RETHINKING MULTI-AGENT COLLABORATION: WHEN MORE IS LESS

Anonymous authors
Paper under double-blind review

ABSTRACT

The rapid advancement of large language models and single-agent harnesses has reshaped the landscape of autonomous systems, raising a critical question of when multi-agent collaboration offers genuine value. As individual agent capabilities continue to scale, multi-agent collaboration faces diminishing returns while incurring growing context overhead. Through systematic analysis, we delineate the capability boundaries of multi-agent collaboration relative to single-agent alternatives, showing that it confers systematic benefits specifically in long-horizon tasks with sparse dependencies, while single-agent harnesses remain superior in tightly coupled, sequential workflows. Building on these insights, we propose SAIGE, a lightweight multi-agent collaboration mechanism based on Semantic-Aware Incremental Graph Evolution. SAIGE models collaboration as a dynamically evolving graph, where nodes are agent instances spawned on demand and edges encode semantic dependencies established through content-based information retrieval. Experiments on long-horizon, complex task benchmarks show that SAIGE achieves a favorable trade-off between context efficiency and task performance, and that scaling the agent pool or deepening the recursion level does not consistently improve outcomes. Our findings suggest that multi-agent superiority is bounded by task structure rather than universal, and that more agents do not necessarily make a system more intelligent.

1 INTRODUCTION

The rapid advancement of large language models (LLMs) has fundamentally reshaped the landscape of autonomous systems. As foundation models grow increasingly capable of handling long-horizon, complex tasks, the accompanying agent harnesses, such as Claude Code (Anthropic, 2025), Codex (OpenAI, 2025a), and DeepSeek Harness (DeepSeek, 2026a), have matured into highly sophisticated infrastructures. These single-agent systems now deliver robust, end-to-end execution in real-world scenarios, setting a formidable baseline for autonomy. In stark contrast, generic multi-agent collaboration frameworks often remain toy-level in practice, lingering on simplistic benchmarks (Paech, 2024; Wang et al., 2024b; Peng et al., 2024) and marginal performance gains (Yun et al., 2026; Alzu'bi et al., 2026). This widening gap compels a critical reexamination of when and why multi-agent collaboration offers genuine value over a well-equipped single-agent harness.

Existing work often attributes the primary advantage of multi-agent systems to context isolation, which partitions a task to prevent context pollution and context rot (Huang et al., 2026; Hong et al., 2025). While intuitively appealing, this explanation remains largely qualitative and lacks a rigorous theoretical foundation. In this work, we aim to delineate the capability boundaries of multi-agent collaboration through graph theory. We formalize multi-agent collaboration by modeling task trajectories as dependency graphs, where subtasks are partitioned by bridge edges. This formalism reveals that multi-agent collaboration is not a universal panacea. Instead, it confers systematic benefits specifically in long-horizon tasks with sparse dependencies, where context isolation can be exploited without incurring prohibitive coordination overhead. Conversely, in tightly coupled, sequential workflows, single-agent harnesses remain superior.

Building on these theoretical insights, we propose SAIGE (Semantic-Aware Incremental Graph Evolution), an elegant and general multi-agent collaboration mechanism. SAIGE models collaboration as a dynamically evolving graph, where nodes are agent instances spawned on demand and edges

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

encode semantic dependencies established through content-based information retrieval. Rather than committing to a static topology upfront, SAIGE grows the subtask tree incrementally as execution unfolds, grounding orchestration decisions in actual execution feedback. We empirically validate our framework on long-horizon, complex task benchmarks, showing that SAIGE achieves a favorable trade-off between context efficiency and task performance compared to existing generic multi-agent methods. Notably, our ablation study indicates that neither a larger agent pool nor a deeper recursion hierarchy consistently improves performance, suggesting that more agents do not necessarily make a system more intelligent. Our findings offer a principled understanding of when collaboration provides meaningful advantage over single-agent execution, and suggest that indiscriminate multi-agent scaling warrants reconsideration.

2 Related Work

2.1 Multi-Agent Collaboration

Multi-agent collaboration governs how agents coordinate, communicate, and align behaviors toward shared objectives, evolving from rigid predefined structures to adaptive, context-aware interactions. Role allocation ranges from static, pre-assigned responsibilities (Chen et al., 2025; Fei et al., 2026) to dynamic selection or runtime instantiation based on task demands (Wu et al., 2023; Chen et al., 2023; Alzu'bi et al., 2026). Communication likewise spans explicit message passing via structured documents or natural language (Hong et al., 2024; Wang et al., 2025c; Mou et al., 2025) and implicit inference from environmental cues (Park et al., 2023; Zou et al., 2026). Information flow further structures these interactions as sequential pipelines or parallel exploration (Shen et al., 2023; Wang et al., 2024a), while interaction patterns ultimately manifest as cooperative or competitive dynamics (Yang et al., 2026; Fu et al., 2023).

2.2 Multi-Agent Topology

Multi-agent topology serves as a critical structural framework for governing agent execution, communication, and interactions. While early attempts often embedded structural organization implicitly (Chan et al., 2023; Wu et al., 2023; Khattab et al., 2023), recent practices have explicitly represented multi-agent organizations as graphs (Liu et al., 2024; Zhuge et al., 2024; Qian et al., 2025). These topologies generally fall into three representative paradigms: centralized, distributed, and hybrid. Centralized topologies leverage a global coordinator for information aggregation and routing (Zhang et al., 2026a; Li et al., 2025); distributed topologies employ peer-to-peer communication and localized routing (Yang et al., 2025; Wang et al., 2025a) to enhance autonomy and scalability; and hybrid topologies combine centralized strategic planning with decentralized execution (Qian et al., 2024; Hong et al., 2024; Zhang et al., 2026b). Concretely, these orchestration strategies manifest as distinct structural morphologies, including chain (Qian et al., 2024; Hong et al., 2024; Holt et al., 2025), star (Wu et al., 2023; Yan et al., 2024), tree (Ishibashi & Nishimura, 2024), and general graph topologies (Qian et al., 2025; Yun et al., 2026).

3 Formalizing Multi-Agent Collaboration

What is the core of multi-agent collaboration, and under what idealized conditions does it genuinely help? We take a deliberately idealized route: we ask what multi-agent collaboration is fundamentally doing to a task, and derive its benefit from that answer, leaving the gap between the idealized and the practical as a source of empirically testable predictions rather than a hidden assumption.

3.1 Preliminaries

3.1.1 Trajectory as a Dependency Directed Acyclic Graph

Consider a task trajectory of length T: $\tau = \{x_1, x_2, \ldots, x_T\}$, where each $x_t = (o_t, a_t)$ denotes the interaction tuple at step t, consisting of the new observation o_t received and the action a_t taken by the agent. Let $\mathcal{H}_t = \{x_1, \ldots, x_{t-1}\}$ denote the historical context available before generating x_t. This history grows monotonically with t, and the agent's decisions may depend on arbitrary subsets of this history.

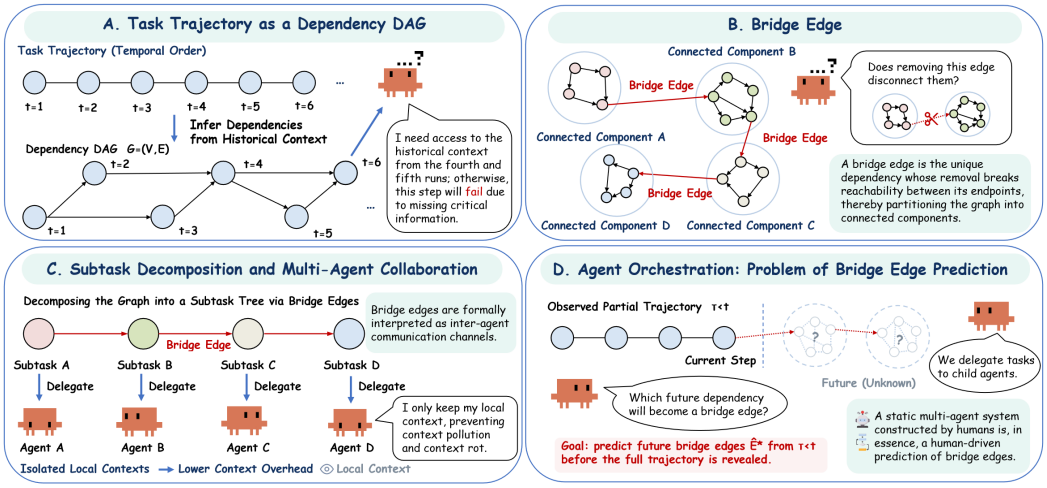

Figure 1: Overview of the theoretical framework for multi-agent collaboration.

To characterize these dependencies formally, we define a value function $Q_t(a_t \mid o_t, \mathcal{H}_t)$ that represents the expected return of taking action a_t given historical context $\mathcal{H}_t$. For any pair of nodes $x_i, x_t \in \tau$ with $i < t$, we say that x_t has a direct dependency on x_i if removing the interaction tuple x_i from the historical context causes a significant drop in the action value at step t:

$$(x_i, x_t) \in \mathcal{E} \iff Q_t(a_t \mid o_t, \mathcal{H}_t \setminus \{x_i\}) < Q_t(a_t \mid o_t, \mathcal{H}_t) - \delta, \tag{1}$$

where $\mathcal{H}_t \setminus \{x_i\}$ denotes the historical context with the contribution of x_i removed, and $\delta > 0$ is a predefined threshold. Since all edges point forward in time, the trajectory naturally induces a directed acyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with node set $\mathcal{V} = \tau$. Each directed edge $(x_i, x_t) \in \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ indicates that node x_t directly depends on node x_i.

We further assume that a task trajectory contains no redundant interactions. Specifically, every node in the trajectory contributes to at least one subsequent decision, implying that each node has out-degree at least one in the dependency graph, with the terminal node serving as the unique sink. Under this assumption, the induced DAG $\mathcal{G}$ is weakly connected. This assumption is mild in practice, as interactions that do not influence any future step can be discarded from the trajectory without affecting the task outcome.

3.1.2 Decomposing Trajectories into Subtasks via Bridge Edges

Within the DAG $\mathcal{G}$, certain edges serve as critical information channels connecting distinct regions of the graph. An edge $(x_i, x_t) \in \mathcal{E}$ is a **bridge edge** if its removal increases the number of weakly connected components of $\mathcal{G}$.

Let $\mathcal{E}^* \subseteq \mathcal{E}$ denote the set of all bridge edges. Removing all bridge edges from $\mathcal{G}$ decomposes the graph into m weakly connected components $\mathcal{G} \setminus \mathcal{E}^* = \mathcal{C}_1 \cup \cdots \cup \mathcal{C}_m$. The corresponding subtask trajectory is defined as the ordered sequence of nodes in $\mathcal{C}_k$: $\tau_{(k)} = \{x_{n_1}, x_{n_2}, \ldots, x_{n_{T_k}}\} \subseteq \tau$, where $T_k = |\mathcal{C}_k|$, and the subtask trajectories collectively partition the original trajectory: $\tau = \tau_{(1)} \cup \tau_{(2)} \cup \cdots \cup \tau_{(m)}$.

In multi-agent systems, this decomposition has a natural interpretation: each subtask $\tau_{(k)}$ is assigned to an individual agent responsible for executing that segment of the trajectory. The bridge edges between components correspond to communication channels between the respective agents, enabling information flow across subtask boundaries when necessary.

Let $\mathcal{V}_\mathcal{S} = \{\tau_{(1)}, \tau_{(2)}, \ldots, \tau_{(m)}\}$ denote the set of subtask trajectories. The directed edges between subtasks are induced by the bridge edges $\mathcal{E}^*$. Since each bridge edge connects nodes from different components, this yields a well-defined relation:

$$(\tau_{(p)}, \tau_{(q)}) \in \mathcal{E}_{\mathcal{S}} \iff \exists (x_p, x_q) \in \mathcal{E}^* \text{ such that } x_p \in \tau_{(p)} \text{ and } x_q \in \tau_{(q)}. \tag{2}$$

The pair $\mathcal{T} = (\mathcal{V}_{\mathcal{S}}, \mathcal{E}_{\mathcal{S}})$ forms a **subtask tree** that captures information flow at the granularity of subtasks.

3.1.3 Context Cost Reduction

Let $c(x)$ denote the non-negative computational overhead of including interaction tuple x in the historical context. For a subtask trajectory $\tau_{(k)} = \{x_{n_1}, \ldots, x_{n_{T_k}}\}$, let $\mathcal{B}_k \subseteq \mathcal{E}^*$ denote the set of bridge edges that enter $\tau_{(k)}$, i.e., $\mathcal{B}_k = \{(x_p, x_q) \in \mathcal{E}^* \mid x_q \in \tau_{(k)}\}$. Let $\mathrm{idx}(x, \tau)$ denote the index of node x within trajectory τ. The total context cost of executing subtask $\tau_{(k)}$ is decomposed as

$$C(\tau_{(k)}) = C_I(\tau_{(k)}) + C_E(\tau_{(k)}) = \sum_{t=1}^{T_k} \sum_{i=1}^{t-1} c(x_{n_i}) + \sum_{(x_p, x_q) \in \mathcal{B}_k} \sum_{i=\mathrm{idx}(x_q, \tau_{(k)})}^{T_k} c(x_p), \tag{3}$$

where $C_I(\tau_{(k)})$ accounts for the cost of accumulating historical context from nodes within the same subtask, and $C_E(\tau_{(k)})$ accounts for the cost of pulling context x_p from predecessor subtasks via incoming bridge edges, starting from the position of the target node x_q through all subsequent steps.

For the original monolithic trajectory τ, there are no incoming bridge edges, so $\mathcal{B} = \varnothing$, and the cost is simply:

$$C(\tau) = \sum_{t=1}^{T} \sum_{i=1}^{t-1} c(x_i). \tag{4}$$

Our key claim is that, under idealized conditions where the complete task trajectory is known in advance, decomposing the trajectory into subtasks reduces the total context cost. The core of the proof is the following lemma, which shows that partitioning a trajectory along a single bridge edge does not increase the total context cost. The detailed proof of Lemma 1 is deferred to Appendix A.

Lemma 1. *For any trajectory $\tau_{(S)} = \tau_{(L)} \cup \tau_{(R)}$ partitioned into two subtask trajectories by a single bridge edge (x_L, x_R) with $x_L \in \tau_{(L)}$ and $x_R \in \tau_{(R)}$, the sum of the context costs of the two subtask trajectories is no greater than the cost of the original trajectory:*

$$C(\tau_{(L)}) + C(\tau_{(R)}) \leq C(\tau_{(S)}).$$

Recursively applying Lemma 1 to $\tau_{(L)}$ and $\tau_{(R)}$ yields:

$$C(\tau) \geq C(\tau_{(L)}) + C(\tau_{(R)}) \geq \cdots \geq \sum_{k=1}^{m} C(\tau_{(k)}).$$

This establishes that, when the task trajectory is known and the decomposition is performed correctly, multi-agent decomposition reduces context cost. It should be noted that this analysis does not account for context compression or other mitigation strategies that may alter the cost structure.

3.1.4 Orchestration as Bridge Edge Prediction

The decomposition above is **post-hoc**: it presumes complete knowledge of the entire trajectory τ and its dependency structure. In practice, the trajectory unfolds sequentially, and the DAG $\mathcal{G}$ is not known in advance. The objective of multi-agent orchestration is therefore to predict the bridge edges $\mathcal{E}^*$ before or during execution, without access to the complete trajectory.

Let $\mathcal{E}^*$ denote the true set of bridge edges induced by the full trajectory, and let $\hat{\mathcal{E}}_t$ be the set of bridge edges predicted by the orchestration strategy at time t based on the observed prefix $\tau_{<t}$. Since the true bridge edges $\mathcal{E}^*$ are only revealed after the entire trajectory is complete, the prediction $\hat{\mathcal{E}}_t$ concerns edges that lie in the future, i.e., edges $e_{i \to j}$ with $j > t$. The orchestration problem is to minimize the expected discrepancy between the true and predicted bridge edge sets:

$$\min_{\hat{\mathcal{E}}_t} \mathbb{E}\left[\mathcal{L}(\mathcal{E}^*, \hat{\mathcal{E}}_t) \mid \tau_{<t}\right], \tag{5}$$

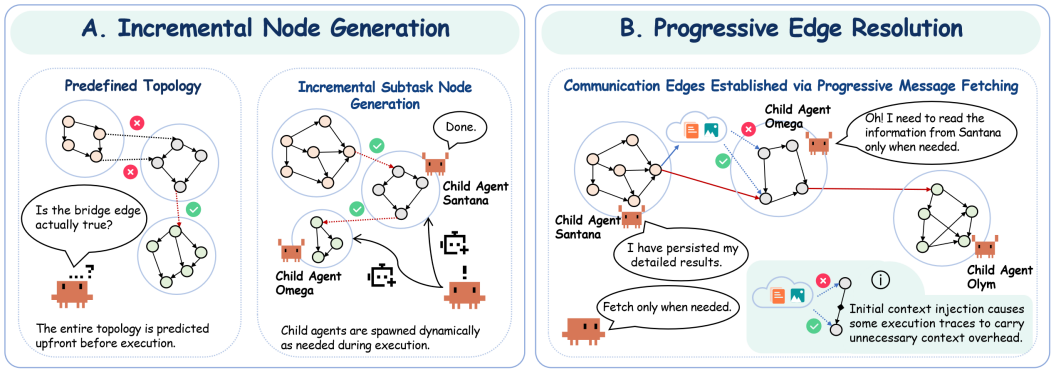

Figure 2: SAIGE: incremental node generation via progressive subtask decomposition, and progressive edge resolution via on-demand message fetching.

where $\mathcal{L}$ is a loss function that measures the dissimilarity between two sets of directed edges over $\mathcal{V}$. A correct prediction partitions the trajectory into the true subtasks; an incorrect prediction results in suboptimal decomposition and diminished context cost reduction.

3.2 Discussion

Our theoretical and empirical results jointly characterize when multi-agent collaboration helps and why its benefits are bounded.

When collaboration pays off. Multi-agent collaboration is effective precisely when a task admits relatively independent subtasks, so that each agent maintains a focused context window and excludes task-irrelevant information. A single-agent system, by contrast, must accumulate observations indiscriminately, degrading the signal-to-noise ratio and increasing the risk of erroneous execution. This benefit is contingent on the accuracy of bridge-edge prediction: a mispredicted edge can trap an agent in a local deadlock while awaiting a dependency that never materializes, or inflate communication overhead as agents recover from an erroneous partition.

The cost of isolation. Context isolation localizes each agent's working memory to its subtask, containing failures within the corresponding scope rather than letting them contaminate the global context. This mitigates the incomplete workflows and premature termination that plague single-agent execution on long-horizon tasks, at the cost of higher aggregate token consumption, since decomposition extends the overall workflow.

Why scaling does not help. Because each bridge edge is predicted from an observed prefix, prediction errors accumulate as the number of subtasks grows, while additional agents incur higher coordination costs in communication, synchronization, and conflict resolution. The marginal benefit of adding agents therefore diminishes, and may eventually be offset by accumulated errors and rising coordination overhead. This explains why more agents do not necessarily make a system more intelligent, and motivates a mechanism that grows its collaboration graph only where the task structure permits.

4 Methodology

We propose Semantic-Aware Incremental Graph Evolution (SAIGE), a lightweight mechanism for decomposing tasks under trajectory uncertainty. Rather than predicting all bridge edges in advance, SAIGE evolves the subtask tree incrementally during execution, growing the structure only when semantic dependencies are revealed by execution feedback.

4.1 Incremental Node Generation

Bridge edges are inherently difficult to predict reliably at the outset, and premature commitment to a static topology risks compounding early mispredictions. Let $\hat{\mathcal{E}}_{t-1}$ and $\hat{\mathcal{E}}_t$ denote the predicted bridge-edge sets at consecutive steps, with $\mathcal{E}^*$ the ground truth. Since SAIGE predicts incrementally, we have $\hat{\mathcal{E}}_{t-1} \subseteq \hat{\mathcal{E}}_t$, and

$$P(\hat{\mathcal{E}}_t \subseteq \mathcal{E}^* \mid \hat{\mathcal{E}}_{t-1} \subseteq \mathcal{E}^*, \tau_{<t}) = \frac{P(\hat{\mathcal{E}}_t \subseteq \mathcal{E}^* \mid \tau_{<t})}{P(\hat{\mathcal{E}}_{t-1} \subseteq \mathcal{E}^* \mid \tau_{<t})} \geq P(\hat{\mathcal{E}}_t \subseteq \mathcal{E}^* \mid \tau_{<t}), \tag{6}$$

which indicates that deferring edge prediction until execution evidence accumulates is more reliable than committing to a topology upfront.

Accordingly, SAIGE spawns subtask nodes on demand rather than pre-allocating them. A running agent delegates a child agent only when it detects a dependency boundary that satisfies the bridge-edge property, that is, a single-step dependency across which the subproblem can be executed independently given a bounded context summary. When such a boundary is present, the subproblem is delegated and the child agent executes autonomously with a self-contained instruction. When no such boundary is present, no child is spawned and the parent agent continues to execute the task monolithically. This conditional delegation is what allows SAIGE to avoid the decomposition penalty on densely coupled tasks.

4.2 Progressive Edge Resolution

Edges between subtask nodes are established through progressive message fetching. Each agent writes its result upon completion and retrieves dependencies only when needed. Let $\mathcal{B}_k$ denote the set of bridge edges entering subtask $\tau_{(k)}$, and let $\mathrm{idx}(x_q, \tau_{(k)})$ denote the step at which dependency on x_p becomes active. Progressive fetching incurs cost

$$\sum_{(x_p,x_q)\in\mathcal{B}_k} \sum_{i=\mathrm{idx}(x_q,\tau_{(k)})}^{T_k} c(x_p) \leq \sum_{(x_p,x_q)\in\mathcal{B}_k} \sum_{i=1}^{T_k} c(x_p), \tag{7}$$

where the right-hand side loads all dependencies at the start. Sparse on-demand retrieval thus dominates push-based communication in context cost. More importantly, progressive fetching establishes communication channels dynamically rather than fixing them at decomposition time, allowing the collaboration graph to adapt to dependencies not fully determined in advance. This converts global topology prediction into localized retrieval decisions grounded in execution feedback.

5 Experiments

5.1 Experimental Setup

5.1.1 Benchmarks

We evaluate on four benchmarks designed to assess agent capabilities on complex, long-horizon, realistic tasks. **Terminal Bench 2.1** (Merrill et al., 2026) tests agents on hard, realistic command-line interface tasks, with resolution rate as the primary metric. **NL2Repo Bench** (Ding et al., 2026) evaluates long-horizon repository generation from a single natural-language requirements document, using the average test pass rate across all tasks as the primary metric. **Deep Research Bench II** (Li et al., 2026) diagnoses deep research agents through rubrics derived from expert reports, scored as the fraction of rubrics passed. **AgentIF-OneDay** (Chen et al., 2026) is a task-level instruction-following benchmark for general AI agents in daily scenarios, employing instance-level, rubric-based scoring with binary evaluation, separated bonus/penalty items, and file-content alignment. Detailed descriptions of all benchmarks are provided in Appendix B.1.

5.1.2 Baselines

Single-Agent executes the full task within a single agent context and serves as the monolithic baseline. It runs on the native **Codex CLI** harness (OpenAI, 2025b). **Task Decomposition and Agent**

Table 1: Main results across four long-horizon agent benchmarks. For each benchmark, we report the primary metric and token usage. The best result for each metric is underlined.

	Terminal Bench 2.1	NL2Repo	Deep Research Bench II	AgentIF-OneDay
Method	**Resolution Rate ↑**	**Pass Rate ↑**	**Total Score ↑**	**Total Score ↑**
Single-Agent	76.71	48.74	51.49	53.57
TDAG	71.49	47.37	54.43	59.94
DynTaskMAS	68.27	44.15	53.44	65.85
GoA	67.47	41.04	48.40	52.67
SAIGE (Ours)	75.50	49.23	55.23	71.80
Token Usage (Input / Output) ↓				
Single-Agent	475M / 8M	1,408M / 10M	2,805M / 7M	189M / 2M
TDAG	542M / 13M	1,562M / 15M	3,895M / 23M	218M / 3M
DynTaskMAS	533M / 12M	1,386M / 15M	3,240M / 21M	233M / 4M
GoA	621M / 16M	1,380M / 14M	3,816M / 23M	222M / 3M
SAIGE (Ours)	501M / 10M	1,282M / 11M	2,745M / 20M	185M / 3M

Generation (TDAG) (Wang et al., 2025b) is a multi-agent framework that decomposes complex tasks into smaller subtasks, assigns each subtask to a specifically generated subagent, and continuously summarizes successful execution patterns into reusable skills for future reference. **DynTaskMAS** (Yu et al., 2025) is a dynamic task graph-driven framework for asynchronous and parallel LLM-based multi-agent systems. **Graph-of-Agents (GoA)** (Yun et al., 2026) models multi-agent LLM communication through a graph-based framework, sampling relevant agents, constructing edges by evaluating response relevance, and aggregating responses via graph-based pooling. Further details on baseline implementations are provided in Appendix B.2.

5.1.3 Implementation

To ensure harness consistency, all methods are instantiated on the Codex CLI harness. For multi-agent methods, token usage is aggregated across all agents. All baselines and our approach are built upon **DeepSeek-V4-Pro** (DeepSeek, 2026c) as the foundation LLM. For all multi-agent methods evaluated in this work, we impose a uniform constraint on agent hierarchy: the **maximum recursion depth is set to 1**, meaning that spawned sub-agents are not permitted to recursively create further sub-agents. Additionally, the **maximum number of agents** is capped at 4. These constraints are imposed uniformly across all multi-agent methods to ensure system stability and prevent unbounded agent proliferation during execution. We examine the effect of relaxing these constraints in the ablation study (Section 5.4) and further verify it on the multi-agent baselines in Appendix B.3.

5.2 Main Results

Table 1 presents the main results across all four benchmarks, reporting the primary metric along with input and output token counts for each benchmark.

The four benchmarks differ in the dependency structure of their execution trajectories. Terminal Bench 2.1 and NL2Repo Bench exhibit dense inter-step dependencies: tool invocations, file reads, and file writes form long upstream-downstream chains in which later steps must condition on earlier outputs, leaving little room for independent execution. Deep Research Bench II and AgentIF-OneDay, by contrast, exhibit sparse dependencies, where large portions of the trajectory can proceed without conditioning on one another. This distinction provides a principled basis for the two regimes of multi-agent behavior observed below, with step-level evidence in Appendix C.

On the densely coupled benchmarks, generic multi-agent methods fall below the single-agent baseline, suggesting that decomposition fragments the dependency chain and that coordination overhead can outweigh the benefit of context isolation. SAIGE does not exhibit the same degradation: it remains comparable to the single-agent baseline on Terminal Bench 2.1 and on NL2Repo Bench, while using fewer tokens than the other multi-agent methods. On the sparsely dependent benchmarks, decomposition appears more beneficial, and SAIGE performs competitively on both, with the most visible margin on AgentIF-OneDay, without the token inflation observed in the other multi-agent

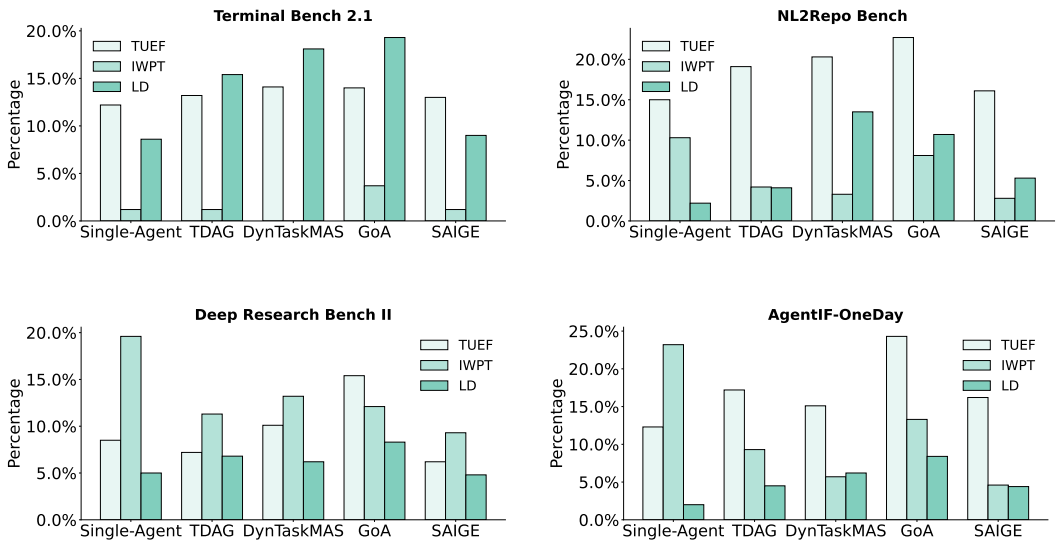

Figure 3: Failure taxonomy across four benchmarks.

variants. GoA, by contrast, employs a more elaborate orchestration design yet yields weaker results on these benchmarks, which suggests that added coordination complexity does not by itself translate into gains when it is not matched to the task's dependency structure.

Taken together, these results are consistent with our central claim that the benefit of multi-agent collaboration is bounded by task structure rather than universal. SAIGE does not dominate the single-agent baseline on every benchmark, and its advantage is most evident on sparsely dependent tasks. Across both regimes, however, it appears to attain a favorable trade-off: it avoids the decomposition penalty on tightly coupled tasks, exploits context isolation on decomposable ones, and grows its collaboration graph in response to execution feedback rather than a pre-committed topology.

5.3 Failure Pattern Analysis

To understand how multi-agent collaboration affects execution, we collect trajectories whose evaluation scores are significantly below the benchmark average and use DeepSeek-V4-Pro to semantically analyze their execution traces, yielding three recurring failure categories: **Task Understanding and Execution Failure (TUEF)**, **Incomplete Workflow and Premature Termination (IWPT)**, and **Local Deadlock (LD)**. Figure 3 summarizes their distribution across the four benchmarks.

On the tightly coupled benchmarks, multi-agent methods are more prone to Local Deadlock: coordination introduces waiting, cyclic dependencies, and resource conflicts that prevent global progress. On the decomposable benchmarks, by contrast, single-agent methods more frequently exhibit Incomplete Workflow and Premature Termination, as the accumulation of context progressively degrades their ability to track the full tool chain, leading them to stop before the workflow is validated or the resolution criterion is satisfied. These two tendencies together clarify why multi-agent collaboration is beneficial only when the task structure admits effective decomposition.

5.4 Ablation Study

We conduct ablation experiments on Deep Research Bench II and AgentIF-OneDay to examine the sensitivity of SAIGE to agent count, recursion depth, and progressive message fetching. Tables 2 and 3 report the primary benchmark score and aggregated token usage across all agents for each configuration. For each recursion depth setting, we vary the maximum number of agents to isolate their joint effects, with "w/o Message Fetching" denoting SAIGE with progressive message fetching disabled under the default recursion depth ($D_{\max} = 1$).

Table 2: Ablation results on Deep Research Bench II. The default configuration is underlined.

Configuration	Max Recursion Depth $D_{\max}=1$	$D_{\max}=2$	$D_{\max}=3$	w/o Message Fetching
Total Score ↑				
$N_{\max}=4$	<u>55.23</u>	54.88	52.21	53.54
$N_{\max}=5$	52.94	52.07	50.89	53.85
$N_{\max}=6$	53.74	48.94	49.07	51.22
Token Usage (Input / Output) ↓				
$N_{\max}=4$	<u>2,745M / 20M</u>	3,102M / 19M	2,993M / 21M	4,027M / 24M
$N_{\max}=5$	3,913M / 25M	4,225M / 24M	4,502M / 25M	5,928M / 26M
$N_{\max}=6$	6,312M / 32M	6,782M / 34M	6,590M / 31M	6,951M / 34M

Table 3: Ablation results on AgentIF-OneDay. The default configuration is underlined.

Configuration	Max Recursion Depth $D_{\max}=1$	$D_{\max}=2$	$D_{\max}=3$	w/o Message Fetching
Total Score ↑				
$N_{\max}=4$	<u>71.80</u>	66.39	63.17	66.22
$N_{\max}=5$	67.32	58.33	58.76	59.15
$N_{\max}=6$	64.02	60.16	56.75	53.50
Token Usage (Input / Output) ↓				
$N_{\max}=4$	<u>185M / 3M</u>	261M / 4M	272M / 4M	237M / 3M
$N_{\max}=5$	239M / 3M	272M / 4M	270M / 3M	268M / 3M
$N_{\max}=6$	352M / 4M	360M / 4M	371M / 4M	366M / 4M

Across the tested configurations, increasing the agent budget does not consistently improve task performance. On both benchmarks, expanding the agent pool beyond the default setting tends to yield lower scores and higher token consumption, with individual configurations deviating from this trend. Deepening the recursion level shows a similar pattern: allowing sub-agents to spawn further sub-agents tends to reduce performance and increase token usage. Disabling progressive message fetching is also associated with a comparable decline, most notably on AgentIF-OneDay. Token usage generally increases with agent count and recursion depth, while task performance does not show a corresponding gain. These results suggest that more agents do not necessarily make a system more intelligent: beyond a certain point, the marginal agent appears to add coordination cost with limited additional problem-solving capacity.

6 Conclusion

In this work, we reexamine the value of multi-agent collaboration in the era of increasingly capable single-agent harnesses. Through a graph-theoretic formalization of task trajectories, we show that multi-agent collaboration is not universally beneficial, but confers systematic advantages specifically on long-horizon tasks with sparse dependencies, where context isolation can be exploited without incurring prohibitive coordination overhead. Building on this insight, we propose SAIGE, a multi-agent collaboration mechanism that models collaboration as a dynamically evolving graph and establishes semantic dependencies through content-based information retrieval. Experiments on long-horizon benchmarks show that SAIGE achieves a favorable trade-off between context efficiency and task performance. These findings suggest that multi-agent superiority is bounded by task structure rather than universal, and that indiscriminate scaling of agents warrants reconsideration.

AI USE STATEMENT

In this work, we used generative AI tools for language polishing and section drafting, with all AI-assisted text reviewed and revised by the authors. We did not use generative AI tools for generating research ideas, designing the methodology, or producing the core theoretical contributions. The experimental pipeline is built on LLM APIs as foundation models for the evaluated methods and baselines, and all API-based components were configured, executed, and verified by the authors. We have reviewed all LLM-generated outputs for factual accuracy and consistency with our results, verified the experimental outputs against the benchmark evaluation protocols, and take responsibility for the final content of this work.

REFERENCES

Salaheddin Alzu'bi, Baran Nama, Arda Kaz, Anushri Eswaran, Weiyuan Chen, Sarvesh Khetan, Rishab Bala, Tu Vu, and Sewoong Oh. Roma: Recursive open meta-agent framework for long-horizon multi-agent systems, 2026. URL https://arxiv.org/abs/2602.01848.

Anthropic. Claude code. https://claude.com/product/claude-code, 2025. Accessed: 2026-09-06.

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. Chateval: Towards better llm-based evaluators through multi-agent debate, 2023. URL https://arxiv.org/abs/2308.07201.

Kaiyuan Chen, Qimin Wu, Taiyu Hou, Tianhao Tang, Xueyu Hu, Yuchen Hou, Bikun Li, Chengming Qian, Guoyin Wang, Haolin Chen, Haotong Tian, Haoye Zhang, Haoyu Bian, Hongbing Pan, Hongkang Zhang, Hongyi Zhou, Jiaqi Cai, Jiewu Rao, Jiyuan Ren, Keduan Huang, Lucia Zhu Huang, Mingyu Yuan, Naixu Guo, Qicheng Tang, Qinyan Zhang, Shuai Chen, Siheng Chen, Ting Ting Li, Xiaoxing Guo, Yaocheng Zuo, Yaoqi Guo, Yinan Wang, Yinzhou Yu, Yize Wang, Yuan Jiang, Yuan Tian, Yuanshuo Zhang, Yuxuan Liu, Yvette Yan Zeng, Zenyu Shan, Zihan Yin, Xiaobo Hu, Yang Liu, Yixin Ren, and Yuan Gong. Agentif-oneday: A task-level instruction-following benchmark for general ai agents in daily scenarios, 2026. URL https://arxiv.org/abs/2601.20613.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors, 2023. URL https://arxiv.org/abs/2308.10848.

Yibin Chen, Yifu Yuan, Zeyu Zhang, Yan Zheng, Jinyi Liu, Fei Ni, Jianye Hao, Hangyu Mao, and Fuzheng Zhang. Sheetagent: Towards a generalist agent for spreadsheet reasoning and manipulation via large language models, 2025. URL https://arxiv.org/abs/2403.03636.

DeepSeek. Deepseek harness. https://www.deepseek.com/harness, 2026a. Accessed: 2026-09-06.

DeepSeek. Deepseek-v4-flash-vision-exp release: Multimodal API now live. https://api-docs.deepseek.com/news/news260821, August 2026b. Accessed: 2026-09-06.

DeepSeek. Deepseek-v4-pro GA release. https://api-docs.deepseek.com/news/news260813/, August 2026c. Accessed: 2026-09-06.

Jingzhe Ding, Shengda Long, Changxin Pu, Huan Zhou, Hongwan Gao, Xiang Gao, Chao He, Yue Hou, Fei Hu, Zhaojian Li, Weiran Shi, Zaiyuan Wang, Daoguang Zan, Chenchen Zhang, Xiaoxu Zhang, Qizhi Chen, Xianfu Cheng, Bo Deng, Qingshui Gu, Kai Hua, Juntao Lin, Pai Liu, Mingchen Li, Xuanguang Pan, Zifan Peng, Yujia Qin, Yong Shan, Zhewen Tan, Weihao Xie, Zihan Wang, Yishuo Yuan, Jiayu Zhang, Enduo Zhao, Yunfei Zhao, He Zhu, Liya Zhu, Chenyang Zou, Ming Ding, Jianpeng Jiao, Jiaheng Liu, Minghao Liu, Qian Liu, Chongyang Tao, Jian Yang, Tong Yang, Zhaoxiang Zhang, Xinjie Chen, Wenhao Huang, and Ge Zhang. Nl2repo-bench: Towards long-horizon repository generation evaluation of coding agents, 2026. URL https://arxiv.org/abs/2512.12730.

Tianxiang Fei, Cheng Chen, Yue Pan, Mao Zheng, and Mingyang Song. Codedelegator: Mitigating context pollution via role separation in code-as-action agents, 2026. URL https://arxiv.org/abs/2601.14914.

Yao Fu, Hao Peng, Tushar Khot, and Mirella Lapata. Improving language model negotiation with self-play and in-context learning from ai feedback, 2023. URL https://arxiv.org/abs/2305.10142.

Samuel Holt, Max Ruiz Luyten, and Mihaela van der Schaar. L2mac: Large language model automatic computer for extensive code generation, 2025. URL https://arxiv.org/abs/2310.02003.

Kelly Hong, Anton Troynikov, and Jeff Huber. Context rot: How increasing input tokens impacts llm performance. Technical report, Chroma, July 2025. URL https://trychroma.com/research/context-rot. Accessed: 2026-09-06.

Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, 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, 2024. URL https://arxiv.org/abs/2308.00352.

Jenny Y. Huang, Leshem Choshen, Wei Sun, Omar Khattab, Ramón Fernandez Astudillo, Mehul Damani, Tamara Broderick, and Jacob Andreas. Do llms benefit from their own words?, 2026. URL https://arxiv.org/abs/2602.24287.

Yoichi Ishibashi and Yoshimasa Nishimura. Self-organized agents: A llm multi-agent framework toward ultra large-scale code generation and optimization, 2024. URL https://arxiv.org/abs/2404.02183.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. Dspy: Compiling declarative language model calls into self-improving pipelines, 2023. URL https://arxiv.org/abs/2310.03714.

Ao Li, Yuexiang Xie, Songze Li, Fugee Tsung, Bolin Ding, and Yaliang Li. Agent-oriented planning in multi-agent systems, 2025. URL https://arxiv.org/abs/2410.02189.

Ruizhe Li, Mingxuan Du, Benfeng Xu, Chiwei Zhu, Xiaorui Wang, and Zhendong Mao. Deepresearch bench ii: Diagnosing deep research agents via rubrics from expert reports, 2026. URL https://arxiv.org/abs/2601.08536.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. A dynamic llm-powered agent network for task-oriented agent collaboration, 2024. URL https://arxiv.org/abs/2310.02170.

Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E. Kelly Buchanan, Junhong Shen, Guanghao Ye, Haowei Lin, Jason Poulos, Maoyu Wang, Marianna Nezhurina, Jenia Jitsev, Di Lu, Orfeas Menis Mastromichalakis, Zhiwei Xu, Zizhao Chen, Yue Liu, Robert Zhang, Leon Liangyu Chen, Anurag Kashyap, Jan-Lucas Uslu, Jeffrey Li, Jianbo Wu, Minghao Yan, Song Bian, Vedang Sharma, Ke Sun, Steven Dillmann, Akshay Anand, Andrew Lanpouthakoun, Bardia Koopah, Changran Hu, Etash Guha, Gabriel H. S. Dreiman, Jiacheng Zhu, Karl Krauth, Li Zhong, Niklas Muennighoff, Robert Amanfu, Shangyin Tan, Shreyas Pimpalgaonkar, Tushar Aggarwal, Xiangning Lin, Xin Lan, Xuandong Zhao, Yiqing Liang, Yuanli Wang, Zilong Wang, Changzhi Zhou, David Heineman, Hange Liu, Harsh Trivedi, John Yang, Junhong Lin, Manish Shetty, Michael Yang, Nabil Omi, Negin Raoof, Shanda Li, Terry Yue Zhuo, Wuwei Lin, Yiwei Dai, Yuxin Wang, Wenhao Chai, Shang Zhou, Dariush Wahdany, Ziyu She, Jiaming Hu, Zhikang Dong, Yuxuan Zhu, Sasha Cui, Ahson Saiyed, Arinbjörn Kolbeinsson, Jesse Hu, Christopher Michael Rytting, Ryan Marten, Yixin Wang, Alex Dimakis, Andy Konwinski, and Ludwig Schmidt. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces, 2026. URL https://arxiv.org/abs/2601.11868.

Xinyi Mou, Chen Qian, Wei Liu, Xuanjing Huang, and Zhongyu Wei. Ecolang: Efficient and effective agent communication language induction for social simulation, 2025. URL https://arxiv.org/abs/2505.06904.

OpenAI. Codex. https://openai.com/codex, 2025a. Accessed: 2026-09-06.

OpenAI. Introducing upgrades to codex. https://openai.com/index/introducing-upgrades-to-codex/, September 2025b. Accessed: 2026-09-06.

Samuel J. Paech. Eq-bench: An emotional intelligence benchmark for large language models, 2024. URL https://arxiv.org/abs/2312.06281.

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, 2023. URL https://arxiv.org/abs/2304.03442.

Qiwei Peng, Yekun Chai, and Xuhong Li. Humaneval-xl: A multilingual code generation benchmark for cross-lingual natural language generalization, 2024. URL https://arxiv.org/abs/2402.16694.

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, 2024. URL https://arxiv.org/abs/2307.07924.

Chen Qian, Zihao Xie, YiFei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Scaling large language model-based multi-agent collaboration, 2025. URL https://arxiv.org/abs/2406.07155.

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, 2023. URL https://arxiv.org/abs/2303.17580.

Ji Wang, Kashing Chen, Xinyuan Song, Ke Zhang, Lynn Ai, Eric Yang, and Bill Shi. Symphony: A decentralized multi-agent framework for scalable collective intelligence, 2025a. URL https://arxiv.org/abs/2508.20019.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities, 2024a. URL https://arxiv.org/abs/2406.04692.

Yaoxiang Wang, Zhiyong Wu, Junfeng Yao, and Jinsong Su. Tdag: A multi-agent framework based on dynamic task decomposition and agent generation, 2025b. URL https://arxiv.org/abs/2402.10178.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark, 2024b. URL https://arxiv.org/abs/2406.01574.

Zhao Wang, Sota Moriyama, Wei-Yao Wang, Briti Gangopadhyay, and Shingo Takamatsu. Talk structurally, act hierarchically: A collaborative framework for llm multi-agent systems, 2025c. URL https://arxiv.org/abs/2502.11098.

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 conversation, 2023. URL https://arxiv.org/abs/2308.08155.

Yikuan Yan, Yaolun Zhang, and Keman Huang. Depending on yourself when you should: Mentoring llm with rl agents to become the master in cybersecurity games, 2024. URL https://arxiv.org/abs/2403.17674.

Bufang Yang, Lilin Xu, Liekang Zeng, Yunqi Guo, Siyang Jiang, Wenrui Lu, Kaiwei Liu, Yixuan Li, Xiaofan Jiang, Guoliang Xing, and Zhenyu Yan. Proagent: Harnessing on-demand sensory contexts for proactive llm agent systems in the wild, 2026. URL https://arxiv.org/abs/2512.06721.

Yingxuan Yang, Huacan Chai, Shuai Shao, Yuanyi Song, Siyuan Qi, Renting Rui, and Weinan Zhang. Agentnet: Decentralized evolutionary coordination for llm-based multi-agent systems, 2025. URL https://arxiv.org/abs/2504.00587.

Junwei Yu, Yepeng Ding, and Hiroyuki Sato. Dyntaskmas: A dynamic task graph-driven framework for asynchronous and parallel llm-based multi-agent systems. *Proceedings of the International Conference on Automated Planning and Scheduling*, 35(1):288–296, 2025. ISSN 2334-0835. doi: 10.1609/icaps.v35i1.36130. URL http://dx.doi.org/10.1609/icaps.v35i1.36130.

Sukwon Yun, Jie Peng, Pingzhi Li, Wendong Fan, Jie Chen, James Zou, Guohao Li, and Tianlong Chen. Graph-of-agents: A graph-based framework for multi-agent llm collaboration, 2026. URL https://arxiv.org/abs/2604.17148.

Ruizhe Zhang, Xinke Jiang, Zhibang Yang, Zhixin Zhang, Jiaran Gao, Yuzhen Xiao, Tao Feng, Yue Fang, Yuxuan Liu, Ruiqing Li, Hongbin Lai, Huheng Huang, Xu Chu, Junfeng Zhao, and Yasha Wang. Stackplanner: A centralized hierarchical multi-agent system with task-experience memory management, 2026a. URL https://arxiv.org/abs/2601.05890.

Wentao Zhang, Liang Zeng, Yuzhen Xiao, Yongcong Li, Ce Cui, Yilei Zhao, Rui Hu, Yang Liu, Yahui Zhou, and Bo An. Agentorchestra: Orchestrating multi-agent intelligence with the tool-environment-agent(tea) protocol, 2026b. URL https://arxiv.org/abs/2506.12508.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. Language agents as optimizable graphs, 2024. URL https://arxiv.org/abs/2402.16823.

Jiaru Zou, Ruizhong Qiu, Gaotang Li, Xiyuan Yang, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, James Zou, Mengdi Wang, and Ling Yang. Latent collaboration in multi-agent systems, 2026. URL https://arxiv.org/abs/2511.20639.

A PROOFS

Theorem 1. *Let $\tau = \{x_1, x_2, \ldots, x_T\}$ be a task trajectory with no redundant interactions, and let $\mathcal{E}$ denote the set of directed edges defined by the dependency criterion. Then the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V} = \tau$ is a weakly connected directed acyclic graph.*

Proof. For any edge $(x_i, x_t) \in \mathcal{E}$, the dependency criterion requires $i < t$, since the value function Q_t is evaluated with respect to the historical context $\mathcal{H}_t = \{x_1, \ldots, x_{t-1}\}$. Thus every edge is oriented from a lower-index vertex to a higher-index vertex.

Define a mapping $\phi : \mathcal{V} \to \mathbb{N}$ by $\phi(x_i) = i$. For every edge $(x_i, x_t) \in \mathcal{E}$, we have

$$\phi(x_i) = i < t = \phi(x_t).$$

Therefore ϕ is a strict order-preserving labeling of the vertices, which is a well-known sufficient condition for acyclicity. Equivalently, any directed path

$$x_{i_1} \to x_{i_2} \to \cdots \to x_{i_\ell}$$

must satisfy

$$i_1 < i_2 < \cdots < i_\ell,$$

precluding the existence of a directed cycle. Hence $\mathcal{G}$ is a directed acyclic graph.

We next establish weak connectivity. By assumption, every node in τ contributes to at least one subsequent decision, implying that every vertex x_i with $i < T$ has out-degree at least one, and the terminal vertex x_T is the unique sink. Hence, for every vertex x_i with $i < T$, there exists a directed path from x_i to x_T: starting from x_i, each step follows an outgoing edge (x_i, x_j) with $j > i$, and since indices strictly increase along edges and the trajectory is finite, the path must terminate at x_T.

Now consider any two vertices $x_i, x_j \in \mathcal{V}$. If $i = j$ the claim is trivial. Otherwise, there exists a directed path P_i from x_i to x_T and a directed path P_j from x_j to x_T. Reversing the orientation of P_j yields an undirected path from x_T to x_j. Concatenating P_i with the reversed P_j gives an undirected walk from x_i to x_j in the underlying undirected graph of $\mathcal{G}$. Therefore any two vertices are connected in the underlying undirected graph, and $\mathcal{G}$ is weakly connected. □

Theorem 2. *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a weakly connected directed acyclic graph, and let $\mathcal{E}^* \subseteq \mathcal{E}$ be the set of bridge edges. Define $\mathcal{V}_\mathcal{S}$ as the set of weakly connected components of $\mathcal{G} \setminus \mathcal{E}^*$, and let*

$$\mathcal{E}_\mathcal{S} = \{(\mathcal{C}_p, \mathcal{C}_q) \mid \exists (u, v) \in \mathcal{E}^* \text{ with } u \in \mathcal{C}_p, v \in \mathcal{C}_q\}.$$

Then $\mathcal{T} = (\mathcal{V}_\mathcal{S}, \mathcal{E}_\mathcal{S})$ is a tree.

Proof. We first prove that $\mathcal{T}$ is connected. Since $\mathcal{G}$ is weakly connected, its underlying undirected graph is connected. Contracting each weakly connected component $\mathcal{C}_i \in \mathcal{V}_\mathcal{S}$ to a single vertex preserves weak connectivity of the quotient graph. The edge set of this quotient graph is precisely $\mathcal{E}_\mathcal{S}$, as all edges between distinct components are bridge edges. Hence $\mathcal{T}$ is weakly connected.

We next prove that $\mathcal{T}$ is acyclic. Suppose, for contradiction, that $\mathcal{T}$ contains an undirected cycle $C_\mathcal{S}$ of length $\ell \geq 2$, with vertices

$$\mathcal{C}_{i_1}, \mathcal{C}_{i_2}, \ldots, \mathcal{C}_{i_\ell}$$

and edges

$$e_1, e_2, \ldots, e_\ell \in \mathcal{E}_\mathcal{S},$$

where $e_j = (\mathcal{C}_{i_j}, \mathcal{C}_{i_{j+1}})$ with indices taken modulo ℓ.

By the definition of $\mathcal{E}_\mathcal{S}$, each e_j corresponds to a bridge edge $(u_j, v_j) \in \mathcal{E}^*$ with

$$u_j \in \mathcal{C}_{i_j} \quad \text{and} \quad v_j \in \mathcal{C}_{i_{j+1}}.$$

Since each $\mathcal{C}_{i_j}$ is weakly connected, there exists an undirected path P_j within $\mathcal{C}_{i_{j+1}}$ connecting v_j to u_{j+1}, with indices modulo ℓ.

Concatenating the bridge edge (u_j, v_j) with the path P_j for each j yields an undirected cycle in $\mathcal{G}$ that contains the bridge edge (u_1, v_1) but does not traverse it twice. The removal of (u_1, v_1) therefore does not disconnect u_1 from v_1, as the alternative route through

$$P_1, e_2, P_2, \ldots, e_\ell$$

remains. This contradicts the defining property of a bridge edge, namely that its removal destroys all paths between its endpoints. Therefore $\mathcal{T}$ contains no undirected cycle.

A weakly connected graph with no undirected cycles is a tree, completing the proof. □

Lemma 1. *For any trajectory $\tau_{(S)} = \tau_{(L)} \cup \tau_{(R)}$ partitioned into two subtask trajectories by a single bridge edge (x_L, x_R) with $x_L \in \tau_{(L)}$ and $x_R \in \tau_{(R)}$, the sum of the context costs of the two subtask trajectories is no greater than the cost of the original trajectory:*

$$C(\tau_{(L)}) + C(\tau_{(R)}) \leq C(\tau_{(S)}).$$

Proof. We proceed by analyzing the internal and external context costs separately, then combine the two bounds to establish the result.

Setup and notation. Both subsequences preserve the original temporal order, so every node in $\tau_{(L)}$ precedes every node in $\tau_{(R)}$. For any trajectory τ and node $x_i \in \tau$, let $\mathrm{idx}(x_i, \tau)$ denote the position of x_i in τ.

For any trajectory τ, the internal context cost can be rewritten as

$$C_I(\tau) = \sum_{x_i \in \tau} c(x_i) \cdot \big|\{x_t \in \tau : \mathrm{idx}(x_i, \tau) < \mathrm{idx}(x_t, \tau)\}\big|,$$

which charges each node x_i for every subsequent node in the same trajectory that must retain x_i in its historical context.

For any trajectory τ with incoming bridge edge set $\mathcal{B}(\tau) \subseteq \mathcal{E}^*$, the external context cost can be rewritten as

$$C_E(\tau) = \sum_{(x_p, x_q) \in \mathcal{B}(\tau)} c(x_p) \cdot \big|\{x_t \in \tau : \mathrm{idx}(x_q, \tau) \leq \mathrm{idx}(x_t, \tau)\}\big|,$$

which charges each predecessor node x_p for every node in τ that must retain x_p after it is pulled across the bridge edge (x_p, x_q).

The total context cost is $C(\tau) = C_I(\tau) + C_E(\tau)$.

Internal cost difference. Since $\tau_{(S)} = \tau_{(L)} \cup \tau_{(R)}$, the internal cost of $\tau_{(S)}$ decomposes as

$$\begin{aligned} C_I(\tau_{(S)}) = C_I(\tau_{(L)}) + C_I(\tau_{(R)}) \\ + \sum_{x_i \in \tau_{(L)}} c(x_i) \cdot \big|\{x_t \in \tau_{(R)} : \mathrm{idx}(x_i, \tau_{(S)}) < \mathrm{idx}(x_t, \tau_{(S)})\}\big| \\ + \sum_{x_j \in \tau_{(R)}} c(x_j) \cdot \big|\{x_t \in \tau_{(L)} : \mathrm{idx}(x_j, \tau_{(S)}) < \mathrm{idx}(x_t, \tau_{(S)})\}\big|. \end{aligned}$$

The first extra term accounts for the fact that in the monolithic trajectory, nodes in $\tau_{(R)}$ must retain preceding nodes in $\tau_{(L)}$ in their context. The second extra term is symmetric: it accounts for nodes in $\tau_{(R)}$ whose context must be retained by subsequent nodes in $\tau_{(L)}$. Since both extra terms are non-negative, dropping the second term yields the lower bound

$$C_I(\tau_{(S)}) - C_I(\tau_{(L)}) - C_I(\tau_{(R)}) \geq \sum_{x_i \in \tau_{(L)}} c(x_i) \cdot \big|\{x_t \in \tau_{(R)} : \mathrm{idx}(x_i, \tau_{(S)}) < \mathrm{idx}(x_t, \tau_{(S)})\}\big|. \quad (8)$$

External cost difference. Let $\mathcal{B}_S$ be the set of bridge edges entering $\tau_{(S)}$. After partitioning by the bridge edge (x_L, x_R), the incoming bridge edge sets for the two subtasks are

$$\mathcal{B}_L = \{(x_p, x_q) \in \mathcal{B}_S \mid x_q \in \tau_{(L)}\}, \qquad \mathcal{B}_R = \{(x_p, x_q) \in \mathcal{B}_S \mid x_q \in \tau_{(R)}\} \cup \{(x_L, x_R)\}.$$

We claim that $\mathcal{B}_S$ decomposes disjointly as

$$\mathcal{B}_S = \mathcal{B}_L \cup \big(\mathcal{B}_R \setminus \{(x_L, x_R)\}\big),$$

with $\mathcal{B}_L \cap \big(\mathcal{B}_R \setminus \{(x_L, x_R)\}\big) = \varnothing$. Indeed, suppose for contradiction that there exists a bridge edge $(x_p, x_q) \in \mathcal{B}_L \cap \big(\mathcal{B}_R \setminus \{(x_L, x_R)\}\big)$. Then x_q would belong to both $\tau_{(L)}$ and $\tau_{(R)}$, contradicting the fact that $\tau_{(L)}$ and $\tau_{(R)}$ are disjoint subtask trajectories obtained by removing the bridge edge (x_L, x_R).

The external cost of the original trajectory is

$$C_E(\tau_{(S)}) = \sum_{(x_p,x_q)\in\mathcal{B}_L} c(x_p) \cdot \big|\{x_t \in \tau_{(S)} : \mathrm{idx}(x_q, \tau_{(S)}) \le \mathrm{idx}(x_t, \tau_{(S)})\}\big| + \sum_{(x_p,x_q)\in\mathcal{B}_R\setminus\{(x_L,x_R)\}} c(x_p) \cdot \big|\{x_t \in \tau_{(S)} : \mathrm{idx}(x_q, \tau_{(S)}) \le \mathrm{idx}(x_t, \tau_{(S)})\}\big|.$$

The external costs of the two subtasks are

$$\begin{aligned} C_E(\tau_{(L)}) + C_E(\tau_{(R)}) = &\sum_{(x_p,x_q)\in\mathcal{B}_L} c(x_p) \cdot \big|\{x_t \in \tau_{(L)} : \mathrm{idx}(x_q, \tau_{(L)}) \le \mathrm{idx}(x_t, \tau_{(L)})\}\big| \\ &+ \sum_{(x_p,x_q)\in\mathcal{B}_R\setminus\{(x_L,x_R)\}} c(x_p) \cdot \big|\{x_t \in \tau_{(R)} : \mathrm{idx}(x_q, \tau_{(R)}) \le \mathrm{idx}(x_t, \tau_{(R)})\}\big| \\ &+ c(x_L) \cdot \big|\{x_t \in \tau_{(R)} : \mathrm{idx}(x_R, \tau_{(R)}) \le \mathrm{idx}(x_t, \tau_{(R)})\}\big|. \end{aligned}$$

For any $(x_p, x_q) \in \mathcal{B}_L$, since $\tau_{(L)} \subseteq \tau_{(S)}$ and the relative positions of nodes within $\tau_{(L)}$ are preserved in $\tau_{(S)}$, we have $\mathrm{idx}(x_q, \tau_{(L)}) = \mathrm{idx}(x_q, \tau_{(S)})$. Consequently,

$$\{x_t \in \tau_{(L)} : \mathrm{idx}(x_q, \tau_{(L)}) \le \mathrm{idx}(x_t, \tau_{(L)})\} \subseteq \{x_t \in \tau_{(S)} : \mathrm{idx}(x_q, \tau_{(S)}) \le \mathrm{idx}(x_t, \tau_{(S)})\},$$

and therefore

$$\begin{aligned} &\big|\{x_t \in \tau_{(L)} : \mathrm{idx}(x_q, \tau_{(L)}) \le \mathrm{idx}(x_t, \tau_{(L)})\}\big| \\ &\quad\le \big|\{x_t \in \tau_{(S)} : \mathrm{idx}(x_q, \tau_{(S)}) \le \mathrm{idx}(x_t, \tau_{(S)})\}\big|. \end{aligned}$$

The same argument applies to any $(x_p, x_q) \in \mathcal{B}_R \setminus \{(x_L, x_R)\}$, yielding

$$\begin{aligned} &\big|\{x_t \in \tau_{(R)} : \mathrm{idx}(x_q, \tau_{(R)}) \le \mathrm{idx}(x_t, \tau_{(R)})\}\big| \\ &\quad\le \big|\{x_t \in \tau_{(S)} : \mathrm{idx}(x_q, \tau_{(S)}) \le \mathrm{idx}(x_t, \tau_{(S)})\}\big|. \end{aligned}$$

Thus, comparing the expressions for $C_E(\tau_{(S)})$ and $C_E(\tau_{(L)}) + C_E(\tau_{(R)})$ term by term, the only term present in the subtask costs but absent from the original cost is the newly introduced bridge edge cost

$$c(x_L) \cdot \big|\{x_t \in \tau_{(R)} : \mathrm{idx}(x_R, \tau_{(R)}) \le \mathrm{idx}(x_t, \tau_{(R)})\}\big|.$$

Therefore,

$$C_E(\tau_{(S)}) - C_E(\tau_{(L)}) - C_E(\tau_{(R)}) \ge -c(x_L) \cdot \big|\{x_t \in \tau_{(R)} : \mathrm{idx}(x_R, \tau_{(R)}) \le \mathrm{idx}(x_t, \tau_{(R)})\}\big|. \quad (9)$$

Combining the bounds. Combining (8) and (9), we have

$$\begin{aligned} &C(\tau_{(S)}) - C(\tau_{(L)}) - C(\tau_{(R)}) \\ &= \big[C_I(\tau_{(S)}) - C_I(\tau_{(L)}) - C_I(\tau_{(R)})\big] \\ &\quad + \big[C_E(\tau_{(S)}) - C_E(\tau_{(L)}) - C_E(\tau_{(R)})\big] \\ &\ge \sum_{x_i\in\tau_{(L)}} c(x_i) \cdot \big|\{x_t \in \tau_{(R)} : \mathrm{idx}(x_i, \tau_{(S)}) < \mathrm{idx}(x_t, \tau_{(S)})\}\big| \\ &\quad - c(x_L) \cdot \big|\{x_t \in \tau_{(R)} : \mathrm{idx}(x_R, \tau_{(R)}) \le \mathrm{idx}(x_t, \tau_{(R)})\}\big|. \end{aligned}$$

Since $x_L \in \tau_{(L)}$, the first sum contains the term corresponding to $x_i = x_L$:

$$\begin{aligned} &\sum_{x_i\in\tau_{(L)}} c(x_i) \cdot \big|\{x_t \in \tau_{(R)} : \mathrm{idx}(x_i, \tau_{(S)}) < \mathrm{idx}(x_t, \tau_{(S)})\}\big| \\ &\qquad\ge c(x_L) \cdot \big|\{x_t \in \tau_{(R)} : \mathrm{idx}(x_L, \tau_{(S)}) < \mathrm{idx}(x_t, \tau_{(S)})\}\big|. \end{aligned}$$

Because (x_L, x_R) is a bridge edge, we have $\mathrm{idx}(x_L, \tau_{(S)}) < \mathrm{idx}(x_R, \tau_{(S)})$, and hence

$$\{x_t \in \tau_{(R)} : \mathrm{idx}(x_R, \tau_{(S)}) \le \mathrm{idx}(x_t, \tau_{(S)})\} \subseteq \{x_t \in \tau_{(R)} : \mathrm{idx}(x_L, \tau_{(S)}) < \mathrm{idx}(x_t, \tau_{(S)})\}.$$

Since $\mathrm{idx}(x_R, \tau_{(R)}) = \mathrm{idx}(x_R, \tau_{(S)})$, this implies

$$\begin{aligned}&\big|\{x_t \in \tau_{(R)} : \mathrm{idx}(x_R, \tau_{(R)}) \le \mathrm{idx}(x_t, \tau_{(R)})\}\big| \\ &\qquad \le \big|\{x_t \in \tau_{(R)} : \mathrm{idx}(x_L, \tau_{(S)}) < \mathrm{idx}(x_t, \tau_{(S)})\}\big|.\end{aligned}$$

Therefore, the first term in the lower bound dominates the second term, and we obtain

$$\begin{aligned}&C(\tau_{(S)}) - C(\tau_{(L)}) - C(\tau_{(R)}) \\ &\ge c(x_L) \cdot \big|\{x_t \in \tau_{(R)} : \mathrm{idx}(x_L, \tau_{(S)}) < \mathrm{idx}(x_t, \tau_{(S)})\}\big| \\ &\quad - c(x_L) \cdot \big|\{x_t \in \tau_{(R)} : \mathrm{idx}(x_R, \tau_{(R)}) \le \mathrm{idx}(x_t, \tau_{(R)})\}\big| \\ &\ge 0.\end{aligned}$$

Thus,

$$C(\tau_{(L)}) + C(\tau_{(R)}) \le C(\tau_{(S)}),$$

which completes the proof. □

B Experimental Details

B.1 Benchmark Details

Terminal Bench 2.1 (Merrill et al., 2026) is a curated hard benchmark composed of tasks in computer terminal environments inspired by real-world workflows. Each task features a unique environment, a human-written solution, and comprehensive tests for verification. Because the framework is interactive, agent and model performance are difficult to decouple; many agent scaffolds are engineered to accommodate the tendencies of specific models, and agents are not explicitly required to use a terminal as their sole tool. To remain faithful to the benchmark's premise, a neutral testbed with a single headless terminal is adopted, and tasks are completed using only Bash commands. The primary metric is the resolution rate. In our evaluation, each task is attempted three times, and the resolution rate is computed as the number of tasks completed in a run divided by the total number of tasks, averaged over the three runs.

NL2Repo Bench (Ding et al., 2026) evaluates repository-level coding abilities with verifiable ground truth. Tasks are derived from real-world Python libraries characterized by modular architectures and authoritative `pytest` suites. Agents receive only a single natural-language specification and must reconstruct the complete repository from scratch, including file structures and functional logic. Correctness is strictly measured by executing the generated code against the original upstream tests. Each task is scored by the fraction of unit tests it passes, and the final benchmark score is the average of these per-task pass rates across all tasks.

Deep Research Bench II (Li et al., 2026) diagnoses deep research agents through rubrics derived from expert reports. The benchmark evaluates agents on information recall, analysis, and presentation, with rubrics constructed through a four-stage pipeline of LLM extraction, self-evaluation iteration, manual revision, and expert review. For each task, the score is the fraction of rubrics it passes, and the final benchmark score is computed as the total number of rubrics passed divided by the total number of rubrics across all tasks.

AgentIF-OneDay (Chen et al., 2026) is a task-level instruction-following benchmark for general AI agents in daily scenarios. It employs instance-level, rubric-based scoring with three key properties: binary scoring, where each rubric item is strictly evaluated as satisfied or not to ensure objectivity; distinction between bonus and penalty items, separating capability assessment from error rates; and file-content alignment, enabling direct evaluation of agent-generated files alongside textual output. Let N be the total number of problems. For the i-th problem, let S_i^+ and S_i^- denote the sum of satisfied bonus points and triggered penalty points, and $S_i^{\max}$ the maximum achievable score. The normalized score is

$$s_i = \frac{\max(0, S_i^+ - S_i^-)}{S_i^{\max}}, \tag{10}$$

and the final benchmark score is the mean of these normalized scores:

$$\text{Score}_{\text{final}} = \frac{1}{N} \sum_{i=1}^{N} s_i. \tag{11}$$

Because AgentIF-OneDay involves multimodal requirements such as image editing and in-depth research, while DeepSeek-V4-Pro (DeepSeek, 2026c) does not natively support multimodal input, we provide a multimodal skill that allows the agent to invoke DeepSeek-V4-Flash-Vision-Exp (DeepSeek, 2026b) for image understanding. This skill is implemented as a script that the agent can invoke directly, so the agent can decide when to delegate visual perception to the vision model without altering the benchmark's evaluation protocol.

B.2 Baseline Details

Single-Agent executes the full task within a single agent context, serving as the monolithic baseline. It runs on the native Codex CLI harness (OpenAI, 2025b).

Task Decomposition and Agent Generation (TDAG) (Wang et al., 2025b) is a multi-agent framework built on dynamic task decomposition and agent generation. It decomposes complex tasks into smaller subtasks and assigns each subtask to a specifically generated subagent, thereby improving adaptability in diverse and unpredictable real-world tasks. The decomposition is not static: subtasks are dynamically adjusted based on the outcomes of preceding tasks, allowing the framework to refine the task structure as execution unfolds. Successful execution patterns are further summarized into reusable skills for future reference.

DynTaskMAS (Yu et al., 2025) is a dynamic task graph-driven framework for asynchronous and parallel LLM-based multi-agent systems. It orchestrates asynchronous and parallel operations through dynamic task graphs, and comprises four key components: a Dynamic Task Graph Generator that decomposes complex tasks while maintaining logical dependencies, an Asynchronous Parallel Execution Engine that optimizes resource utilization through efficient task scheduling, a Semantic-Aware Context Management System that enables efficient information sharing among agents, and an Adaptive Workflow Manager that dynamically optimizes system performance. This design supports scalable, high-performance multi-agent execution on complex, dynamic tasks.

Graph-of-Agents (GoA) (Yun et al., 2026) models multi-agent LLM communication through a graph-based framework. It first samples relevant agents from a candidate pool, then constructs edges between them by evaluating the relevance of their responses to one another, and finally aggregates the refined responses via graph-based pooling. Message passing proceeds in two stages, source-to-target followed by target-to-source, allowing highly relevant agents to refine the responses of less relevant ones and vice versa. To ensure consistency of the base model across all evaluated methods, we do not adopt GoA's original multi-source LLM design, in which different agents are backed by different foundation models. Instead, we instantiate all agents with the same foundation LLM and differentiate them by assigning distinct roles, so that the comparison isolates the effect of orchestration rather than that of heterogeneous model capabilities.

B.3 Detailed Experimental Results

B.3.1 Detailed Main Results

Beyond the aggregate metrics in Table 1, we report the detailed breakdown when a benchmark provides finer-grained evaluation dimensions.

Table 4: Detailed results on Deep Research Bench II.

Method	Info Recall	Analysis	Presentation	Total Score
Single-Agent	47.57	52.08	92.49	51.94
TDAG	50.72	53.56	91.15	54.43
DynTaskMAS	48.85	54.69	93.57	53.44
GoA	43.11	50.77	92.49	48.40
SAIGE (Ours)	51.40	54.33	93.16	55.23

Table 5: Detailed results on AgentIF-OneDay.

Method	Content	Form	Execution	Total Score
Single-Agent	51.76	47.57	43.71	53.57
TDAG	60.92	59.42	48.12	59.94
DynTaskMAS	66.18	59.35	48.39	65.85
GoA	53.04	48.99	41.08	52.67
SAIGE (Ours)	68.77	60.95	54.81	71.80

B.3.2 Further Ablation Study

We further examine the ablation results under the five configurations for the multi-agent baselines. As shown in Figure 4, increasing the agent count or deepening the recursion level generally reduces the task score while increasing token consumption. Across both benchmarks and both baselines, i.e., TDAG and DynTaskMAS, the default configuration $(N = 4, D = 1)$ attains the most favorable overall trade-off between task performance and token cost, whereas the remaining configurations fall into the lower-right region relative to the default. This indicates that additional agents or deeper recursion introduce coordination overhead without improving problem-solving capacity.

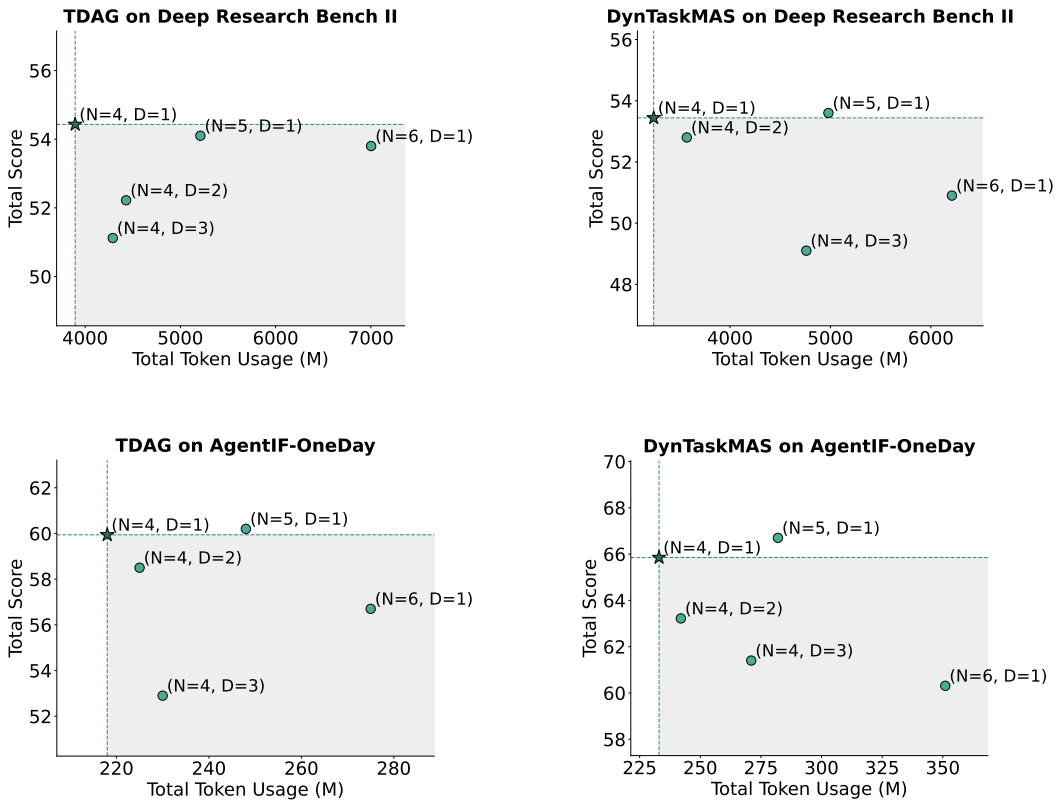

Figure 4: Ablation results under five configurations. Each point corresponds to one configuration.

B.3.3 Fine-Grained Analysis

To examine whether SAIGE performs conditional subtask decomposition according to task dependency structure rather than spawning agents indiscriminately, we record the average number of agents instantiated per task on each benchmark. We further evaluate whether the predicted bridge edges correspond to the ground-truth dependency boundaries.

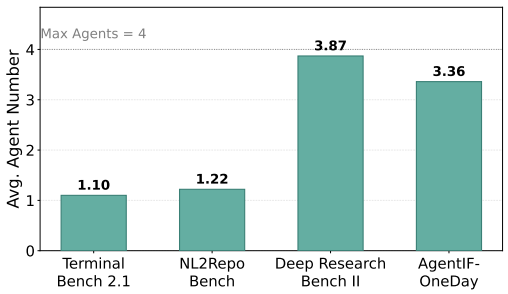

Figure 5: Average agent number per task across four benchmarks.

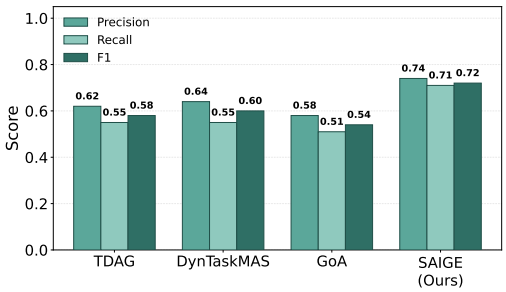

Figure 6: Bridge-edge decomposition accuracy on AgentIF-OneDay.

As shown in Figure 5, SAIGE instantiates substantially fewer agents on Terminal Bench 2.1 and NL2Repo Bench, where the trajectory exhibits dense inter-step dependencies and most subtasks cannot be executed independently. On Deep Research Bench II and AgentIF-OneDay, whose trajectories are sparsely dependent, SAIGE spawns more agents and exploits context isolation more aggressively. The agent count therefore tracks the dependency structure of the benchmark rather than a fixed allocation policy, which is consistent with the conditional delegation criterion. Notably, the average agent number remains below the maximum budget on all four benchmarks, indicating that SAIGE does not expand its collaboration graph beyond what the task structure permits.

To obtain a ground-truth decomposition for AgentIF-OneDay, we first collect the successful trajectories produced by the Single-Agent baseline on this benchmark. For each such trajectory, we construct a semantic dependency matrix with DeepSeek-V4-Pro: for every step t and each earlier step $i < t$, the model judges whether step t semantically conditions on the output of step i, i.e., whether the action or observation at step t would change if x_i were removed from the history. This yields a binary dependency matrix. Bridge edges are then identified as the single-step dependencies whose removal partitions the trajectory into weakly connected components, and the resulting components define the ground-truth decomposition points at which the task can be split into independently executable subtasks.

We then evaluate the decomposition produced by each multi-agent method against this ground truth. For every sub-agent trajectory generated by a method, we semantically match its instruction and executed steps to the ground-truth subtask components using DeepSeek-V4-Pro, and count a predicted bridge edge as correct if it connects the same pair of subtask components as a ground-truth bridge edge. Precision, recall, and F1 are computed over the matched bridge edges across all tasks.

Figure 6 reports the results on AgentIF-OneDay. SAIGE achieves the most accurate decomposition among all multi-agent methods, indicating that its incremental, execution-grounded edge prediction recovers the true dependency boundaries more faithfully than static or pre-committed decomposition strategies. The improvement is most pronounced in recall, suggesting that SAIGE is less prone to omitting bridge edges, which is precisely the failure mode that produces local deadlocks. These results provide direct evidence that the performance gains of SAIGE on sparsely dependent benchmarks are accompanied by, and plausibly attributable to, more accurate bridge-edge prediction.

C Case Study

C.1 Task Dependency Patterns

To illustrate the task dependency patterns that underlie our analysis, we select one representative single-agent trajectory from Terminal Bench 2.1 and one from AgentIF-OneDay, and construct their dependency matrices through semantic dependency analysis. Specifically, for each step t in the trajectory, we take the history $\mathcal{H}_t = \{x_1, \ldots, x_{t-1}\}$ and perform semantic analysis with DeepSeek-V4-Pro to assess, for each earlier step $i < t$, whether step t semantically conditions on the output of step i, i.e., whether the action or observation at step t would change if x_i were removed from the

history. This yields a binary dependency matrix $A \in \{0,1\}^{T\times T}$ with $A_{t,i} = 1$ if step t depends on step i, and $A_{t,i} = 0$ otherwise.

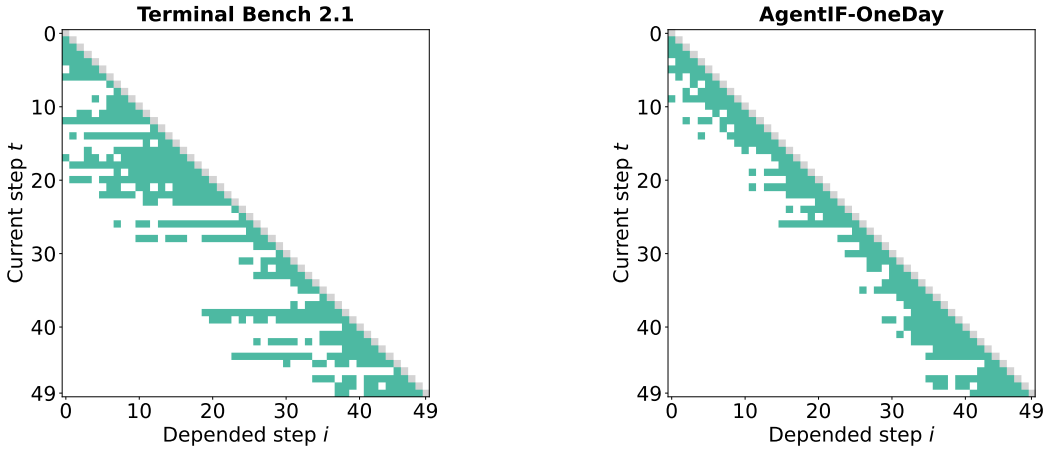

Figure 7: Dependency matrices of representative single-agent trajectories.

Figure 7 shows the resulting dependency matrices for the two trajectories. The Terminal Bench 2.1 trajectory exhibits a dense dependency pattern in which a non-negligible fraction of steps depend on distant predecessors, producing long-range dependencies that span a wide range of earlier steps. This is consistent with the long upstream-downstream chains induced by continuous tool invocations, file reads, and file writes, where later steps must consume the outputs of earlier ones and independent execution is rarely possible. The AgentIF-OneDay trajectory, by contrast, exhibits a sparse and localized pattern: non-zero entries concentrate near the diagonal, indicating that most steps depend only on their immediate predecessors and that large portions of the trajectory can proceed without conditioning on one another. This contrast directly reflects the two structural regimes identified in our analysis, and provides a step-level view of why multi-agent decomposition is constrained on tightly coupled benchmarks while remaining feasible on sparsely dependent ones.

C.2 Bridge Edge Mis-Prediction: A Local Deadlock Case

We illustrate the effect of bridge edge mis-prediction on a representative task from Terminal Bench 2.1, `make-doom-for-mips`, in which the agent must build a MIPS ELF that runs under a supplied JavaScript MIPS VM. The task contains a long upstream-downstream chain: the VM defines the syscall ABI and the memory layout, the build must target a compatible MIPS ABI, and the resulting ELF must be validated by running it inside the VM. A correct decomposition must preserve the bridge edges that connect these stages.

```
// Ground-truth dependency structure
1  inspect vm.js
2      -> identify syscall ABI
3      -> identify memory layout
4  inspect doomgeneric/
5      -> identify build targets
6  configure MIPS toolchain
7      -> select ABI / soft-float
8  build doomgeneric_mips
9  run under vm.js
10     -> observe failure
11 -> adjust build flags
12 -> rebuild
13 -> validate execution
```

```
// Predicted dependency structure
1  inspect vm.js
2      -> identify syscall ABI
3  inspect doomgeneric/
4      -> identify build targets
5  configure MIPS toolchain
6      -> select ABI / soft-float
7  build doomgeneric_mips
8  run under vm.js
9      -> observe failure
10 -> adjust build flags
11 -> rebuild
12 -> validate execution
```

Figure 8: Ground-truth and mis-predicted dependency structures on `make-doom-for-mips`.

The error is one omitted bridge edge: the dependency from VM memory layout (step 3) to build configuration (step 6) in the ground-truth dependency structure. In the ground-truth structure, this edge carries the constraint that determines the correct ABI and memory model. In the predicted structure, the edge is absent.

This omission produces a local deadlock. The build agent configures the MIPS toolchain without the VM memory layout, while the validation agent repeatedly rebuilds and waits for a build that satisfies the omitted constraint. Neither agent can break the cycle: the build agent adjusts flags that are not the actual bottleneck, and the validation agent keeps issuing rebuild-and-run cycles.

Two costs follow. Communication overhead rises because repeated build-and-run messages do not converge, and redundant execution accumulates because each cycle reruns the same build and the same VM validation.

If the bridge edge is predicted correctly, the build configuration is conditioned on the VM memory layout and syscall ABI, and the correct ABI and soft-float configuration are selected before the first build. The validation stage then observes a consistent ELF, and the trajectory terminates after a small number of build-and-run cycles. This contrast shows why orchestration is naturally formulated as bridge edge prediction: an omitted bridge edge does not merely reduce context cost reduction, but can turn a convergent workflow into a local deadlock.

D IMPACT STATEMENT

This work studies when multi-agent collaboration is beneficial for long-horizon agentic tasks and when it is not. Its primary impact is conceptual and practical: it clarifies that multi-agent superiority is bounded by task structure rather than universal, and that blindly scaling the number of agents can increase coordination overhead without improving task performance. This message is relevant to practitioners who build agentic systems, since it suggests that collaboration should be introduced selectively, based on the dependency structure of the target task, rather than adopted as a default design choice.

On the positive side, the proposed SAIGE mechanism grows its collaboration graph in response to execution feedback and establishes semantic dependencies through content-based information retrieval. This may help reduce unnecessary context accumulation and improve the reliability of long-horizon execution, which is relevant to applications such as software engineering agents, repository-level code generation, and deep research assistants. By making the cost-performance trade-off of multi-agent decomposition explicit, our analysis may also encourage more disciplined evaluation and reporting practices in this area.

On the risk side, more capable and autonomous agentic systems can be misused for harmful purposes, including automated exploitation of software vulnerabilities, unauthorized access to computing resources, or large-scale generation of misleading content. The benchmarks used in this work are designed for research evaluation and operate in controlled environments, but the underlying techniques are not inherently restricted to benign settings. Multi-agent decomposition may also introduce failure modes that are harder to audit, such as local deadlocks, mispredicted dependencies, and opaque coordination among agents, which can complicate debugging, accountability, and oversight.