InfiAgent: An Infinite-Horizon Framework for General-Purpose Autonomous Agents

Chenglin Yu[1,2], Yuchen Wang[2], Songmiao Wang[2], Hongxia Yang[2], Ming Li[2,3]*
[1]The University of Hong Kong
[2]The Hong Kong Polytechnic University
[3]The Hong Kong Polytechnic University-Daya Bay Technology and Innovation Research Institute

Abstract

LLM agents can reason and use tools, but they often break down on long-horizon tasks due to unbounded context growth and accumulated errors. Common remedies such as context compression or retrieval-augmented prompting introduce trade-offs between information fidelity and reasoning stability. We present **InfiAgent**, a general-purpose framework that keeps the agent's reasoning context strictly bounded regardless of task duration by externalizing persistent state into a *file-centric state abstraction*. At each step, the agent reconstructs context from a workspace state snapshot plus a fixed window of recent actions. Experiments on DeepResearch and an 80-paper literature review task show that, without task-specific fine-tuning, InfiAgent with a 20B open-source model is competitive with larger proprietary systems and maintains substantially higher long-horizon coverage than context-centric baselines. These results support explicit state externalization as a practical foundation for stable long-horizon agents.

Keywords: Large Language Model Agents, Infinite-Horizon Reasoning, File-Centric State Management, Autonomous Research, Multi-Agent Collaboration

1 Introduction

Large Language Models (LLMs) have increasingly been deployed as autonomous agents capable of planning, tool use, and multi-step reasoning across a wide range of tasks (Naveed et al., 2025; Tran et al., 2025). Recent agent-based systems demonstrate promising performance in domains such as scientific research, software engineering, and information synthesis, suggesting that LLMs can serve as general-purpose problem solvers that coordinate external tools and resources. Despite this progress, existing LLM agents remain brittle when deployed over long task horizons.

A core challenge arises from how agent state is represented and maintained. Most current agent frameworks implicitly treat the LLM prompt as the primary carrier of state, accumulating dialogue history, tool traces, intermediate plans, and partial results directly within the context window. As task duration increases, this design leads to unbounded context growth, forcing systems to rely on truncation, summarization, or heuristic retrieval to remain within model limits. These mechanisms introduce well-known failure modes, including information loss, interference from irrelevant tokens, and increased sensitivity to early errors, ultimately resulting in unstable behavior over long horizons.

Retrieval-Augmented Generation (RAG) and long-context models partially mitigate context length limitations by enabling access to external memory or larger windows. However, these approaches still entangle long-term task state with the agent's immediate reasoning context, placing increasing cognitive load on the LLM as execution progresses. Recent benchmark evidence on deep research agents shows that long-horizon report generation remains brittle despite strong short-horizon capabilities (Du et al., 2025). Consequently, simply extending context length does not fundamentally resolve the long-horizon stability problem.

Recent work such as MAKER (Meyerson et al., 2025) demonstrates that near-infinite execution is possible in highly structured domains through extreme task decomposition. While effective for logic-based problems with predefined subtask boundaries, such approaches rely on specialized micro-agents and rigid workflows, limiting their applicability to open-ended domains like scientific research, where task structure, intermediate goals, and relevant information sources are not known a priori.

In this work, we argue that achieving stable long-

*Corresponding author: ming.li@polyu.edu.hk

Findings of the Association for Computational Linguistics: ACL 2026, pages 35884–35894

horizon behavior in LLM agents requires an explicit separation between *persistent task state* and *bounded reasoning context*. We introduce **InfiAgent**, a general-purpose agent framework that externalizes long-term state into a file-centric representation. Rather than storing historical information implicitly within the prompt, InfiAgent treats the file system as the authoritative and persistent record of the agent's actions, environment, and intermediate artifacts. At each decision step, the agent reconstructs its reasoning context solely from a snapshot of this externalized state and a fixed-size window of recent actions, ensuring that the context size remains strictly bounded regardless of task duration.

Building on this state abstraction, InfiAgent employs a hierarchical agent architecture that enforces structured task decomposition and controlled tool invocation. High-level planning agents operate over abstract goals and state summaries, while lower-level agents execute domain-specific or atomic actions. This design reduces error propagation commonly observed in flat multi-agent systems and enables consistent behavior across extended execution horizons. In addition, InfiAgent introduces an external attention mechanism that processes large documents and heavy information outside the main reasoning context, injecting only task-relevant outputs back into the agent's state.

We evaluate InfiAgent on DeepResearch Bench (Du et al., 2025) and a long-horizon literature review task involving up to 80 academic papers. Without task-specific fine-tuning, InfiAgent equipped with a 20B-parameter open-source model achieves competitive performance against substantially larger proprietary research agents on the benchmark. In extended evaluations spanning hundreds of execution steps, the proposed file-centric state abstraction enables reliable task completion, whereas baseline agents frequently degrade due to context limitations. These results suggest that explicit state externalization provides a practical and effective foundation for building stable, general-purpose LLM agents capable of operating over arbitrarily long horizons.

2 Related Work

2.1 Multi-Agent System Architectures

Traditional multi-agent systems have primarily focused on peer-to-peer collaboration models. While effective for simple coordination, these approaches struggle with complex, hierarchical task decomposition. Recent work has explored hierarchical organization (Liu et al., 2025). Moore (Moore, 2025) categorized Hierarchical Multi-Agent Systems (HMAS) highlighting trade-offs between global efficiency and local autonomy. Comprehensive reviews note a persistent challenge in designing unified agents that seamlessly integrate cognition, planning, and interaction (Qu et al., 2025). InfiAgent addresses these by enforcing stability through strict compositional constraints and a file-centric state.

2.2 Task Decomposition in Multi-Agent Systems

Task decomposition has been a central challenge. Existing approaches include goal-oriented (Li et al., 2023), constraint-based (Cai et al., 2018), and learning-based decomposition (Woo et al., 2022). Frameworks like AGENTiGraph leverage knowledge graphs (KGs) for decomposition (Zhao et al., 2025). However, these approaches often lack guarantees regarding system stability over long horizons. MAKER (Meyerson et al., 2025) introduced extreme decomposition for logic tasks, but its applicability to open-ended research remains limited. InfiAgent employs a recursive DAG-based decomposition that is flexible enough for general tasks while maintaining strict parent-child control.

2.3 Long-Context and Autonomous Research Agents

Benchmark suites such as DeepResearch Bench (Du et al., 2025) highlight the difficulty of sustained research-oriented agent execution. Agents like AgentGym (Xi et al., 2025) and Richelieu (Guan et al., 2024) focus on self-evolution, while systems like the AI Scientist attempt end-to-end automation. However, many rely on extended context windows or manual templates. InfiAgent's approach of file-based state offers a robust alternative to these context-heavy methods.

3 Formalizing File-Centric State for Long-Horizon Agents

We formalize the long-horizon agent execution problem by distinguishing between *persistent task state* and *bounded reasoning context*. This separation clarifies the limitations of context-centric agent designs and motivates the file-centric state abstraction adopted by InfiAgent.

3.1 Agent Execution as a State-Conditioned Decision Process

We consider an autonomous LLM agent operating over discrete time steps $t = 1, 2, \ldots$. At each step, the agent selects an action $a_t \in \mathcal{A}$ conditioned on an internal representation of task state. In conventional agent frameworks, this state is implicitly represented by the accumulated prompt context:

$$c_t = \langle o_1, a_1, \ldots, o_{t-1}, a_{t-1}, o_t \rangle, \tag{1}$$

where o_t denotes observations such as user instructions, tool outputs, or intermediate results. As t increases, the context length $|c_t|$ grows unbounded, necessitating truncation or compression to fit within a finite context window.

We define this design as *context-centric state representation*, where long-term task information and short-term reasoning signals are entangled within a single sequence. This entanglement introduces an inherent trade-off between information retention and reasoning stability, particularly in long-horizon tasks.

3.2 Persistent State Externalization

To decouple long-term memory from bounded reasoning context, we introduce an explicit *persistent state* representation S_t that evolves over time:

$$S_t = \mathcal{F}_t, \tag{2}$$

where $\mathcal{F}_t$ denotes the set of files and structured artifacts stored in the agent's workspace at step t. These artifacts may include intermediate results, plans, summaries, datasets, or generated code, and collectively serve as the authoritative record of task progress.

The persistent state evolves through state-transition operators induced by agent actions:

$$\mathcal{F}_{t+1} = \mathcal{T}(\mathcal{F}_t, a_t), \tag{3}$$

where $\mathcal{T}$ captures file creation, modification, or deletion. Importantly, $\mathcal{F}_t$ is not subject to context window constraints and can grow with task complexity and duration.

3.3 Bounded Reasoning Context Reconstruction

At each step t, the agent constructs a bounded reasoning context c_t^{bounded} by querying the persistent state:

$$c_t^{\text{bounded}} = g(\mathcal{F}_t, a_{t-k:t-1}), \tag{4}$$

where $a_{t-k:t-1}$ denotes a fixed-size window of the most recent k actions, and $g(\cdot)$ is a deterministic context-construction function. In InfiAgent, k is fixed as a small constant for a given run, ensuring that $|c_t^{\text{bounded}}| = \mathcal{O}(1)$ with respect to the task horizon.

This formulation guarantees that the agent's reasoning context remains strictly bounded for all t, while full task history is preserved implicitly through the persistent state $\mathcal{F}_t$. Unlike summarization-based approaches, no information is discarded from the authoritative state; instead, relevance is determined dynamically at each step via state inspection.

In the implementation, $g(\cdot)$ is a fixed-schema builder rather than a free-form directory dump. A dedicated thinking model periodically rewrites a persistent thinking record M_t from the current workspace and the most recent action window. This record contains four required parts: a task-level todo list, descriptions of relevant files, pinned carry-over state (workspace structure, rules, failures, and content that must survive window resets), and the next k tool-level steps. The execution context concatenates this record with the latest user instruction, recent task history, agent-hierarchy state, and the current action window. After each thinking update, the visible action window is reset, so long trajectories are replaced by the updated M_t rather than accumulated in the prompt. Per-agent action traces and the latest thinking record are also persisted separately, enabling checkpoint-based resume after interruption.

3.4 Comparison and Implications

Compared to context-centric agents, which encode all task state directly in the prompt, and retrieval-augmented agents, which partially externalize memory but re-inject retrieved text into the context, the proposed file-centric abstraction treats persistent state as a first-class object. By reconstructing reasoning context through a bounded interface, InfiAgent eliminates unbounded context growth and reduces interference between long-term artifacts and short-term decision-making.

As a result, reasoning errors are less likely to accumulate implicitly over time, and the agent can scale execution horizon without increasing cognitive load on the underlying language model.

4 Methodology: The InfiAgent Framework

Building on the file-centric state formulation in Section 3, we describe InfiAgent as a concrete system that instantiates persistent state externalization and bounded context reconstruction. This section focuses on architectural and implementation choices, rather than re-defining the state abstraction. Figure 1 provides an overview.

4.1 File-Centric State Management and Task Memory

InfiAgent materializes the persistent state $\mathcal{F}_t$ as a structured workspace on the file system. Each task is assigned a dedicated workspace directory that stores plans, intermediate artifacts, tool outputs, and validation logs. This workspace serves as the authoritative record of task progress and persists across execution steps and sessions.

To support bounded reasoning, InfiAgent separates durable workspace artifacts from a compact thinking record that serves as the bounded interface to the main LLM. The runtime exposes a small fixed action window together with a matched thinking interval; after each interval, a dedicated thinking model rewrites the thinking record and the visible action buffer is cleared.

Periodic State Consolidation. To sustain long-horizon execution, InfiAgent periodically consolidates recent progress into persistent state. Operationally, the thinking record is not a raw directory listing. It is a fixed-schema summary containing current to-do items, file descriptions likely to be reused, pinned state that should survive history truncation, and the next tool-level steps. This makes context construction deterministic at the level of inputs, ordering, and budget, even though the textual contents are model-generated.

If execution is interrupted, the runtime preserves both the latest thinking record and the full action trace in task-local state files and archives interrupted runs into task history. Recovery therefore resumes from the last saved checkpoint rather than replaying the full dialogue.

4.2 Multi-Level Agent Hierarchy

InfiAgent organizes agents into a tree-structured hierarchy (DAG):

- **Level 3 (Alpha Agent):** The orchestrator responsible for high-level planning and decomposing user requests into subtasks. It acts as the root of the decision tree.
- **Level 2 (Domain Agents):** Specialists such as `data_collection_agent`, `data_analysis_agent`, `coder_agent`, or `material_to_document_agent`. They execute specific workflows delegated by the Alpha Agent.
- **Level 1 (Atomic Agents):** Single-purpose executors such as `answer_from_papers`, `get_data_from_arxiv`, or `get_data_from_web_search`.

This hierarchy allows for serial execution with clear boundaries. Higher-level agents invoke lower-level agents as callable tools (Agent-as-a-Tool), preventing "tool calling chaos" often seen in flat multi-agent systems where agents compete for execution.

4.3 External Attention Pipeline

To handle massive information (e.g., reading 80 papers) without context bloat, InfiAgent uses an *External Attention Pipeline*. When an agent needs information from a document, it does not load the document into its context. Instead, it calls a specialized document-reading agent (e.g., `answer_from_papers`). This auxiliary call spins up an isolated LLM process to query the document and returns only the extracted answer.

$$C_{main} \leftarrow C_{main} \cup Tool(Query, Document) \tag{5}$$

This effectively offloads the "reading" cost to the tool layer, keeping the main agent's cognitive load low and focused on decision-making. This mechanism acts as an application-layer attention head, selecting only the relevant information from massive external data sources.

5 Experiments

The goal of our experiments is to evaluate whether explicit state externalization improves the stability and reliability of LLM agents in long-horizon tasks. Rather than optimizing for peak performance on short interactions, we focus on execution robustness under extended task duration, large document collections, and repeated tool usage.

We evaluate InfiAgent along two complementary dimensions. First, we assess general research capability using DeepResearch Bench (Du et al.,

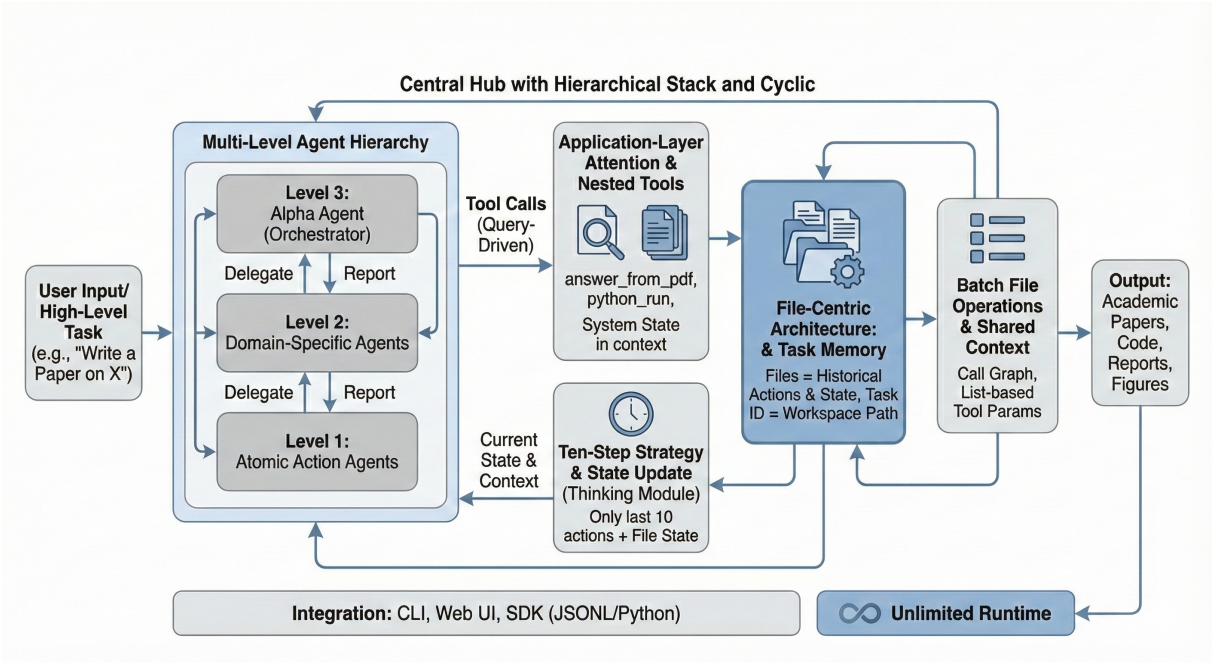

Figure 1: **The InfiAgent Framework.** InfiAgent implements a hierarchical execution stack over a file-centric persistent state. Files serve as the authoritative task memory, while an external attention mechanism processes heavy documents outside the bounded reasoning context. Periodic state consolidation refreshes the agent's context from the workspace snapshot.

2025), a standardized evaluation designed to measure multi-step research quality. Second, we design a long-horizon literature review task to directly stress-test agent stability over hundreds of execution steps. Across all experiments, we use identical model backbones and comparable prompting budgets when applicable, and we report averaged results over multiple runs to reduce variance.

5.1 DeepResearch Benchmark

We evaluate InfiAgent on DeepResearch Bench (Du et al., 2025), which is designed to assess an agent's ability to conduct multi-step research involving information gathering, synthesis, and structured reporting. The benchmark scores agents along four dimensions: comprehensiveness, insight, instruction following, and readability.

Setup. We configure InfiAgent using a 20B-parameter open-source model (`gpt-oss-20b`) without any task-specific fine-tuning. To isolate the impact of agent architecture, we follow the standard benchmark protocol and adopt default task instructions wherever possible. All reported scores are taken directly from the benchmark evaluation to ensure comparability across systems.

Overall performance. On DeepResearch Bench, InfiAgent achieves an overall score of 41.45 with a 20B open-source model. Appendix 1 shows that this places our configuration close to several substantially larger proprietary research agents on the public benchmark. We do not interpret this as an intra-system scaling result for InfiAgent, because InfiAgent is evaluated at one model size only.

Component-wise analysis. Figure 2 presents a breakdown across evaluation dimensions. InfiAgent performs particularly well on instruction following and readability. We attribute this behavior to the explicit file-centric state and structured execution pipeline, which encourage adherence to task instructions and consistent output formats. Performance on insight and comprehensiveness is competitive with larger models, suggesting that stable state management supports sustained reasoning over extended interactions.

Overall, these results suggest that explicit externalization of agent state can improve the reliability and efficiency of multi-step research behavior, even when using smaller backbone models.

5.2 Long-Term Literature Review Task

To evaluate long-horizon stability, we design a literature review task that requires an agent to iterate over a large paper collection. Each run provides 80 local PDFs plus a scoring rubric and instructs

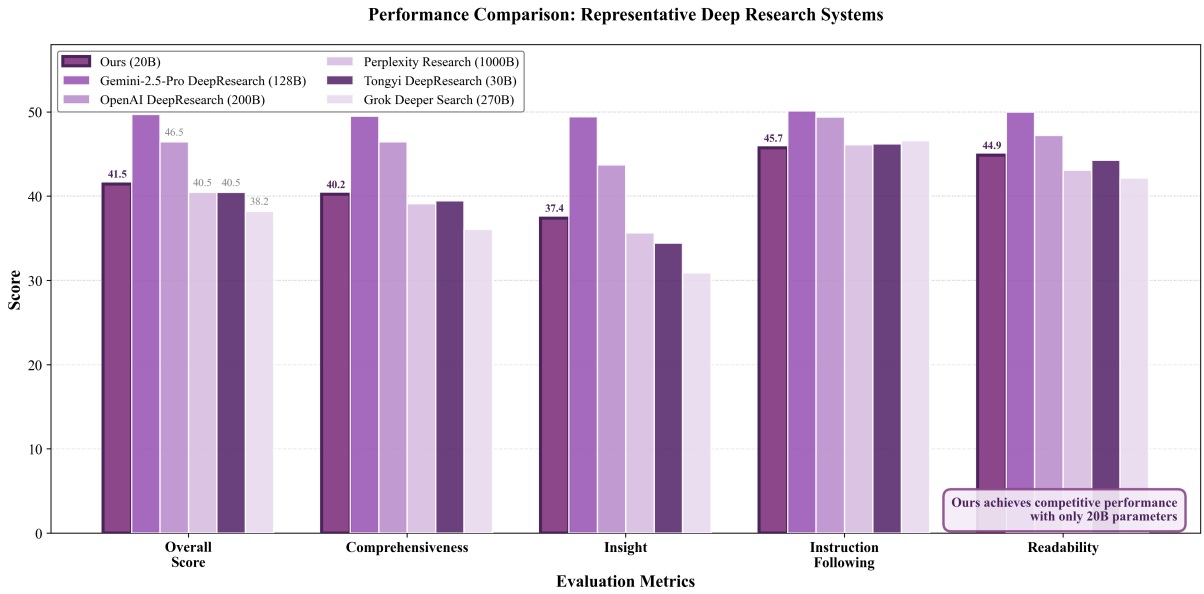

Figure 2: **Component-wise comparison on DeepResearch.** Scores are broken down by evaluation dimension. InfiAgent shows strong performance on instruction following and readability, which are closely related to structured state management and output control.

the agent to (i) read each paper, (ii) produce a short summary, and (iii) assign a relevance score. This task stresses sustained tool usage and state tracking over hundreds of steps and requires reliable local-file operation rather than web retrieval.

Evaluation metric (coverage). We focus on *task completion reliability* rather than summary quality. Specifically, we measure **coverage**, defined by log-based auditing: a paper counts as covered only if the trace contains an actual PDF-body read or extraction for that document and the final output includes a non-empty content-grounded note beyond title/metadata. We report the maximum, minimum, and average coverage over repeated runs under the same setting.

Results. Detailed numbers are reported in Appendix D. With file-centric state, InfiAgent reaches 67.1 average coverage with GPT-OSS-20B and 80.0 with stronger backbones. Under comparable local-workspace interfaces, Claude Code averages 29.1 coverage, while Cursor averages 1.0 and 0.1 with Claude-4.5-Sonnet and Gemini-3-Flash, respectively. This behavior is consistent with the proposed file-centric state abstraction, which enables the agent to iterate through long document lists without relying on an ever-growing prompt history.

In contrast, baseline agents exhibit substantially lower average coverage and high variance across runs. Manual inspection suggests that failures often manifest as early termination, skipping items, or producing summaries that primarily restate titles rather than paper content.

Ablation analysis. Removing the file-centric state and relying on compressed long-context prompts substantially degrades coverage across all models. Average coverage drops to 3.2, 21.1, and 27.7 for GPT-OSS-20B, Gemini-3-Flash, and Claude-4.5-Sonnet, respectively. This supports the claim that explicit persistent state externalization is a key contributor to long-horizon stability and cannot be fully replaced by long-context compression alone.

Limitations of the comparison. We do not claim absolute superiority over proprietary systems. Commercial agents may employ undocumented optimizations or termination heuristics that trade off completeness for latency or cost. Our goal is to highlight a qualitative difference in long-horizon behavior: explicit externalization of persistent state is associated with higher and more stable coverage under extended execution horizons.

6 Discussion

What Explicit State Externalization Does Not Solve. While the proposed file-centric state abstraction significantly improves long-horizon stability, it does not inherently enhance the reasoning

capability of the underlying language model. If a backbone model produces incorrect intermediate conclusions or flawed artifacts, these errors may still be written into persistent state and propagated across subsequent steps. Although separating persistent state from bounded context enables inspection and potential correction, it does not eliminate the need for validation mechanisms or human oversight in high-stakes settings.

Long Context Is Not a Substitute for Persistent State. The ablation results in Section 5.2 highlight an important distinction between long context and persistent state. Even when using models with large context windows, replacing file-centric state with compressed long-context prompts leads to substantial degradation in coverage and increased variance across runs. This suggests that increased context length alone is insufficient for reliable long-horizon execution, and that explicit state externalization plays a qualitatively different role than context expansion.

Efficiency and Latency Trade-offs. Externalizing state and enforcing hierarchical execution introduce additional overhead compared to single-pass, context-centric agents. File system operations, periodic state consolidation, and serialized agent execution may increase latency, particularly for tasks requiring rapid responses. As a result, the proposed framework is better suited for long-running, knowledge-intensive workflows than for real-time interactive applications. Exploring asynchronous updates and partial parallelism remains an important direction for future work.

Scope and Generalization. Our empirical evaluation focuses on research-oriented tasks involving extended document processing and multi-step information synthesis. While these settings expose long-horizon failure modes, they do not fully represent other agent scenarios such as reactive dialogue, embodied interaction, or environments with rapidly changing external state. Further evaluation across a broader range of tasks is needed to assess generality.

Broader Implications. Despite these limitations, the results suggest a general design principle for long-horizon agents: persistent task state should be treated as a first-class object, distinct from the bounded reasoning context of the language model. This separation enables systematic analysis of agent behavior over time and provides a foundation for future work on verification, correction, and collaborative multi-agent systems.

7 Limitations

While InfiAgent demonstrates robust capabilities in long-horizon tasks, several limitations remain. First, the multi-level agent hierarchy introduces **latency overhead**. As the depth of the agent tree increases, the time required for task delegation and result aggregation grows, potentially impacting real-time responsiveness. Second, despite the file-centric state management mitigating context loss, **hallucination accumulation** is still a risk, particularly when using smaller models (e.g., 20B) for extremely long tasks. If a sub-agent writes incorrect information to the file system that is not caught by the validation mechanism, downstream agents may propagate this error. Third, the current architecture strictly enforces serial execution to ensure state consistency, which means it **does not support parallel processing**. This limits the system's efficiency in tasks that are inherently parallelizable, such as simultaneous literature review and experiment coding.

8 Conclusion

InfiAgent proposes a shift from context-centric agents to file-centric agents. By externalizing persistent state and periodically rewriting a bounded thinking record, InfiAgent maintains a fixed reasoning interface over long horizons. Our results show that this **training-free** architecture enables a 20B open-source model to remain competitive with larger proprietary research agents while substantially improving long-horizon coverage.

Acknowledgments

This paper is partially supported by three grants from the Research Grants Council of the Hong Kong Special Administrative Region, China (No. PolyU15208824,PolyU15215325 and T41-517/25-N) and two Innovation and Technology Funds from the Innovation and Technology Commission of the Hong Kong Special Administrative Region, China (No.ITP/003/26LP and ITP/013/26TI).

AI Usage Declaration

In accordance with ACL policies, we declare the use of AI assistance in the preparation of this manuscript. The framework illustration (Figure

1) was generated using the Gemini Nano Banana model. The text of this paper was polished for clarity and grammar using Gemini 1.5 Flash. Appendix E describes and links to raw agent-generated artifacts preserved as outputs of the evaluated system; their internal text, citation markers, and reference lists are not author-provided citations for this manuscript. All scientific claims, experimental designs, and results were verified by human authors.

References

Xinye Cai, Zhiwei Mei, Zhun Fan, and Qingfu Zhang. 2018. A constrained decomposition approach with grids for evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 22(4):564–577.

Mingxuan Du, Benfeng Xu, Chiwei Zhu, Xiaorui Wang, and Zhendong Mao. 2025. DeepResearch Bench: A comprehensive benchmark for deep research agents. *arXiv preprint arXiv:2506.11763*.

Zhenyu Guan, Xiangyu Kong, Fangwei Zhong, and Yizhou Wang. 2024. Richelieu: Self-evolving LLM-based agents for AI diplomacy. In *Advances in Neural Information Processing Systems*, volume 37, pages 123471–123497. Curran Associates, Inc.

Wenhao Li, Dan Qiao, Baoxiang Wang, Xiangfeng Wang, Bo Jin, and Hongyuan Zha. 2023. Semantically aligned task decomposition in multi-agent reinforcement learning. *arXiv preprint arXiv:2305.10865*.

Haowei Liu, Xi Zhang, Haiyang Xu, Yuyang Wanyan, Junyang Wang, Ming Yan, Ji Zhang, Chunfeng Yuan, Changsheng Xu, Weiming Hu, and Fei Huang. 2025. PC-Agent: A hierarchical multi-agent collaboration framework for complex task automation on PC. *arXiv preprint arXiv:2502.14282*.

Elliot Meyerson, Giuseppe Paolo, Roberto Dailey, Hormoz Shahrzad, Olivier Francon, Conor F. Hayes, Xin Qiu, Babak Hodjat, and Risto Miikkulainen. 2025. Solving a million-step LLM task with zero errors. *arXiv preprint arXiv:2511.09030*.

David J. Moore. 2025. A taxonomy of hierarchical multi-agent systems: Design patterns, coordination mechanisms, and industrial applications. *arXiv preprint arXiv:2508.12683*.

Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2025. A comprehensive overview of large language models. *ACM Transactions on Intelligent Systems and Technology*, 16(5):1–72.

Xiaodong Qu, Andrews Damoah, Joshua Sherwood, Peiyan Liu, Christian Shun Jin, Lulu Chen, Minjie Shen, Nawwaf Aleisa, Zeyuan Hou, Chenyu Zhang, Lifu Gao, Yanshu Li, Qikai Yang, Qun Wang, and Cristabelle De Souza. 2025. A comprehensive review of AI agents: Transforming possibilities in technology and beyond. *arXiv preprint arXiv:2508.11957*.

Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O'Sullivan, and Hoang D. Nguyen. 2025. Multi-agent collaboration mechanisms: A survey of LLMs. *arXiv preprint arXiv:2501.06322*.

Honguk Woo, Gwangpyo Yoo, and Minjong Yoo. 2022. Structure learning-based task decomposition for reinforcement learning in non-stationary environments. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(8):8657–8665.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Xin Guo, Dingwen Yang, Chenyang Liao, Wei He, Songyang Gao, Lu Chen, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, Xipeng Qiu, Xuanjing Huang, Zuxuan Wu, and Yu-Gang Jiang. 2025. AgentGym: Evaluating and training large language model-based agents across diverse environments. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 27914–27961, Vienna, Austria. Association for Computational Linguistics.

Xinjie Zhao, Moritz Blum, Fan Gao, Yingjian Chen, Boming Yang, Luis Marquez-Carpintero, Mónica Pina-Navarro, Yanran Fu, So Morikawa, Yusuke Iwasawa, Yutaka Matsuo, Chanjun Park, and Irene Li. 2025. AGENTiGraph: A multi-agent knowledge graph framework for interactive, domain-specific LLM chatbots. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pages 6757–6761. Demo track.

A Appendix: DeepResearch Detailed Results

Table 1 provides the complete evaluation results for InfiAgent and various baseline systems on the DeepResearch benchmark.

B Appendix: Operational Context Construction

Shared runtime state. InfiAgent persists task state in two complementary layers. The first layer is the workspace itself, which stores durable artifacts such as plans, code, summaries, and logs. The second layer is a task-level share-context record that stores the latest instruction, agent hierarchy, runtime metadata, and the latest thinking state for each running agent.

Model/Agent	Overall	Comp.	Insight	Inst. Fol.	Read.	Cit. Acc.	Eff. Cit.	Params (B)
tavily-research (GPT-5)	52.44	52.84	53.59	51.92	49.21	-	-	1000
thinkdepthai-deepresearch (GPT-5)	52.43	52.02	53.88	52.04	50.12	-	-	1000
cellcog (GPT-4o)	51.94	52.17	51.90	51.37	51.94	-	-	200
salesforce-air-deep-research (GPT-4o)	50.65	50.00	51.09	50.77	50.32	-	-	200
gensee-search-gpt-5 (GPT-5)	50.60	50.06	50.76	51.31	49.72	32.94	21.06	1000
gemini-2.5-pro-deepresearch	49.71	49.51	49.45	50.12	50.00	78.30	165.34	128
langchain-open-deep-research (GPT-5)	49.33	49.80	47.34	51.05	48.99	34.74	22.44	1000
openai-deepresearch (GPT-4o)	46.45	46.46	43.73	49.39	47.22	75.01	39.79	200
claude-research (Claude-3.7-Sonnet)	45.00	45.34	42.79	47.58	44.66	-	-	175
kimi-researcher (Kimi-K2)	44.64	44.96	41.97	47.14	45.59	-	-	1000
doubao-deepresearch (Doubao-1.5-Pro)	44.34	44.84	40.56	47.95	44.69	52.86	52.62	300
langchain-open-deep-research (GPT-4o)	43.44	42.97	39.17	48.09	45.22	49.10	29.49	200
ours (`gpt-oss-20b`)	**41.45**	**40.22**	**37.39**	**45.72**	**44.87**	-	-	20
nvidia-aiq-research-assistant	40.52	37.98	38.39	44.59	42.63	-	-	70
tongyi-deepresearch-30B-A3B	40.46	39.46	34.44	46.22	44.27	-	-	30
perplexity-Research (GPT-5)	40.46	39.10	35.65	46.11	43.08	82.63	31.20	1000
grok-deeper-search (Grok-2)	38.22	36.08	30.89	46.59	42.17	73.08	8.58	270
sonar-reasoning-pro	37.76	34.96	31.65	44.93	42.42	45.19	9.39	670
sonar-reasoning	37.75	34.73	32.59	44.42	42.39	52.58	13.37	70
claude-3-7-sonnet-with-search	36.63	35.95	31.29	44.05	36.07	87.32	24.51	-
sonar-pro	36.19	33.92	29.69	43.39	41.07	79.72	16.75	70
gemini-2.5-pro-preview	31.90	31.75	24.61	40.24	32.76	-	-	128
gpt-4o-search-preview	30.74	27.81	20.44	41.01	37.60	86.63	5.05	200
sonar	30.64	27.14	21.62	40.70	37.46	76.41	10.68	20
gpt-4.1	29.31	25.59	18.42	40.63	36.49	89.85	4.27	200
gemini-2.5-flash-preview	29.19	28.97	21.62	37.80	29.97	-	-	128
gpt-4o-mini-search-preview	27.62	24.24	16.62	38.59	35.27	81.69	4.62	200
gpt-4.1-mini	26.62	22.86	15.39	38.18	34.49	84.54	4.10	200
claude-3-5-sonnet-with-search	23.95	21.28	16.20	32.41	29.87	94.06	9.35	175

Table 1: Detailed performance breakdown on the DeepResearch benchmark. Scores for comprehensiveness (Comp.), insight, instruction following (Inst. Fol.), and readability (Read.) are normalized. Parameters are estimated based on public reports.

Thinking trigger and window reset. The runtime keeps a fixed small window of recent actions. At initialization, and whenever the tool-call counter crosses a multiple of the thinking interval, a dedicated thinking model reads the current system context together with the visible action history and rewrites the current thinking record. After this update, the visible action buffer is cleared while the rewritten thinking record is retained. This is the mechanism that prevents unbounded growth of prompt-visible history.

Thinking schema. The thinking record follows a fixed four-part schema: a task-level to-do list, descriptions of files likely to be reused, pinned state that must survive history resets, and the next tool-level steps for the following window. This schema is fixed across runs even though the textual contents are generated by the thinking model.

Prompt assembly. For main execution, the bounded context is assembled from: (i) the shared system prompt, (ii) the latest user input, (iii) recent task history, (iv) the structured agent-call tree, (v) the current task input, (vi) the latest thinking record, and (vii) the current action window. For the thinking call, the same recent actions are rendered as structured history in the context so that the thinking model can rewrite the persistent record.

Failure and resume behavior. Each agent additionally persists its action history, latest thinking state, tool-call counter, and full system prompt in a task-local actions file. On interruption, the runtime archives interrupted agents together with their latest thinking state and completed child outputs; on restart, execution resumes from the last saved checkpoint rather than replaying the full dialogue history.

C Appendix: Prompt Templates

Shared scaffold. All agents receive a shared system-prompt scaffold that injects the latest user input, recent user-agent history, a history-search hint, the current agent name, workspace path, visible skills path, structured call information, the current task, and the latest thinking record. The thinking call additionally receives the recent action window rendered as structured history.

Alpha, domain, and atomic roles. The level-3 alpha agent is responsible for orchestration and delegation. Level-2 domain agents instantiate role-specific workflows such as data collection, analy-

sis, coding, figure creation, and document writing. Level-1 atomic agents are single-purpose executors such as document reading and retrieval. The public implementation instantiates these roles from fixed YAML prompt files with separate responsibility and workflow fields.

Thinking prompt. The dedicated thinking prompt enforces the four-block schema described above and requires tool-level next-step plans over the next fixed action window. This prompt is responsible for converting recent action traces into the bounded persistent summary that will be consumed by subsequent execution steps.

English translation of the thinking prompt template.

> You are a context-management specialist for an autonomous agent. Before the recent action history is discarded, you must rewrite the agent's persistent progress state. The previous progress state, if any, appears in `<Current Progress Thinking>`. You must output only the contents of the rewritten progress state, using the language of the latest user input rather than the language of this meta-prompt.
>
> `<todo_list>`: maintain a fine-grained task list for the current agent only. Update completed items to done, keep unfinished items as `waiting`, and mark partially advanced items as `ongoing`. Each todo should be small enough to finish within the next action window.
>
> `<Valid File Descriptions>`: list all files that may be reused in future steps, together with what each file is for and when it should be consulted. Remove stale file entries when they are no longer useful.
>
> `<Persistent State>`: preserve any information that must survive the next history reset. This block must include the current workspace structure, task rules, failure counts or failed attempts that should not be repeated, and any file contents or summaries that will still matter in the next window. If previous steps only read files without producing progress, warn about that and pin the useful contents here. If debugging attempts failed, record the failed strategy so the agent does not loop.
>
> `<next_n_steps>`: produce a concrete tool-level plan for the next fixed action window. Each step should name one specific tool action rather than a vague high-level intention. The plan should first gather any missing information, then read the minimum necessary context, then create an incremental artifact, and finally validate it when appropriate. Avoid vague phrases such as "process documents one by one." If the agent is currently following a skill, the first step of the next window should reread the skill file so the agent does not drift.

Public implementation defaults. The released runtime exposes `action_window_steps`, `thinking_interval`, and `thinking_steps` as configurable parameters. In the current public app configuration, all three defaults are set to 30, but they can be overridden per task or per run.

D Appendix: Long-Horizon Task Protocol and Results

Each long-horizon run places 80 local PDFs and a scoring rubric in the workspace. The agent must produce a structured review for each paper consisting of a score and a short justification. We compare only against agents that can operate over comparable local-workspace interfaces. Each run persists per-agent action histories, latest thinking snapshots, and task-level share-context metadata; coverage is computed from these traces by checking whether each target PDF has at least one read or extraction event together with a non-empty grounded output entry.

E Appendix: Raw End-to-End Generated Paper Artifacts

To further document the end-to-end capability discussed in the revision, we make three representative manuscripts generated in full research-assistant runs available outside the proceedings PDF at https://github.com/polyuiislab/infiAgent_Outputs. They are raw, unedited artifacts of the evaluated agent framework, not supporting literature for the main manuscript and not quantitative evidence for the paper's scientific claims.

The raw artifact PDFs are not embedded in this camera-ready proceedings PDF because they use their original page geometry and are not formatted as ACL manuscript pages. Keeping them external ensures that every page of the proceedings PDF follows the ACL margin requirements while preserving access to the original generated outputs.

Important note on internal references. The citation markers, bibliographies, and reference formatting inside the three external artifacts are part of the generated agent outputs. They have been intentionally preserved without post-hoc correction so that the artifacts reflect the behavior and limitations of the system. These internal references are not part of the manuscript's author-provided bibliography, have not been independently curated as supporting citations by the authors, and should not be interpreted as evidence for claims in the main paper. The author-provided references supporting this

Setting	Model	Max	Min	Avg
Main results (with file-centric state; InfiAgent vs. baselines)				
InfiAgent	GPT-OSS-20B	**80**	15	67.1
InfiAgent	Gemini-3-Flash	**80**	**80**	**80.0**
InfiAgent	Claude-4.5-Sonnet	**80**	**80**	**80.0**
Claude Code	Claude-4.5-Sonnet	80	11	29.1
Cursor	Claude-4.5-Sonnet	5	0	1.0
Cursor	Gemini-3-Flash	1	0	0.1
Ablation (remove file-centric state; compressed long-context prompts)				
No File State (Compressed Context)	GPT-OSS-20B	7	1	3.2
No File State (Compressed Context)	Gemini-3-Flash	25	20	21.1
No File State (Compressed Context)	Claude-4.5-Sonnet	77	11	27.7

Table 2: **Long-horizon task reliability (coverage) and ablation.** Coverage on the 80-paper literature review task. Top: main results with file-centric state (InfiAgent vs. baselines). Bottom: ablation removing file-centric state and replacing it with compressed long-context prompts. We report max/min/avg coverage across repeated runs.

manuscript appear only in the References section
above.