Lemon: Evidence-Risk-Aware Adaptive Organization for Long-Horizon LLM Agents

Zimo Yin[1] **Haipeng Jiang**[1] **Kailong Ren**[1] **Zhetao Sun**[1]
Guangyi Lv[1] **Congli Yin**[1] **Likang Wu**[2] **Hongke Zhao**[2]
Ming He[1,*] **Peng Wang**[1] **Jianping Fan**[3]
[1]AI Lab, Lenovo Research, Beijing, China
[2]College of Management and Economics, Tianjin University, Tianjin, China
[3]College of Intelligent Robotics and Advanced Manufacturing,
Fudan University, Shanghai, China

Abstract

Large language model agents now combine tools, long context, memory, and multi-step workflows, yet they still fail when execution loses track of the evidence required for a reliable answer. Common failure modes include missing support, unresolved contradictions, and evidence lost under context pressure. We present Lemon, an evidence-risk-aware framework that uses evidence risk to guide execution-time organization. At the intra-instance level, a prompt-based, state-conditioned controller coordinates reasoning, tool invocation, worker expansion, and verification, while a recoverable evidence substrate pairs anchored context compression with reusable semantic memory, preserving raw tool outputs while storing transferable process fragments. We further extend Lemon to an inter-instance setting in which heterogeneous personalized agents expose expertise, exchange evidence-bearing proposals, critique unsupported claims, and internalize useful collaboration artifacts. Empirically, Lemon reaches 91.36% on the GAIA test leaderboard and 80% in an in-house xbench-DeepSearch-2510 evaluation. With a shared GPT-5 backbone, it uses 50–88% fewer tokens per task than three open-source agent systems, and on a 165-task collaboration stress set, inter-instance organization raises accuracy from 80.6% to 90.9% at about 3.4× the tokens per correct answer.

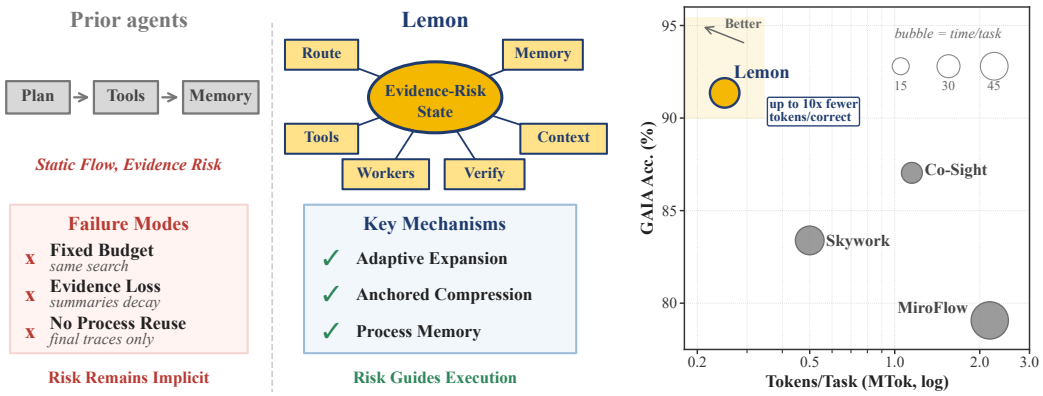
1 Introduction

Large language model agents now combine tool use, memory, workflow search, and long-context execution for search, coding, multimodal reasoning, and long-horizon tasks [1–4]. Stronger backbones and richer tool sets do not alone guarantee reliable execution. Agents still waste computation on redundant search, lose decisive evidence to repeated summarization, store past experience in forms that cannot be reused, or fail to invoke the specialized capability a task actually requires. We refer to these execution-time failures collectively as *evidence risk*, which encompasses missing support, unresolved contradictions, stale assumptions, and context-induced evidence loss.

Evidence risk turns long-horizon execution into a two-level organization problem. Within a single agent, routing, worker expansion, tool invocation, verification, context compression, and memory operations are typically governed by separate heuristics rather than by a shared view of what evidence is currently sufficient [5, 6]. Across personalized agents, recent assistants such as OpenClaw and Hermes [7, 8] accumulate distinct prompts, tools, memories, and usage histories, so collaboration

*Corresponding author: heming01@foxmail.com

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



(a) Architectural contrast. (b) Accuracy-efficiency profile.

Figure 1: Overview of Lemon. (a) Schematic of evidence-risk-guided organization versus a static workflow. (b) GAIA accuracy–token profile, measured as described in Section 6.2.

requires that expertise and supporting evidence be made visible when claims are incomplete or in conflict [9, 10]. These costs are visible in everyday use. A user of a personalized assistant such as OpenClaw often spends substantial effort searching and filtering through a large library of skills before a task can even start, and the selected skill may still need to be edited because its evidence requirements do not match the current task. As a result, a capable backbone is routinely paired with a brittle outer loop, and answer quality degrades for organizational rather than modeling reasons. Therefore, the shared control problem at both levels is allocating model calls, tools, sub-agents, context, memory, and peer communication while keeping evidence sufficiency in view.

We introduce **Lemon**, an evidence-risk-aware framework named for the segmented fruit because its agents, memory, tools, and evidence modules remain distinct while operating as one system. At the intra-instance level, one state-conditioned controller coordinates routing, tools, worker expansion, verification, context compression, and memory from evidence gaps, unsupported claims, context pressure, retrieved memory, available tools, and remaining computation. A recoverable evidence substrate preserves decisive raw evidence through truncation anchors and retrospective refinement, while reusable semantic memory and specialized visual, geospatial, file-processing, and multi-source tools share one interface. At the inter-instance level, Lemon applies the same evidence-risk view to heterogeneous personalized agents: instances expose expertise and uncertainty, submit evidence-bearing proposals, critique unsupported peer fragments, score claim-level consensus, and internalize useful artifacts for later single-instance reuse. Since both levels update state after each action, missing claims trigger retrieval, worker expansion, or peer consultation, contradictions trigger verification or critique, and context pressure triggers recoverable anchoring rather than lossy summarization. Figure 1 contrasts this design with static workflow composition and shows the resulting accuracy and token profile on GAIA.

Empirically, Lemon reaches 91.36% on the GAIA test leaderboard and 80% in an in-house xbench-DeepSearch-2510 evaluation. With GPT-5 as the shared controller and worker backbone, it uses 50–88% fewer tokens per task than MiroFlow, Co-Sight, and Skywork. The ablations in Section 6.3 identify where recoverable context, semantic memory, and specialized tools matter. On a 165-task collaboration stress set in Section 6.4, inter-instance organization raises accuracy from 80.6% to 90.9% at about 3.4× the tokens per correct answer. Our contribution is the execution-time organization of established agent primitives around evidence needs.

Our key contributions are summarized as follows:

- We formulate long-horizon LLM agent execution as evidence-risk-aware control over routing, tools, workers, verification, context, memory, and residual evidence risk.
- We develop the intra-instance Lemon architecture, which combines state-conditioned scheduling, recoverable evidence anchors, reusable semantic memory, and specialized tool invocation under a single execution loop.

- We extend Lemon to personalized inter-instance collaboration through expertise visibility, evidence-bearing proposal exchange, critique, consensus scoring, and artifact internalization.

2 Related work

Tool-augmented agents and workflow control. Reasoning-and-acting frameworks and tool-learning systems connect language models to external actions, APIs, and callable functions [1–3, 11], while workflow-search and orchestration methods optimize reusable execution graphs, agent structures, or model and tool routing [4, 12, 13]. These approaches decide which tool, workflow, or model to use under different scopes. Lemon instead coordinates tool use, worker expansion, verification, recoverable context, and memory during execution, conditioning each step on a shared evidence-risk state that summarizes evidence gaps, contradictions, context pressure, retrieved memory, available tools, and remaining computation, so a retrieved source, a contradiction, or a missing claim can directly change the next action.

Reflection, memory, and long-context management. Reflective agents and memory-augmented systems revise prior experience or store interaction traces for later reuse [5, 6, 14], and long-context retrieval has become a standard mechanism for grounding generation in external evidence. Many of these systems, however, plug memory into a fixed agent workflow, where retrieval is decoupled from the rest of the control loop. In Lemon, memory is one component of a broader control loop. Retrieval competes with direct reasoning, tool use, context recovery, verification, and delegated search, and memory writing stores reusable process fragments only after evidence-bearing execution. This integration lets retrieval and writing respond to the current evidence state instead of operating as an independent add-on.

Multi-agent collaboration. Role-based and conversational frameworks instantiate several collaborators within one workflow and aggregate their outputs through debate, voting, or coordinator prompts [15–17]. Participants are typically instances of a shared backbone distinguished by assigned roles. Frameworks such as AutoGen also allow agents with different models, tools, or human participants, but leave expertise discovery, evidence accountability, and reuse of collaboration outcomes to the application designer. Lemon addresses a setting in which heterogeneous personalized instances already differ in tools, memories, and usage histories before any task begins. This calls for explicit coordination primitives, namely expertise visibility, evidence-bearing proposals, critique of unsupported fragments, claim-level consensus, and artifact internalization. Appendix G provides additional citations and a detailed comparison along all three directions discussed above.

3 Problem setup and framework overview

We study task execution by personalized LLM agents under limited computation, context, and tool-use budgets. Let x denote a user task and let $A_i = (M_i, \mathcal{T}_i, \mathcal{M}_i, \pi_i, B_i, \theta_i)$ denote an agent instance with backbone model M_i, tool set $\mathcal{T}_i$, memory store $\mathcal{M}_i$, controller π_i, budget vector B_i, and personalization factors θ_i. At step t, the execution state is

$$s_t = (x, h_t, e_t, m_t, b_t),$$

where h_t is the trajectory history, e_t is the current evidence state, m_t is retrieved memory, and b_t is the remaining budget. The controller selects among direct reasoning, tool invocation, worker delegation, context compression, memory read or write, and inter-instance communication. Adaptive organization aims to improve task quality while controlling execution cost and residual risk:

$$\pi_i^\star = \arg\max_{\pi_i} \mathbb{E}[Q(y \mid x, s_H) - \lambda_c C_H - \lambda_l L_H - \lambda_r R_H],$$

where H is the execution horizon, Q measures answer quality, C_H cost, L_H latency, R_H residual risk such as unsupported claims or unresolved contradictions, and $\lambda_c, \lambda_l, \lambda_r \geq 0$ are tradeoff coefficients. Lemon instantiates this objective at two levels. The intra-instance level organizes one agent's routing, scheduling, tools, context, and memory. The inter-instance level organizes communication among heterogeneous personalized instances through expertise visibility, proposal exchange, critique, consensus, and artifact internalization.

Operationally, Lemon realizes this objective with a prompt-based controller rather than a trained policy or a numerical optimizer: its prompts ask the model to identify missing, questionable, or

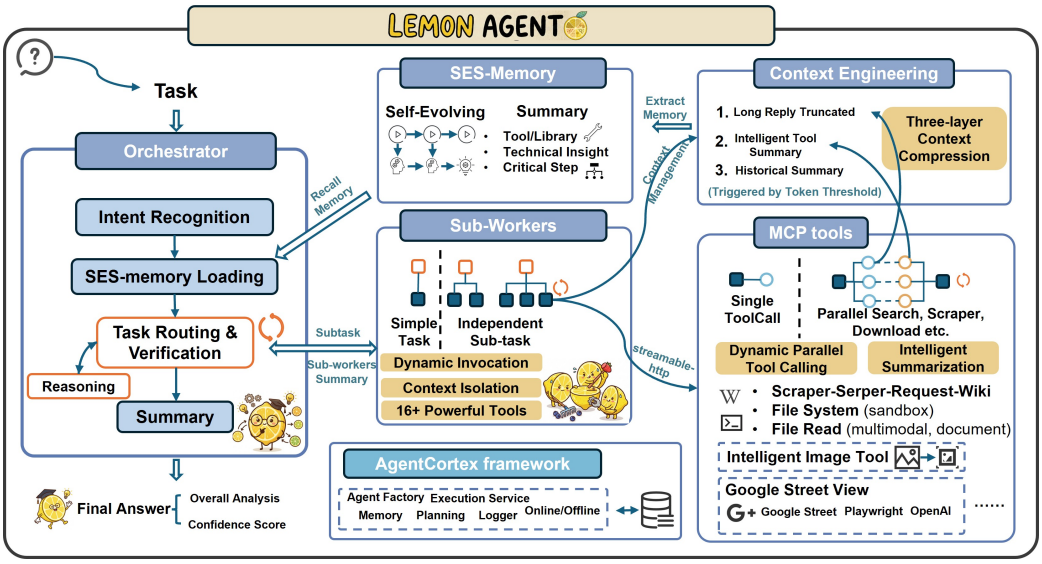


Figure 2: Intra-instance execution structure of Lemon. The controller coordinates task routing, hierarchical scheduling, context management, memory evolution, and tool use in a single agent. The "Token Threshold" in the figure corresponds to the character-count thresholds that trigger compression.

unreliable information and to gather disambiguating evidence before committing, while character thresholds and round limits are enforced programmatically. The objective therefore states design criteria, and Appendix A.1 relates it to the implemented interfaces.

4 Intra-instance adaptive organization

The intra-instance level describes how one Lemon instance turns model and tool capacity into task capability. Given the state s_t in Section 3, the controller chooses reasoning, tool use, worker delegation, raw-context recovery, and memory retrieval or writing. Character-threshold guards trigger compression. Figure 2 summarizes this structure, and Algorithm 1 gives the pipeline.

Operationally, one Lemon run proceeds through a recurrent execution loop rather than a single prompt template. The instance first interprets the task, initializes the available tools, retrieves potentially relevant memory fragments, and constructs an initial evidence state. During execution, the controller alternates between route selection, worker allocation, tool invocation, evidence integration, context compression, and verification. After answer synthesis, the same trace is inspected for reusable procedural fragments that can be written back to semantic memory. This design lets routing, context, memory, and tool use influence one another.

4.1 State-conditioned execution control

Lemon treats routing and scheduling as budgeted control rather than a fixed workflow. The controller first decides whether a task should remain in one trajectory or be expanded into specialized workers. It then decides the degree of tool parallelism, verification depth, and context management within the chosen path. For a candidate action $a \in \mathcal{A}$, the design criteria can be summarized by the conceptual utility

$$a^\star = \arg\max_{a\in\mathcal{A}} U(a \mid s_t) = \arg\max_{a\in\mathcal{A}} \big(\hat{p}_{\text{prog}}(a \mid s_t) + \lambda_r \widehat{\Delta R}(a \mid s_t) - \lambda_c C(a) - \lambda_l L(a) - \lambda_o O(a)\big),$$

where $\hat{p}_{\text{prog}}$ is an ordinal progress estimate, $\widehat{\Delta R}$ estimates reduction in evidence risk, $C(a)$ cost, $L(a)$ latency, $O(a)$ coordination overhead, and all λ coefficients are nonnegative. These estimates condition on task difficulty, unresolved evidence gaps, contradiction flags, retrieved memory, available tools, context pressure, and remaining budget. Thus Lemon can stay parsimonious on simple tasks, expand parallel search for separable evidence needs, and prefer sequential verification for tightly dependent reasoning chains.

Implementation. The utility U is not computed numerically. At each planning turn, the main-agent prompt asks the model to analyze missing, questionable, or unreliable information and, when several interpretations remain plausible, to gather disambiguating evidence before deciding. The remaining controls are numeric budgets and programmatic guards: tool outputs above 100,000 characters are anchored, current-round and history summaries are triggered at 400,000 and 600,000 characters, memory retrieval returns at most two entries with cosine similarity ≥ 0.95, and the controller and workers are capped at 10 and 20 rounds, as specified in Appendices B and E.1. The 0–100 confidence produced at answer extraction is computed after execution and is not used for control.

4.2 Recoverable context and self-evolving memory

Long-horizon tasks often generate tool outputs much larger than the evidence needed for final reasoning. Lemon therefore compresses context with recoverable truncation anchors and cross-round retrospective refinement. When an output exceeds a threshold, Lemon stores the raw output externally, inserts a compact anchor at the original message position, and later refines from the raw content under the current task context. This differs from repeated forward summarization because later compression can reconsider the original evidence rather than only a summary of a summary.

Design Contract 1 (Recoverability under Valid Anchoring and Retrieval)**.** *Let d be a decisive evidence atom reused across K compression events. If a forward-only pipeline retains d with conditional probability at most $p_s < 1$ at each event and cannot recover discarded evidence, its recoverability is at most p_s^K. If raw outputs remain accessible, an anchor is valid with probability at least $1 - \epsilon_a$, and retrieval from a valid anchor recovers d with probability at least $1 - \epsilon_r$, then recoverability is at least $(1-\epsilon_a)(1-\epsilon_r)$. This bound is independent of K only when these assumptions hold uniformly over the considered horizon.*

The contract states sufficient conditions rather than an empirical guarantee for Lemon: decisive evidence need not survive every summary as long as a valid anchor and a successful recovery remain available.

Self-evolving semantic memory (SES-Memory) operates across tasks. Instead of storing only complete successful trajectories, Lemon stores process-level fragments such as decomposition templates, evidence patterns, tool-selection heuristics, code snippets, and failure-repair notes from both successful and partially successful attempts. Retrieved memory can then influence later routing, tool selection, and verification. Detailed memory retrieval and write rules are given in Appendix B.

4.3 Specialized tool invocation

Lemon organizes specialized external capabilities as structured tools rather than as an uncurated action list. In our implementation, these tools are exposed through Model Context Protocol (MCP)-compatible interfaces, each packaging a task-specialized procedure behind a description and input-output schema. The current tool suite includes localized image refinement for fine-grained visual evidence, street-view navigation for geospatial reasoning, file processing, and multi-source search for high-fidelity information grounding. This design allows the controller to access evidence channels that generic tool calls cannot easily provide, while keeping the capabilities compatible with the same routing, context, and memory loop. Appendix F provides detailed case studies.

5 Inter-instance collective organization

The inter-instance level extends Lemon from one personalized agent to a collective of heterogeneous personalized agents. The goal is not to create another centrally scripted role-play team, but to translate social-organization principles into implementable coordination rules. Building on transactive memory and collective sensemaking, Lemon exposes expertise, exchanges evidence-bearing proposals, critiques unsupported claims, forms consensus, and internalizes traces.

This setting differs from conventional agent teams and parallel sub-agent workflows. A conventional worker is usually instantiated for the current workflow. An agent instance may have accumulated distinct tools, memories, and preferences before collaboration begins. The problem is therefore not only how to parallelize subtasks, but also how to discover differentiated expertise, make evidence accountable, and turn successful collaboration into reusable individual capability. A detailed comparison is provided in Appendix G.6.

Definition 1 (Heterogeneous Personalized Agent Collective)**.** *Let $\mathcal{I} = \{A_i\}_{i=1}^n$ denote a set of personalized agent instances. Each instance is represented as*

$$A_i = (M_i, \mathcal{T}_i, \mathcal{M}_i, \pi_i, B_i, \theta_i),$$

where M_i is the backbone model, $\mathcal{T}_i$ is the tool set, $\mathcal{M}_i$ is the memory store, π_i is the controller, B_i is the budget vector, and θ_i denotes personalization factors such as prompting style, preferences, and usage history. The collective is heterogeneous when instances differ in at least one capability-bearing component such as tools, memories, controllers, budgets, or personalization factors.

5.1 Expertise coverage

Lemon begins collaboration by estimating which instances provide complementary expertise. For a task-relevant domain $k \in \mathcal{K}(x)$, each instance exposes a visibility signal

$$v_{i,k} = \sigma(w_s s_{i,k} + w_t t_{i,k} + w_m m_{i,k} + w_q q_{i,k}),$$

where $s_{i,k}$ is historical success on related tasks, $t_{i,k}$ measures tool coverage, $m_{i,k}$ measures relevant memory retrieval, $q_{i,k}$ is current self-estimated confidence, $w_s, w_t, w_m, w_q \geq 0$, and $\sigma : \mathbb{R} \to [0, 1]$ maps the score to a nonnegative bounded visibility value.

Definition 2 (Collective Capability Coverage)**.** *For a task x with relevant domains $\mathcal{K}(x)$ and nonnegative domain weights ω_k, the coverage of an instance subset $S \subseteq \mathcal{I}$ is*

$$F(S, x) = \sum_{k \in \mathcal{K}(x)} \omega_k \max_{A_i \in S} v_{i,k}.$$

We set the maximum over an empty set to zero. The coverage gain of adding instance A_j to S is $\Delta_j(S, x) = F(S \cup \{A_j\}, x) - F(S, x)$.

Proposition 1 (Coverage Gain from Greedy Expertise Routing)**.** *For nonnegative weights and visibility scores $v_{i,k} \in [0, 1]$, $F(S, x)$ is monotone and submodular in S. Under the standard cardinality-constrained greedy rule [18], if one selects b instances by repeatedly adding the instance with the largest marginal coverage gain, the selected set S_g satisfies*

$$F(S_g, x) \geq (1 - 1/e) F(S^\star, x),$$

where $S^\star$ is the best subset of size b under the same coverage objective.

This is the standard monotone-submodular guarantee applied to the coverage objective. It motivates complementary expertise selection and does not by itself bound answer accuracy.

5.2 Proposal, critique, and evidence-aware consensus

After expertise exposure, each participating instance produces a structured proposal artifact containing a candidate answer or partial hypothesis, supporting evidence, confidence, uncertainty notes, requested checks, and a cost trace. Other instances can then request missing evidence, flag contradictions, and revise weak fragments before synthesis. Consensus is formed over claim fragments rather than opaque final answers. Operationally, Lemon decomposes proposals into atomic schema-aligned claim fragments, canonicalizes equivalent fragments, merges duplicates, and retains supported contradictions as competing candidates. Each candidate keeps pointers to source evidence, provenance groups, proposing instances, critique notes, and cost traces. Appendix C.1 gives the normalized scoring implementation. For a candidate claim z, Lemon scores

$$S(z) = \alpha_e E(z) + \alpha_c C_{\text{ind}}(z) + \alpha_v V(z) - \alpha_d D(z) - \alpha_u U(z) - \alpha_l L(z),$$

where $E(z)$ measures evidence support, $C_{\text{ind}}(z)$ measures corroboration from distinct provenance groups, $V(z)$ measures verification completeness, $D(z)$ measures unresolved contradiction mass, $U(z)$ measures unsupported portions, $L(z)$ is a cost or latency penalty, and all α coefficients are nonnegative. The independence term reduces repeated-source corroboration. Provenance groups are approximated from source identifiers, so this term de-duplicates sources rather than guaranteeing independent evidence. Appendix C.1 gives the scoring rules of the released reference implementation.

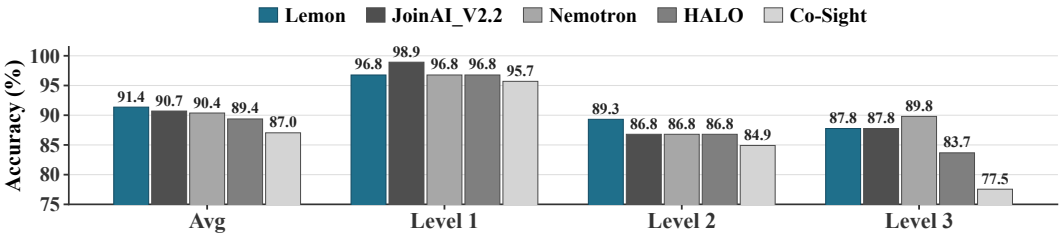


Figure 3: Level-wise GAIA test accuracy across three difficulty tiers: short single-tool tasks (Level 1), multi-tool workflows of 5–10 steps (Level 2), and open-ended tasks with unbounded steps and tools (Level 3). The snapshot of February 6, 2026 covers Lemon, JoinAI_V2.2, Nemotron-ToolOrchestra-0106/0107 [13], HALO V1217-1, and Co-Sight_v2.1.0, as listed in Table 1(a). Entries differ in models, tools, and budgets.

Design Contract 2 (Consensus under Uniform Calibration)**.** *Let $\mathcal{J}^\star(z)$ be the true decision loss of candidate claim z under the same evidence, contradiction, unsupportedness, verification, and normalized cost criteria used for consensus, and let Lemon use the computable surrogate*

$$\widehat{\mathcal{J}}(z) = B_0 - S(z),$$

where B_0 is a constant large enough to keep the surrogate nonnegative and does not affect the argmax. If the surrogate is uniformly calibrated, so that $|\widehat{\mathcal{J}}(z) - \mathcal{J}^\star(z)| \le \epsilon$ for every candidate claim $z \in \mathcal{Z}$, then the Lemon consensus choice $\hat{z} = \arg\max_z S(z)$ satisfies

$$\mathcal{J}^\star(\hat{z}) \le \min_{z \in \mathcal{Z}} \mathcal{J}^\star(z) + 2\epsilon.$$

Uniform calibration is a sufficient assumption, not an experimentally established property of the reported consensus scores.

The final stage retains accepted claims, their evidence provenance, and critique records in participating instances' semantic memories for explicit later reuse. Extracting reusable decomposition plans and critique templates is a further extension. Appendix C provides conditional analyses of artifact utility, and measuring gains on later tasks is left to future work.

6 Experiments

6.1 Main results on agent benchmarks

GAIA [19] evaluates general-purpose AI assistants on web retrieval, multi-step reasoning, multimodal perception, and heterogeneous file processing. The benchmark has 466 tasks, 165 for validation and 301 for test [20, 21]. Lemon's submission of February 6, 2026 solves 275 of 301 tasks, or 91.36%, with 96.77%, 89.31%, and 87.76% on Levels 1–3, that is, 90/93, 142/159, and 43/49 tasks [22]. Appendix E.4 identifies the submitted run and the repeated-run protocol. Table 1(a) lists selected top-ranked entries of the same snapshot, and Figure 3 compares level-wise accuracy. The entries differ in backbone models, tools, and test-time compute, so the snapshot reflects standing rather than the effect of any single component. The top entries are within a few tasks of one another, and the 95% Wilson interval of Lemon's score is 87.6–94.0%.

On xbench-DeepSearch-2510 [23], Lemon scores 80/100 in an in-house run of April 16, 2026 with the official `xbench-evals` harness, as detailed in Appendix E.8. Public systems were not rerun under this protocol, so Table 1(b) lists the result separately from the public leaderboard [24] and no ranking is claimed. We report it only as evidence that the approach applies beyond GAIA.

6.2 Efficiency analysis

We compare Lemon with MiroFlow [25], Co-Sight [26], and Skywork [27]. All four systems run on the same GAIA task list, with one attempt per task and GPT-5 at high reasoning effort as controller and worker backbone. Each baseline keeps its released defaults, and Lemon uses 10 controller and

Table 1: Benchmark context. (a) Selected top-ranked GAIA test leaderboard entries as of February 6, 2026, plus Co-Sight_v2.1.0. Counts are solved/total tasks, and Table 19 lists the submitters' registered models [22]. (b) Selected public xbench-DeepSearch entries, accessed May 5, 2026, and our in-house run, which follows a different protocol. Accuracy is in percent.

(a) GAIA test, snapshot of February 6, 2026

Entry (date)	Overall	L1	L2	L3
Lemon (2026-02-06)	**275 (91.36)**	90/93	142/159	43/49
JoinAI_V2.2 (2026-01-14)	273 (90.70)	92/93	138/159	43/49
Nemotron-ToolOrchestra-0106 (2026-01-06)	272 (90.37)	90/93	138/159	44/49
JoinAI_V2.1 (2026-01-13)	271 (90.03)	92/93	138/159	41/49
SU Zero–Shuqian Series Pro MAX (2025-12-23)	271 (90.03)	92/93	138/159	41/49
HALO V1217-1 (2025-12-17)	269 (89.37)	90/93	138/159	41/49
Co-Sight_v2.1.0 (2025-10-13)	262 (87.04)	89/93	135/159	38/49

(b) xbench-DeepSearch

Product	Organization	Accuracy	Evaluation Date
StepFun Research	StepFun	38	2025-08-28
Minimax Agent	Minimax	38	2025-11-04
SuperGrok Expert	xAI	42	2025-08-28
Kimi K2.5 (Thinking)	Moonshot AI	46	2026-01-29
Gemini Pro	Google	53	2026-02-16
ChatGPT-5-Pro	OpenAI	79	2025-08-28
In-house DeepSearch-2510 evaluation (separate protocol)			
Lemon Agent	This work	**80**	2026-04-16

Table 2: Resource comparison on GAIA. Acc. is the public test leaderboard score of each system's entry, with solved tasks out of 301. Time in minutes and Tokens in MTok are per-task averages of the comparison harness, and /Correct divides them by accuracy.

Method	Acc. (solved/301)	Time/Task (min)	Tokens/Task (MTok)	Time/Correct (min)	Tokens/Correct (MTok)
MiroFlow	79.07% (238)	44.7	2.17	56.5	2.74
Co-Sight	87.04% (262)	**11.8**	1.15	**13.6**	1.32
Skywork	83.39% (251)	25.1	0.50	30.1	0.60
Lemon	**91.36% (275)**	17.5	**0.25**	19.2	**0.27**

20 worker rounds, as detailed in Appendix E.1. Lemon additionally routes image, street-view, audio, reasoning, and answer-extraction calls to the other hosted models listed in Table 15.

Lemon has the highest leaderboard accuracy and the lowest token use. At 0.25 MTok per task, it uses 50%, 78%, and 88% fewer tokens than Skywork at 0.50, Co-Sight at 1.15, and MiroFlow at 2.17, or about 2.2×, 4.9×, and 10.1× fewer tokens per correct answer. Co-Sight is faster, at 11.8 versus 17.5 minutes per task. Even if every baseline were credited with 100% accuracy, its tokens per correct answer would still be 1.9–8.0× Lemon's 0.27.

Repeated-run stability. We repeat each system three times in an identical configuration on the same task list, with one answer per task and no exposed random seed. Lemon reaches $91.25 \pm 0.51\%$, Co-Sight $86.93 \pm 0.51\%$, Skywork $83.50 \pm 0.84\%$, and MiroFlow $79.07 \pm 0.66\%$ as mean $\pm$ sample SD, with the same accuracy ordering in every run. Appendix E.4 gives the per-run results, and Lemon's leaderboard submission is Run 2, the median of its three runs.

6.3 Intra-instance ablation

The intra-instance ablations isolate three mechanisms: recoverable context compression, self-evolving semantic memory, and specialized tools. We evaluate each removal on a matched stress slice, not the full GAIA benchmark, because these mechanisms are not expected to affect every GAIA task uniformly. The matched stress slices focus on evidence loss under long outputs, repeated search

in reusable task families, fine-grained visual inspection, and Street-View evidence, as specified in Appendix E.2.

Table 3: Matched task-slice ablations for intra-instance Lemon. ΔAcc. is computed from task counts.

Removed module	**Matched Task Slice (Task Num.)**	**Slice Acc.**	**ΔAcc.**	**MTok**
Recoverable evidence substrate				
w/o context recovery	Context pressure (68)	94.1% → 80.9%	**-13.2 pp**	0.41 → 0.53
w/o SES-Memory	Procedural reuse (82)	91.5% → 81.7%	**-9.8 pp**	0.39 → 0.52
Specialized evidence tools				
w/o image tool	Fine-grained vision (59)	91.5% → 76.3%	**-15.3 pp**	0.38 → 0.31
w/o street-view tool	Street-View (10)	90.0% → 70.0%	**-20.0 pp**	0.52 → 0.44

Table 3 measures localized effects where each mechanism is exercised. Context recovery supports revisiting compressed evidence, SES-Memory avoids repeated search, and specialized tools extract visual evidence. Token use rises when removal causes compensatory search and falls when disabling a costly tool, even if accuracy declines.

Table 4: Adaptive parallel search efficiency ablation. Reductions are relative to single-worker search; negative values indicate increased resource use. Reductions are computed from unrounded values.

Configuration	**Time/Task (min)**	**Time reduction vs. single**	**Tokens/Task (MTok)**	**Token reduction vs. single**
Single sub-agent search	21.7	–	1.24	–
Fixed parallel search (2 workers)	49.0	-125.4%	1.74	-40.7%
Adaptive parallel search	**16.4**	**24.5%**	**0.63**	**49.2%**

Table 4 compares worker-allocation policies. Forced two-worker search adds redundant work, while adaptive allocation reduces time by 24.5% and tokens by 49.2% relative to single-worker search. Accuracy is not reported for these arms, so this comparison concerns resource use only.

Full-benchmark removals and sensitivity. Table 5 applies the component removals to all 301 tasks. The joint removal loses 29 tasks, slightly fewer than the 30-task sum of individual removals; slice effects should not be read as benchmark-wide effects. Halving or doubling six numerical controls one at a time changes accuracy by at most 1.33 points, whereas reducing the controller cap from 10 to 5 rounds lowers accuracy from 91.36% to 88.04% (Appendix E.7).

Table 5: Full-benchmark removals. Accuracy is in percent with solved counts out of 301, and changes are in percentage points relative to the reference of 91.36%, or 275 tasks.

Removed component	Slice n	Slice Δ (pp)	Full Acc. (%)	Full Δ (pp)
Context recovery	68	-13.2	88.04 (265)	-3.32
SES-Memory	82	-9.8	88.37 (266)	-2.99
Image tool	59	-15.3	88.37 (266)	-2.99
Street-view tool	10	-20.0	90.70 (273)	-0.66
All four	–	–	81.73 (246)	-9.63

Controller interventions. Component removals do not isolate the controller. Table 6 instead changes its evidence clauses while fixing the backbone, tools, and caps. Accuracy drops by 4.32 and 6.64 points for B and C, and token use rises. The placebo changes one task, and the A–B contrast retains its direction on a second backbone. Appendix E.6 gives the prompt changes and categorizes the 19 tasks solved by A but missed by B.

6.4 Inter-instance study

The inter-instance study uses a fixed 165-task GAIA-style stress set selected for multi-source synthesis, entity disambiguation, or visual/geospatial verification, with selection rules and implementation notes

Table 6: Controller interventions on the full benchmark. Each arm is a single run. Accuracy is in percent with solved counts out of 301, and Δ is in percentage points relative to A, or to A_2 on the second backbone. Discordant counts are tasks solved only by the reference / only by the arm, and p is from an exact McNemar test.

Arm	Acc. (%)	Δ	MTok/task	MTok/correct	Discordant	p
A (shipped)	91.36 (275)	–	0.25	0.27	–	–
A′ (placebo)	91.03 (274)	-0.33	0.25	0.27	4/3	1.00
B (state not used)	87.04 (262)	-4.32	0.31	0.36	19/6	.015
C (clauses removed)	84.72 (255)	-6.64	0.38	0.45	25/5	.0003
Second backbone						
A_2 (as A)	88.04 (265)	–	–	–	–	–
B_2 (as B)	84.39 (254)	-3.65	–	–	17/6	.035

Table 7: Inter-instance evaluation on the fixed 165-task stress set. Δ is the accuracy change in percentage points relative to isolated Lemon, and p is from an exact McNemar test on paired outcomes.

Configuration	**Solved**	**Acc.**	Δ	**95% CI (Acc.)**	p
Isolated Lemon	133/165	80.6%	–	73.9–85.9	–
Direct communication	139/165	84.2%	+3.6 pp	77.9–89.0	.180
Fixed-role organization	143/165	86.7%	+6.1 pp	80.6–91.0	.021
Full inter-instance Lemon	**150/165**	**90.9%**	**+10.3 pp**	**85.5–94.4**	**<.001**

in Appendix D. All configurations share the paired task set, candidate pool, backbone, evaluator, and per-instance limits, while total budget scales with the number of instances.

The isolated reference has no communication and also differs in tools, memories, and total budget. The three multi-instance arms share a heterogeneous profile pool. Direct communication exchanges answers, fixed-role organization assigns predefined roles, and full Lemon adds expertise routing, structured proposals, peer critique, claim-level consensus, and artifact internalization.

Full Lemon gains 10.3 points over the isolated reference on this stress set (Table 7), at 3.4× the tokens per correct answer, 1.08 versus 0.32 MTok. Our efficiency claim therefore concerns the single-instance comparison in Table 2. Four homogeneous instances reach 85.5% and an alternative heterogeneous pool reaches 89.7%, showing contributions from both protocol and profiles. Gains decrease from 25.0 to 18.2 to 7.6 points for tasks meeting three, two, or one selection rule. The 131 one-rule tasks still benefit; Appendix D.1 gives the details.

7 Conclusion and limitations

We present Lemon, a two-level framework for adaptive organization in LLM agent systems. Within an instance, it controls routing, worker expansion, recoverable context compression, reusable semantic memory, and specialized tool invocation from a shared evidence-risk state under a single prompt-based controller. Across instances, the same view coordinates heterogeneous personalized agents through expertise visibility, evidence-bearing proposals, critique, consensus scoring, and artifact internalization. The GAIA and xbench results, the resource comparison, and the ablations are consistent with the value of evidence-directed execution, and inter-instance organization improves accuracy on a collaboration stress set at a higher token cost.

Limitations. Three limitations remain. First, the controller is a hand-specified prompt rather than a learned policy, and the design contracts rest on assumptions that we do not verify empirically. Second, the evidence comes from two benchmarks and one collaboration stress set with limited repetition, and systems on the leaderboards differ in models, tools, and compute. Third, memory internalization and artifact sharing raise privacy, provenance, and over-reliance risks. We are addressing these points in ongoing work and leave a full treatment to future work, including learned control policies, larger repeated evaluations in interactive, multilingual, and domain-specific settings, and safeguards such as bounded permissions, provenance tracking, and artifact filtering.

Acknowledgments and Disclosure of Funding

We thank Xin Gan, Hong Pan, Changwen Zhang, Shan Tong, Zhengyu Xu, Zeping Chen, Yubin Huangfu, Yanzhi Xu, Xing Su, Qin Feng, and Dong An for their contributions to the Lemon Agent system. This work was supported by the National Natural Science Foundation of China under Grant No. 62502340 and the Natural Science Foundation of Anhui Province under Grant No. 2508085MF176. Apart from current and past employment at the affiliations listed above, the authors declare no competing interests.

References

[1] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

[2] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, 2023.

[3] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations*, 2024.

[4] Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. In *The Thirteenth International Conference on Learning Representations*, 2025.

[5] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

[6] Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *The Thirteenth International Conference on Learning Representations*, 2025.

[7] OpenClaw Team. OpenClaw: Personal AI assistant, 2026. URL `https://github.com/openclaw/openclaw`. GitHub repository.

[8] Nous Research. Hermes Agent, 2026. URL `https://github.com/NousResearch/Hermes-Agent`. GitHub repository.

[9] Beatrice Biancardi, Maurizio Mancini, Brian Ravenet, and Giovanna Varni. Modelling the "transactive memory system" in multimodal multiparty interactions. *Journal on Multimodal User Interfaces*, 18:103–117, 2024.

[10] Vitaliy Popov, Catherine Gabelica, Sarah Tomaka, and Theodora Danciu. Making the invisible visible: how multisource feedback and guided facilitation affect team reflection. *Cognition, Technology & Work*, 28:135–154, 2026. doi: 10.1007/s10111-025-00835-4.

[11] Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive APIs. In *Advances in Neural Information Processing Systems*, 2024.

[12] Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. AgentSquare: Automatic LLM agent search in modular design space. In *The Thirteenth International Conference on Learning Representations*, 2025.

[13] Hongjin Su, Shizhe Diao, Ximing Lu, Mingjie Liu, Jiacheng Xu, Xin Dong, Yonggan Fu, Peter Belcak, Hanrong Ye, Hongxu Yin, Yi Dong, Evelina Bakhturina, Tao Yu, Yejin Choi, Jan Kautz, and Pavlo Molchanov. ToolOrchestra: Elevating intelligence via efficient model and tool orchestration. In *International Conference on Machine Learning*, 2026.

[14] Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. arXiv preprint arXiv:2310.08560, 2023. URL https://arxiv.org/abs/2310.08560v2.

[15] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, 2024.

[16] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversations. In *Conference on Language Modeling*, 2024.

[17] Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *International Conference on Machine Learning*, 2024.

[18] George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming*, 14(1): 265–294, 1978. doi: 10.1007/BF01588971.

[19] Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for general AI assistants. In *International Conference on Learning Representations*, 2024.

[20] GAIA Benchmark. GAIA benchmark dataset. https://huggingface.co/datasets/gaia-benchmark/GAIA, 2024. Official Hugging Face gated-access dataset page; accessed 2026-04-30.

[21] GAIA Benchmark. GAIA leaderboard. https://huggingface.co/spaces/gaia-benchmark/leaderboard, 2026. Accessed 2026-09-26.

[22] GAIA Benchmark. GAIA public submission results, 2026. URL https://huggingface.co/datasets/gaia-benchmark/results_public. Accessed 2026-10-23.

[23] 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.

[24] xbench. xbench AI Search Leaderboard. https://xbench.org/agi/aisearch, 2026. Accessed 2026-05-05.

[25] MiroMind AI Team. MiroFlow: A high-performance open-source research agent framework. https://github.com/MiroMindAI/MiroFlow, 2025.

[26] Hongwei Zhang, Ji Lu, Shiqing Jiang, Chenxiang Zhu, Li Xie, Chen Zhong, Haoran Chen, Yurui Zhu, Yongsheng Du, Yanqin Gao, Lingjun Huang, Baoli Wang, Fang Tan, and Peng Zou. Co-Sight: Enhancing LLM-based agents via conflict-aware meta-verification and trustworthy reasoning with structured facts. *arXiv preprint arXiv:2510.21557*, 2025. URL https://arxiv.org/abs/2510.21557.

[27] Skywork AI. DeepResearchAgent. https://github.com/SkyworkAI/DeepResearchAgent, 2025.

[28] xbench. xbench-DeepSearch dataset. https://huggingface.co/datasets/xbench/DeepSearch-2510, 2025. Hugging Face dataset page, MIT license; accessed 2026-04-30.

[29] xbench. xbench-evals. https://github.com/xbench-ai/xbench-evals, 2025. GitHub repository, MIT license; accessed 2026-04-30.

[30] ZTE-AICloud. Co-Sight. https://github.com/ZTE-AICloud/Co-Sight, 2025. GitHub repository, Apache-2.0 license; accessed 2026-11-04.

[31] CAMEL-AI. CAMEL: Communicative agents, 2026. URL https://github.com/camel-ai/camel. Apache-2.0 repository; accessed 2026-10-23.

[32] MiroMind AI Team. MiroFlow README (commit fcb2425). https://github.com/MiroMindAI/MiroFlow/blob/fcb2425e87386d9a8ceada000c64e75c7140e9e4/README.md, 2025. Accessed 2026-09-27.

[33] Skywork AI. Skywork launches Skywork Super Agents globally, 2025. Press release. URL https://www.prnewswire.com/news-releases/skywork-launches-skywork-super-agents-globally-the-ai-powered-office-suite-built-on-deep-research-302463133.html.

[34] Skywork AI. DeepResearchAgent README (commit 5b8711a). https://github.com/SkyworkAI/DeepResearchAgent/blob/5b8711aaa0c564ac69f847e046c5c3002b0ead60/README.md, 2025. Accessed 2026-11-04.

[35] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. HuggingGPT: Solving AI tasks with ChatGPT and its friends in Hugging Face. In *Advances in Neural Information Processing Systems*, 2023.

[36] Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. Chameleon: Plug-and-play compositional reasoning with large language models. In *Advances in Neural Information Processing Systems*, 2023.

[37] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

[38] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, 2023.

[39] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *International Conference on Machine Learning*, 2023.

[40] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023.

[41] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations*, 2024.

[42] Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024.

[43] Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik R. Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *The Thirteenth International Conference on Learning Representations*, 2025.

[44] Chris Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Toyama, Robert Berry, Divya Tyamagundlu, Timothy Lillicrap, and Oriana Riva. AndroidWorld: A dynamic benchmarking environment for autonomous agents. In *The Thirteenth International Conference on Learning Representations*, 2025.

[45] Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Aleksander Madry, and Lilian Weng. MLE-bench: Evaluating machine learning agents on machine learning engineering. In *The Thirteenth International Conference on Learning Representations*, 2025.

[46] Ziru Chen, Shijie Chen, Yuting Ning, Qianheng Zhang, Boshi Wang, Botao Yu, Yifei Li, Zeyi Liao, Chen Wei, Zitong Lu, Vishal Dey, Mingyi Xue, Frazier N. Baker, Benjamin Burns, Daniel Adu-Ampratwum, Xuhui Huang, Xia Ning, Song Gao, Yu Su, and Huan Sun. ScienceAgentBench: Toward rigorous assessment of language agents for data-driven scientific discovery. In *The Thirteenth International Conference on Learning Representations*, 2025.

[47] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, 2020.

[48] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George B. M. van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning*, 2022.

[49] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Annual ACM Symposium on User Interface Software and Technology*, 2023.

[50] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024.

[51] Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: Communicative agents for mind exploration of large language model society. In *Advances in Neural Information Processing Systems*, 2023.

[52] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. ChatDev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15174–15186, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.810.

[53] Xiaohe Bo, Zeyu Zhang, Quanyu Dai, Xueyang Feng, Lei Wang, Rui Li, Xu Chen, and Ji-Rong Wen. Reflective multi-agent collaboration based on large language models. In *Advances in Neural Information Processing Systems*, 2024.

[54] Kunlun Zhu, Hongyi Du, Zhaochen Hong, Xiaocheng Yang, Shuyi Guo, Zhe Wang, Zhenhailong Wang, Cheng Qian, Xiangru Tang, Heng Ji, and Jiaxuan You. MultiAgentBench: Evaluating the collaboration and competition of LLM agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8580–8622, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.421.

[55] Yanwei Yue, Guibin Zhang, Boyang Liu, Guancheng Wan, Kun Wang, Dawei Cheng, and Yiyan Qi. MasRouter: Learning to route LLMs for multi-agent systems. In *Proceedings of the 63rd*

Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 15549–15572, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.757.

[56] Zhexuan Wang, Yutong Wang, Xuebo Liu, Liang Ding, Miao Zhang, Jie Liu, and Min Zhang. AgentDropout: Dynamic agent elimination for token-efficient and high-performance LLM-based multi-agent collaboration. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 24013–24035, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.1170.

[57] Jen-tse Huang, Jiaxu Zhou, Tailin Jin, Xuhui Zhou, Zixi Chen, Wenxuan Wang, Youliang Yuan, Michael R. Lyu, and Maarten Sap. On the resilience of LLM-based multi-agent collaboration with faulty agents. In *International Conference on Machine Learning*, 2025.

[58] Daniel M. Wegner, Toni Giuliano, and Paula T. Hertel. Cognitive interdependence in close relationships. In William J. Ickes, editor, *Compatible and Incompatible Relationships*, pages 253–276. Springer, 1985.

[59] Kyle Lewis, Donald Lange, and Lynette Gillis. Transactive memory systems, learning, and learning transfer. *Organization Science*, 16(6):581–598, 2005. doi: 10.1287/orsc.1050.0143.

[60] Karl E. Weick, Kathleen M. Sutcliffe, and David Obstfeld. Organizing and the process of sensemaking. *Organization Science*, 16(4):409–421, 2005.

[61] Anita Williams Woolley, Christopher F. Chabris, Alex Pentland, Nada Hashmi, and Thomas W. Malone. Evidence for a collective intelligence factor in the performance of human groups. *Science*, 330(6004):686–688, 2010.

[62] Tina Comes. Sensemaking AI: Introducing a research and design agenda for human–AI networks. *EPJ Data Science*, 15:42, 2026. doi: 10.1140/epjds/s13688-026-00634-5. URL `https://link.springer.com/article/10.1140/epjds/s13688-026-00634-5`.

Appendix

A Complete intra-instance execution procedure

This appendix gives an algorithmic overview of the operational pipeline for the intra-instance Lemon agent summarized in Section 4. Algorithm 1 presents the end-to-end loop. The procedure integrates the architectural components introduced in the main text. Lemon first initializes its core execution services, interprets the user task, retrieves relevant SES-Memory entries, and loads task-relevant tools and functional libraries. It then enters a reasoning loop in which the controller repeatedly performs task routing, worker allocation, tool invocation, context compression, verification, and evidence integration.

Algorithm 1 Intra-Instance Lemon Execution Pipeline

Require: User task T; thresholds $\tau_{\max}$, τ_{multi}, τ_{context} in characters
Ensure: Final answer A with confidence score S_{conf}

Initialize core services for agents, execution, tools, and memory.
Build the initial task state s_0 by interpreting task intent and constraints.
Retrieve top-k relevant SES-Memory entries $\mathcal{M}_{\text{SES}}(T)$.
Load task-relevant tools and functional libraries suggested by the task state and retrieved memory.

Optionally refine the task goal, evidence needs, and subtask options.
Initialize worker set $\mathcal{W} \leftarrow \emptyset$, evidence state e_0, and context state h_0.
while T is not resolved and the remaining budget is sufficient **do**
    Describe task needs, dependencies, and missing, questionable, or unreliable evidence qualitatively.
    Select the next action through the controller prompt and exposed tool interfaces.
    if the selected action is direct reasoning **then**
        Continue the main reasoning trajectory and update the evidence state.
    else if the selected action is delegated execution **then**
        Route subtasks to workers $\mathcal{W} = \{w_1, \ldots, w_n\}$ according to task structure.
        for each worker $w_i \in \mathcal{W}$, sequentially or with configured concurrency **do**
            Initialize worker context C_i from the subtask, memory, and evidence.
            while w_i has not terminated and its local budget remains **do**
                Perform independent reasoning on the assigned subtask.
                Select tools $\mathcal{T}_i = \{t_1, \ldots, t_m\}$ according to the worker state.
                Execute either a single call or dynamic parallel calls and collect outputs $\mathcal{R}_i$.
                Apply tool-level truncation anchors to any output $r_j \in \mathcal{R}_i$ with $|r_j| > \tau_{\max}$.
                If multi-tool outputs exceed τ_{multi}, persist every raw output and retain its anchor before summarizing the round.
                If $|C_i| > \tau_{\text{context}}$, apply cross-round retrospective context compression.
                Update C_i and the worker evidence state with the processed results.
            end while
            Verify worker evidence when verification is selected or evidence risk remains high.
            Return compact answer fragments, evidence, and confidence.
        end for
        Integrate worker outputs into the main evidence state.
    else if the selected action is memory or context management **then**
        Retrieve memory, write a reusable trace, or recover raw text through an authorized anchor.
    end if
    Update the global task state $s_t = (T, h_t, e_t, m_t, b_t)$ and the remaining budget.
end while
Extract final answer A and post-hoc confidence S_{conf} in a final extraction call (S_{conf} is not used for control).
Extract reusable procedural fragments from the trace and write accepted items into SES-Memory.

return A, S_{conf}

Within the loop, Lemon uses macro-level routing to determine whether the task should remain in a single reasoning trajectory or be expanded into delegated workers. When delegated workers are used, each worker maintains its own working context, performs independent reasoning on its assigned subtask, and selects between sequential tool use and dynamic parallel tool calls. Specialized tools such as localized image refinement, street-view navigation, file processing, and multi-source search are invoked when the task requires the corresponding evidence channel.

The procedure also couples execution with context and memory management. Lemon monitors accumulated tool outputs and applies the hierarchical compression mechanism when threshold conditions are met. This includes tool-level truncation anchors, intra-round synthesis for multiple tool outputs, and cross-round retrospective refinement when the session context grows large. After answer synthesis, Lemon extracts high-value procedural fragments from the execution trace and writes reusable tool procedures, evidence patterns, or repair traces into SES-Memory.

A.1 Operational state and action interface

This subsection relates the conceptual quantities of Sections 3–4 to the implementation. The controller receives the task transcript, tool descriptions, retrieved memory, and prompt instructions, and Table 8 groups the information relevant to its decisions. Evidence needs and uncertainty are expressed in natural language, whereas context lengths and round limits are numeric. The quantities defined below are analytical abstractions for the design analysis and are not computed by the controller.

Table 8: Conceptual categories of controller information.

Feature	**Definition**	**Use in control**
Task route	Coarse task type, dependency structure, and estimated difficulty	Determines whether to stay in one trajectory or expand workers
Evidence gap	Required evidence items not yet supported by accepted sources	Triggers search, tool use, or worker delegation
Claim support	Supported, unsupported, and partially supported claim fragments	Guides verification and final synthesis readiness
Contradiction status	Conflicting claims or conflicting sources that remain unresolved	Triggers critique, additional retrieval, or down-weighting
Context pressure	Current context length relative to compression thresholds	Triggers anchored compression or retrospective refinement
Memory relevance	Top-k retrieved memory scores and reusable process fragments	Biases decomposition, tool choice, and repair strategy
Tool match	Match between task evidence needs and available tool descriptions	Selects specialized visual, geospatial, file, or search tools
Budget remaining	Configured controller/worker round limits and output cap	Bounds execution

For the conditional analysis, let $G_t, U_t, D_t, X_t \in [0, 1]$ denote abstract unresolved evidence requirements, unsupported claim mass, contradiction mass, and context-loss risk. A possible scalar summary is

$$R_t = w_g G_t + w_u U_t + w_d D_t + w_x X_t, \qquad w_g, w_u, w_d, w_x \geq 0, \quad \sum_{\ell \in \{g,u,d,x\}} w_\ell = 1.$$

One bookkeeping abstraction expresses changes using resolved and newly created masses: r_g, n_g for gaps, r_u, n_u for unsupported claims, r_d, n_d for contradictions, and r_x, n_x for context loss:

$$\begin{aligned} G_{t+1} &= \mathrm{clip}_{[0,1]}(G_t - r_g + n_g), \\ U_{t+1} &= \mathrm{clip}_{[0,1]}(U_t - r_u + n_u), \\ D_{t+1} &= \mathrm{clip}_{[0,1]}(D_t - r_d + n_d), \\ X_{t+1} &= \mathrm{clip}_{[0,1]}(X_t - r_x + n_x). \end{aligned}$$

This abstraction defines $\Delta R(a_t \mid s_t) = R_t - \mathbb{E}[R_{t+1} \mid s_t, a_t]$.

Table 9 groups actions into conceptual families whose implemented interfaces differ. Tools return structured outputs, workers return messages, and context and memory operations are triggered programmatically.

Table 9: Conceptual action families, typical triggers, and resulting information.

Action	Typical trigger	Returned update
Direct reasoning	Low evidence gap and low contradiction risk	Draft answer fragment and confidence note
Tool invocation	A concrete external query or computation is needed	Source output, provenance, and evidence atoms
Specialized tool invocation	Specialized evidence channel needed	Structured tool output and refined evidence state
Worker delegation	Evidence needs are separable or search space is broad	Worker summaries, evidence, cost, and uncertainty notes
Claim verification	A candidate answer exists but support is incomplete	Verified, rejected, or weakened claim fragments
Context compression	Context pressure exceeds threshold	Anchor identifiers, summaries, and raw-output pointers
Memory retrieval	Retrieved experience may alter routing or tool choice	Ranked memory fragments and relevance scores
Memory writing	A reusable process fragment is detected after execution	Accepted memory item or rejected write candidate
Stop and answer	Evidence risk is low or budget is exhausted	Final answer, confidence, and residual-risk note

For a candidate action, the conceptual utility summarizes the intended tradeoffs:

$$U(a \mid s_t) = \hat{p}_{\mathrm{prog}}(a \mid s_t) + \lambda_r \widehat{\Delta R}(a \mid s_t) - \lambda_c C(a) - \lambda_l L(a) - \lambda_o O(a).$$

Here the terms express design preferences among progress, evidence improvement, cost, latency, and coordination. The prompt controller does not evaluate the expression numerically.

In the implementation, these preferences are conveyed by the main-agent prompt, which asks the model to analyze missing, questionable, or unreliable information and, when several interpretations remain plausible, to gather disambiguating evidence before deciding, as Appendix E.6 describes. The confidence produced at answer extraction is computed after execution and is not part of the controller state.

B Intra-instance mechanism details

B.1 Recoverable context compression

The reference implementation stores raw UTF-8 text before every lossy tool-output replacement, including aggregate compression of individually sub-threshold outputs. Each anchor retains a SHA-256 hash, identifier, raw-output path, original length, and a prefix preview of 4,000 characters by default. Main and worker sessions maintain separate context states. The exposed `recover_context` tool checks the hash and session access, then returns a query-located or offset-based excerpt of at most 20,000 characters with pagination. `compress_context` refines a visible anchor for the current query while preserving its raw reference.

The configuration uses character thresholds of 100,000 for tool-output compression, 400,000 for current-round summarization, and 600,000 for historical-context summarization. These are character counts, not token counts. These values are the configuration defaults, and Appendix E.7 sweeps each of the numeric controls at half and double its value.

The recovery contract is conditional on valid stored raw outputs and successful retrieval. It does not assume that a summary preserves every decisive detail. Regression tests in the reference implementation check recoverability.

B.2 Self-evolving semantic memory

The memory index embeds task text with normalized `text-embedding-3-large` embeddings, and FAISS returns the squared L_2 distance, which for unit vectors is $d^2 = 2 - 2\cos\theta$. The reported experiments retrieve the top $k = 2$ entries with $d^2 \leq 0.1$, which corresponds to cosine similarity ≥ 0.95. Under this conservative setting only closely matching past tasks are retrieved, and a write gate skips saving when the preceding retrieval already returned k such entries. The reference release exposes this cutoff as a parameter with a default of 0.6.

The released memory bank is empty. In the reference release, ordinary requests start from immutable seed bytes copied into a fresh working file, and collective evaluation additionally isolates each task, attempt, and profile. Nonempty collective seeds require source-task and split metadata, and exact task-ID or question overlaps are rejected before model execution. Memory entries serve as procedural guidance, not as answer evidence.

B.3 Specialized tool suite

The current implementation exposes these specialized tools through MCP-compatible schemas, while the controller treats them as ordinary admissible actions in the same evidence-risk loop. The intelligent image tool improves localized visual evidence extraction by identifying a target region, mapping normalized coordinates to original pixel coordinates, cropping the high-resolution region, and re-running visual reasoning on the localized patch. The street-view tool supports geospatial reasoning by translating high-level navigation goals into API-driven spatial actions, constructing panoramic views, performing coordinate-based movement, and using localized visual refinement to inspect candidate targets. The multi-source search tool suite combines general web search, revision-aware retrieval, focused crawling, retry logic, query sanitization, redundancy filtering, and diagnostic feedback for high-fidelity information grounding.

C Detailed derivations and collaboration example

This section gives conditional analyses for Sections 4 and 5 and an illustrative collaboration example. The statements hold under their stated assumptions, namely valid recovery, calibrated scores, risk-reducing actions, and stationary reuse feedback, and serve as design rationale rather than end-to-end accuracy guarantees.

C.1 Consensus scoring implementation

Scope. The reference implementation in the supplement executes same-task heterogeneous profiles with expertise routing, structured proposals, independent peer critique, six-term consensus, and artifact retention, with offline regression tests for these mechanisms. Appendix D.2 describes its relation to the experiments.

This subsection specifies how the consensus terms in Section 5 are computed from structured proposal and critique records. Let P_i denote the proposal artifact produced by participating instance A_i, and let C_j denote a critique artifact. Lemon reads explicit claim slots and values from proposals, such as entity identity, location, date, numerical value, or boolean status. The candidate set is

$$\mathcal{Z} = \mathrm{Canon}\left(\bigcup_i \mathrm{Claims}(P_i)\right),$$

where Canon normalizes case and whitespace without assuming general semantic equivalence. Distinct values in the same exclusive answer slot remain competing candidates. The complete proposed answer is also retained as an answer-slot candidate requiring its own peer verification, since support for atomic fragments does not automatically verify that answer. Critiques assess existing candidates, while introducing new repair candidates is a further extension. Each $z \in \mathcal{Z}$ stores evidence links, source provenance groups $g(e)$, required verification steps $\mathcal{H}_z$, atomic subclaims $\mathcal{A}_z$, and available cost traces. Support must quote an actually observed tool record, because a generated citation alone does not establish evidence. Peer support and reliability scores are bounded verifier estimates, not calibrated truth probabilities.

The evidence-support term aggregates source reliability and verifier support after deduplicating records with the same provenance group and content:

$$E(z) = \mathrm{clip}_{[0,1]}\left(\sum_{e \in \mathcal{E}_z^+} \rho(e)\,\eta(e,z)\right),$$

where $\rho(e) \in [0,1]$ is the reliability weight of evidence source e and $\eta(e,z) \in [0,1]$ is the structured support score returned by the evidence verifier. Independent corroboration de-duplicates repeated evidence sources:

$$C_{\mathrm{ind}}(z) = \min\left(1, \frac{|\{g(e) : e \in \mathcal{E}_z^+, \eta(e,z) \geq \tau_e\}|}{K_z}\right),$$

where $K_z > 0$ is the target number of independent provenance groups for the claim type. Verification completeness is the fraction of required verification steps that have been satisfied:

$$V(z) = \frac{|\{h \in \mathcal{H}_z : h \text{ is verified}\}|}{\max(1, |\mathcal{H}_z|)}.$$

Contradiction mass combines supported alternatives and evidence-backed peer refutations:

$$D(z) = \min\left(1, \max\left\{R^-(z), \max_{z' \in \mathrm{Conf}(z)} E(z')\,\chi(z,z')\right\}\right),$$

where $\chi(z,z') = 1$ for distinct exclusive values in the same slot and the maximum over an empty conflict set is zero. $R^-(z)$ is the clipped sum of reliability times refutation support over observed peer evidence, deduplicated by provenance group and content. Negative comments without a matching observation contribute zero. Unsupported mass is the weighted fraction of subclaims without accepted evidence:

$$U(z) = 1 - \frac{\sum_{a \in \mathcal{A}_z} \omega_a\,\mathbb{I}[\mathrm{supp}_z(a) \geq \tau_s]}{\max(\epsilon, \sum_{a \in \mathcal{A}_z} \omega_a)}.$$

Here $\omega_a \geq 0$, $\epsilon > 0$, and $\mathrm{supp}_z(a) \in [0,1]$ is the accepted evidence-support score for subclaim a within candidate z. Finally, cost is normalized by the task budget:

$$L(z) = \mathrm{clip}_{[0,1]}\left(\lambda_{\mathrm{tok}}\frac{\mathrm{tokens}(z)}{B_{\mathrm{tok}}} + (1-\lambda_{\mathrm{tok}})\frac{\mathrm{time}(z)}{B_{\mathrm{time}}}\right).$$

Here $\lambda_{\mathrm{tok}} \in [0,1]$ and $B_{\mathrm{tok}}, B_{\mathrm{time}} > 0$. The reference defaults use $\lambda_{\mathrm{tok}} = 0$ and $B_{\mathrm{time}} = 600$ s.

The reference coefficients are $(\alpha_e, \alpha_c, \alpha_v, \alpha_d, \alpha_u, \alpha_l) = (.40, .20, .15, .25, .25, .05)$, with support threshold .5 and two target provenance groups. Accepted candidates require $E \geq .25$, $S \geq 0$, and $R^- < .25$, and unresolved slots or tied exclusive alternatives cause abstention. Visibility weights are .25 each, with clipping to $[0,1]$. Missing historical success and empty seed memory contribute zero. Greedy routing selects up to three of four profiles, leaving a non-proposer available for independent critique. These values are the defaults of the reference implementation.

C.2 Additional formal statements

Design Contract 3 (Risk Drift under Per-Step Reduction Assumptions)**.** *Let R_t be the abstract evidence-risk quantity defined in Appendix A.1, and let $\Delta R(a \mid s_t) = R_t - \mathbb{E}[R_{t+1} \mid s_t, a]$ denote the true expected one-step risk reduction of action a. The conceptual utility also includes progress, cost, latency, and coordination terms, so the following is a conditional risk-control statement. Suppose that at every step there exists an admissible action with $\Delta R(a \mid s_t) \geq \rho R_t - \beta$ for constants $\rho \in (0,1]$ and $\beta \geq 0$. If Lemon's selected action has true risk reduction within ξ of the best admissible risk reduction after the progress and budget filters have defined the admissible set, then after H steps*

$$\mathbb{E}[R_H] \leq (1-\rho)^H R_0 + \frac{\beta + \xi}{\rho}\big(1 - (1-\rho)^H\big).$$

Under these assumptions, the recurrence contracts expected risk to a floor determined by β and ξ. The statement is conditional, and testing whether the prompted controller meets the per-step assumptions is left to future work.

Proposition 2 (Margin-Based Collaboration Activation)**.** *Let $J_{\mathrm{single}}(x)$ be the expected utility of solving task x with one instance and let $J_{\mathrm{coll}}(x, S)$ be the expected utility after invoking the inter-instance protocol over S. Consider a hypothetical collaboration-gain estimator*

$$\widehat{\Delta}_{\mathrm{coll}}(x, S) = \widehat{G}_{\mathrm{cov}} + \widehat{G}_{\mathrm{crit}} + \widehat{G}_{\mathrm{reuse}} - \widehat{C}_{\mathrm{comm}}.$$

If $|\widehat{\Delta}_{\mathrm{coll}}(x, S) - (J_{\mathrm{coll}}(x, S) - J_{\mathrm{single}}(x))| \leq \epsilon_{\mathrm{coll}}$, then activating collaboration whenever $\widehat{\Delta}_{\mathrm{coll}}(x, S) > \epsilon_{\mathrm{coll}}$ guarantees positive true expected utility gain. Conversely, suppressing collaboration whenever $\widehat{\Delta}_{\mathrm{coll}}(x, S) < -\epsilon_{\mathrm{coll}}$ avoids a negative true expected utility gain.

The proposition motivates a margin-based activation policy, whereas the released runner selects the protocol explicitly.

Lemma 1 (Critique Contracts Risk to a Noise Floor)**.** *Let $R_t^{\mathrm{crit}} = \alpha_u U_t + \alpha_d D_t$ denote the unsupported and contradictory risk mass after critique round t. Suppose each critique round satisfies*

$$\mathbb{E}[R_{t+1}^{\mathrm{crit}} \mid R_t^{\mathrm{crit}}] \leq (1 - \rho) R_t^{\mathrm{crit}} + \beta,$$

where $\rho \in (0, 1]$ is the expected critique effectiveness and $\beta \geq 0$ captures residual noise from missed contradictions, ambiguous evidence, or false critique alarms. Then after K critique rounds,

$$\mathbb{E}[R_K^{\mathrm{crit}}] \leq (1 - \rho)^K R_0^{\mathrm{crit}} + \frac{\beta}{\rho}\big(1 - (1 - \rho)^K\big).$$

Thus critique need not be perfect to be useful: it contracts unsupported and contradictory mass until the residual noise floor β/ρ dominates.

Proposition 3 (Expected Artifact Utility under Stationary Feedback)**.** *Let $u_{i,k}^{(t)}(a)$ denote the utility estimate of artifact a for instance i on task family k after round t, and let*

$$u_{i,k}^{(t+1)}(a) = (1 - \lambda) u_{i,k}^{(t)}(a) + \eta \hat{r}_{i,k}^{(t)}(a).$$

Assume $\lambda \in (0, 1]$, $0 \leq \eta \leq \lambda$, and the reuse feedback has conditional mean $\mu_{i,k}(a) \in [0, 1]$. If this mean is stationary over a related task family, then

$$\mathbb{E}[u_{i,k}^{(t)}(a)] = (1 - \lambda)^t u_{i,k}^{(0)}(a) + \frac{\eta \mu_{i,k}(a)}{\lambda}\big(1 - (1 - \lambda)^t\big),$$

and therefore $\mathbb{E}[u_{i,k}^{(t)}(a)]$ converges to $\eta \mu_{i,k}(a)/\lambda$. This is convergence in expectation to a scaled value. It does not imply convergence of a noisy sample path, or convergence to $\mu_{i,k}(a)$ unless $\eta = \lambda$.

C.3 Proof of Design Contract 1

For a forward-only summarization pipeline, let $\mathcal{S}_k$ denote the event that d remains recoverable after the k-th compression. The assumption gives $\Pr(\mathcal{S}_k \mid \mathcal{S}_{1:k-1}) \leq p_s$ for each k. By the chain rule,

$$\Pr(\mathcal{S}_{1:K}) = \prod_{k=1}^{K} \Pr(\mathcal{S}_k \mid \mathcal{S}_{1:k-1}) \leq p_s^K.$$

For Lemon, let $\mathcal{A}$ denote the event that a valid truncation anchor is created and linked to the raw tool output, and let $\mathcal{R}$ denote the event that retrospective refinement recovers d from that valid anchor. By assumption, $\Pr(\mathcal{A}) \geq 1 - \epsilon_a$ and $\Pr(\mathcal{R} \mid \mathcal{A}) \geq 1 - \epsilon_r$. Therefore

$$\Pr(d \text{ is recoverable}) \geq \Pr(\mathcal{A} \cap \mathcal{R}) = \Pr(\mathcal{A}) \Pr(\mathcal{R} \mid \mathcal{A}) \geq (1 - \epsilon_a)(1 - \epsilon_r).$$

This bound depends on anchor validity and retrospective recovery, not on the number of intermediate compression events K. The bound holds uniformly in K only when raw-output accessibility and both probability bounds do so.

C.4 Proof of Design Contract 3

Let a_t be the action selected by Lemon. Since a stabilizing admissible action exists, $\max_a \Delta R(a \mid s_t) \geq \rho R_t - \beta$. The ξ-optimality assumption gives

$$\Delta R(a_t \mid s_t) \geq \rho R_t - \beta - \xi.$$

By definition of ΔR,

$$\mathbb{E}[R_{t+1} \mid s_t, a_t] = R_t - \Delta R(a_t \mid s_t) \leq (1-\rho)R_t + \beta + \xi.$$

Taking total expectation and unrolling the recurrence for H steps yields

$$\mathbb{E}[R_H] \leq (1-\rho)^H R_0 + (\beta+\xi)\sum_{\tau=0}^{H-1}(1-\rho)^\tau = (1-\rho)^H R_0 + \frac{\beta+\xi}{\rho}\big(1-(1-\rho)^H\big).$$

C.5 Proof of Proposition 1

For each domain k, define $f_k(S) = \max_{A_i \in S} v_{i,k}$. If $S \subseteq T$, then $f_k(S) \leq f_k(T)$, so each f_k is monotone. For submodularity, consider adding an instance A_j to $S \subseteq T$. The marginal gain for domain k is

$$f_k(S \cup \{A_j\}) - f_k(S) = \max\{0, v_{j,k} - f_k(S)\}.$$

Since $f_k(T) \geq f_k(S)$, we have

$$\max\{0, v_{j,k} - f_k(S)\} \geq \max\{0, v_{j,k} - f_k(T)\},$$

which is the diminishing-returns property. A nonnegative weighted sum of monotone submodular functions is also monotone submodular, so $F(S,x) = \sum_k \omega_k f_k(S)$ is monotone submodular.

The following is the standard cardinality-constrained greedy argument [18]. Let S_t be the greedy set after t selections and let $S^\star$ be an optimal set of size b. By submodularity, at least one element in $S^\star \setminus S_t$ has marginal gain at least $(F(S^\star, x) - F(S_t, x))/b$. Greedy selects an element with no smaller marginal gain, so

$$F(S_{t+1}, x) - F(S_t, x) \geq \frac{F(S^\star, x) - F(S_t, x)}{b}.$$

Solving this recurrence gives

$$F(S_b, x) \geq \left(1 - \left(1 - \frac{1}{b}\right)^b\right) F(S^\star, x) \geq (1 - 1/e) F(S^\star, x).$$

C.6 Proof of Proposition 2

Let $\Delta^\star_{\text{coll}}(x, S) = J_{\text{coll}}(x, S) - J_{\text{single}}(x)$ denote the true expected collaboration gain. If $\widehat{\Delta}_{\text{coll}}(x, S) > \epsilon_{\text{coll}}$ and $|\widehat{\Delta}_{\text{coll}}(x, S) - \Delta^\star_{\text{coll}}(x, S)| \leq \epsilon_{\text{coll}}$, then

$$\Delta^\star_{\text{coll}}(x, S) \geq \widehat{\Delta}_{\text{coll}}(x, S) - \epsilon_{\text{coll}} > 0.$$

Thus collaboration has positive true expected utility gain. The suppression statement follows similarly: if $\widehat{\Delta}_{\text{coll}}(x, S) < -\epsilon_{\text{coll}}$, then $\Delta^\star_{\text{coll}}(x, S) \leq \widehat{\Delta}_{\text{coll}}(x, S) + \epsilon_{\text{coll}} < 0$, so invoking collaboration would have negative true expected utility.

C.7 Proof of Lemma 1

The assumed drift inequality gives

$$\mathbb{E}[R^{\text{crit}}_{t+1} \mid R^{\text{crit}}_t] \leq (1-\rho)R^{\text{crit}}_t + \beta.$$

Taking expectation on both sides yields

$$\mathbb{E}[R^{\text{crit}}_{t+1}] \leq (1-\rho)\mathbb{E}[R^{\text{crit}}_t] + \beta.$$

Unrolling the recurrence for K rounds gives

$$\mathbb{E}[R^{\text{crit}}_K] \leq (1-\rho)^K R^{\text{crit}}_0 + \beta\sum_{\tau=0}^{K-1}(1-\rho)^\tau = (1-\rho)^K R^{\text{crit}}_0 + \frac{\beta}{\rho}\big(1-(1-\rho)^K\big).$$

C.8 Proof of Design Contract 2

Because $\widehat{\mathcal{J}}(z) = B_0 - S(z)$ and B_0 is constant over the candidate set, maximizing $S(z)$ is equivalent to minimizing $\widehat{\mathcal{J}}(z)$. Let $z^\star = \arg\min_z \mathcal{J}^\star(z)$ and let $\hat{z} = \arg\min_z \widehat{\mathcal{J}}(z)$. Uniform calibration gives

$$\mathcal{J}^\star(\hat{z}) \le \widehat{\mathcal{J}}(\hat{z}) + \epsilon \le \widehat{\mathcal{J}}(z^\star) + \epsilon \le \mathcal{J}^\star(z^\star) + 2\epsilon.$$

Since $z^\star$ is the true minimum-loss claim under the stated decision criteria, this proves the stated near-optimality bound.

C.9 Proof of Proposition 3

Taking conditional expectation in the update rule gives

$$\mathbb{E}[u_{i,k}^{(t+1)}(a)] = (1-\lambda)\mathbb{E}[u_{i,k}^{(t)}(a)] + \eta\mu_{i,k}(a).$$

Unrolling the recurrence yields

$$\mathbb{E}[u_{i,k}^{(t)}(a)] = (1-\lambda)^t u_{i,k}^{(0)}(a) + \eta\mu_{i,k}(a)\sum_{\tau=0}^{t-1}(1-\lambda)^\tau.$$

The geometric sum equals $\big(1-(1-\lambda)^t\big)/\lambda$, which gives the stated expression. Since $\lambda \in (0,1]$, $(1-\lambda)^t$ converges to zero, and the expectation converges to $\eta\mu_{i,k}(a)/\lambda$. If this limit is above a retrieval threshold, the expected utility eventually exceeds that threshold for sufficiently large t.

C.10 GAIA-style collaboration example

Consider a representative GAIA-style geospatial question that asks an agent to identify a business from a low-resolution street-view clue and verify a time-sensitive fact about it. A single agent may need to locate visual evidence, disambiguate similar business names, search web sources, and verify that the final answer refers to the same entity.

In Lemon, the task is broadcast to several personalized instances. In this constructed example, one instance has street-view navigation tools and proposes a candidate location based on road signs and storefront geometry. A second instance has web retrieval tools and retrieves official pages, map records, and recent web snippets. A third instance has experience with image-refinement tools and crops the storefront or signboard to extract fine-grained text. Each instance returns a proposal artifact containing its candidate answer, evidence, confidence, uncertainty notes, and requested checks.

During critique, the retrieval-oriented instance may flag that the candidate business name appears in multiple branches. The image-oriented instance may request a second crop because the storefront text is ambiguous. The street-view instance may revise the candidate location after comparing road geometry and sign orientation. These critiques expose weak or conflicting fragments early.

Consensus is then formed at the claim level. A location claim supported by street-view geometry, cropped visual text, and an independent official source receives higher evidence and independent corroboration scores. A claim that relies on a single search result but conflicts with visual evidence receives a higher contradiction penalty. The final answer uses the better-supported claim set.

After the task, Lemon can extract reusable artifacts. Examples include a decomposition plan for street-view localization followed by official-source verification, an evidence pattern for distinguishing same-name business branches, and a repair trace that triggers additional image cropping when map records and visual text disagree. These artifacts can be written into the participating instances' semantic memories and reused on later geospatial tasks.

D Detailed inter-instance evaluation

The reported inter-instance evaluation uses a fixed 165-task collaboration stress set selected before running the four comparison protocols. The set is built from GAIA-style tasks whose text or attachments satisfy at least one deterministic rule in Table 10. The categories can overlap, so the final

Table 10: Task selection rules for the inter-instance collaboration stress set.

Criterion	Operational rule	#Tasks	Expected collaboration pressure
Multi-source synthesis	The answer requires reconciling at least two independent web, document, or database evidence sources.	96	Source triangulation and duplicate-source control
Entity disambiguation	The task contains ambiguous people, places, organizations, products, or events that require entity-level verification.	61	Competing hypotheses and critique
Visual/geospatial verification	The task includes image, map, street-view, or location evidence that must be linked to textual sources.	54	Complementary tool expertise
Union after de-duplication	A task is included if any rule above is satisfied.	165	Fixed paired evaluation set

size is the union after de-duplication rather than the sum of category counts. Selection does not use Lemon outcomes.

The multi-instance configurations use the same four fixed Lemon profiles in Table 11. The isolated baseline uses only the general retrieval profile. The profiles are defined before evaluation and are not selected per task. They differ in exposed tools, memory seeds, and system-level preference prompts, while sharing the same backbone, evaluator, stopping limits, and answer format.

Table 11: Fixed instance profiles used in the inter-instance evaluation.

Profile	Tool exposure	Memory seed	Preference prompt
General retrieval	Web retrieval, crawling, multi-source search	Search and source-triangulation traces	Prefer independent corroboration
Visual/geospatial	Image refinement, street-view, web retrieval	Visual repair and location-disambiguation traces	Prefer spatial and visual checks
File/data reasoning	File parsing, code execution, web retrieval	Table, document, and attachment-processing traces	Prefer structured extraction
Critique verifier	Retrieval, verifier, contradiction tracking	Unsupported-claim and contradiction templates	Prefer claim-level critique

The reported accuracy is micro accuracy over the same 165 paired tasks, not a macro average over categories. Exact McNemar tests compare each multi-instance protocol with the isolated baseline using task-level discordant pairs. Direct communication has 10 isolated-wrong/direct-correct and 4 isolated-correct/direct-wrong tasks, fixed-role organization has 13 and 3, and full inter-instance Lemon has 20 and 3. These paired counts yield the p values in Table 7.

The multi-instance configurations were run with four profiles whose memory seeds are listed in Table 11, while the reference implementation ships the same four profiles with empty memory seeds.

D.1 Stratification, profile controls, and cost

Stratification. We partition the 165 paired outcomes by the number of selection rules a task satisfies, as shown in Tables 10 and 12. The one-rule stratum holds 79% of the set and still gains 7.6 points, so the gain is not confined to the most demanding tasks.

Profile controls. Two further configurations keep the collaboration protocol and change the profile pool: four homogeneous instances, and four alternative heterogeneous profiles defined independently of Table 11. On the same 165 tasks they reach 85.5% and 89.7%, against 80.6% for the isolated reference and 90.9% for the original profiles, as listed in Table 13. The homogeneous pool therefore retains $+4.9$ of the $+10.3$ points and the alternative heterogeneous pool retains $+9.1$, or 88% of the

Table 12: Stratification of the 165 paired tasks by the number of selection rules satisfied. Solved counts are given with accuracy in percent, and Δ is in percentage points.

Rules satisfied	Tasks	Isolated	Full inter-instance	Δ
3	12	8 (66.7)	11 (91.7)	+25.0
2	22	16 (72.7)	20 (90.9)	+18.2
1	131	109 (83.2)	119 (90.8)	+7.6
All	165	133 (80.6)	150 (90.9)	+10.3

gain, so the gain is not tied to the original profile set, and both the protocol and the heterogeneity contribute. Because the comparisons are nested and the number of instances also changes total resources, the two contributions are not additive.

Table 13: Profile controls on the 165 paired tasks. Accuracy is percent.

Configuration	Accuracy (%)
Isolated Lemon	80.6
Four homogeneous instances	85.5
Four alternative heterogeneous profiles	89.7
Four original heterogeneous profiles	90.9

Cost and peer retrieval. On the same tasks, isolated Lemon uses 0.26 MTok per task and 0.32 per correct answer, and the four-instance configuration uses 0.98 and 1.08, about $3.4\times$ the tokens per correct answer, as Table 14 shows. Peer retrieval uses `top_k`$= 5$ and `max_claims_per_run`$= 5$. Roughly halving `top_k` gives 89.7%, and doubling it leaves 90.9% unchanged.

Table 14: Collaboration cost on the 165 paired tasks, from runs separate from those of Table 2.

Configuration	Accuracy (%)	MTok/task	MTok/correct
Isolated Lemon	80.6	0.26	0.32
Four-instance Lemon	90.9	0.98	1.08

D.2 Reference implementation

The supplementary material contains a reference implementation of the four protocols, namely isolated, direct communication, fixed-role, and full collective, with explicit mode switches, per-task isolation of memory and blackboard, and offline regression tests with mocked services. The results in Table 7 were obtained with the research prototype used for the experiments, and reproducing them with the reference implementation is left to future work.

E Detailed single-instance evaluation

E.1 Models and execution settings

The controller and workers use GPT-5 at high reasoning effort, with round caps of 10 and 20. Hosted calls use temperature 1.0 and a maximum output of 128,000 tokens where supported. No model is trained or fine-tuned. Table 15 lists all model roles of the implementation. In the reference release, adaptive delegated execution with concurrency at most three is the default, and serial and fixed-two presets are explicit alternatives.

The orchestration host runs Ubuntu 24.04.2 LTS with Linux 6.8.0-53-generic, two Intel Xeon Gold 5118 CPUs (24 physical cores, 48 threads), 503 GiB RAM, and 8 GiB swap. It is used for orchestration, tool execution, and logging, while inference uses hosted services. Reported wall-clock times include local orchestration, tool calls, hosted inference latency, delegated workers, and

Table 15: Model roles in the implementation, given as provider model identifiers.

Role	Model identifier / configuration
Controller and workers	`gpt-5`, high reasoning effort
Final answer extraction and reasoning tool	`o3`, high reasoning effort
Image, street view, and video	`gemini-3-pro-preview`
Audio	`gpt-4o-transcribe`; `gpt-4o-audio-preview` for audio reasoning
Memory embeddings	`text-embedding-3-large`
Agent summarization	`gpt-5`, medium reasoning effort
Tool/memory summaries	`gpt-4o` in the experiments; configurable in the reference release (default GPT-5, medium)

summarization, and reported token counts aggregate all LLM calls made by the controller, delegated workers, and summarization module.

For task-level uncertainty we treat each task as a Bernoulli outcome and report 95% Wilson intervals: 275/301 gives 91.36% with an interval of 87.64–94.04%, and Levels 1–3 with 90/93, 142/159, and 43/49 give 90.94–98.90%, 83.54–93.22%, and 75.76–94.27%. These intervals describe task-level binomial uncertainty and do not measure run-to-run variation. Appendix E.4 reports repeated runs of the resource comparison.

E.2 Ablation setup

For the component ablations of Table 3, we use matched stress slices rather than the full GAIA set: context-pressure tasks for context recovery, procedural-reuse tasks for SES-Memory, fine-grained visual tasks for the image tool, and explicit Street-View tasks for the street-view tool. Slices are selected from task text and attachment metadata so that the evaluated tasks exercise the removed mechanism. The no-context-recovery variant disables truncation anchors, cross-round retrospective refinement, and raw-evidence recovery. The no-memory variant disables SES-Memory retrieval and writing. The tool variants disable the corresponding image or street-view tool while preserving the general tool interface. Accuracy and token use are averaged only over the matched slice, so token use can increase from compensatory search or decrease when a costly specialized tool is unavailable. The search ablation of Table 4 compares a single delegated search worker, two workers launched for every task, and adaptive allocation in which the controller adjusts the number and type of workers according to uncertainty, evidence gaps, and remaining budget. The full-benchmark removals in Table 5 and numerical sweeps in Appendix E.7 apply the same variants to the full benchmark task list.

E.3 Existing assets, licenses, and terms of use

GAIA is credited to its original authors [19]. Its official dataset card restricts resharing outside gated/private access and crawlable redistribution [20]. Appendix F includes qualitative case-study figures; access to the underlying benchmark dataset remains separate.

xbench-DeepSearch [23] and its evaluator list MIT licensing [28, 29]. We cite aggregate results and do not redistribute task data. MiroFlow and Co-Sight are Apache-2.0 projects, and Skywork DeepResearchAgent is MIT licensed [25, 27, 30]. CAMEL-AI is also credited for tool adaptations under Apache-2.0 [31]. The supplement retains upstream copyright headers and includes third-party notices and license texts, including MiroMind AI SPDX notices in several utility modules.

The supplement provides a reference implementation that records source hashes in its run summaries and offers nine runtime presets matching the ablation arms, namely full, no_context_recovery, no_memory, no_image, no_street_view, no_all_four, single, fixed_2, and adaptive. It is a re-implementation rather than the exact code used for Section 6.

E.4 Repeated runs and evaluation scope

We run each system three times in an identical configuration on the same task list and harness, with one answer per task. No random seed is exposed, so these are repeated runs and not controlled seeds. Table 16 gives accuracies with solved counts out of 301. Lemon has the highest accuracy in every run, and the ordering of the four systems is the same in all three. Lemon's Run 2 is the leaderboard submission of Section 6.1. Cross-run SD reflects execution variability, whereas Wilson intervals reflect the finite task set.

Table 16: Repeated-run accuracies in percent, with solved counts out of 301 in parentheses. SD is the sample standard deviation across the three runs.

System	Run 1	Run 2	Run 3	Mean $\pm$ SD
MiroFlow	79.07 (238)	78.41 (236)	79.73 (240)	79.07 ± 0.66
Co-Sight	87.04 (262)	87.38 (263)	86.38 (260)	86.93 ± 0.51
Skywork	83.39 (251)	82.72 (249)	84.39 (254)	83.50 ± 0.84
Lemon	91.69 (276)	91.36 (275)	90.70 (273)	91.25 ± 0.51

Table 17: Per-task resource use of the repeated-run comparison. /Correct divides tokens per task by the mean accuracy.

System	MTok/task	MTok/correct	Time/task (min)
MiroFlow	2.19	2.77	45.1
Co-Sight	1.17	1.35	12.0
Skywork	0.51	0.61	25.4
Lemon	0.26	0.28	17.8

As Table 17 shows, Lemon has the lowest token use of the four systems, 0.26 MTok per task against 0.51 to 2.19 for the baselines, and Co-Sight the lowest wall-clock time. These figures agree with the single comparison of Table 2 and Figure 1b to within 0.02 MTok and 0.4 minutes per task.

E.5 Baselines and public sources

The resource comparison uses the GPT-5 controller/worker backbone and decoding settings of Appendix E.1, with one harness, one GAIA task list, and one attempt per task. Each baseline retains its released defaults, so tools, utility models, and total budgets differ. Undisclosed settings are not inferred. The accuracy column of Table 2 uses each team's public leaderboard result. Table 18 audits the three comparison baselines and other published figures, which differ in split, aggregation, and models; Lemon's submission is reported in Table 1(a).

Table 18: Public accuracy figures for the compared baselines and their scope. Entries differ in split and protocol.

System / source	Acc. (%)	Scope and registered models
MiroFlow, historical README	82.4	GAIA validation, majority vote, Claude Sonnet 3.7 [32]
MiroFlow (Reproducible!)	79.07	Test leaderboard, 238/301, 2025-09-16: GPT-5 (agent), o3, Gemini-2.5-Pro, Claude [22]
Co-Sight 2.0, report	84.4	Technical report; GAIA test [26]
Co-Sight_v2.1.0	87.04	Test leaderboard, 262/301, 2025-10-13: ZTE Nebula LLM, Claude Sonnet 4, Gemini 2.5 Pro [22]
Skywork, May 2025	82.42	Company announcement [33]; GAIA validation per README [34]
Skywork Deep Research Agent v2	83.39	Test leaderboard, 251/301, 2025-08-13: skywork-agent-model, gpt-4.1, claude-3.7-sonnet, gemini-2.5-pro [22]

Exact McNemar tests on the paired per-task outcomes of the comparison runs give $p = .024$ against Co-Sight and $p < .001$ against MiroFlow and Skywork. Table 19 lists the registered models of the leaderboard entries in Table 1(a), as self-reported by the submitters.

Table 19: Registered models of the GAIA entries in Table 1(a), as listed by the submitters [22]. Entries: MiroFlow (Reproducible!) and the other baselines are dated in Table 18.

Entry	Registered models
Lemon	GPT5, Gemini-3-pro, o3
JoinAI_V2.2	GPT 5, Gemini 3 Pro, DeepSeek 3.1, Qwen 3
Nemotron-ToolOrchestra-0106 / 0107	Nemotron-ToolOrchestrator-8B, GPT-5, Claude Opus 4.1, Qwen2.5-Math-72B
JoinAI_V2.1	GPT, Gemini, DeepSeek, Qwen
SU Zero–Shuqian Series Pro MAX	Self Consistency 35
HALO V1217-1	none registered
Co-Sight_v2.1.0	ZTE Nebula LLM, Claude Sonnet 4, Gemini 2.5 Pro
MiroFlow (Reproducible!)	GPT-5 (agent), o3, Gemini-2.5-Pro, Claude
Skywork Deep Research Agent v2	skywork-agent-model, gpt-4.1, claude-3.7-sonnet, gemini-2.5-pro

E.6 Controller prompt interventions

This experiment tests whether two evidence-oriented instructions in the controller prompt affect performance when tools and execution limits are fixed. The first asks the model to "analyze missing, questionable, or unreliable information." The second says, "If multiple interpretations remain plausible, gather disambiguating evidence before deciding." Both occur in the serial and parallel prompt variants; each intervention changes both occurrences.

A: original prompt.
Keep both instructions.

A′: placebo edit.
Rewrite an unrelated sentence of comparable length with equivalent content, to check sensitivity to an incidental prompt edit.

B: disable the disambiguation instruction.
Keep the first instruction, but replace the second with "If multiple interpretations remain plausible, proceed with the most likely one." The prompt still requests an evidence-gap description but no longer explicitly requires disambiguation before deciding.

C: remove both instructions.
Delete the evidence-analysis clause and the disambiguation sentence.

Backbone, tools, worker availability, round caps, evaluator, and answer extraction are identical across arms, and each arm is a single run on the full benchmark, with results in Table 6. The supplementary material provides the A-to-B and A-to-C diffs, reconstructed from this specification against the released prompt file, but not the exact placebo sentence. The interventions do not remove latent uncertainty reasoning or every evidence-related instruction.

The placebo edit changes accuracy by one task with $p = 1.00$, whereas B and C lower it by 4.32 and 6.64 points, and token use rises from 0.25 to 0.31 and 0.38 MTok per task as the evidence clauses are weakened. Of the 19 tasks that A solves and B does not, B's trajectory diverged in four ways: answering while a flagged contradiction was still open in 8 tasks, answering from a single unverified source when a second was available in 5, compressing away evidence that was later needed without triggering recovery in 4, and repeating a completed search instead of verifying an existing claim in 2. None of these was a tool failure, a parsing failure, or budget exhaustion. Repeating the A–B contrast on a second frontier backbone at high reasoning effort, with prompts, tools, caps, and evaluator otherwise unchanged, gives 88.04% versus 84.39% with $p = .035$ on this single alternative backbone.

E.7 Numerical controls and round-cap sensitivity

We halve or double one numerical control at a time and hold the other settings fixed. Table 20 gives accuracy with solved counts against the 91.36% reference.

The largest decrease is 1.33 points, or 4 tasks, for a 2,000-character preview.

Table 21 varies the round caps. Accuracy changes little beyond the default caps: doubling the controller cap adds one task, or 0.33 points, and doubling the worker cap leaves it unchanged.

Table 20: One-factor-at-a-time sweeps of the numerical controls, with accuracy in percent and solved counts out of 301 in parentheses. The first four controls count characters. The memory cutoff is a distance threshold, where smaller distance indicates greater similarity.

Control	Original	Half: value / Acc.	Double: value / Acc.
Tool compression (characters)	100,000	50,000 / 90.70 (273)	200,000 / 91.03 (274)
Current summary (characters)	400,000	200,000 / 91.03 (274)	800,000 / 90.70 (273)
History summary (characters)	600,000	300,000 / 91.03 (274)	1,200,000 / 91.36 (275)
Anchor preview (characters)	4,000	2,000 / 90.03 (271)	8,000 / 91.03 (274)
Memory top-k	2	1 / 91.03 (274)	4 / 90.70 (273)
Memory distance cutoff	0.1	0.05 / 91.03 (274)	0.2 / 90.70 (273)

Table 21: Round-cap sweeps, changing one cap at a time, with accuracy in percent and solved counts out of 301 in parentheses. The original controller and worker caps are 10 and 20.

Cap varied	Half: cap / Acc.	Original: cap / Acc.	Double: cap / Acc.
Controller	5 / 88.04 (265)	10 / 91.36 (275)	20 / 91.69 (276)
Worker	10 / 90.03 (271)	20 / 91.36 (275)	40 / 91.36 (275)

E.8 xbench-DeepSearch evaluation

We evaluated Lemon on the 100 questions of DeepSearch-2510 on April 16, 2026 with the official `xbench-evals` harness [29], using a single attempt without retries and the same controller and prompts as for GAIA, and it solved 80 questions. Public systems were not rerun under this protocol.

F Case study details

These qualitative cases explain procedural reuse and specialized evidence extraction; they do not add benchmark measurements. The repeated attempts in the memory example illustrate how procedural fragments are combined and do not specify the single-attempt protocol of the reported benchmark comparison.

F.1 SES-Memory case study

The SES-Memory case illustrates how Lemon converts execution traces into reusable procedural knowledge. The task challenge is that a successful trajectory may depend on intermediate fragments discovered across several attempts rather than on a single complete solution. A memory system that stores only the final successful trace can miss useful extraction strategies, tool-selection heuristics, and repair patterns found in earlier failed or partially successful attempts.

Figure 4 illustrates this process with an audio-identification task. During the first attempt, the agent used `ffmpeg` to extract music clips and a Shazam-based method to identify them. During the second attempt, it tried spectral-flux extraction and ASR. During the final successful attempt, the agent reused transferable pieces from both previous attempts by combining spectral flux for robust extraction with Shazam-based matching for identification. The evidence improvement comes from reusing process-level fragments rather than replaying an entire previous trajectory.

Operationally, stored artifacts include compact procedural templates such as extraction and recognition pipelines, tool-selection heuristics conditioned on input structure, and failure-repair notes such as when to switch extraction strategy before recognition. These artifacts improve reuse efficiency while avoiding full-trace replay.

F.2 Intelligent image tool case study

The intelligent image tool case illustrates why a specialized tool interface can improve fine-grained visual evidence extraction. The task challenge is a GAIA-style clock-reading question in which the target object occupies a small region of the image. Direct full-image inference can fail because global resizing removes details such as clock hands or small text.

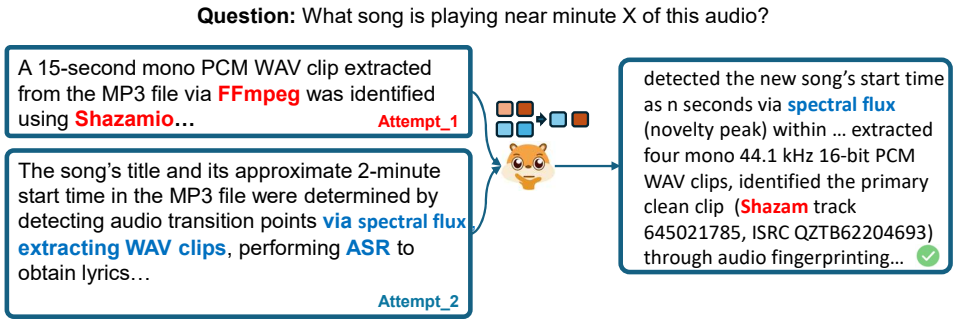


Figure 4: SES-Memory case study. The agent extracts reusable procedural fragments from previous attempts and recombines them in a later successful execution.

Lemon first predicts a localized region, maps normalized coordinates to original pixel coordinates, crops the high-resolution region, and re-runs visual reasoning on the cropped patch. Figure 5 illustrates this localized refinement process. This action turns a global image-understanding problem into a targeted evidence-extraction step. The evidence improvement is that the downstream reasoning step receives a higher-resolution view of the relevant object rather than a lossy global representation.

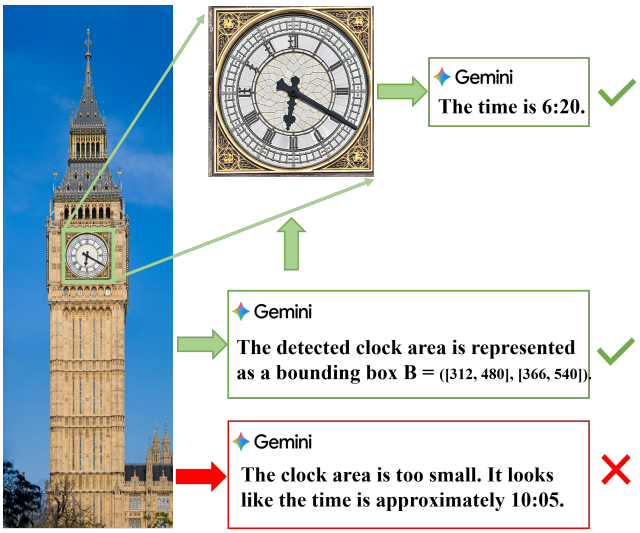


Figure 5: Intelligent image tool case study. Localized cropping provides a higher-resolution view of the relevant region before visual reasoning is repeated.

This case supports the role of specialized tool interfaces in Lemon's intra-instance organization. The controller does not need to solve the visual limitation directly. Instead, it invokes a tool that exposes structured inputs, localized outputs, and a refined evidence state for downstream reasoning.

F.3 Street-view tool case study

The street-view tool case illustrates how Lemon separates high-level geospatial reasoning from low-level navigation and visual refinement. The task challenge is to extract information from a signboard in a street-view environment. Browser-based navigation can require many interaction rounds and can return redundant or poorly aligned views.

Lemon's street-view tool first retrieves a panoramic scene, narrows candidate regions, and estimates movement direction from localized coordinates. It then repositions to a clearer viewpoint before invoking localized visual extraction on the target sign. Figure 6 shows the corresponding geospatial refinement workflow. Evidence improves through panoramic perception, coordinate-based movement, and target-centric visual refinement before final reasoning.



Figure 6: Street-view tool case study. The tool combines panoramic perception, coordinate-based movement, and localized visual refinement to obtain high-quality spatial evidence.

This case demonstrates why specialized capabilities are organized as tools rather than as an unstructured action list. The controller receives structured spatial evidence while the tool absorbs repeated navigation, viewpoint selection, and local visual extraction.

G Extended related work and comparisons

G.1 Tool-augmented LLM agents and workflow control

Tool-augmented LLM agents provide the immediate technical context for Lemon. Reasoning-and-acting methods connect intermediate reasoning with external actions [1], while self-supervised tool-use and API-oriented systems broaden the set of callable operations available to a model [2, 3, 11]. Multimodal tool systems and plug-and-play module frameworks further show that model capability can be expanded by routing requests to specialized visual, retrieval, or computational modules [35, 36]. Chain-of-thought, self-consistency, program-aided, and tree-search methods show that intermediate reasoning, aggregation over sampled paths, external computation, or search can improve complex reasoning beyond direct generation [37–40]. These systems establish the value of external actions, but tool selection, context management, worker expansion, and verification are often handled as separate prompt or workflow decisions.

Lemon supplies the evolving transcript, tool descriptions, and retrieved memory to a prompt-guided controller. Its evidence-directed instructions can influence reasoning, tool calls, delegation, and verification, while separate programmatic guards trigger context compression. A newly retrieved source may close an evidence gap, expose a contradiction, or lead the model to seek additional verification. This interaction among execution choices motivates the organization studied here.

G.2 Agent benchmarks and long-horizon evaluation

Agent benchmarks increasingly evaluate whether LLM systems can operate under realistic tool, interaction, and environment constraints. AgentBench and WebArena study multi-environment and web-control abilities [41, 42]. τ-bench emphasizes tool-agent-user interaction in realistic domains [43]. AndroidWorld evaluates dynamic computer-control behavior [44]. GAIA evaluates retrieval,

multimodal inspection, file processing, and long-horizon synthesis [19]. Scientific and machine-learning workflow benchmarks add another form of long-horizon pressure by requiring agents to plan, execute, inspect, and repair experimental procedures [45, 46]. These benchmarks make clear that agent performance depends not only on the base model but also on how the system allocates computation, preserves evidence, and handles uncertainty across many steps.

Lemon is designed for this long-horizon evaluation regime. Its prompts ask the model to assess evidence gaps, unsupported claims, and contradictions qualitatively, alongside recorded context pressure, retrieved memory, available tools, and execution limits. This makes the evaluation question more specific than whether a tool-using agent can solve a benchmark: the paper asks whether evidence-aware control allocates search, verification, context recovery, and tool use more selectively than fixed workflows.

G.3 Context management, memory, and tool reuse

Long-horizon execution also depends on preserving evidence and reusing experience across turns or tasks. Retrieval-augmented and retrieval-enhanced models show how external evidence can support generation [47, 48]. Agent memory systems and reflective agents extend this idea from document retrieval to interactive trajectories, storing experiences, reflections, or operating-system-like memory states that can influence later behavior [5, 14, 49]. Long-term memory benchmarks highlight that memory quality must be evaluated under realistic multi-session use, where irrelevant memories and stale assumptions can be harmful as well as helpful [6].

Embodied agents and procedural libraries provide a complementary view. Rather than only storing facts, an agent can store reusable procedures. Voyager, for example, shows how executable procedures can accumulate and improve later task solving in an embodied setting [50]. Lemon adopts the procedural-reuse motivation, but integrates it with evidence-risk control and tool selection. SES-Memory stores decomposition templates, evidence patterns, tool-selection heuristics, and repair traces, while the controller decides when retrieved memory should alter routing, tool selection, or verification. This differs from memory-only approaches because memory retrieval is not the final intervention. It is an input to a broader control state.

Context management is another bottleneck that becomes visible only in long trajectories. Repeated summarization can gradually remove decisive evidence even when the agent observed it earlier. Lemon addresses this with recoverable truncation anchors and retrospective refinement over raw tool outputs. The distinction is not simply shorter context. It is recoverable context. A compressed state can still point back to the raw evidence needed for later verification, which lets context management participate in evidence preservation rather than only token reduction.

G.4 Workflow search and agent orchestration

Automated workflow generation and modular agent search study how reusable agent structures can be generated, searched, or recombined [4, 12]. These methods are important because fixed prompt templates and hand-designed pipelines often underfit task diversity. Recent orchestration systems also study how models and tools can be organized into hierarchical or efficient execution structures [13]. Such work is closest to Lemon's intra-instance contribution because it treats agent performance as a property of the surrounding workflow, not only the base model.

The difference is that Lemon does not primarily search for a global workflow before execution. Instead, it performs local, state-conditioned organization during execution. The same task can stay single-trajectory, spawn workers, invoke specialized tools, compress context, retrieve memory, or verify claims depending on the current evidence state and budget. This online control view is useful when evidence changes the problem structure: a search result may make a task easy, reveal a missing modality, or expose a contradiction that requires a different route. Lemon therefore complements workflow search by studying online adaptation.

Our focus is the combination of evidence-directed organization across action families with recoverable context, and Table 22 summarizes the closest perspectives.

Table 22: Closest organizational perspectives. Each row summarizes the work's primary focus.

Work	Main decision studied	Relation to this paper's focus
ReAct / Toolformer / ToolLLM [1–3]	Reasoning, tool-use decisions, and tool learning	Establish tool-using agent primitives; Lemon studies their coordination with verification, memory, and recoverable context
AFlow / AgentSquare [4, 12]	Workflow construction and modular agent search	Optimize agent structures; Lemon emphasizes evidence-conditioned decisions during execution
ToolOrchestra [13]	Online model/tool orchestration	Learns orchestration policies; Lemon studies a prompt-based controller and its evidence-directed interventions
MemGPT [14]	Context and memory management	Manages memory tiers and context; Lemon combines recoverable raw-evidence access with other execution choices

G.5 Multi-agent debate, agent teams, and aggregation

Multi-agent LLM methods instantiate several agents to decompose tasks, debate candidate answers, or aggregate independent reasoning traces. Role-based and communicative systems such as CAMEL, MetaGPT, AutoGen, and ChatDev demonstrate that structured interaction can improve planning, software development, and complex task execution [15, 16, 51, 52]. Debate and reflective collaboration methods show that disagreement can expose errors and improve factuality or reasoning when agents critique one another's claims [17, 53]. Benchmarks and routing methods then evaluate when multi-agent collaboration helps, how agents should be routed, and how redundant or harmful participants can be pruned [54–56]. Resilience analyses further show that collaboration structure matters when some agents provide faulty or low-quality contributions [57].

Lemon shares the motivation that multiple agents can expose missing evidence and reduce blind spots, but it differs in the unit of collaboration. Conventional multi-agent teams are usually instantiated inside one workflow, with roles assigned by the system designer for the current task. Lemon instead studies peer personalized instances that may have accumulated different tool collections, memories, and preferences before collaboration begins. This makes expertise discovery, evidence exposure, and critique-driven confidence updates central. Frameworks such as AutoGen [16] also support agents with different models, tools, or human participants. Lemon differs in the coordination primitives it specifies and evaluates.

G.6 Inter-instance personalized agent collaboration

Lemon focuses on independently personalized agent instances. These instances may share a similar backbone model, yet diverge through different tools, memories, preferences, and usage histories. The central issue is therefore not only how to split a task into subproblems. It is also how to discover differentiated expertise, exchange evidence-bearing proposals, critique unsupported claims, reconcile conflicting fragments, and preserve useful collaboration traces for later single-instance execution.

This setting is informed by transactive memory systems, where groups benefit from knowing who has relevant expertise [58, 59]. It is also informed by collective sensemaking and collective intelligence, where disagreement, coordination, and complementary expertise shape group performance [60, 61]. Recent human–AI and team-reflection studies further motivate making expertise and feedback visible in collaborative settings [10, 62]. Lemon operationalizes these ideas through expertise visibility, proposal artifacts, critique rounds, consensus, and artifact writing.

This translation is intentionally limited. Lemon does not claim that artificial agents reproduce human social systems, nor that social theory alone guarantees better answers. Instead, these concepts motivate concrete coordination rules: expose expertise before routing, require proposals to carry evidence and uncertainty, critique unsupported fragments before consensus, de-duplicate dependent evidence sources, and write useful collaboration artifacts back into individual memory. The contribution is

a reusable pattern for coordinating the current task and retaining artifacts, and its effects on later individual performance are left to future work.

G.7 Comparison summary

Table 7 reports the accuracy of the four protocols on the collaboration stress set, and Table 14 reports the cost of the isolated and four-instance configurations. Reduced manual tool-configuration effort, longitudinal artifact reuse, and downstream single-instance gains are left for future evaluation. Table 23 summarizes the intended organizational distinctions.

Table 23: Comparison between conventional agent teams and Lemon inter-instance organization.

Aspect	**Conventional agent team**	**Lemon organization**
Participant origin	Sub-agents are instantiated within one workflow	Profiles differ before collaboration (constructed profiles in our experiments)
Source of heterogeneity	Roles and prompts are assigned by the system designer	Profiles vary tools, preferences, and memory seeds
Primary objective	Improve the current task through decomposition or parallelism	Improve the current task and create reusable traces for future tasks
Coordination signal	Role labels, task partitions, debate, or voting	Expertise, proposals, critique, and consensus
Post-task effect	Final answer is usually the main output	Artifacts are internalized into memory

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 introduction state the leaderboard, resource, ablation, and stress-set results with their scope, and Section 6 states the scope of each result.

   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: Section 7 discusses the prompt-based controller and the unverified assumptions of the design contracts, the scope and repetition of the evaluation, and the risks of memory sharing, and it outlines ongoing work on each.

   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: [Yes]

Justification: The formal statements are given as conditional design contracts with their assumptions, the proofs are in Appendix C, and the standard coverage result is credited to Nemhauser et al.

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: [No]

Justification: Models, decoding settings, prompts, numeric controls, and protocols are specified, and a reference implementation with presets and offline tests is provided. The experiment code and run logs are not released, so the reported numbers cannot be regenerated exactly from the release. Appendix E gives the details.

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: [No]

   Justification: The supplement contains a reference implementation and instructions, but not the code that produced the reported numbers, and benchmark data cannot be redistributed under their terms.

   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: Section 6 and Appendices D and E specify models, decoding settings, round caps, numeric controls, task slices, selection rules, profiles, and evaluation protocols. No model is trained.

   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: Wilson intervals are reported for the GAIA score, three-run means and standard deviations for the resource comparison, and 95% intervals for the inter-instance study. Exact McNemar tests are reported for that study and the controller interventions. Slice ablations, sweeps, and controller interventions are limited-repetition measurements without error bars, and the leaderboard result is a single submission.

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: The orchestration hardware, per-task time and token use, and the hosted-model roles are reported in Appendix E. Inference uses hosted APIs and no local accelerators. Compute for preliminary runs is not separately reported.

   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 paper discusses bounded tool access, privacy, provenance, and benchmark redistribution terms. Appendix F includes qualitative case figures from benchmark-task execution.

   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: [Yes]

    Justification: Section 7 discusses privacy, provenance, and over-reliance risks of tool use and memory sharing, and the positive impact is more reliable long-horizon agent execution under bounded compute.

    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: The release contains a reference implementation and case-study figures, but no new pretrained model or scraped dataset.

    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: GAIA, xbench, the compared agents, and the tool code adapted from CAMEL-AI, MiroFlow, and Co-Sight are credited and their license or access terms are stated in Appendix E.3. The supplement retains upstream license headers and third-party notices.

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 reference implementation is documented in Appendices B, D.2, and E.3 with configuration, protocols, memory policy, presets, offline tests, and the limits of what it reproduces.

    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 reports no crowdsourcing or human-subject study.

    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 human-subject research requiring IRB disclosure.

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: Appendix E lists the controller, worker, extraction, multimodal, embedding, and summarization models. No model is trained or fine-tuned.

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.