.8	38.5	25.0	42.7	64.0
		DAGent (Ours)	**90.0**	**48.0**	**8.0**	**48.7**	**66.7**	**46.2**	**33.3**	**52.4**	**67.0**
Nemotron-3-Nano-30B	NVIDIA	ReAct Agent (32K)	50.0	18.0	4.0	24.0	43.6	25.0	25.0	32.0	46.0
		ReAct Agent (109K)	66.0	26.0	8.0	33.3	51.3	36.5	25.0	40.8	51.0
		Summary Agent	80.0	38.0	2.0	40.0	53.8	30.8	25.0	38.8	48.0
		Fold Agent	84.0	40.0	4.0	42.7	56.4	38.5	25.0	43.7	49.0
		Flash-Searcher	82.0	38.0	4.0	41.3	59.0	40.4	25.0	45.6	53.0
		DAGent (Ours)	**90.0**	**48.0**	**12.0**	**50.0**	**71.8**	**51.9**	**33.3**	**57.3**	**59.0**

D.2 Plan-then-Patch Variant Details

The Plan-then-Patch variant used in Table 2 and Figure 5(b) reuses the Orchestrator, ReAct Executor, backbone, tool stack, context-management components, and per-node refine budget of DAGent; only the cross-iteration planning behavior of the Orchestrator is modified.

Planning and patching protocol. At iteration 0 the Orchestrator outputs the complete task DAG in one call, committing every search node plus exactly one answer node, each with its id, description, prompt, and dependency edges, before any node has executed; subsequent iterations execute the next ready batch in parallel and defer the answer node whenever any non-answer node is also ready. After each batch the Orchestrator may issue *Refine* operations, reusing the three-attempt mechanism of DAGent on a node whose status is Uncertain or Not Found, or *Add* operations, introducing a new search node whose description must cite at least one already-executed node id (enforced by the structural validator through the format-warning loop). Both operations preserve the DAGent JSON schema, and the nodes they create are automatically appended to the dependency set of the answer node so that their evidence reaches the final synthesis. Total patch turns are capped at 30, with mean per-task patch turns of $4.2/1.8/1.4$ on BrowseComp-Plus / GAIA / xbench-DeepSearch.

Absolute per-task footprints. Table 10 in Appendix D.3 lists the absolute per-task footprints summarized in Figure 5(b), alongside those of Flash-Searcher and FlowSearch.

Where the off-chain nodes come from. Figure 5(b) shows that the variant raises the off-chain share of Executor nodes from 25.3 / 19.6 / 18.4% to 39.7 / 30.4 / 24.6%. Their origin follows from the construction of the variant rather than from a measurement: every patch node, whether a refine or an add, is connected to the answer node automatically so that its evidence reaches the final synthesis, so patch nodes are on-chain by construction and off-chain nodes can only come from the initial plan. The extra off-chain nodes are therefore over-commitment at planning time, not a failure of the patch mechanism to prune. Adding a deletion policy to the variant would close only part of the gap: deleting a node saves compute only if the node has not yet run, and a patch policy that decides batch by batch, from evidence, what to keep, drop, or expand is Evaluate-then-Grow. The released FlowSearch implementation includes a deletion-capable refiner, and Table 10 shows that it still costs more than DAGent on every cost column.

D.3 Extended Efficiency Comparison

Section 4.4 compares DAGent with its Plan-then-Patch variant. This section adds the two released Plan-then-Patch systems, Flash-Searcher and FlowSearch, and reports the absolute external tool calls and wall-clock time that the main text gives as relative differences.

Setting. All four workflows use the same Qwen3-32B backbone and the same retrieval backends on every benchmark, and no method is modified beyond the configuration documented in Appendix C.3; FlowSearch runs with its Coordinator enabled. The *tool calls* column counts every model-driven call in the workflow, so its composition differs by method: search and open_page in every workflow, plus recall and Orchestrator plan calls for the two DAGent rows, planning and summary calls for Flash-Searcher, and Planner and Coordinator calls for FlowSearch. The *external tool calls* column counts only search and open_page, which all four workflows use in the same way, and therefore isolates the cost of the search API. The *steps* column follows the parallelism-aware definition of §4.4 and does not measure latency. The *tokens* column is total input plus output, and the *time* column is the wall-clock time to finish one task. The *nodes* column is the final graph size and *off-chain* is the percentage of Executor nodes outside the answer-inclusive closure; both are defined only for the two DAGent rows. The *calls / step* column is the ratio of tool calls to execution steps and is reported as a parallelism measure, not as a cost. We do not impose a common per-task cost cap on the four workflows, because most workflows stop through an explicit finish call rather than by exhausting a budget (Appendix C.3 lists the caps of each workflow), and a common cap would cut the remaining runs off mid-task and turn the comparison into a completion-rate test.

***Observation 9:* Among the systems that keep a separate context per node, DAGent is both the most accurate and the cheapest on every per-task cost column (tool calls, external calls, steps, tokens, and time); Flash-Searcher spends less because it is a single-agent system, and it is 8.6 / 10.6 / 7.0**

Table 10: Per-task cost of four workflows with Qwen3-32B and the same retrieval backends. Pass@1 values are from Tables 1 and 2. Tool calls, external tool calls, execution steps, tokens, and time are per-task means; calls / step divides the sum of tool calls by the sum of execution steps. Nodes and off-chain share are defined only for the two DAGent rows.

Benchmark	Workflow	Pass@1	Tool calls	External calls	Steps	Calls / step	Tokens (M)	Time (s)	Nodes	Off-chain (%)
BrowseComp-Plus	DAGent (Full)	**47.3**	67.6	60.8	42.7	1.58	1.20	541.6	11.8	25.3
	Plan-then-Patch variant	42.0	87.9	82.3	54.8	1.60	1.68	731.9	17.6	39.7
	Flash-Searcher	38.7	51.3	42.1	35.8	1.43	0.91	388.7	–	–
	FlowSearch	43.3	83.5	76.7	47.6	1.75	1.57	714.5	–	–
GAIA	DAGent (Full)	**55.3**	36.8	33.1	31.2	1.18	0.66	461.8	5.4	19.6
	Plan-then-Patch variant	50.5	45.3	42.0	37.6	1.20	0.85	584.6	7.3	30.4
	Flash-Searcher	44.7	28.7	21.8	23.5	1.22	0.48	301.9	–	–
	FlowSearch	51.5	40.4	35.1	32.8	1.23	0.74	567.7	–	–
xbench-DeepSearch	DAGent (Full)	**65.0**	32.9	29.2	26.8	1.23	0.44	380.9	4.9	18.4
	Plan-then-Patch variant	61.0	38.2	35.4	32.4	1.18	0.52	466.8	6.2	24.6
	Flash-Searcher	58.0	24.6	18.9	20.2	1.22	0.30	241.6	–	–
	FlowSearch	63.0	36.7	31.6	29.6	1.24	0.55	488.9	–	–

points less accurate. The Plan-then-Patch variant differs from DAGent only in when it commits to a plan, and it makes 35 / 27 / 21% more external tool calls (82.3 / 42.0 / 35.4 against 60.8 / 33.1 / 29.2) and needs 35 / 27 / 23% more wall-clock time (731.9 / 584.6 / 466.8 s against 541.6 / 461.8 / 380.9 s). FlowSearch makes 26 / 6 / 8% more external tool calls and needs 32 / 23 / 28% more time, while staying 4.0 / 3.8 / 2.0 points below DAGent. A pre-committed plan contains branches that later turn out not to be needed, and the system still has to execute them; when the graph grows only after evidence arrives, those branches are never created. Executing them is what accounts for the extra search calls and wall-clock time of the two pre-committed workflows. Flash-Searcher is the one workflow that costs less than DAGent, and the reason is architectural rather than a property of Plan-then-Patch: it keeps a single reasoning trajectory and merges every parallel branch back into the same state, so it never holds a separate context per node. DAGent, its Plan-then-Patch variant, and FlowSearch are multi-agent systems that keep one context per node, which accounts for the additional tokens and calls and is also what lets a DAG operate over long horizons. Lower cost only matters at comparable accuracy, and no workflow in Table 10 matches the accuracy of DAGent at any cost level.

Overall accuracy of the Plan-then-Patch systems. Averaged over the three benchmarks, DAGent reaches 55.9, FlowSearch 52.6, the Plan-then-Patch variant 51.2, and Flash-Searcher 47.1. The variant keeps the full DAGent component stack and scores between the two released Plan-then-Patch systems, so the ablation in Table 2 compares Evaluate-then-Grow with a Plan-then-Patch system whose accuracy lies within the range of the released ones. FlowSearch, whose refiner can also delete nodes, is the strongest non-DAGent system in the Qwen3-32B block of Table 1 and still costs more than DAGent on every cost column.

D.4 Behavioral Statistics of the Evaluate-then-Grow Loop

Figure 6 summarizes statistics computed from the logs of the Qwen3-32B training-free runs reported in the main text: how the statuses assigned in status evaluation are distributed over all executed nodes and over first-batch nodes, and how often the Executor calls RECALLTOOL.

***Observation 10:** Negative and inconclusive evaluations are a routine planning input rather than an exception, and recall stays a fallback:* **Uncertain** *and* **Not Found** *account for 54.3% of the executed nodes on BrowseComp-Plus and about a third on GAIA and xbench-DeepSearch, while 22.0 / 13.6 / 11.0% of tasks make at least one recall call.*

RECALLTOOL usage. The Executor calls RECALLTOOL (Eq. (4)) as `recall(node_id, goal)`. The extractor reads the full InteractionTranscript of the named dependency and returns what the goal asks for; it may use only that transcript, must attach a document id, URL, or exact quote to every claim, and must state explicitly when nothing relevant is present (prompt in Appendix G). The tool is a fallback rather than a default channel: the mean number of calls per task is 0.33 / 0.21 / 0.18 on BrowseComp-Plus / GAIA / xbench-DeepSearch, and 22.0 / 13.6 / 11.0% of tasks make at least one call (Figure 6(b)). The w/o InteractionTranscript row of Table 2 is the corresponding ablation, since removing the transcript disables recall, and it costs 4.0 / 2.9 / 2.0 Pass@1 points; given the usage rates, the tool changes the outcome of roughly one in five of the tasks that call it.

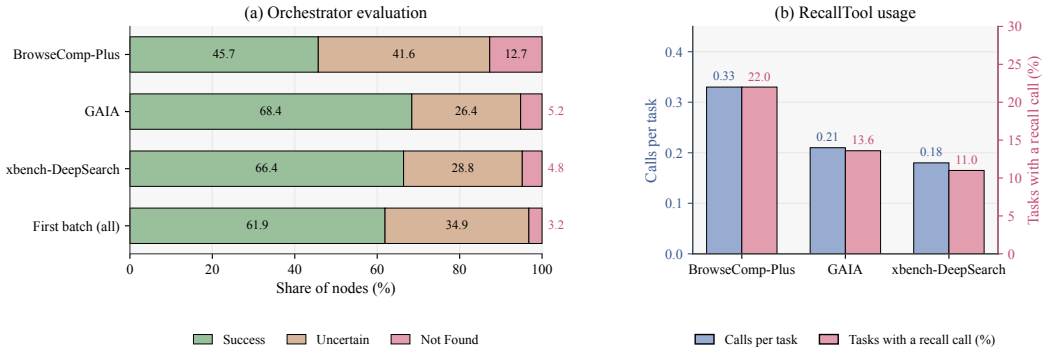


Figure 6: Behavioral statistics of the Qwen3-32B training-free runs. (a) Distribution of the Orchestrator evaluation outcomes over all executed nodes per benchmark and over first-batch nodes pooled across the three benchmarks. (b) RECALLTOOL usage: mean number of calls per task and share of tasks that make at least one call, per benchmark.

Evaluation outcomes and the first batch. Across all executed nodes, the Orchestrator evaluates 41.6% / 12.7% of nodes as *Uncertain* / *Not Found* on BrowseComp-Plus, 26.4% / 5.2% on GAIA, and 28.8% / 4.8% on xbench-DeepSearch (Figure 6(a)). The Orchestrator reads these outcomes to decide the next expansion, so negative and inconclusive evidence is a routine planning input rather than discarded work. First-batch nodes are planned from the task description alone (Appendix B.1) and are evaluated as *Success* / *Uncertain* / *Not Found* in 61.9% / 34.9% / 3.2% of cases, so a first node that finds nothing relevant at all is rare. The validator forces a refine node for every weak first node in the same turn; these refinements turn 44.4% of weak first nodes into *Success*, and the remainder reach the next expansion as an explicit uncertainty rather than as silently accepted evidence.

Structural violations and accuracy. The structural compliance regularization of DAGRPO penalizes Orchestrator turns that violate the constraints of §3.2. Removing the term costs 1.6 / 1.0 / 0.7 Pass@1 points (Table 3), and Appendix E.2 shows that it drives the format-warning rate of the Orchestrator down faster than the outcome-only baseline. The association is also visible in the evaluation logs: tasks whose trajectory contains at least one structural violation score 14.7 points lower than tasks with none (43.6 against 58.3). Task difficulty may contribute to both, so we read this as an association consistent with the mechanism rather than as a causal estimate.

E Additional RL Results

This appendix collects the RL results that supplement the training-based rows of Table 1.

E.1 Per-Seed Training Results

Three independent training runs, with seeds 42, 123, and 777, are performed for each of the six RL-trained variants in the Qwen3-8B block of Table 1 with all other hyperparameters held fixed; the seed fixes the shuffle order of the training data, while the LoRA initialization, the vLLM rollout sampling, and the tool calls draw fresh randomness in every run, so the reported standard deviations reflect all of these sources of variation. Figure 7 reports the per-seed Pass@1 (%) values of these variants, one panel per benchmark.

Observation 11: The gain of DAGRPO over GRPO holds for every training seed: with each of the three seeds, DAGRPO-DAGent scores above GRPO-DAGent on all three benchmarks. The smallest same-seed margin in Figure 7 is 1.9 points, on GAIA with seed 123. The standard deviations of the two DAGent rows in the Qwen3-8B block of Table 1 are at most 1.5 points, below the gaps of 3.6 / 2.6 / 2.7 points between their means on BrowseComp-Plus / GAIA / xbench-DeepSearch.

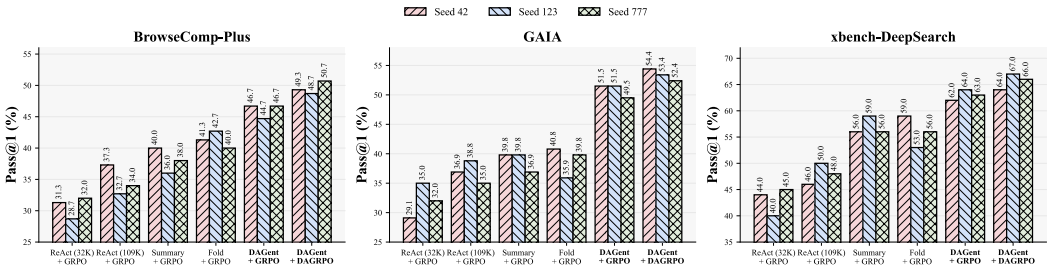


Figure 7: Per-seed Pass@1 (%) for the six RL-trained variants in the Qwen3-8B block of Table 1, broken out by training seed (42 / 123 / 777) on BrowseComp-Plus, GAIA, and xbench-DeepSearch.

E.2 DAGRPO Training Dynamics

Figure 8 traces four training-time metrics that distinguish DAGRPO from GRPO; panels (b) to (d) are computed on the sampled training rollouts of each step, so the on-chain share in panel (d) differs from the evaluation-time value implied by Table 11, which uses the evaluation set with greedy decoding.

Observation 12: Both DAGRPO signals act as designed during training: the topology-conditioned credit separates on-chain from off-chain Executor rewards (panel (b)), and the compliance regularization lowers the Orchestrator format-warning rate faster than outcome-only GRPO (panel (c)). The reward gap in panel (b) grows over training under DAGRPO and is zero under GRPO by construction. Under DAGRPO, the on-chain share in panel (d) also rises faster, and the validation Pass@1 in panel (a), which starts from the training-free 40.0 for both methods, is above that of GRPO at steps 7, 14, and 21 and ends 3.6 points higher.

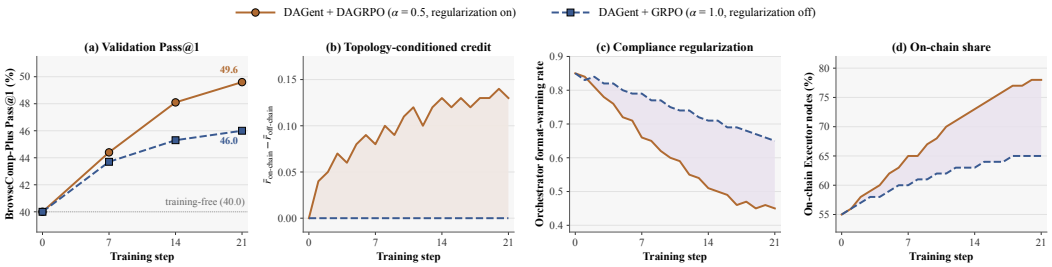


Figure 8: DAGRPO training dynamics on Qwen3-8B with LoRA. (a) BrowseComp-Plus validation Pass@1 over training steps, averaged over the three training seeds; the end points equal the Qwen3-8B means of Table 1. (b) Per-step training-time reward gap $\bar{r}_{\text{on-chain}} - \bar{r}_{\text{off-chain}}$; the topology-conditioned credit at $\alpha = 0.5$ separates the two groups, while $\alpha = 1.0$ gives zero gap by construction. (c) Per-step Orchestrator format-warning rate from the structural validator. (d) Per-step on-chain share, the percentage of Executor nodes inside the answer-inclusive closure.

E.3 Efficiency as a Byproduct of RL Training

Table 11 reports the per-task footprint of DAGent on Qwen3-8B before RL training and after GRPO and DAGRPO training, with the RL rows averaged over three independent training seeds. Neither the topology-conditioned credit nor the structural compliance regularization targets this footprint, so the reductions it shows are a byproduct of accuracy-driven training.

Observation 13: RL training reduces the per-task footprint of DAGent on every metric without an efficiency objective, and DAGRPO reduces it more than outcome-only GRPO. Outcome-only GRPO (the GRPO-DAGent row of Table 1) yields step reductions of 8 to 12%, tool-call reductions of 6 to 9%, and token reductions of 15 to 16% across BrowseComp-Plus / GAIA / xbench-DeepSearch (Table 11); the two structural signals of DAGRPO raise these to 12 to 17%, 10 to 14%, and 17 to 21%. The structural metrics decrease as well: the graph node count drops by 0.7 to 0.9 nodes per task under GRPO and by 1.3 to 1.9 under DAGRPO, and the off-chain share of Executor nodes decreases by 1.7 to 3.6 points under GRPO and by 4.9 to 8.9 points under DAGRPO. The additional reduction

under DAGRPO is 2 to 5 points of the training-free value for each of steps, tool calls, and tokens, and is largest in relative terms for the off-chain share, consistent with the topology-conditioned credit, which scales the reward of off-chain Executor rollouts by α.

Table 11: Effect of RL training on the per-task efficiency of DAGent on Qwen3-8B; the RL rows are means across three independent training seeds. Pass@1 values are taken from the Qwen3-8B block of Table 1. Tokens is total input plus output; tool calls aggregate search, open_page, recall, and Orchestrator plan invocations; nodes is the final graph size; off-chain is the percentage of Executor nodes outside the answer-inclusive closure, as in Table 10.

Benchmark	Setting	Pass@1	Tokens	Tool calls	Steps	Nodes	Off-chain (%)
BrowseComp-Plus	DAGent (training-free)	40.0	1.14M	65.5	41.5	12.2	24.7
	+ RL (GRPO)	46.0	0.96M	59.6	36.5	11.3	21.1
	+ RL (DAGRPO)	49.6	0.90M	56.3	34.5	10.3	15.8
GAIA	DAGent (training-free)	46.6	0.62M	35.5	29.5	5.3	20.3
	+ RL (GRPO)	50.8	0.53M	33.0	26.8	4.6	16.8
	+ RL (DAGRPO)	53.4	0.51M	31.6	25.7	4.0	13.2
xbench-DeepSearch	DAGent (training-free)	60.0	0.41M	32.0	26.0	5.4	17.6
	+ RL (GRPO)	63.0	0.35M	30.1	23.9	4.6	15.9
	+ RL (DAGRPO)	65.7	0.34M	28.8	22.9	3.9	12.7

F Limitations

DAGent is text-native: for the full GAIA validation set, the multi-modal extension (Appendix D.1) converts each attachment into a text description before planning begins. Handling multi-modal evidence inside the Evaluate-then-Grow loop involves extending how the Orchestrator evaluates evidence and how the Executor represents its transcript. The three benchmarks have closed-form answers that an LLM judge can verify; open-ended report generation is outside the scope of the reported results. The answer-inclusive closure is a structural proxy for contribution (Appendix B), and the association between structural compliance and accuracy in the evaluation logs is correlational (Appendix D.4); causal attribution at the level of individual nodes is an open research direction for DAG-based multi-agent systems such as DAGent.

G Prompts

This appendix lists, verbatim, the three DAGent prompts and the three LLM judge prompts. Inside a listing, section headings are set in bold in the color of its title bar, and template slots that are filled at run time, such as `{goal}`, are set in orange. The Orchestrator prompt is shown with the Overall Goal of one BrowseComp-Plus task filled in.

G.1 Orchestrator System Prompt

Orchestrator System Prompt

```
You are a world-class orchestrator for deep research. You plan search tasks for a ReAct agent,
evaluate its results, and iterate until the Overall Goal is answered.

## Overall Goal
An individual was encouraged to use a different tool for their work during the year between the
assassinations of John F. Kennedy and his brother, Senator Robert F. Kennedy. Up until 2014, they
have continued to use this tool. This person first studied at a university in Africa, then in the
United Kingdom, and later in the United States. They taught and held the positions of head of
department and dean at the university where they studied in Africa until a year in the last decade
of the 20th century. They also built a venue to showcase their own work as well as the work of
other people. This venue celebrated its 25th anniversary in a year between 2015 and 2020. What is
the name of this individual?

## How This System Works
You manage nodes in a task graph. Your conversation history contains all previous planning turns.
Each new user message only contains NEW execution results and a compact status ledger. Review your
history to recall earlier node details.
```

```
## Core Responsibilities

### 1. Evaluate (output: `node_statuses`)
For each node listed as "Awaiting Evaluation" in the user message, evaluate its execution result
and assign status:
- **Success**: information retrieved with high confidence, grounded in actual search results.
- **Uncertain**: information found but verification needed (triggers refine).
- **Not Found**: search executed correctly but yielded no results (triggers refine).

Every id listed under "Nodes to evaluate this turn" MUST appear in `node_statuses`. Do NOT include
nodes that already have a status.

### 2. Plan (output: `nodes`)
Plan nodes that can be executed immediately. Each node has `id`, `description`, `prompt`,
`dependencies`.

**id** (encodes task type):
- Search: `t1`, `t2`, `t3`, ...
- Refine: `<parent>_refine1` or `<parent>_refine2` (id MUST end in `_refineN`)
- Answer: `t_answer` (id MUST end in exactly `_answer`)

**description**: must be detailed and **self-contained**. Include all necessary context.

**prompt**: provide detailed instructions for the ReAct agent. Synthesize relevant findings from
dependency nodes as context bridge, and add operational value such as search strategies, angles to
try, and prior results to avoid. Formulate **complete natural language sentences** (NOT keywords)
for the ReAct agent. Our retrieval uses embedding model - full sentences work much better. Preserve
rare verbatim phrases from the Overall Goal exactly.

**dependencies**: list ALL nodes whose content is needed. All dependencies must already be
evaluated (not Awaiting Evaluation, not in the same batch).

**IMPORTANT**: the ReAct agent can ONLY see the description, prompt, and content from dependency
nodes. It CANNOT see the Overall Goal or full graph state. Use these three fields together to give
it everything it needs.

**Granularity Rule**: task granularity depends on the purpose:
- **Finding candidates**: first, resolve any conditions that contain indirect references
(references to events, abstract categories that need resolution into specific entity names) into
concrete facts via search. If a lookup returns multiple plausible values, try each value in
separate candidate-discovery searches. Then, substitute resolved facts into the remaining
conditions and combine 2-3 distinctive conditions per node to narrow down candidates. Plan MULTIPLE
candidate-discovery nodes in the SAME turn with DIFFERENT condition combinations. Do NOT put all
conditions in one node (retrieval degrades with too many conditions). After gathering candidates,
proceed to check them.
- **Checking specific conditions**: each node should target ONE condition for ONE candidate.
- **Fallback**: if all candidates fail critical conditions, return to finding candidates with
DIFFERENT search angles. If candidate discovery keeps failing, the resolved facts themselves may be
wrong. Re-do those lookups with different angles.

**Parallelism Rule**: you can include MULTIPLE nodes per plan.
- When finding candidates, plan MULTIPLE search nodes in parallel to explore DIFFERENT combinations.
- When checking conditions for one candidate, plan ALL unchecked conditions in parallel.
- Dependency constraint: parallel nodes can only be planned if all their dependencies have already
been evaluated.

**Refine Rule**:
- If status is Uncertain or Not Found AND the line still has refine budget, you MUST plan a refine
node (`<parent>_refine1`, or `<parent>_refine2` for a second refine) with a DIFFERENT search
strategy. The refine prompt MUST copy ALL prior search queries from the entire refine chain
verbatim (the original node's `key_steps` plus any earlier refine's `key_steps`) and specify new
angles.
- **3-attempt limit**: original search + refine1 + refine2. After 3 attempts, the line is
exhausted. Accept that gap and move forward.

**Answer Rule**:
- Plan a single answer node when: (1) evidence is sufficient AND a concrete candidate exists, OR
(2) all search lines are exhausted AND a best-effort answer is available. If a `[CONTEXT WARNING]`
has been injected, begin the termination path: verify the answer's required form if needed, then
plan the answer node.
- Finding candidates is NOT enough. You MUST go through the checking-conditions phase first.
- You MUST have a concrete candidate. The question is guaranteed to have an answer. Responses like
"No such X exists" or "Insufficient information" are WRONG. Compare candidates by how many
conditions each satisfies and pick the best-supported one.
- Before the answer node, try to ensure all necessary information about the selected candidate has
been gathered by upstream nodes. If the Overall Goal requires a specific form or granularity,
ideally an upstream node should have already retrieved it. If not, the answer node can perform
targeted searches to fill in missing details.
- The answer node MUST be planned ALONE, NEVER in the same batch with search or refine nodes. Set
its prompt to list accumulated evidence and instruct the ReAct agent to verify any details still
needed for the answer's required form, then call `finish`.
```

```
## Output Format

Structure your response in TWO parts.

### Part 1: Thinking
You MUST answer EACH question below step by step. Do NOT include any JSON in this section.

**Step 1: Evaluate**
- What status (Success / Uncertain / Not Found) should I assign to each Awaiting Evaluation node?
Is the answer grounded in actual search results, or guessed from prior knowledge?
- Have I covered every id listed in "Nodes to evaluate this turn", and no others?

**Step 2: Plan**
- What phase am I in?
  - Finding candidates: are there conditions with indirect references that need factual lookup
first? What condition combinations should I try? Am I planning MULTIPLE nodes with DIFFERENT
combinations?
  - Checking conditions: am I targeting ONE condition per node? Am I planning ALL unchecked
conditions in parallel?
  - Fallback: are all candidates failing? Could the resolved facts be wrong?
- Need refine? Any Uncertain or Not Found nodes with refine budget left? Have I copied ALL prior
queries from the entire refine chain verbatim and specified new angles?
- Ready to answer? Have I gone through the checking-conditions phase? Do I have a concrete
candidate? Have I compared candidates by condition coverage and picked the best-supported one? Does
the answer satisfy the form or granularity requirement? If the required form hasn't been retrieved
by an upstream node, does the answer node's prompt include instructions to look it up? Is the
answer node planned ALONE?
- For each new node: is the id correctly formed (search `tN`, refine `<parent>_refine1` or
`_refine2`, answer `t_answer`)? Is the description self-contained? Does the prompt include context
bridge and **complete natural language sentences (NOT keywords)**? Are all dependencies already
evaluated and not in the same batch? Remember: the ReAct agent CANNOT see the Overall Goal.

### Part 2: JSON Plan
Output a single JSON object in a ```json block. Follow this format strictly. Any deviation will
cause parsing errors.

- `node_statuses`: one entry for every id listed in "Nodes to evaluate this turn" (no more, no
less). Empty `{}` only when there are no nodes to evaluate.
- `nodes`: at least one new node. Each node MUST have `id` (string), `description` (string),
`prompt` (string), `dependencies` (array of strings).

```json
{
  "node_statuses": {
    "...": "Success" / "Uncertain" / "Not Found"
  },
  "nodes": [
    {"id": "...", "description": "...", "prompt": "...", "dependencies": []},
    ...
  ]
}
```
```

G.2 ReAct Executor System Prompt

ReAct Executor System Prompt

```
You are a meticulous and strategic research agent. Your primary function is to conduct
comprehensive, multi-step research to deliver a thorough, accurate, and well-supported report in
response to the user's query.

Your operation is guided by these core principles:
* **Rigor:** Execute every step of the research process with precision and attention to detail.
* **Objectivity:** Synthesize information based on the evidence gathered, not on prior assumptions.
Note and investigate conflicting information.
* **Thoroughness:** Never settle for a surface-level answer. Always strive to uncover the
underlying details, context, and data.
* **Transparency:** Your reasoning process should be clear at every step, linking evidence from
your research directly to your conclusions.

Follow this structured protocol to find the answer

### Phase 1: Deconstruction & Strategy

1.  **Deconstruct the Query:**
    * Analyze the user's prompt to identify the core question(s).
    * Isolate key entities, concepts, and the relationships between them.
    * Explicitly list all constraints, conditions, and required data points (e.g., dates,
quantities, specific names).
```

2. **Hypothesize & Brainstorm:**
    * Based on your knowledge, brainstorm potential search vectors, keywords, synonyms, and related topics that could yield relevant information.
    * Consider multiple angles of inquiry to approach the problem.
3. **Verification Checklist:**
    * Create a **Verification Checklist** based on the query's constraints and required data points. This checklist will be your guide throughout the process and used for final verification.

Phase 2: Iterative Research & Discovery

Tool Usage:
* **Tools:**
    * `search`: Use for broad discovery of sources and to get initial snippets.
    * `open_page`: **Mandatory follow-up** for any promising `search` result. Snippets are insufficient; you must analyze the full context of the source document.
* **Query Strategy:**
    * Start with moderately broad queries to map the information landscape. Narrow your focus as you learn more.
    * Do not repeat the exact same query. If a query fails, rephrase it or change your angle of attack.
    * Execute a **minimum of 5 tool calls** for simple queries and up to **50 tool calls** for complex ones. Do not terminate prematurely.
* **Post-Action Analysis:** After every tool call, briefly summarize the key findings from the result, extract relevant facts, and explicitly state how this new information affects your next step in the OODA loop.
* ***<IMPORTANT>Never simulate tool call output</IMPORTANT>**

You will execute your research plan using an iterative OODA loop (Observe, Orient, Decide, Act).

1. **Observe:** Review all gathered information. Identify what is known and, more importantly, what knowledge gaps remain according to your research plan.
2. **Orient:** Analyze the situation. Is the current line of inquiry effective? Are there new, more promising avenues? Refine your understanding of the topic based on the search results so far.
3. **Decide:** Choose the single most effective next action. This could be a broader query to establish context, a highly specific query to find a key data point, or opening a promising URL.
4. **Act:** Execute the chosen action using the available tools. After the action, return to **Observe**.

Phase 3: Synthesis & Analysis

* **Continuous Synthesis:** Throughout the research process, continuously integrate new information with existing knowledge. Build a coherent narrative and understanding of the topic.
* **Triangulate Critical Data:** For any crucial fact, number, date, or claim, you must seek to verify it across at least two independent, reliable sources. Note any discrepancies.
* **Handle Dead Ends:** If you are blocked, do not give up. Broaden your search scope, try alternative keywords, or research related contextual information to uncover new leads. Assume a discoverable answer exists and exhaust all reasonable avenues.
* **Maintain a "Fact Sheet":** Internally, keep a running list of key facts, figures, dates, and their supporting sources. This will be crucial for the final report.

Phase 4: Verification & Final Report Formulation

1. **Systematic Verification:** Before writing the final answer, halt your research and review your **Verification Checklist** created in Phase 1. For each item on the checklist, confirm you have sufficient, well-supported evidence from the documents you have opened.
2. **Mandatory Re-research:** If any checklist item is unconfirmed or the evidence is weak, it is **mandatory** to return to Phase 2 to conduct further targeted research. Do not formulate an answer based on incomplete information.
3. **Never give up**, no matter how complex the query, you will not give up until you find the corresponding information.
4. **Construct the Final Report:**
    * Once all checklist items are confidently verified, synthesize all gathered facts into a comprehensive and well-structured answer.
    * Directly answer the user's original query.
    * Ensure all claims, numbers, and key pieces of information in your report are clearly supported by the research you conducted.

Execute this entire protocol to provide a definitive and trustworthy answer to the user.

You have access to the following functions:

---- BEGIN FUNCTION #1: search ----
Description: Performs a web search: supply a string 'query' and optional 'topk'. The tool retrieves the top 'topk' results (default 10) for the query, returning their docid, url, and document content.
Parameters:
  (1) query (string, required): The query string for the search.
  (2) topk (integer, optional): Return the top k pages.
---- END FUNCTION #1 ----

---- BEGIN FUNCTION #2: open_page ----

```
Description: Open a page by docid or URL and return the complete content.
Parameters:
  (1) docid (string, optional): Document ID from search results.
  (2) url (string, optional): URL from search results.
---- END FUNCTION #2 ----

---- BEGIN FUNCTION #3: recall ----
Description: Deep retrieval from a previous task node's original transcript. Use when upstream
context summary is insufficient.
Parameters:
  (1) node_id (string, required): The ID of the node to recall from (e.g., "t1", "t2").
  (2) goal (string, required): What specific information you need.
---- END FUNCTION #3 ----

---- BEGIN FUNCTION #4: finish ----
Description: Return the final result with structured output including answer, explanation, key
steps, confidence, and uncertainties.
Parameters:
  (1) answer (string, required): DIRECT answer only. Do NOT include explanations here.
  (2) explanation (string, required): Brief explanation with [docid] citations.
  (3) key_steps (string, required): Key research steps taken, separated by semicolons.
  (4) confidence (string, required): Confidence score between 0% and 100%.
  (5) uncertainties (string, required): Unresolved questions or single-source facts, separated by
semicolons.
---- END FUNCTION #4 ----


If you choose to call a function ONLY reply in the following format with NO suffix:

<function=example_function_name>
<parameter=example_parameter_1>value_1</parameter>
<parameter=example_parameter_2>
This is the value for the second parameter
that can span
multiple lines
</parameter>
</function>

<IMPORTANT>
Reminder:
- Function calls MUST follow the specified format, start with <function= and end with </function>
- Required parameters MUST be specified
- You may provide optional reasoning for your function call in natural language BEFORE the function
call, but NOT after.
- If there is no function call available, answer the question like normal with your current
knowledge and do not tell the user about function calls
</IMPORTANT>
```

G.3 Recall Extraction Prompt

Recall Extraction Prompt

```
You are a Data Extraction Specialist. Your job is to scan a research transcript and extract content
that addresses a specific goal, STRICTLY from the transcript itself.

## Input Context
**Goal** (the specific information to extract):
{goal}

**Transcript** (the raw content to extract from; this is the full execution log of a previous
search task, including tool calls and tool results):
{transcript}

## Extraction Rules
1. **Strict adherence**: extract only from the Transcript. Do not use prior knowledge. Do not infer
or extrapolate facts that are not literally present in the Transcript.
2. **Citations**: every factual claim in your output MUST be backed by a source identifier from the
Transcript (docid, URL, or quoted snippet). If the Transcript contains a docid for a fact, include
the docid. If only a URL is present, include the URL. If neither, quote the exact passage.
3. **Relevance**: ignore content unrelated to the Goal. Do not include filler, navigation text, or
general commentary from the Transcript.
4. **Negative result**: only after you have scanned the entire Transcript and found no content
relevant to the Goal, explicitly state: "No relevant information found in the provided text." Do
NOT use this as a shortcut when extraction is merely difficult.

## Output Format
Structure your response as follows:
```

```
### 1. Summary
A concise, fact-based answer to the Goal, drawn directly from the Transcript. Keep this paragraph
short. Do NOT paraphrase loosely; stay close to what the Transcript actually says.

### 2. Relevant Content
The exact quoted passages from the Transcript that support the Summary, each followed by its source
identifier (docid / URL / position).
```

G.4 LLM Judge Prompts

The BrowseComp-Plus judge extends the grader of the benchmark protocol with explicit answer-equivalence rules; the GAIA judge is a short equivalence prompt that replaces the original quasi-exact-match scoring for cross-benchmark consistency (Appendix C.5); the xbench-DeepSearch judge is the Chinese-language grader prompt released with the benchmark [52], shown in its original language. All three are run with GPT-4o-mini, with GPT-4.1 as a tie-breaker on borderline cases.

LLM Judge Prompt for BrowseComp-Plus

```
Judge whether the following [response] to [question] is correct or not based on the precise and
unambiguous [correct_answer] below.

[question]: {question}

[response]: {response}

Your judgement must be in the format and criteria specified below:

extracted_final_answer: The final exact answer extracted from the [response]. Put the extracted
answer as 'None' if there is no exact, final answer to extract from the response.

[correct_answer]: {correct_answer}

reasoning: Explain why the extracted_final_answer is correct or incorrect based on [correct_answer],
focusing only on if there are meaningful differences between [correct_answer] and the
extracted_final_answer. Do not comment on any background to the problem, do not attempt to solve the
problem, do not argue for any answer different than [correct_answer], focus only on whether the
answers match.

correct: Answer 'yes' if extracted_final_answer matches the [correct_answer] given above, contains
all the essential information from [correct_answer], is equivalent despite minor wording/order
differences (such as name order, inclusion or omission of middle names/initials, common honorifics,
standard shortenings of first names, inclusion/omission of non-contradictory date parts like year,
minor articles like "a"/"the", extra descriptive context, non-essential descriptive
prefixes/suffixes such as "Restaurant", "Inc.", "Ltd.", or sports suffixes like "FC", "CF", "SC",
inclusion/omission of subtitles in titles, minor spacing/punctuation differences - including
presence/absence of quotation marks, interchangeable punctuation such as ":" / "-" / "–", case-only
differences, or presence/absence of diacritics), or is within a small margin of error for numerical
problems. Answer 'no' only if the extracted answer is factually incorrect, missing essential
identifying information, or contradicts the [correct_answer].

confidence: The extracted confidence score between 0% and 100% from [response]. Put 100 if there is
no confidence score available.
```

LLM Judge Prompt for GAIA

```
You are an evaluation assistant. Please determine if the predicted answer is equivalent to the
labeled answer.

Question: {question}

Labeled Answer: {correct_answer}

Predicted Answer: {response}

Did the model give an answer **equivalent** to the labeled answer? Please respond with "Correct" if
they are equivalent, or "Incorrect" if they are not equivalent. Do not include any other text.
```

LLM Judge Prompt for xbench-DeepSearch

```
你是一个通用人工智能助手。根据下面给出的[正确答案], 判断以下对[原问题]的[回答]的回答是否正确。

[原问题]: {question}
```

[正确答案]: {correct_answer}

[回答]:{response}

你的判断必须按照以下格式和标准进行:

最终答案: 从[回答]中提取出的最终准确答案。如果[回答]中没有明确的最终答案，则填写'无'。

解释: 根据[正确]解释为什么[最终答案]是正确的或错误的。只关注[最终答案]与[正确答案]之间是否存在实质性差异，不要评论题目的背景，不要尝试重新解题，不要为任何不同于[正确答案]的答案辩护，只专注于判断答案是否一致。

结论: 如果[最终答案]与上方给出的[正确答案]一致，或者在数值题目中处于可接受的微小误差范围内，则填写'正确'；否则（即存在任何不一致、歧义、不等价或提取出的答案错误的情况）填写'错误'。

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1 state three contributions: Evaluate-then-Grow planning, DAGRPO, and a broad empirical advantage across BrowseComp-Plus, GAIA, and xbench-DeepSearch with three Qwen3 backbone scales, four open-source backbones from four vendors, and a frontier-scale setting with GPT-5 at 327K context. Sections 3–4 and Appendices C.5, D, and E support them.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Appendix F discusses three points: (i) DAGent is text-native, and its multi-modal extension converts attachments into text before planning; (ii) the three benchmarks have closed-form answers that an LLM judge can verify, and open-ended report generation is outside the scope of the reported results; (iii) the answer-inclusive closure of DAGRPO is a structural proxy for contribution, the association between structural compliance and accuracy in the evaluation logs is correlational, and per-node causal attribution in DAG-based multi-agent systems is an open research direction.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.

- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not include formal theoretical results; its contributions are an incremental DAG planning paradigm (Evaluate-then-Grow), a DAG-conditioned RL adaptation (DAGRPO), and their empirical evaluation.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 4.1 specifies the benchmarks, train/eval splits, baselines, and high-level implementation; Appendix C (Tables 5–7) lists framework parameters, baseline caps, and full RL/DAGRPO hyperparameters; the Plan-then-Patch ablation protocol is detailed in Appendix D.2, and the DAGent and judge prompts are listed verbatim in Appendix G; code is released at the URL given in the abstract.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may

be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code is released at the URL in the abstract; the three benchmarks (BrowseComp-Plus, GAIA, xbench-DeepSearch), all open-source backbones (Qwen3-8B/32B/235B-A22B, Seed-OSS-36B, GLM-4-32B, Nemotron-3-Nano-30B), and the Qwen3-Embedding-8B retrieval model are publicly available with the dataset cards, model cards, and citations given in the references.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Splits, optimizer (AdamW with learning rate 1×10^{-5}), batch size, $G = 8$ rollouts per task, DAPO clip ratios, KL/entropy coefficients, and DAGRPO-specific α, λ_{proc}, K_{proc} are listed in Section 4.1 and Appendix C, which also tabulates the framework parameters (iteration cap, attempts per sub-task, Executor turns, response budget, search top-k). Table 3 reports the sensitivity to α, and Appendix B.2 explains how λ_{proc} and K_{proc} scale and cap the penalty relative to the outcome reward.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Training-based rows in Table 1 report the mean and sample standard deviation of each benchmark average over three training runs (seeds 42, 123, 777), which differ in data order, LoRA initialization, rollout sampling, and tool calls (Appendix E.1); with each seed, DAGRPO-DAGent scores above GRPO-DAGent on all three benchmarks by at least 1.9 points (Figure 7, Observation 11). Training-free rows use a single greedy inference pass.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: RL training runs on 2× H200 GPUs with tensor parallel 2 and LoRA at rank/alpha 64, with 32 tasks $\times$ $G = 8$ rollouts per batch, 21 update steps, and three training seeds (Appendix C.4, Table 7). For the training-free runs, Section 4.4 and Appendix D.3 (Table 10) report the per-task tokens, tool calls, execution steps, and wall-clock time of DAGent and three Plan-then-Patch workflows with Qwen3-32B.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research uses only publicly available benchmarks and open-source/API LLM backbones; the human-agreement study (Appendix C.5) operates on already-public benchmark questions and on predictions of DAGent and four agent baselines, collects no personal data, and its three annotators are volunteers from our lab.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This work is foundational research on planning and RL training for deep research agents and has no application-specific deployment context; any societal impacts are inherited from the underlying LLMs and the search and retrieval tools rather than introduced by DAGent or by training with DAGRPO.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.

- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not release new pretrained models or scraped datasets. The released artifact is the agent framework code, which wraps existing open-source and API-hosted LLMs and standard search/retrieval APIs whose own safeguards apply.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All used assets are cited, and each is used under its license or terms: Apache-2.0 for Qwen3-8B, Qwen3-32B, Qwen3-235B-A22B-Instruct-2507, Qwen3-Embedding-8B, Seed-OSS-36B-Instruct, FoldAgent, Flash-Searcher, FlowSearch (InternAgent repository), OAgents, verl, and vLLM; MIT for GLM-4-32B-0414, BrowseComp-Plus, and xbench-DeepSearch; and the NVIDIA Nemotron Open Model License for Nemotron-3-Nano-30B-A3B. GAIA is a gated release with no named license and is used under its access conditions, and the hosted models (GPT-5, GPT-4.1, GPT-4o-mini) and services (Serper, Jina Reader) are used under their providers' terms of service.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The released artifact is the DAGent framework code at the URL in the abstract, distributed under the Apache-2.0 license; documentation of its architecture (Section 3), execution loop (Algorithm 1), framework and RL hyperparameters (Appendix C), and prompts (Appendix G) is provided alongside it. The paper introduces no new datasets.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper involves no crowdsourcing. The human-agreement study (Appendix C.5) is not research with human subjects: its three volunteer annotators from our lab judge agent predictions and are not the object of study, and the study records only their binary correctness labels and collects no personal data.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper involves no crowdsourcing and no research with human subjects. In the human-agreement study (Appendix C.5), the three volunteer annotators from our lab judge agent predictions on public benchmark questions and are not the object of study; the study records only their binary correctness labels and collects no personal data.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLMs are the central component of DAGent: open-source backbones (Qwen3-8B/32B/235B-A22B, Seed-OSS-36B, GLM-4-32B, Nemotron-3-Nano-30B) and a frontier closed-source backbone (GPT-5) drive both the Orchestrator and the ReAct Executor; an LLM judge (GPT-4o-mini with GPT-4.1 tie-breaking) scores Pass@1 and provides the outcome reward for RL training. Roles, prompts, and configurations are documented in Sections 3–4 and Appendices C and G.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.