CoAct: A Minimal Coordination Substrate for Decentralized LLM Agents

Zhenting Qi [*1] **Jack Fan** [*1] **Karly Hou** [*2] **Daisy Xinlei Lin** [3] **Satyaki Chakraborty** [3] **Mete Kemertas** [3] **Yu Yao** [3]
Sophia Simeng Han [4] **Han Zheng** [5] **Zihan Wang** [6] **Xiangjun Wang** [3] **Himabindu Lakkaraju** [1] **Yilun Du** [1]

Abstract

Large language models are increasingly deployed as tool-using agents, motivating multi-agent systems that can explore in parallel, share discoveries, and cross-check decisions. However, decentralized collaboration introduces two core challenges: agents must make local discoveries available to the team, and they must coordinate before executing stateful actions that modify shared external environments. We introduce CoAct, a minimal coordination substrate for decentralized LLM agents based on two primitives: compact shared-memory communication and synchronized stateful commitment. CoAct preserves the standard ReAct-style agent loop while allowing agents to independently use stateless tools, write decision-relevant messages to a shared memory, and jointly select stateful actions before execution. Across four agentic task domains spanning math reasoning, web research, financial analysis, and stateful user assistance, CoAct improves over single-agent execution, non-communicating parallelization, and multi-agent system with unstructured communication, while using communication more token-efficiently than free-form discussion. We further show that scaling beyond small teams leads to non-monotonic advantage: increasing the number of agents or the capability of the base model does not always improve performance. Through empirical scaling studies, probabilistic analysis, and case studies, we identify when communication helps or hurts decentralized agent teams, suggesting that progress in multi-agent LLM systems depends not only on scaling teams but also improving effective communication.

[1]Kempner Institute for the Study of Natural and Artificial Intelligence at Harvard University, Boston, MA [2]Unconventional AI [3]Amazon AGI Lab, San Francisco, CA [4]Stanford University, Stanford, CA [5]Massachusetts Institute of Technology, Cambridge, MA [6]Abaka AI. Correspondence to: Zhenting Qi <zhentingqi@g.harvard.edu>.

Proceedings of the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

1. Introduction

Large language models (LLMs) are increasingly deployed as agents that reason, use tools, and interact with external environments (Yao et al., 2022; Li et al., 2023; Patil et al., 2024; Yang et al., 2024; Xie et al., 2024). Many agentic tasks require gathering dispersed information, checking intermediate conclusions, and revising decisions mid-trajectory, which motivates multi-agent systems (MAS): parallel agents explore diverse trajectories, uncover information more efficiently, and cross-check reasoning. However, coordination can introduce overhead, noise, and interference that degrade performance (Cemri et al., 2025; Kim et al., 2025; Yadav et al., 2026).

Centralized approaches impose structure via orchestrators, sub-agent decomposition, or task-specific workflows (Qian et al., 2024; Wu et al., 2023; Hong et al., 2024), but require bespoke coordination logic, introduce bottlenecks, and create single points of failure (Wang et al., 2025a; Kim et al., 2025; Yang et al., 2025; Cemri et al., 2025). Decentralized systems remove the central controller, but most target closed or simplified settings like text games (Du et al., 2024; Subramaniam et al., 2025; Yang et al., 2025; Wang et al., 2025a). Tool-using environments differ: agents acquire information incrementally through their own tool calls, and some actions modify shared or externally visible state.

We argue that decentralized tool-using agents face two bottlenecks: *information availability* (each agent observes only its own trajectory, so discoveries cannot help the team unless communicated) and *synchronized commitment* (uncoordinated stateful actions cause duplication, inconsistency, or race conditions). These motivate two design principles: a communication mechanism that makes local discoveries team-usable, and a synchronization mechanism that ensures stateful actions execute consistently.

Building upon these, we introduce CoAct, a minimal decentralized generalization of ReAct-style tool-using agents built on exactly these two primitives. Tools are partitioned into *stateless* tools (used concurrently) and *stateful* tools (requiring synchronized commitment); agents communicate via compact shared-memory messages visible to all in subsequent steps. CoAct prescribes no roles, hierarchies, or task-specific workflows.

Across four agentic domains—math reasoning, web research, financial analysis, and stateful user assistance—two-to-three-agent CoAct consistently outperforms single-agent execution, non-communicating parallelization, and unstructured debate, while using communication more token-efficiently. Scaling beyond small teams, however, produces non-monotonic gains: communication repairs wrong trajectories when agents contribute decision-relevant information, but redundant or low-quality messages limit gains and complicate synchronization. We characterize this behavior through empirical scaling experiments, probabilistic analysis, and case studies.

This work makes three contributions: (1) CoAct, a minimal task-agnostic coordination substrate based on compact shared memory and synchronized stateful commitment; (2) empirical results showing consistent improvements in the two-to-three-agent regime with token-efficient communication; and (3) a characterization of non-monotonic scaling behavior identifying when inter-agent communication helps or hurts.

2. CoAct: A Minimal Multi-Agent Substrate

CoAct (Figure 1) is a drop-in generalization of ReAct-style tool-using LLM agents to decentralized teams. It preserves the standard local Think → Act → Observe loop but runs K agents in parallel over the same task environment, adding two coordination mechanisms: compact shared-memory communication and synchronized execution of stateful actions. Agents maintain private trajectories and assume no prior knowledge of one another's observations, reasoning, capabilities, or roles.

This design targets two bottlenecks. *Information availability*: each agent observes only its local trajectory, so discoveries help the team only if communicated in a usable form. *Synchronized commitment*: uncoordinated stateful actions cause duplication, inconsistent updates, or premature final answers.

2.1. Execution Loop

CoAct alternates between two phases: Generate and Synchronize.

Generation. Each agent independently follows a ReAct-style Think → Act → Observe loop, restricted to *stateless tools* $\mathcal{T}_{\text{stateless}}$ that do not modify shared state (e.g., search, retrieval, calculation, document reading). Each agent may also call `<share>`(m) to append a compact, decision-relevant message to shared global memory $\mathcal{C}$, injected into every agent's context at each subsequent decision point. Messages are bounded by a fixed token budget $B \ll L$, where L is the mean per-agent trajectory length, keeping per-agent context cost at $O(LT + KBT)$ versus $O(KLT)$ under full trajectory sharing—a reduction by a factor of L/B.

Synchronization. *Stateful tools* $\mathcal{T}_{\text{stateful}}$ (e.g., submitting a final answer, updating a database) cannot be executed during generation. Each agent proposes a candidate intent or abstains; a synchronization module aggregates candidates into at most one team-level selection, executed once, with the observation broadcast to all agents. If no candidate is selected, the system either defers and re-enters generation, or falls back to uniform random selection over $\mathcal{Y}_t$. Random fallback is used to prevent indefinite re-entry and cost explosion; fallback also forces commitment and allows the agents to continue in a generally effective and cost-efficient manner. We also introduce a terminal stateful action $u^{\text{term}} \in \mathcal{T}_{\text{stateful}}$, which ends execution. Aggregation rules are discussed in Section B.

2.2. Formal Setup

We model decentralized execution as decomposing into two necessary conditions for success. Let M denote *information compositionality*: the number of decision-relevant factors that must be correctly established across the team to determine the correct next stateful action or final answer. Importantly, M is not an input to CoAct; it is an analysis abstraction that captures how much task-relevant information must be discovered and communicated. A run succeeds only if: (E_{info}) all required factor values become correctly available to the team through local discovery and communication, and (E_{sync}) synchronization selects the correct stateful action given that information. Together,

$$\begin{aligned} E_{\text{succ}} &= E_{\text{info}} \cap E_{\text{sync}}, \\ \Pr(E_{\text{succ}}) &= \Pr(E_{\text{sync}} \mid E_{\text{info}}) \, \Pr(E_{\text{info}}) \end{aligned} \tag{1}$$

This decomposition isolates where each coordination primitive intervenes. Shared memory targets $\Pr(E_{\text{info}})$: if agent i writes m_t^i at round t, then $\mathcal{C}_{t+1} = \mathcal{C}_t \cup \{m_t^i\}$, with messages from multiple agents unioned, and the token budget B directly governs communication fidelity — larger B increases $\Pr(E_{\text{info}})$ at the cost of context overhead, while $B \ll L$ risks lossy compression of decision-relevant content. Synchronization targets $\Pr(E_{\text{sync}} \mid E_{\text{info}})$: each agent proposes a candidate $y_t^i \in \mathcal{Y}_{\text{stateful}} \cup \{\varnothing\}$, the candidate set $\mathcal{Y}_t = \{y_t^i : y_t^i \neq \varnothing,\ i \in [K]\}$ is passed to an aggregation rule

$$y_t^\star = A(\mathcal{Y}_t, \mathcal{C}_t), \qquad A : 2^{\mathcal{Y}_{\text{stateful}}} \times \mathfrak{C} \to \mathcal{Y}_{\text{stateful}} \cup \{\varnothing\},$$

and a well-designed A recovers the correct intent from the candidate set while a poorly designed one discards it. We evaluate deployable and diagnostic choices of A in Section B. The full interaction-process formalization is deferred to Section D.

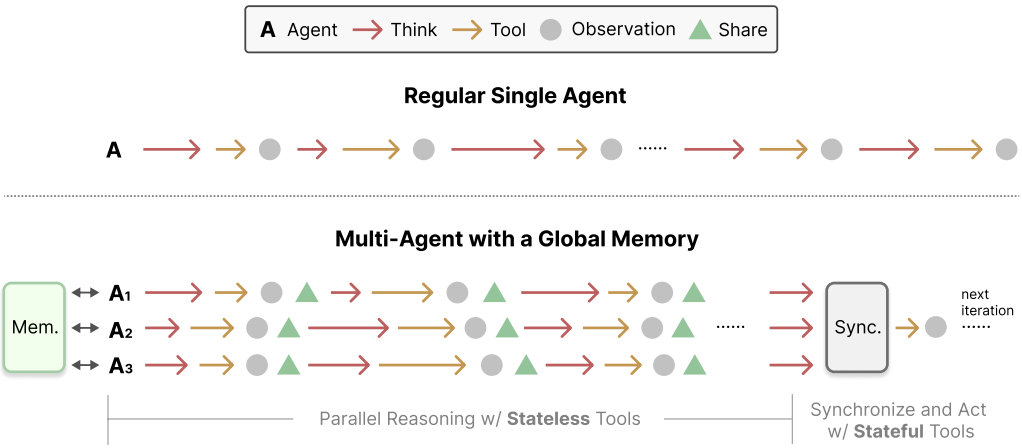


Figure 1. **CoAct overview.** CoAct generalizes a ReAct-style tool-using agent into a decentralized team, where agents reason and use stateless tools independently, communicate compact discoveries through a shared global memory, and synchronize before executing stateful tools that modify the shared environment.

3. Experiments

We evaluate CoAct as a minimal coordination substrate for decentralized tool-using agents. The experiments are organized around the two primitives introduced in Section 2: compact shared-memory communication and synchronized commitment. We first ask whether these primitives are sufficient to improve over single-agent execution, independent parallelism, and unstructured debate. We then leave a dedicated cost analysis for communication overhead. Finally, we study scaling beyond the small-team regime by varying the number of agents, task structure, and base-model capability. Across these settings, we find that multi-agent gains are non-monotonic rather than automatic.

3.1. Setup

Benchmarks. We evaluate on four primary benchmarks spanning math reasoning, web research (Wei et al., 2025), financial analysis (Bigeard et al., 2025), and stateful user assistance (Barres et al., 2025). These benchmarks differ in domain, task compositionality, single-agent strength, and synchronization structure. Table 1 summarizes the evaluation setting, and Appendix G provides additional benchmark details.

We distinguish between single-sync and multiple-sync tasks. In single-sync tasks, including HMMT, BrowseComp, and FinanceAgent, the only stateful action is final-answer submission: once the team submits an answer, the task terminates. By contrast, τ^2-retail requires multiple synchronization points because agents must coordinate a sequence of user-facing or order-modifying actions across a multi-turn interaction.

Methods and baselines. We compare CoAct against three baselines that isolate different possible sources of multi-agent gains. **Single agent** runs one ReAct-style tool-using agent and measures whether multi-agent execution helps at all. **Naive parallel** runs K independent agents without communication and aggregates only their final answers or stateful-action proposals, testing whether gains come merely from sampling additional independent trajectories. **Debate** (Du et al., 2024) runs K agents with unstructured free-form inter-agent communication before selecting a final decision, testing whether unconstrained communication is sufficient for coordination.

CoAct runs K ReAct-style agents in parallel using the execution loop defined in Section 2.1. Agents independently use stateless tools, write compact decision-relevant messages to shared memory through `<share>`, and synchronize before executing stateful tools. Thus, the comparison with naive parallel isolates the value of communication, while the comparison with debate isolates the value of compact structured communication and explicit synchronization. Full implementation details, including model versions, decoding parameters, step budgets, shared-memory budgets, and debate budgets, are provided in Appendix E.

Aggregation methods. At synchronization time, each agent proposes a candidate stateful intent, and an aggregation rule selects at most one to execute. The main CoAct results use ranked aggregation: agents rank the candidate set, and the one with the highest Borda count is selected. We use ranked aggregation because it is deployable and can compare semantically distinct candidate intents without requiring exact string agreement. Other aggregation rules, including agreement, majority, random, and oracle aggregation are discussed in Appendix B.

Table 1. **Benchmarks.** The four primary benchmarks are HMMT 2026, BrowseComp, FinanceAgent, and τ^2-retail. AIME 2025 is used as an auxiliary math benchmark for additional team-size analysis.

Benchmark	Domain	Stateful Actions	Synchronization	Difficulty	Evaluation set
AIME 2025	Math reasoning	Final answer	Single	competition math reasoning	full set
HMMT 2026	Math reasoning	Final answer	Single	competition math reasoning	full set
BrowseComp	Web research	Final answer	Single	sparse and fragmented evidence	100-task subset
FinanceAgent	Financial analysis	Final answer	Single	source selection and extraction	50-task public split
τ^2-retail	User assistance	All tools	Multiple	irreversible ordered actions	114-task retail subset

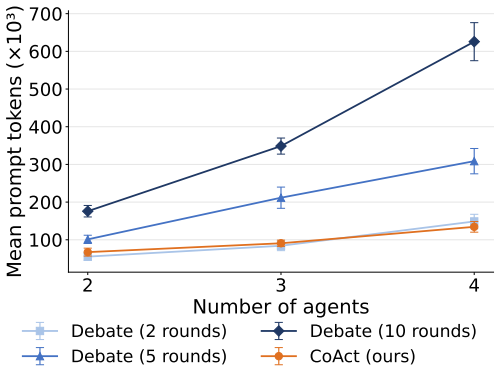
Metrics and fairness. All methods use the same underlying models, tool sets, task splits, and task-level stopping budgets within each benchmark. For multi-agent methods, the per-agent prompt and tool interface are fixed except for the communication and synchronization mechanisms required by each method. This ensures that performance differences primarily reflect how agents coordinate rather than differences in tools, data, or model access. For cost analysis, we report token cost directly, focusing on communication tokens and total context tokens.

3.2. Main Results: CoAct Improves in the Small-Team Regime

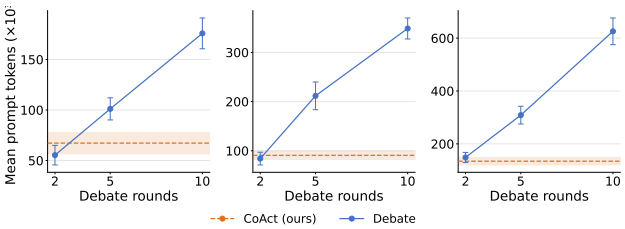
Table 2 summarizes the main results across the four primary benchmarks. CoAct provides the most consistent improvements in the small-team regime. With three agents, CoAct obtains the best result across benchmarks, improving over single-agent execution, naive parallelism, and debate. On BrowseComp, CoAct with three agents reaches 29.0% accuracy, compared to 13.0% for a single agent. On τ^2-retail, CoAct achieves 78.1% compared to 57.0% for a single agent. CoAct also increases HMMT performance from 37.0% to 45.5% and FinanceAgent from 23.3% to 25.3%.

Naive parallelism and debate are less reliable. Naive parallelism sometimes improves, but also degrades performance in several settings, indicating that CoAct's gains are not simply due to sampling more trajectories. Debate performs poorly on BrowseComp and τ^2-retail, falling below the single-agent baseline with three agents. This suggests that unconstrained communication can introduce noise, distract agents with low-quality alternatives, or increase context overhead without producing a better shared decision state.

Overall, Table 2 supports a bounded claim: CoAct is an effective small-team coordination substrate. Two to three agents often improve over single-agent execution and common multi-agent baselines, but these results do not imply that performance should improve monotonically with more agents. The rest of this section therefore examines what changes when we scale beyond the small-team regime.



(a) Tokens per task



(b) Total tokens

Figure 2. **Communication-token cost analysis.** Debate uses free-form inter-agent discussion following (Du et al., 2024), while CoAct communicates through bounded shared-memory messages; as a result, CoAct grows communication context more slowly as team size increases, shown here on AIME 25' with GPT-5-mini.

3.3. Communication Cost: Compact Shared Memory Reduces Token Overhead

Figure 2 compares the per-task and total prompt-token cost of CoAct and debate on AIME2025. Debate's token cost grows super-linearly with both the number of debate rounds and the number of agents, since each additional round appends full agent responses to every participant's context. CoAct communicates through fixed-length shared-memory messages, so context grows at a substantially slower rate as agents and rounds increase. At 10 debate rounds with 4 agents, debate incurs over $4\times$ the total token cost of CoAct, without a corresponding accuracy gain, showing that bounded shared memory decouples communication richness from context overhead.

Table 2. **Main results.** All results are reported as percentages and rounded to one decimal place, comparing single-agent baselines, naive parallel ensembles, debate, and CoAct across four benchmarks with their respective backbone models.

Method	**HMMT 26'** (GPT-4.1-mini)	**BrowseComp** (GPT-5-mini)	**FinanceAgent** (GPT-4.1-mini)	**τ^2-retail** (GPT-5.2)
Single agent	37.0	13.0	23.3	50.0
Naive parallel ×2	30.3	13.0	**25.3**	42.1
Naive parallel ×3	36.4	21.0	18.7	59.6
Debate ×2	39.4	9.0	22.7	45.6
Debate ×3	42.4	10.0	23.3	36.0
CoAct ×2	42.4	22.0	24.7	50.9
CoAct ×3	**45.5**	**29.0**	**25.3**	**78.1**

3.4. Scaling Beyond Small Teams Is Non-Monotonic

We stress-test by increasing team size, varying task structure, and changing the capability of the base model. Across these settings, the same qualitative pattern emerges: decentralized scaling helps only when additional agents contribute complementary information that can be communicated and selected reliably. Once the baseline model performs at a strong level alone or a large number of proposals become redundant or noisy, larger teams can saturate or degrade. See Appendix H for case studies.

Scaling the number of agents. On HMMT, AIME, and τ^2-retail, increasing K does not produce monotone gains. Figure 3 shows this on HMMT. CoAct improves over the single-agent baseline and no-communication parallelism for small teams, but gains plateau and eventually decline as K grows. Oracle aggregation shows that additional agents do increase the chance that a correct proposal appears, but this benefit is bounded and can be offset by noisier communication and harder selection among more candidates.

AIME shows the same pattern in a simpler single-sync math setting. As shown in Table 3, CoAct improves from two to three agents, rising from 30.0 to 36.7 pass@1, but drops to 33.3 with four agents. This rules out a pure sample-size explanation: if larger K only increased independent coverage, performance should not decline. Instead, additional agents can introduce weaker proposals or misleading intermediate reasoning that make the final synchronized decision harder.

The effect is stronger on τ^2-retail, where agents must coordinate multiple irreversible actions rather than one final answer. Figure 4 shows that CoAct performs best at three agents, while four agents reduce success. This matches the benchmark structure: extra agents can help discover user constraints, policy requirements, and valid tool arguments, but can also introduce additional opportunities for premature commitment, duplicated intents, and disagreement about action order. In stateful settings, coordination cost grows with K because the team must maintain a consistent shared action plan across several synchronization points.

Table 3. **Results on AIME 25'.** Pass@1 accuracy (%) for GPT-4.1 and GPT-4o under text-only, ReAct, and CoAct with varying team sizes; standard errors across runs are shown in parentheses.

Model	**Method**	**Pass@1**
GPT-4.1	Text-only	34.7
GPT-4o	Text-only	11.0
	ReAct	20.0
	CoAct (2 agents)	30.0 (3.3)
	CoAct (3 agents)	**36.7** (2.0)
	CoAct (4 agents)	33.3 (2.5)

Changing task compositionality. Holding the base model constant, FinanceAgent and τ^2-retail exhibit different scaling behavior, suggesting that the value of additional agents depends strongly on task compositionality and synchronization structure. On τ^2-retail, shown in Figure 4 (right), the task decomposes into multiple interacting constraints: agents must identify the user, inspect orders, check product variants, respect payment preferences, and execute state-changing tools in the correct order. This creates many opportunities for complementary partial discoveries, so moving from two to three agents yields a large gain. However, because these discoveries must be converted into a consistent sequence of stateful actions, further scaling eventually increases coordination burdens.

FinanceAgent behaves differently. Figure 4 (left) shows that CoAct remains close to the no-communication baseline and does not obtain a large positive scaling advantage. We hypothesize that this benchmark has more correlated failure modes: agents often search the same filings, extract from the same documents, and make similar source-selection or pars-

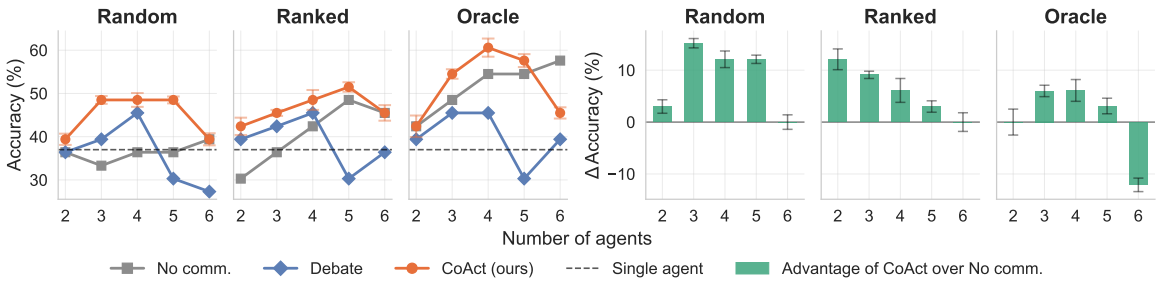


Figure 3. **Team-size scaling on HMMT 26'.** Accuracy of single agent, naive parallel, debate, and CoAct as team size increases from 1 to 4 agents, evaluated on HMMT 26' using GPT-4.1-mini.

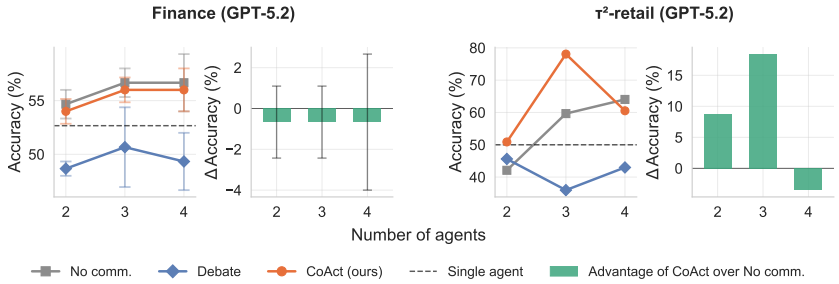


Figure 4. **Team-size scaling on FinanceAgent and τ^2-retail.** Accuracy of single agent, naive parallel, debate, and CoAct as team size increases, evaluated on FinanceAgent and τ^2-retail using GPT-5.2.

ing mistakes. When agents fail in correlated ways, scaling K increases redundancy rather than coverage. Moreover, if the bottleneck is precise extraction from a single authoritative source, communication provides less benefit than in tasks where different agents can uncover distinct missing factors. This contrast supports the view that multi-agent gains require complementary decomposition, not merely a larger number of trajectories.

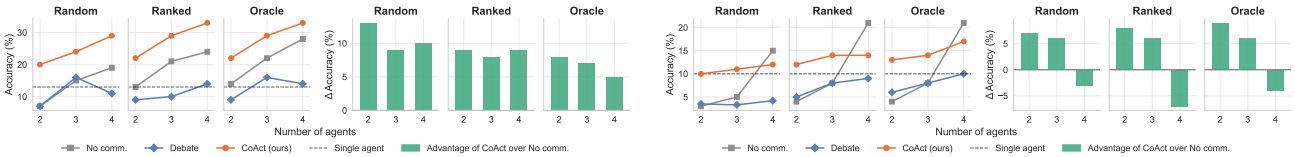
Increasing base-model capability. Finally, we test how scaling changes with base-model capability by comparing GPT-5-mini and GPT-5.2 on BrowseComp. BrowseComp is a sparse web-research task where success often requires locating fragmented evidence and correctly resolving ambiguous entity chains. With GPT-5-mini, Figure 5a shows that CoAct benefits consistently from increasing team size from two to four agents. In this weak-capability regime, individual agents often recover only partial evidence, so compact shared memory helps pool discoveries and ranked synchronization helps select among competing interpretations.

With GPT-5.2, the trend changes. Figure 5b shows that CoAct still outperforms debate and improves over no communication at small K, but the advantage shrinks and eventually becomes negative as K increases. This suggests stronger agents move the system into a regime where independent performance is already high, reducing the marginal value of shared evidence, while imperfect communication still introduces omissions, spurious reasoning, and selection noise. Thus, stronger base models do not automatically make larger teams better; beyond some point, communication overhead can dominate information-pooling gains.

4. Theoretical motivations: When do multi-agent systems succeed or fail?

CoAct improves over baselines in several settings, but not always—adding agents can sometimes hurt. We now examine a probabilistic framework that offers theoretical grounding for the observed empirical patterns. We analyze a simplified information-pooling model to isolate the information-availability bottleneck identified in Section 2. Let $X_1, \ldots, X_M$ denote the M decision-relevant factors that must be correctly identified, where M serves as a proxy for *information compositionality*. Let F_j denote the event that factor X_j is correctly discovered and processed during a generation phase. We focus on E_{info}, the event where all required factors become correctly available to the team.

Reduced-model assumptions. Formally, let D_{ij} denote the event that agent i discovers and correctly processes factor X_j, with $\Pr(D_{ij}) = q_{ij}(M)$. For tractability, we assume the events $\{D_{ij}\}_{i,j}$ are mutually independent. Under imperfect communication, let A_{ij} denote the event that agent i's correct local discovery of factor X_j becomes team-usable.



(a) GPT-5-mini *(b)* GPT-5.2

Figure 5. **Team-size scaling on BrowseComp.** Accuracy of single agent, naive parallel, debate, and CoAct as team size increases, evaluated on BrowseComp using GPT-5-mini (left) and GPT-5.2 (right).

Table 4. **Probabilistic model quantities and experimental proxies.** Each row maps a theoretical quantity from the probabilistic model to its interpretation and the corresponding experimental proxy used in empirical evaluation.

Quantity	Interpretation	Experimental proxy
K	team size	number of agents
$q(M)$	discovery strength at compositionality M	single-agent accuracy by benchmark
$\epsilon(M)$	processing reliability at compositionality M	single-agent accuracy by benchmark
$\eta(K, M)$	team-usable communication rate under scale	no communication vs. debate vs. shared memory
M	information compositionality	benchmark tasks

We model

$$\Pr(A_{ij}) = a_{ij}(K, M) := \eta_{ij}(K, M) q_{ij}(M) \tag{2}$$

where $\eta_{ij}(K, M) \in [0, 1]$ captures communication loss: even when an agent locally identifies a factor correctly, that information may fail to become usable by the rest of the team. Proofs, alternative aggregation rules, and threshold conditions for this section are deferred to Appendix F.

4.1. No communication

Without communication, success requires at least one agent to independently discover and correctly process all required factors. Under the reduced model,

$$P_{\text{no-comm}}(K, M) = 1 - \prod_{i=1}^{K} \left(1 - \prod_{j=1}^{M} q_{ij}(M) \right). \tag{3}$$

Proposition 4.1 (Monotonicity in team size under no communication)**.** *Fix M and the factorwise success probabilities $\{q_{ij}(M)\}$ of the existing agents. Under the reduced model in (3), $P_{\text{no-comm}}(K, M)$ is monotone non-decreasing in K.*

4.2. Perfect communication

Under perfect communication, every correct local discovery becomes team-usable. The team therefore succeeds if every required factor is discovered by at least one agent. Under the reduced model,

$$P^{\text{perf}}_{\text{comm}}(K, M) = \prod_{j=1}^{M} \left(1 - \prod_{i=1}^{K} (1 - q_{ij}(M)) \right), \tag{4}$$

and the perfect-communication advantage is $\Delta^{\text{perf}}_{K,M} = P^{\text{perf}}_{\text{comm}}(K, M) - P_{\text{no-comm}}(K, M)$.

Proposition 4.2 (Monotonicity in team size under perfect communication)**.** *Fix M and the factorwise success probabilities $\{q_{ij}(M)\}$ of the existing agents. Under the reduced model in (4), $P^{\text{perf}}_{\text{comm}}(K, M)$ is monotone non-decreasing in K.*

Proposition 4.3 (Perfect communication never hurts)**.** *For every K, M, and $\{q_{ij}(M)\}$, $\Delta^{\text{perf}}_{K,M} \geq 0$.*

Theorem 4.4 (Perfect-communication advantage is non–monotone in single-agent factor strength)**.** *Suppose $K > 1$ and $M > 1$. In the symmetric setting $q_{ij}(M) = q$ for all i, j, the function $\Delta^{\text{perf}}_{K,M}(q)$ is non-monotone on $q \in [0, 1]$. In particular, $\Delta^{\text{perf}}_{K,M}(0) = \Delta^{\text{perf}}_{K,M}(1) = 0$, while $\Delta^{\text{perf}}_{K,M}(q) > 0$ for sufficiently small $q > 0$.*

Thus perfect communication helps most at intermediate capability levels: if q is too small, correct discoveries are scarce; if q is too large, the no-communication baseline is already near saturation.

4.3. Imperfect communication

We now relax the idealized communication setting. The quantity $\eta_{ij}(K, M)$ captures loss in turning a correct local identification into team-usable information. Under the reduced model,

$$P^{\text{imp}}_{\text{comm}}(K, M) = \prod_{j=1}^{M} \left(1 - \prod_{i=1}^{K} \big(1 - a_{ij}(K, M)\big) \right), \tag{5}$$

while the corresponding no-communication baseline remains Equation (3).

We define $\Delta^{\text{imp}}_{K,M} = P^{\text{imp}}_{\text{comm}}(K, M) - P_{\text{no-comm}}(K, M)$. Unlike the perfect-communication case, imperfect communication does not admit a uniform improvement guarantee.

Proposition 4.5 (No general monotonicity of imperfect–communication success probability)**.** *When η_{ij} is allowed to depend on K, the quantity $P_{\text{comm}}^{\text{imp}}(K, M)$ is not monotone in general with respect to team size.*

Proposition 4.6 (Imperfect communication can help or hurt)**.** *Under imperfect communication, the advantage of communication may be positive, zero, or negative depending on K, M, $\{q_{ij}(M)\}$, and $\{\eta_{ij}(K, M)\}$.*

To make the dependence on team size and capability explicit, we consider the symmetric setting.

Theorem 4.7 (Imperfect-communication advantage is non–monotone in general)**.** *Consider the symmetric setting where $q_{ij}(M) = q, \eta_{ij}(K, M) = \eta(K, M)$ for all i, j. Then the following hold.*

1. ***Dependence on agent capability.*** *For fixed K, M, and $\eta(K, M)$, $\Delta_{K,M}^{\text{imp}}$ can be non-monotone as a function of the single-agent factor strength q. In particular, communication can be beneficial in weak-capability regimes when effective shared coverage is sufficiently high, but can become harmful in strong-capability regimes when communication is lossy.*

2. ***Dependence on team size.*** *If communication quality $\eta(K, M)$ deteriorates sufficiently with K, then increasing the number of agents does not necessarily improve $P_{\text{comm}}^{\text{imp}}(K, M)$ or $\Delta_{K,M}^{\text{imp}}$.*

Therefore, under imperfect communication, adding more agents does not guarantee positive communication advantage. In weak-agent regimes, communication helps by pooling scarce partial discoveries when effective shared coverage is large enough. In strong-agent regimes, the no-communication baseline is already near saturation, so lossy sharing can become an additional bottleneck; and if communication quality degrades with team size, scaling can further hurt performance. Team composition matters as well: heterogeneity helps when it is *complementary*, not merely different.

5. Related Work

Several works have explored methods for improving single-agent performance (Yao et al., 2022; Shinn et al., 2023; Wang et al., 2023; Stiennon et al., 2020; Snell et al., 2025), but their fundamental limitations (Huang et al., 2024; Kamoi et al., 2024; Stechly et al., 2024) motivate a shift toward multi-agent systems. However, multi-agent systems can be fragile, with failure modes like coordination overhead, error amplification, misalignment, and verification errors (Cemri et al., 2025; Zhang et al., 2020; Khatua et al.; Hammond et al., 2025) highlighting the need for complementary coordination. Empirical works on primarily closed-ended tasks show that performance scaling is non-monotone (Chen et al., 2024; Qian et al., 2025; Kim et al., 2025). We extend this perspective to analyze specific quantities and conditions for positive communication advantage and study stateful, realistic agent settings. A wide range of multi-agent system architectures have been explored, including centralized (Hong et al., 2024; Wang et al., 2025b; Wu et al., 2023), decentralized (Du et al., 2024; Yang et al., 2025; Dang et al., 2025), and automated (Zhuge et al., 2024; Zhang et al., 2025b; Liu et al., 2024; Zhang et al., 2025c; Hu et al., 2024; Ye et al., 2025) designs. However, many approaches introduce bespoke workflows and focus on stateless settings. CoAct aims to provide a fixed, task-agnostic coordination substrate orthogonal to design choices explored in prior systems. Effective multi-agent execution must address coupled design problems: efficient information sharing and safe action commitment. Prior work explores communication topology, content, and timing (Li et al., 2024; Gao & Zhang, 2024; Rezazadeh et al., 2025; Han & Zhang, 2025; Zhang et al., 2025a; Song et al., 2025), and consensus strategies (Wang et al., 2023; Chen et al., 2025; Ai et al., 2025; Lifshitz et al., 2025; Kaesberg et al., 2025) primarily across post-hoc aggregation. CoAct explicitly gates stateful actions with structured consensus, making coordination tractable without a central orchestrator. We provide an extended discussion of related work in Appendix A.

6. Limitations, Conclusions, and Future Work

We introduced CoAct, a minimal coordination substrate for decentralized tool-using LLM agents. CoAct uses compact shared memory to make local discoveries team-usable and synchronized commitment to prevent conflicting stateful actions. Across diverse benchmarks, CoAct with two to three agents improves over single-agent execution, naive parallelism, and unstructured debate, while using communication more token-efficiently. However, scaling is non-monotonic: larger teams, stronger models, and different task structures can change whether communication helps or hurts.

This work has several limitations. Our analysis assumes independent agent discoveries $\{D_{ij}\}$, while real agents often share models, tools, prompts, and search spaces, inducing correlated failures. Compositionality M is an abstraction rather than an observable task property, and communication loss $\eta(K, M)$ collapses many failure modes into one quantity. Empirically, our evaluation covers a limited set of models and mostly homogeneous teams.

Future work may improve communication and synchronization by learning when and what agents should share, designing better retrieval over shared memory, developing adaptive intent selection, and post-training agents to identify conflicts, defer premature commitments, and produce more useful shared messages.

References

Ai, R., Pan, Y., Simchi-Levi, D., Tambe, M., and Xu, H. Beyond majority voting: Llm aggregation by leveraging higher-order information, 2025. URL https://arxiv.org/abs/2510.01499.

Barres, V., Dong, H., Ray, S., Si, X., and Narasimhan, K. τ^2-bench: Evaluating conversational agents in a dual-control environment, 2025. URL https://arxiv.org/abs/2506.07982.

Bigeard, A., Nashold, L., Krishnan, R., and Wu, S. Finance agent benchmark: Benchmarking llms on real-world financial research tasks. Vals AI, May 2025. URL https://arxiv.org/abs/2508.00828.

Cemri, M., Pan, M. Z., Yang, S., Agrawal, L. A., Chopra, B., Tiwari, R., Keutzer, K., Parameswaran, A., Klein, D., Ramchandran, K., Zaharia, M., Gonzalez, J. E., and Stoica, I. Why do multi-agent llm systems fail?, 2025. URL https://arxiv.org/abs/2503.13657.

Chen, J. C.-Y., Saha, S., and Bansal, M. Reconcile: Round-table conference improves reasoning via consensus among diverse llms. In *The 62nd Annual Meeting of the Association for Computational Linguistics*, 2025. URL https://arxiv.org/abs/2309.13007.

Chen, L., Davis, J. Q., Hanin, B., Bailis, P., Stoica, I., Zaharia, M., and Zou, J. Are more llm calls all you need? towards scaling laws of compound inference systems. In *NeurIPS*, 2024. URL https://arxiv.org/abs/2403.02419.

Dang, Y., Qian, C., Luo, X., Fan, J., Xie, Z., Shi, R., Chen, W., Yang, C., Che, X., Tian, Y., Xiong, X., Han, L., Liu, Z., and Sun, M. Multi-agent collaboration via evolving orchestration. In *The 42nd International Conference on Machine Learning*, 2025. URL https://arxiv.org/abs/2505.19591.

Du, Y., Li, S., Torralba, A., Tenenbaum, J. B., and Mordatch, I. Improving factuality and reasoning in language models through multiagent debate. In *The 41st International Conference on Machine Learning*, 2024. URL https://openreview.net/pdf?id=zj7YuTE4t8.

Gao, H. and Zhang, Y. Memory sharing for large language model based agents, 2024. URL https://arxiv.org/pdf/2404.09982.

Hammond, L., Chan, A., Clifton, J., Hoelscher-Obermaier, J., Khan, A., McLean, E., Smith, C., Barfuss, W., Foerster, J., Gavenciak, T., Han, T. A., Hughes, E., Kovark, V., Kulveit, J., Leibo, J. Z., Oesterheld, C., de Witt, C. S., Shah, N., Wellman, M., Bova, P., Cimpeanu, T., Ezell, C., Feuillade-Montixi, Q., Franklin, M., Kran, E., Krawczuk, I., Lamparth, M., Lauffer, N., Meinke, A., Motwani, S., Reuel, A., Conitzer, V., Dennis, M., Gabriel, I., Gleave, A., Hadfield, G., Haghtalab, N., Kasirzadeh, A., Krier, S., Larson, K., Lehman, J., Parkes, D. C., Piliouras, G., and Rahwan, I. Multi-agent risks from advanced ai, 2025. URL https://arxiv.org/pdf/2502.14143.

Han, B. and Zhang, S. Exploring advanced llm multi-agent systems based on blackboard architecture, 2025. URL https://arxiv.org/pdf/2507.01701.

Hong, S., Zhuge, M., Chen, J., Zheng, X., Cheng, Y., Wang, J., Zhang, C., Wang, Z., Yau, S. K. S., Lin, Z., Zhou, L., Ran, C., Xiao, L., Wu, C., and Schmidhuber, J. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VtmBAGCN7o.

Hu, Y., Cai, Y., Du, Y., Zhu, X., Liu, X., and Yu, Z. Self-evolving multi-agent collaboration networks for software development, 2024. URL https://arxiv.org/html/2410.16946v1.

Huang, J., Chen, X., Mishra, S., Zheng, H. S., Yu, A. W., Song, X., and Zhou, D. Large language models cannot self-correct reasoning yet. In *ICLR*, 2024. URL https://arxiv.org/abs/2310.01798.

Kaesberg, L. B., Becker, J., Wahle, J. P., Ruas, T., and Gipp, B. Voting or consensus? decision-making in multi-agent debate. In *Findings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025. URL https://arxiv.org/pdf/2502.19130.

Kamoi, R., Zhang, Y., Zhang, N., Han, J., and Zhang, R. When can llms actually correct their own mistakes? a critical survey of self-correction of llms. In *TACL*, 2024. URL https://arxiv.org/abs/2406.01297.

Khatua, A., Zhu, H., Tran, P., Prabhudesai, A., Sadrieh, F., Lieberwirth, J. K., Yu, X., Fu, Y., Ryan, M. J., Pei, J., and Yang, D. Cooperbench: Why coding agents cannot be your teammates yet. URL https://cooperbench.com/.

Kim, Y., Gu, K., Park, C., Park, C., Schmidgall, S., Heydari, A. A., Yan, Y., Zhang, Z., Zhuang, Y., Malhotra, M., et al. Towards a science of scaling agent systems. *arXiv preprint arXiv:2512.08296*, 2025.

Li, G., Hammoud, H., Itani, H., Khizbullin, D., and Ghanem, B. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in neural information processing systems*, 36:51991–52008, 2023.

Li, Y., Du, Y., Zhang, J., Hou, L., Grabowski, P., Li, Y., and Ie, E. Improving multi-agent debate with sparse communication topology. In *EMNLP Findings*, 2024. URL https://aclanthology.org/2024.findings-emnlp.427.pdf.

Lifshitz, S., McIlraith, S. A., and Du, Y. Multi-agent verification: Scaling test-time compute with multiple verifiers, 2025. URL https://arxiv.org/pdf/2502.20379.

Liu, Z., Zhang, Y., Li, P., Liu, Y., and Yang, D. A dynamic llm-powered agent network for task-oriented agent collaboration. In *COLM*, 2024. URL https://arxiv.org/abs/2310.02170.

Patil, S. G., Zhang, T., Wang, X., and Gonzalez, J. E. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37: 126544–126565, 2024.

Qian, C., Liu, W., Liu, H., Chen, N., Dang, Y., Li, J., Yang, C., Chen, W., Su, Y., Cong, X., et al. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 15174–15186, 2024.

Qian, C., Xie, Z., Wang, Y., Liu, W., Zhu, K., Xia, H., Dang, Y., Du, Z., Chen, W., Yang, C., Liu, Z., and Sun, M. Scaling large language model-based multi-agent collaboration. In *ICLR*, 2025. URL https://arxiv.org/abs/2406.07155.

Rezazadeh, A., Li, Z., Lou, A., Zhao, Y., Wei, W., and Bao, Y. Collaborative memory: Multi-user memory sharing in llm agents with dynamic access control, 2025. URL https://arxiv.org/pdf/2505.18279.

Shinn, N., Cassano, F., Berman, E., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2303.11366.

Snell, C., Lee, J., Xu, K., and Kumar, A. Scaling llm test-time compute optimally can be more effective than scaling model parameters. In *ICLR*, 2025. URL https://arxiv.org/abs/2408.03314.

Song, X., Wang, Z., Wu, S., Shi, T., and Ai, L. Gradientsys: A multi-agent llm scheduler with react orchestration, 2025. URL https://arxiv.org/abs/2507.06520.

Stechly, K., Valmeekam, K., and Kambhampati, S. Chain of thoughtlessness? an analysis of cot in planning. In *NeurIPS*, 2024. URL https://arxiv.org/abs/2405.04776.

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D. M., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. Learning to summarize from human feedback. In *NeurIPS*, 2020. URL https://arxiv.org/abs/2009.01325.

Subramaniam, V., Du, Y., Tenenbaum, J. B., Torralba, A., Li, S., and Mordatch, I. Multiagent finetuning: Self improvement with diverse reasoning chains. *arXiv preprint arXiv:2501.05707*, 2025.

Wang, J., Chen, K., Song, X., Zhang, K., Ai, L., Yang, E., and Shi, B. Symphony: A decentralized multi-agent framework for scalable collective intelligence. *arXiv preprint arXiv:2508.20019*, 2025a.

Wang, Q., Wang, T., Tang, Z., Li, Q., Chen, N., Liang, J., and He, B. Megaagent: A large-scale autonomous llm-based multi-agent system without predefined sops. In *Findings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025b.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. In *The 11th International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2203.11171.

Wei, J., Sun, Z., Papay, S., McKinney, S., Han, J., Fulford, I., Won Chung, H., Tachard Passos, L., Fedus, W., and Glaese, A. Browsecomp: A simple yet challenging benchmark for browsing agents, April 2025. URL https://arxiv.org/abs/2504.12516.

Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., Jiang, L., Zhang, X., Zhang, S., Liu, J., Hassan, Awadallah, A. H., White, R. W., Burger, D., and Wang, C. Autogen: Enabling next-gen llm applications via multi-agent conversation, 2023. URL https://arxiv.org/abs/2308.08155.

Xie, T., Zhang, D., Chen, J., Li, X., Zhao, S., Cao, R., Hua, T. J., Cheng, Z., Shin, D., Lei, F., et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

Yadav, A., Black, S., and Sourbut, O. More capable, less cooperative? when llms fail at zero-cost collaboration. *arXiv preprint arXiv:2604.07821*, 2026.

Yang, J., Jimenez, C. E., Wettig, A., Lieret, K., Yao, S., Narasimhan, K., and Press, O. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

Yang, Y., Chai, H., Shao, S., Song, Y., Qi, S., Rui, R., and Zhang, W. Agentnet: Decentralized evolutionary coordination for llm-based multi-agent systems. In Belgrave, D., Zhang, C., Lin, H., Pascanu, R., Koniusz, P., Ghassemi, M., and Chen, N. (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 107309–107336. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/9a379c1b05793d1c42dc832269834515-Paper-Conference.pdf.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. R., and Cao, Y. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022.

Ye, R., Tang, S., Ge, R., Du, Y., Yin, Z., Chen, S., and Shao, J. Mas-gpt: Training llms to build llm-based multi-agent systems, 2025. URL https://arxiv.org/pdf/2503.03686.

Zhang, E., Zhu, E., Bansal, G., Fourney, A., Mozannar, H., and Gerrits, J. Optimizing sequential multi-step tasks with parallel llm agents, 2025a. URL https://arxiv.org/abs/2507.08944.

Zhang, G., Yue, Y., Sun, X., Wan, G., Yu, M., Fang, J., Wang, K., Chen, T., and Cheng, D. G-designer: Architecting multi-agent communication topologies via graph neural networks. In *The 42nd International Conference on Machine Learning*, 2025b. URL https://arxiv.org/abs/2410.11782.

Zhang, J., Xiang, J., Yu, Z., Teng, F., Chen, X., Chen, J., Zhuge, M., Cheng, X., Hong, S., Wang, J., Zheng, B., Liu, B., Luo, Y., and Wu, C. Aflow: Automating agentic workflow generation. In *ICLR*, 2025c. URL https://arxiv.org/abs/2410.10762.

Zhang, K., Yang, Z., and Başar, T. Multi-agent reinforcement learning: A selective overview of theories and algorithms. In Bertsekas, D. P., Borkar, V., et al. (eds.), *Handbook of Reinforcement Learning and Control*, Studies in Systems, Decision and Control. Springer, 2020. Invited survey.

Zhuge, M., Wang, W., Kirsch, L., Faccio, F., Khizbullin, D., and Schmidhuber, J. Gptswarm: Language agents as optimizable graphs. In *The 41st International Conference on Machine Learning*, 2024. URL https://arxiv.org/html/2402.16823v3.

A. Extended Related Work

A.1. Single-Agent Reasoning and Test-Time Scaling

A natural starting point is the single-agent reasoning paradigm. Frameworks like ReAct (Yao et al., 2022) interleave chain-of-thought reasoning with tool invocation, enabling agents to plan, gather external information, and act within an environment over multiple steps. Reflexion (Shinn et al., 2023) extends this with verbal self-feedback across episodes, allowing agents to incorporate past failures into subsequent attempts. A parallel line of work has explored scaling inference-time computation within a single agent: self-consistency (Wang et al., 2023) samples multiple reasoning paths and selects the most frequent answer, best-of-N sampling (Stiennon et al., 2020) generates N candidate responses and selects among them using a verifier or reward model, and Snell et al. (2025) show that optimal allocation of test-time compute can outperform models 14x larger. However, both paradigms face limitations on complex tasks: Huang et al. (2024); Kamoi et al. (2024) show LLMs cannot reliably self-correct reasoning errors without external feedback, while Stechly et al. (2024) finds chain-of-thought gains are confined to demonstrations narrowly tailored to the problem class, with performance degrading rapidly on out-of-distribution instances. Test-time scaling methods share a related limitation: compute benefit is bounded by what a single agent can discover in isolation. These limitations motivate a shift toward multi-agent systems, where inputs from other agents can provide external corrective signals unavailable through self-reflection alone.

A.2. Formal Analysis of Multi-Agent Systems and Multi-Agent Failures

Theoretical and empirical work has studied the properties of scaling multi-agent systems. Chen et al. (2024) show that performance under majority voting is non-monotone in the number of calls: adding more agents initially improves accuracy but eventually degrades it, with the inflection point determined by query difficulty. Qian et al. (2025) show in an orchestrated system that performance grows logistically as agents scale, with performance eventually saturating. Both results are established in settings where agents operate independently or in simple topologies, and where the task is closed-ended with a single answer to verify. Our framework extends this perspective to settings where agents interact with a stateful external environment over multiple steps, where the outcome depends not only on how many agents attempt a task but on whether their partial discoveries can be shared and combined effectively. A second body of work empirically derives scaling laws: Kim et al. (2025) evaluate 260 configurations across different coordination architectures and find that coordination yields diminishing or negative returns once single-agent accuracy exceeds roughly 45%, independent agents amplify errors at much higher rates than coordinated ones, and performance scaling is strongly architecture- and task-dependent. Our framework provides an analytical expression for these dependencies, making explicit conditions for positive communication advantages. Several other works have examined failure modes of multi-agent systems, identifying coordination overhead, error amplification, inter-agent misalignment, and task verification errors (Cemri et al., 2025; Zhang et al., 2020). For example, Khatua et al. finds that teams of coding agents perform worse than individual agents due to 1) vague, ill-timed, and inaccurate messages flooding communication channels, 2) discrepancies between agent commitments and actions, and 3) incorrect agent expectations about others' plans and observations. Hammond et al. (2025) examines risks and case studies across miscoordination, conflict, and collusion. These failure modes underscore the fragility of many multi-agent systems, and the need for safe and robust coordination frameworks that allow agents to complement rather than impede one another.

A.3. LLM Multi-Agent Architectures

A wide range of multi-agent system architectures have been explored, differing in how control and communication are specified.

Manually designed systems. Centralized frameworks like MetaGPT (Hong et al., 2024) and MegaAgent (Wang et al., 2025b) encode fixed role assignments and sequenced pipelines, with an orchestrator responsible for task decomposition and output aggregation. AutoGen (Wu et al., 2023) provides a conversation-centric programming interface through which developers compose agents with configurable interaction patterns, tool access, and human-in-the-loop support. These systems demonstrate strong performance in some structured domains, but reliance on specified workflows means coordination logic must be redesigned for each new task class.

Decentralized and peer-to-peer systems. An alternative line of work removes the central orchestrator in favor of peer interaction. Du et al. (2024) show that iterative debate among agents can improve factual accuracy and reasoning on closed-ended tasks. AgentNet (Yang et al., 2025) and related systems (Dang et al., 2025) explore dynamic topologies and

hybrid coordination patterns. These approaches are more flexible but often depend on unconstrained full message exchange and synchronous turn-taking, which introduces high communication overhead that scales poorly as agent count grows.

Automated system design. A third line of work treats the MAS structure itself as an optimization target. GPTSwarm (Zhuge et al., 2024) represents agents as nodes in a graph and uses RL to optimize communication edges. G-Designer (Zhang et al., 2025b) extends this with graph neural networks over agent topology. DyLAN (Liu et al., 2024) selects participating agents based on performance in a preliminary trial. AFlow (Zhang et al., 2025c) reformulates workflow generation as a code-level search problem, iteratively refining agent programs from execution feedback. EvoMAC (Hu et al., 2024) applies a textual analogue of backpropagation to iteratively update both agent prompts and inter-agent connections during test time; this yields strong results on software benchmarks but requires a task-specific verification oracle to generate feedback. MAS-GPT (Ye et al., 2025) trains a model to generate executable multi-agent programs in a single forward pass conditioned on the task description. These methods reduce human design effort and introduce flexibility but introduce their own constraints: optimization-based approaches are run per task class, learned generators require supervised fine-tuning data, and these frameworks alone do not address coordinating agents that must act on a shared stateful environment. CoAct aims to provide a fixed, task-agnostic coordination substrate that any task can use without modification. Its primary architectural contributions—differentiating stateless and stateful tools, isolating the minimal communication components, and gating stateful actions through collective consensus—are orthogonal to the design choices explored in the above systems.

A.4. Coordination, Concurrency, Consensus, and Commitment

Effective multi-agent execution must address coupled design problems: how agents share information during parallel execution, and how they commit to actions—particularly irreversible ones—without conflicting or duplicating effort.

Communication. Several axes govern information sharing: topology (who talks to whom), content (full reasoning traces vs condensed artifacts), and timing (synchronous rounds vs asynchronous posting). Li et al. (2024) find that sparse communication topologies perform comparably to fully connected ones, Gao & Zhang (2024) aggregate agent interactions into a shared retrievable memory that agents can query during in-context learning, and Rezazadeh et al. (2025) separate private from selectively shared memories to control what info propagates across agents. On timing, Han & Zhang (2025) use a blackboard architecture for synchronous task selection among agents, (Zhang et al., 2025a) run multiple agent teams in parallel with async event-driven messaging to exploit plan diversity and reduce latency, and (Song et al., 2025) combine synchronous and synchronous execution with a retry-and-replan mechanism to handle failures.

Consensus. A separate body of work studies how to aggregate the outputs of multiple agents into a single decision. Self-consistency (Wang et al., 2023) selects the most frequent answer across sampled reasoning chains. Reconcile (Chen et al., 2025) uses multi-agent debate to reach consensus among diverse models. More recent work explores weighted aggregation using confidence signals (Ai et al., 2025) and combinations of multiple verifiers (Lifshitz et al., 2025). Kaesberg et al. (2025) systematically compare voting and debate as consensus mechanisms. These methods improve accuracy on structured reasoning benchmarks, but operate at the level of post-hoc output aggregation: each agent completes an independent trajectory and a final answer is selected afterward. They do not address the problem of coordinating agents that must act during execution in a shared environment where actions have side effects and cannot be freely retried. CoAct explicitly gates stateful actions with a structured proposal and vote before execution, making coordination tractable without requiring a central orchestrator.

B. Aggregators

Table 5. Aggregation rules used for synchronized stateful execution.

Rule	Definition	Role
Agreement	Select a candidate only when all non-stopping agents produce the same candidate; otherwise return $\varnothing$ and defer commitment.	Deployable
Majority	Select the candidate with the most support among agents' candidates; ties are broken by a fixed deterministic rule.	Deployable
Ranked	Each agent ranks the candidate set; the aggregator selects the candidate with the best aggregate rank, with ties broken by a fixed deterministic rule.	Deployable
Random	Uniformly select one candidate from the candidate set.	Reference
Oracle	Post-hoc select a successful candidate if one exists; otherwise return $\varnothing$.	Reference

C. CoAct Execution Loop

Algorithm 1 CoAct Execution Loop

K agents $\{a_1, \ldots, a_K\}$, input x, stateless tools $\mathcal{T}_{\text{stateless}}$, stateful tools $\mathcal{T}_{\text{stateful}}$, shared-message budget B, aggregation rule A Final observation or STOP Initialize shared memory $\mathcal{C} \leftarrow \emptyset$ Initialize candidate set $\mathcal{Y} \leftarrow \emptyset$ each agent a_i **in parallel** reason over input x, local history, and shared memory $\mathcal{C}$ use tools in $\mathcal{T}_{\text{stateless}}$ and observe local results optionally call <share>(m_i) with $|m_i| \leq B$, updating $\mathcal{C}$ agent proposes candidate y_i for a stateful tool or STOP a_i proposes a candidate y_i $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{y_i\}$ $\mathcal{Y} = \emptyset$ **continue** $y^\star \leftarrow A(\mathcal{Y}, \mathcal{C})$ $y^\star \neq \varnothing$ submit $y^\star$ through the corresponding stateful tool, observe o, and broadcast o to all agents termination

D. Full Execution Formalization

All notation is consistent with the main text. We begin by defining the environment model that governs the full execution process, then walk through the Generate and Synchronize phases in order, introducing each formal object when it first becomes relevant.

D.1. Interaction Process

We model a decentralized tool-using task as

$$\mathcal{M} = \big(\mathcal{S},\ \{\Omega_i\}_{i=1}^{K},\ \mathcal{T}_{\text{stateless}},\ \mathcal{T}_{\text{stateful}},\ P,\ \{O_i\}_{i=1}^{K}\big),$$

where $\mathcal{S}$ is the shared latent state space, Ω_i is the observation space of agent i, P is the environment transition kernel, and O_i is the observation kernel for agent i. At each step t, the latent state is $s_t \in \mathcal{S}$ and agent i receives observation $o_t^i \sim O_i(\cdot \mid s_t)$. To connect s_t to the information compositionality quantity M from the main text, we write $s_t = (e_t, x_t)$ and $x_t = (X_{1,t}, \ldots, X_{M,t})$, where e_t denotes the remaining environment state and $X_{j,t}$ denotes the j-th decision-relevant factor. Examples include a key source and target fact in a browsing task, a financial metric and modeling assumption in a financial analysis task, or a user constraint, policy rule, and valid next action in a stateful user-assistance task. E_{info} is therefore the event that all $X_{j,t}$ become correctly available to the team before commitment. The interaction process alternates between two phases, Generate and Synchronize, which we now formalize in turn.

D.2. Generate

At the start of each generation round, each agent i holds a *local information state*

$$h_t^i = \big(q,\ \tau_{1:t}^i,\ \mathcal{C}_{1:t},\ u_{1:t-1}\big),$$

where q is the task input, $\tau_{1:t}^i$ is agent i's private trajectory of reasoning, stateless tool calls, and local observations, $\mathcal{C}_{1:t}$ is the shared-memory history available up to round t, and $u_{1:t-1}$ are the previously committed stateful actions and their broadcast

observations. No agent's h_t^i contains another agent's private trajectory $\tau_{1:t}^{i'}$; all inter-agent information enters exclusively through $\mathcal{C}_{1:t}$. Conditioning on h_t^i, each agent independently follows a THINK $\rightarrow$ ACT $\rightarrow$ OBSERVE loop restricted to stateless tools $\mathcal{T}_{\text{stateless}}$, which produce local observations without modifying s_t in a way that affects other agents.

Shared memory dynamics. In addition to stateless tool calls, each agent may call `<share>`(m_t^i) to append a message to shared memory. The update rule is $\mathcal{C}_{t+1} = \mathcal{C}_t \cup \{m_t^i\}$; messages from multiple agents writing in the same round are unioned into $\mathcal{C}_{t+1}$. The token budget B bounds each m_t^i, directly governing communication fidelity: larger B increases the probability that a correct local discovery survives compression into a team-usable message, while $B \ll L$ risks lossy truncation, where L is the mean per-agent trajectory length.

Discovery and communication. Let $\hat{X}_{j,t}^i$ denote agent i's estimate of $X_{j,t}$ formed from h_t^i. A *correct discovery* occurs when $\hat{X}_{j,t}^i = X_{j,t}$; an *incorrect discovery* occurs when $\hat{X}_{j,t}^i \neq X_{j,t}$. A correct discovery contributes to E_{info} only if it is written to $\mathcal{C}$ in a form that other agents can condition on: decentralized execution increases the probability that some agent correctly identifies a factor, but the team benefits only if communication preserves it through the budget constraint.

D.3. Synchronize

When generation reaches a point where a stateful action may be needed, each agent proposes a candidate $y_t^i \in \mathcal{Y}_{\text{stateful}} \cup \{\varnothing\}$, where $\mathcal{Y}_{\text{stateful}}$ denotes the set of stateful tools and their valid inputs. The non-empty candidate set is $\mathcal{Y}_t = \{y_t^i : y_t^i \neq \varnothing,\ i \in [K]\}$. A synchronization module applies the aggregation rule

$$y_t^\star = A(\mathcal{Y}_t, \mathcal{C}_t), \qquad A : 2^{\mathcal{Y}_{\text{stateful}}} \times \mathfrak{C} \rightarrow \mathcal{Y}_{\text{stateful}} \cup \{\varnothing\},$$

where $\mathfrak{C}$ denotes the space of shared memory histories. If $y_t^\star \neq \varnothing$, the selected tool call is executed once, the shared environment updates according to $s_{t+1} \sim P(\cdot \mid s_t, u_t)$ where $u_t = y_t^\star$, and the resulting observation is broadcast to all agents and incorporated into their subsequent h_t^i as part of $u_{1:t}$. If $y_t^\star = \varnothing$, the team defers commitment and continues generation. The choice of A directly governs $\Pr(E_{\text{sync}} \mid E_{\text{info}})$: a well-designed aggregator recovers the correct intent from the candidate set; a poorly designed one discards it.

Termination. We designate a terminal action $u^{\text{term}} \in \mathcal{T}_{\text{stateful}}$ that commits the final answer. If $y_t^\star$ invokes this action, execution terminates:

$$y_t^\star \mapsto u^{\text{term}} \ \Rightarrow\ s_{t+1} \in \mathcal{S}_{\text{term}}.$$

The environment may also terminate through external stopping conditions such as a maximum step budget, context limit, or task-level failure condition. In both cases, task success is evaluated against the terminal state s_{t+1}.

E. Implementation Details

Models. HMMT experiments use GPT-4.1-mini (`gpt-4.1-mini-2025-04-14`). BrowseComp main results use GPT-5-mini; BrowseComp scaling experiments additionally evaluate GPT-5.2 (`gpt-5.2-2025-12-11`). AIME 25' experiments use GPT-4.1 (text-only baseline) and GPT-4o (ReAct, CoAct, naive parallel). FinanceAgent main results use GPT-4.1-mini; the GPT-5 scaling condition uses GPT-5.2. τ^2-retail experiments use GPT-5-mini. For all GPT models, we use the standard OpenAI API to generate text.

Sampling and generation. All agents sample at temperature 0.7. Maximum generation length is $4{,}096$ tokens per response for web-research tasks (BrowseComp, FinanceAgent) and $2{,}048$ tokens otherwise.

Step budgets. Each agent is allocated 20 ReAct iterations per orchestrator step (50 for BrowseComp). The orchestrator runs for at most 50 total steps (15 for BrowseComp; 100 for τ^2-retail to accommodate the multi-turn interaction structure). All multi-agent methods receive the same per-agent budget; total compute scales linearly with K.

Debate configuration. Debate implements the Du et al. (2024) protocol: agents reason independently in parallel each step, then each receives the full reasoning traces of all peers appended to its conversation history before the next step. Main results use first-wins intent selection with no minimum round constraint. Token-sweep comparisons (§3.3) enforce a minimum round budget R before any answer submission is permitted, using majority selection; $R \in \{2, 5, 10\}$ for $K \in \{2, 3, 4\}$

agents respectively, with the total step budget set to $R + 10$ to allow answer submission once the minimum is satisfied. This gives debate a controlled per-agent communication budget of $O(K \cdot R)$ peer-context tokens before commitment.

F. Appendix for Section 4

F.1. Alternative aggregation rules

The main text adopts the any-agent aggregation rule: once the required factor values are available, an instance is counted as correct if at least one agent can produce the correct final answer. This is an optimistic upper bound on downstream decision quality. Other aggregation rules can behave differently.

One simple alternative is a random-agent rule, where after all agents finish, one agent is selected uniformly at random and its answer is submitted. Under no communication, the corresponding success probability is

$$P_{\text{rand,no-comm}}(K, M) = \frac{1}{K} \sum_{i=1}^{K} \prod_{j=1}^{M} q_{ij}(M). \tag{6}$$

Unlike the any-agent rule, this quantity is not necessarily monotone in K: adding weak agents can dilute a strong one.

F.2. Proof of Proposition 4.1

Write

$$p_i(M) := \prod_{j=1}^{M} q_{ij}(M), \qquad P_{\text{no-comm}}(K, M) = 1 - \prod_{i=1}^{K} (1 - p_i(M)).$$

Then

$$P_{\text{no-comm}}(K + 1, M) = 1 - (1 - p_{K+1}(M)) \prod_{i=1}^{K} (1 - p_i(M)).$$

Since $1 - p_{K+1}(M) \in [0, 1]$, we have

$$(1 - p_{K+1}(M)) \prod_{i=1}^{K} (1 - p_i(M)) \leq \prod_{i=1}^{K} (1 - p_i(M)),$$

hence

$$P_{\text{no-comm}}(K + 1, M) \geq P_{\text{no-comm}}(K, M).$$

This proves monotonicity.

F.3. Perfect communication: proofs and asymptotics

F.3.1. Proof of Proposition 4.2

For each factor j, define

$$r_j^{(K)}(M) := 1 - \prod_{i=1}^{K} (1 - q_{ij}(M)).$$

Then

$$r_j^{(K+1)}(M) = 1 - (1 - q_{K+1,j}(M)) \prod_{i=1}^{K} (1 - q_{ij}(M)) \geq r_j^{(K)}(M).$$

Since

$$P_{\text{comm}}^{\text{perf}}(K, M) = \prod_{j=1}^{M} r_j^{(K)}(M),$$

and each factorwise term is non-decreasing in K, $P_{\text{comm}}^{\text{perf}}(K, M)$ is monotone non-decreasing in K.

F.3.2. Proof of Proposition 4.3

Let

$$A_i := \{\text{agent } i \text{ discovers all } M \text{ required factors}\}, \qquad A := \bigcup_{i=1}^{K} A_i.$$

Let

$$B_j := \{\text{at least one agent discovers factor } F_j\}, \qquad B := \bigcap_{j=1}^{M} B_j.$$

If some agent discovers all required factors, then for every j, at least one agent discovers factor F_j. Hence $A \subseteq B$, and therefore

$$P_{\text{no-comm}}(K, M) = \Pr(A) \le \Pr(B) = P^{\text{perf}}_{\text{comm}}(K, M).$$

Thus $\Delta^{\text{perf}}_{K,M} \ge 0$.

F.3.3. Proof of Theorem 4.4

In the symmetric setting $q_{ij}(M) = q$,

$$P^{\text{perf}}_{\text{comm}}(K, M) = \big(1 - (1-q)^K\big)^M, \qquad P_{\text{no-comm}}(K, M) = 1 - \big(1 - q^M\big)^K. \tag{7}$$

Hence

$$\Delta^{\text{perf}}_{K,M}(q) = \big(1 - (1-q)^K\big)^M - \Big[1 - \big(1 - q^M\big)^K\Big]. \tag{8}$$

At the endpoints, $\Delta^{\text{perf}}_{K,M}(0) = 0$ and $\Delta^{\text{perf}}_{K,M}(1) = 0$. In the low-$q$ regime,

$$(1-q)^K = 1 - Kq + O(q^2), \qquad (1 - q^M)^K = 1 - Kq^M + O(q^{2M}),$$

so

$$P^{\text{perf}}_{\text{comm}}(K, M) = (Kq)^M + O(q^{M+1}), \qquad P_{\text{no-comm}}(K, M) = Kq^M + O(q^{2M}), \tag{9}$$

and therefore

$$\Delta^{\text{perf}}_{K,M}(q) = q^M\big(K^M - K\big) + o(q^M). \tag{10}$$

For $K > 1$ and $M > 1$, $K^M - K > 0$, so $\Delta^{\text{perf}}_{K,M}(q) > 0$ for all sufficiently small $q > 0$. Since $\Delta^{\text{perf}}_{K,M}(0) = \Delta^{\text{perf}}_{K,M}(1) = 0$ and the function is continuous, it cannot be monotone on $[0, 1]$.

F.4. Imperfect communication: basic facts

F.4.1. Proof of Proposition 4.5

It suffices to give a counterexample. Consider the symmetric setting with any fixed $M \ge 1$, fixed $q \in (0, 1]$, and

$$\eta(K, M) = \frac{c}{K^2}, \qquad c \in (0, 1].$$

Then

$$P^{\text{imp}}_{\text{comm}}(K, M) = \left(1 - \left(1 - \frac{cq}{K^2}\right)^K\right)^M.$$

For $K = 1$,

$$P^{\text{imp}}_{\text{comm}}(1, M) = (cq)^M.$$

For $K = 2$,

$$P^{\text{imp}}_{\text{comm}}(2, M) = \left(1 - \left(1 - \frac{cq}{4}\right)^2\right)^M = \left(\frac{cq}{2} - \frac{c^2q^2}{16}\right)^M.$$

Because

$$\frac{cq}{2} - \frac{c^2q^2}{16} < cq,$$

we have

$$P^{\text{imp}}_{\text{comm}}(2, M) < P^{\text{imp}}_{\text{comm}}(1, M).$$

Thus $P^{\text{imp}}_{\text{comm}}(K, M)$ is not monotone in K in general.

F.4.2. Proof of Proposition 4.6

We show that $\Delta^{\text{imp}}_{K,M}$ can be positive, negative, or zero by explicit examples in the symmetric setting with $K = 2$ and $M = 2$. In this setting,

$$P^{\text{imp}}_{\text{comm}} = \big(1 - (1 - \eta q)^2\big)^2, \qquad P_{\text{no-comm}} = 1 - (1 - q^2)^2.$$

Positive advantage. Let $q = 0.1$ and $\eta = 0.9$. Then

$$P^{\text{imp}}_{\text{comm}} = \big(1 - 0.91^2\big)^2 \approx 0.0295, \qquad P_{\text{no-comm}} = 1 - 0.99^2 \approx 0.0199.$$

Hence $\Delta^{\text{imp}}_{K,M} > 0$.

Negative advantage. Let $q = 1$ and $\eta = 0.5$. Then

$$P^{\text{imp}}_{\text{comm}} = \big(1 - 0.5^2\big)^2 = 0.5625, \qquad P_{\text{no-comm}} = 1.$$

Hence $\Delta^{\text{imp}}_{K,M} < 0$.

Zero advantage. For a nontrivial zero-advantage example, fix $q = 0.5$ and view the difference as a continuous function of $\eta \in [0, 1]$:

$$g(\eta) = \big(1 - (1 - \eta q)^2\big)^2 - \big[1 - (1 - q^2)^2\big].$$

At $\eta = 0$, $g(0) < 0$. At $\eta = 1$, Proposition 4.3 gives $g(1) > 0$ for $K = 2$, $M = 2$, and $q = 0.5$. By continuity, there exists $\eta^\star \in (0, 1)$ such that $g(\eta^\star) = 0$. Therefore $\Delta^{\text{imp}}_{K,M}$ may also be zero.

F.5. Proof of Theorem 4.7

In the symmetric setting,

$$a_{ij}(K, M) = \eta(K, M)q.$$

Hence

$$P^{\text{imp}}_{\text{comm}}(K, M) = \big(1 - (1 - \eta(K, M)q)^K\big)^M, \tag{11}$$

and

$$P_{\text{no-comm}}(K, M) = 1 - \big(1 - q^M\big)^K. \tag{12}$$

Therefore

$$\Delta^{\text{imp}}_{K,M}(q) = \big(1 - (1 - \eta(K, M)q)^K\big)^M - \Big[1 - \big(1 - q^M\big)^K\Big]. \tag{13}$$

Low-q regime. Fix K, M, and $\eta = \eta(K, M)$. As $q \to 0$,

$$(1 - \eta q)^K = 1 - K\eta q + O(q^2), \qquad (1 - q^M)^K = 1 - Kq^M + O(q^{2M}).$$

Thus

$$P^{\text{imp}}_{\text{comm}}(K, M) = (K\eta q)^M + O(q^{M+1}), \qquad P_{\text{no-comm}}(K, M) = Kq^M + O(q^{2M}), \tag{14}$$

and

$$\Delta^{\text{imp}}_{K,M}(q) = q^M\big((K\eta)^M - K\big) + o(q^M). \tag{15}$$

Therefore communication is beneficial for sufficiently small positive q iff

$$(K\eta)^M > K, \tag{16}$$

equivalently $\eta > K^{1/M-1}$.

High-q regime. As $q \to 1$,

$$P_{\text{no-comm}}(K, M) \to 1.$$

Meanwhile,

$$P^{\text{imp}}_{\text{comm}}(K, M) \to \big(1 - (1 - \eta)^K\big)^M. \tag{17}$$

Whenever $\eta < 1$, the right-hand side of (17) is strictly below 1. Hence $\Delta^{\text{imp}}_{K,M}(q) < 0$ for all sufficiently large q close to 1.

Non-monotonicity in factor strength. Assume $K > 1$, $M > 1$, and $K^{1/M-1} < \eta < 1$. The lower bound on η is equivalent to $(K\eta)^M > K$, so by (15), $\Delta^{\text{imp}}_{K,M}(q) > 0$ for sufficiently small $q > 0$. By the high-q argument above, $\Delta^{\text{imp}}_{K,M}(q) < 0$ for sufficiently large q close to 1. Since $\Delta^{\text{imp}}_{K,M}(0) = 0$ and the function is continuous, $\Delta^{\text{imp}}_{K,M}(q)$ is non-monotone on $[0, 1]$.

Non-monotonicity in team size. It suffices to give explicit counterexamples. First, Proposition 4.5 already gives a setting in which $P^{\text{imp}}_{\text{comm}}(K, M)$ decreases from $K = 1$ to $K = 2$, so $P^{\text{imp}}_{\text{comm}}(K, M)$ is not monotone in K in general.

For the communication advantage $\Delta^{\text{imp}}_{K,M}$, take $M = 2$, $q = 0.1$, and define

$$\eta(1, 2) = 1, \qquad \eta(2, 2) = 1, \qquad \eta(3, 2) = 0.01.$$

At $K = 1$, communication and no communication coincide, so $\Delta^{\text{imp}}_{1,2} = 0$. At $K = 2$,

$$P^{\text{imp}}_{\text{comm}}(2, 2) = \left(1 - 0.9^2\right)^2 = 0.0361,$$

while

$$P_{\text{no-comm}}(2, 2) = 1 - (1 - 0.01)^2 = 0.0199,$$

so $\Delta^{\text{imp}}_{2,2} = 0.0162 > 0$. At $K = 3$,

$$P^{\text{imp}}_{\text{comm}}(3, 2) = \left(1 - (1 - 0.001)^3\right)^2 \approx 8.98 \times 10^{-6},$$

whereas

$$P_{\text{no-comm}}(3, 2) = 1 - (1 - 0.01)^3 \approx 0.0297.$$

Thus $\Delta^{\text{imp}}_{3,2} < 0$. Hence $\Delta^{\text{imp}}_{K,M}$ is not monotone in K in general.

G. Benchmarks

We evaluate on five benchmarks spanning stateless and stateful agentic tasks, chosen to vary in task compositionality M and expected single-agent capability $q(M)$, allowing empirical tests of theoretical predictions in Section 4.

BrowseComp (Wei et al., 2025) is a challenging web-browsing benchmark requiring agents to locate hard-to-find information through persisten web search, query reformulation, and reasoning over fragmented evidence across many sources. Single-agent accuracy is low across models, placing this in the weak-capability, high-compositionality regime. We evaluate on a 100-question subset and grade the submitted answer for problem against the provided GT answer using a gpt-5-mini judge.

FinanceAgentBench (Bigeard et al., 2025) evaluates agents on expert-authored financial analysis tasks including fact retrieval, market research, projections, and financial modeling. Task categories provide within-benchmark variation in compositionality. We evaluate on the 50-question public split. Each problem has a rubric with correctness (answer must contain a required fact, calculation, or claim) and contradiction (answer must not contradict the reference) items. The evaluator asks a gpt-5-mini judge yes/no three times for each rubric item and takes the mode. A problem is considered correct if it passes every rubric item.

τ^2-bench (Barres et al., 2025) is a user-assistance benchmark in which agents complete multi-turn tasks requiring information gathering and irreversible state-changing actions in the correct order. It is the primary stateful benchmark in our evaluation, directly exercising CoAct's synchronization protocol. We evaluate on the 114-question retail subset using the provided evaluator, which tests several criteria within an environment.

For **math reasoning**, we evaluate on HMMT 2026 and AIME 2025, which are challenging math competition benchmarks. We use the full test set for both benchmarks, grade the submitted answer against the ground truth value, and report pass@1.

H. Case Studies

H.1. τ^2-bench

H.2. BrowseComp

H.3. FinanceAgentBench

τ^2-Bench Task 99: CoAct trace with ranked voting, 2 GPT-5.2 agents. User: Sofia Li, ZIP 78260. Goals: 1) Exchange bicycle to a larger frame for her kid, 2) Exchange jigsaw puzzle in the same order for one of the same difficulty, but 1000 more pieces, and prefer art over the animal theme if both are available, 3) Exchange camera to lower resolution, 4) Prefer Visa card as payment or refund, 5) Cancel the skateboard in one of her orders.

Aliases.

- `B`: bicycle+puzzle order `#W3916020`; `C`: camera order `#W4689314`; `S`: skateboard order `#W8855135`; `V`: Visa `credit_card_3951670`.
- Item swaps: bike item `7758198585` → `5606522780`; puzzle item `4068787148` → `5546244844`; camera item `5996159312` → `8363011723`.

Step 1: Authenticate and enumerate orders.

- Both agents call `find_user_id_by_name_zip` and retrieve `sofia_li_9219`.
- Both call `get_user_details`; saved payment methods include `V`, Visa ending 6791.
- Agents identify relevant orders: `B`, `C`, and `S`.

Step 2: Inspect order contents.

- `B`: delivered; contains Bicycle `7758198585` and Puzzle `4068787148`.
- `C`: delivered; contains Camera `5996159312`.
- `S`: pending; contains Skateboard `3098764622`.

Step 3: Retrieve product variants and share findings.

- Bicycle: both agents find that the current bicycle is medium/green/road. The only available larger-frame variant changes other options to large/red/mountain. They share this caveat and the refund amount: $-\$14.54$.
- Puzzle: both agents find two available 1500-piece intermediate variants: art and animals. Since the user prefers art if both are available, they select the art variant. Refund: $-\$0.42$.
- Camera: both agents find a lower-resolution camera that keeps the requested fixed options: 20MP/3x/SD instead of 24MP/3x/SD. Refund: $-\$71.59$.
- Price calculation: one agent computes the combined expected refund and shares:

$$-14.54 - 0.42 - 71.59 = -86.55.$$

Step 4: Ask user for explicit confirmation.

- Both agents propose `talk_to_user`, listing all three swaps and payment method `V`.
- Ranked vote selects the more complete confirmation message.

✓ `talk_to_user: confirm B and C exchanges using V`

Step 5: Execute first exchange, order `B`.

- After confirmation, both agents propose the same unsafe action: exchange bike+puzzle in `B`.
- Vote is unanimous because proposals are identical.

✓ `exchange(B): [bike,puzzle] -> [new-bike,new-puzzle]`

- Result: `B` becomes `exchange requested`; refund $-\$14.96$ to `V`.

Step 6: User asks for completion/status confirmation.

- User asks whether exchanges for `B` and `C` were submitted, refund totals, return instructions, and skateboard status.
- Current state: `B` is complete; `C` is still delivered and unexchanged.

Step 7: Meaningful proposal disagreement.

- `agent_1` proposes `exchange(C)` for the camera.
- `agent_0` proposes `talk_to_user`, explicitly saying `B` is submitted and `C` is not yet submitted.

Step 8: Ranked vote selects communication over immediate mutation.

- Both agents review the two proposals and reason that the user should be addressed before continuing.
- `talk_to_user`: Borda 4, selected.
- `exchange(C)`: Borda 2, not selected.
- Selected message preserves the unfinished camera exchange as the next action.

✓ `talk_to_user: B done; C pending; C is next`

Step 9: Re-verify camera order and replacement.

- Agents re-check `C`: status still `delivered`; item `5996159312` still present; no exchange fields set.
- Agents re-check camera variants: replacement `8363011723` still available.

Step 10: Execute camera exchange, order `C`.

- One agent proposes `exchange(C)`; both agents rank it first.

✓ `exchange(C): camera -> new-camera`

- Result: `C` becomes `exchange requested`; refund $-\$71.59$ to `V`.

Step 11: Validation.

Required action	Oracle	Ranked CoAct
Exchange B	1	1 ✓
Exchange C	1	1 ✓

Result: ✓ PASS

BrowseComp Task 664: CoAct trace w/ ranked voting vs. naive parallel, 3 GPT-5.2 agents. "An episode in a patriarchal society before the first millennium's close incited a wave of indignation among the people of that area. Central to this event was an individual born in the same region and within the same century. This individual, more than four decades after their birth, was entrusted with a position of territorial authority. A few years later, the individual received another position of prominence, which they held for less than a decade. The second successor to this individual was a figure who reinstated a historical practice designed to obliterate certain individuals from official records and public memory. Identify the individual who initially entrusted the initial individual with their 1st territorial position of authority."

Aliases.

- `C`: Cadaver Synod, the 897 posthumous trial that caused public outrage.
- `F`: Pope Formosus, the central figure in the correct chain.
- `N`: Pope Nicholas I, who appointed `F` to his first territorial post.
- `S`: Pope Stephen VI, Formosus's second successor, who conducted the Cadaver Synod.
- `P`: Porto, Formosus's first territorial authority as Cardinal-Bishop of Porto.
- `A`: Anagni, the bishopric used in an incorrect alternative chain.

Step 1: Agents search the historical clue set.

- All three agents search using the "public indignation," patriarchal society, successor, and *damnatio memoriae* clues.
- `agent_1` and `agent_0` converge on `C`, the Cadaver Synod; `agent_2` finds only generic *damnatio memoriae* evidence and does not propose an answer.
- The key ambiguity between the two successful searches is which person is the central individual in the prompt: `F` or `S`.

Step 2: One agent proposes a plausible but wrong chain.

- `agent_1` identifies `C`, but treats `S` as the central individual.
- It verifies that `F` consecrated `S` as Bishop of `A`.
- Under this interpretation, the answer is `F`.

× `agent_1: submit_answer(Pope Formosus)`

Step 3: Another agent finds the correct role assignment.

- `agent_0` also identifies `C`, but maps the roles differently.
- It reasons that `F` is the central figure: born in the relevant century, appointed Cardinal-Bishop of `P` in 864, and later pope from 891–896.
- It observes that `S`, the second successor after `F`, is the figure who conducted the Cadaver Synod.
- Therefore, the requested individual is the pope who gave `F` his first territorial authority: `N`.

✓ `agent_0: submit_answer(Pope Nicholas I)`

Step 4: CoAct exposes the disagreement through shared memory.

- The channel contains both proposed chains: `F` as the answer from `agent_1`, and `N` as the answer from `agent_0`.
- The disagreement is meaningful: both agents found the same event, but only one correctly assigned the prompt's "central individual" and "second successor" roles.

Step 5: Ranked voting selects the corrected interpretation.

- The orchestrator collects two intents: `submit_answer(Pope Formosus)` and `submit_answer(Pope Nicholas I)`.
- Each agent reviews both proposals while seeing the shared memory log and independently ranks `Pope Nicholas I` first.
- Example reasoning: "I1 fits the question's target: the initial territorial post was Formosus being made cardinal-bishop of Porto (864), and multiple sources attribute that appointment to Pope Nicholas I; thus Nicholas I is the person who initially entrusted him with territorial authority. I2 instead names Pope Formosus, which confuses the roles (Formosus appointed Stephen VI as bishop of Anagni, but the question's chain points to Formosus as the central individual."
- Ranked voting assigns the correct proposal the higher Borda score: 6 vs. 3.

✓ `submit_answer(Pope Nicholas I)`

Step 6: Naive parallel fails on the discarded candidate.

- In the naive-parallel run, agents do not share the competing role analyses.
- Two agents come up with Pope Formosus, and one agent proposes Pope Nicholas I.
- The system takes a majority and submits `Pope Formosus`, the same candidate that CoAct considered and rejected.
- This answer is marked incorrect because the gold answer is `Pope Nicholas I`.

× `naive_parallel: submit_answer(Pope Formosus)`

Step 7: Validation.

System	Submitted answer	Oracle	Result
Naive parallel	Pope Formosus	Pope Nicholas I	×
Ranked CoAct	Pope Nicholas I	Pope Nicholas I	✓

Result: CoAct turns a plausible-but-wrong independent answer into a useful contrastive proposal. Shared evidence exposes the role-assignment error, and ranked voting selects the correct answer.

FinanceBench Task 11: CoAct w/ ranked voting, 3 GPT-5.2 agents.
Question: "How many common stock shares are outstanding for ABNB? Format `Class X: X shares` add line break."

Ground truth.
- `Class A`: 432,876,657 shares; `Class B`: 188,462,942 shares; `Class C`: 0 shares; `Class H`: 9,200,000 shares.

Step 1: Agents independently locate the relevant filing.
- All agents search EDGAR for Airbnb filings using CIK `0001559720`.
- `agent_2` identifies latest annual report: FY2024 Form 10-K, accession `0001559720-25-000010`, filed 2025-02-13.
- `agent_1` independently confirms the same 10-K and also enumerates the 2024 quarterly filings.
- `agent_0` finds both the FY2024 10-K and the 2024 Q3 10-Q, identifying the annual filing as the likely source for cover-page shares outstanding.

Step 2: One agent creates shared extraction state.
- `agent_2` parses the FY2024 10-K HTML from SEC EDGAR.
- The parsed document is stored under key `abnb_10k_2024`.
- `agent_2` shares the storage key with the team so other agents can query the same filing rather than re-discover it.

✓ `parse_html_page(ABNB FY2024 10-K) -> abnb_10k_2024`

Step 3: Agents extract conflicting evidence.
- `agent_2` initially queries the wrong character range and fails to extract the cover-page sentence.
- `agent_0` queries the full parsed 10-K and extracts the correct cover-page sentence:

$$\texttt{Class A} = 432{,}876{,}657, \quad \texttt{Class B} = 188{,}462{,}942, \quad \texttt{Class C} = 0, \quad \texttt{Class H} = 9{,}200{,}000.$$

- `agent_1` queries an insufficient range and obtains a contradictory extraction:

$$\texttt{Class A} = 619{,}085{,}761, \quad \texttt{Class B} = 16{,}860{,}476, \quad \texttt{Class C} = 0, \quad \texttt{Class H} = 0.$$

Step 4: Shared channel and shared storage expose the conflict.
- `agent_2` parses the FY2024 10-K into shared storage key `abnb_10k_2024`, then later shares correct `Class A/B` counts from a targeted cover-page range.
- `agent_0` queries the same shared key and shares the fuller extraction, including `Class C`=0 and `Class H`=9,200,000.
- `agent_1` also queries the shared key, but with an insufficient range, and shares a conflicting extraction: `Class A`=619,085,761, `Class B`=16,860,476, `Class C`=0, and `Class H`=0.

Step 5: Channel-aware re-verification corrects the bad extraction.
- Seeing the conflict in the shared channel, `agent_1` explicitly decides not to submit immediately.
- To independently verify, it reparses the same SEC filing under a new key, `abnb_10k_fy2024_agent1`.
- It then queries the cover-page region and retrieves the exact statement:

`432,876,657 Class A; 188,462,942 Class B; no Class C; 9,200,000 Class H.`

- `agent_1` explicitly notes that this contradicts its earlier extraction.

✓ `shared conflict -> re-verify(cover page) -> corrected Class A/B/C/H counts`

Step 6: Proposal disagreement.
- `agent_0` proposes an incomplete answer containing only the two largest classes:

`Class A: 432,876,657.00 shares; Class B: 188,462,942.00 shares`

- `agent_1` proposes the complete answer with all common stock classes:

`Class A: 432,876,657.00; Class B: 188,462,942.00; Class C: 0.00; Class H: 9,200,000.00`

Step 7: Ranked voting selects the complete answer.
- The orchestrator collects two intents: incomplete `A/B` and complete `A/B/C/H`.
- All three agents rank the complete `A/B/C/H` answer first.
- Ranked voting assigns the complete answer the higher Borda score: 6 vs. 3.

✓ `submit_answer(Class A, Class B, Class C, Class H)`

Step 8: Validation.

Required field	Oracle	Naive Parallel	Ranked CoAct
Class A shares	432,876,657	432,876,657 ✓	432,876,657 ✓
Class B shares	188,462,942	188,462,942 ✓	188,462,942 ✓
Class C shares	0	omitted ×	0 ✓
Class H shares	9,200,000	omitted ×	9,200,000 ✓

Result: CoAct ✓ PASS, Naive Parallel × FAIL

FinanceBench Task 25: Scaling from 3 to 4 GPT-5.2 agents under CoAct degrades performance.
Question: Summarize the regulatory risks Paylocity's (NASDAQ: PCTY) lists in its FY 2024 10-K.

Aliases.
- `P`: Paylocity Holding Corp. (`PCTY`).
- `K`: FY2024 Form 10-K, accession `0001591698-24-000151`, filed 2024-08-02.
- Rubric-critical themes: privacy laws; GDPR/foreign laws; changing tax/benefit/privacy laws; money transmitter licensing; third-party/supply-chain risk; cybersecurity trust damage; increasing cybersecurity scrutiny.

Both runs locate the correct filing.
- Agents identify `K`, Paylocity's FY2024 Form 10-K for fiscal year ended 2024-06-30.
- Agents parse the filing under the shared key `pcty_10k_fy2024`.

The 3-agent run builds a complete shared summary.
- `agent_0` shares a broad regulatory-risk extraction from `K`.
- It includes changing payroll, tax, wage/hour, employment, and benefits laws; HIPAA/PHI; privacy laws; cybersecurity regulation; third-party and supply-chain exposure; money transmitter/MSB risk; AI/ML regulation; tax law changes; international privacy; and SEC/internal-control risk.
- `agent_2` independently extracts named regimes: FTC authority, HIPAA, GDPR, CCPA/CPRA, state breach-notification laws, Illinois BIPA, money transmitter/MSB statutes, and vendor cybersecurity risk.
- `agent_1` separately re-extracts regulatory categories from the same filing and confirms the named regimes and third-party/vendor dependencies.

The 3-agent proposal preserves the rubric-critical details.
- `agent_0` proposes a consolidated answer with separate bullets for: privacy/data protection, cybersecurity scrutiny, third-party/vendor security weaknesses, money transmitter/MSB licensing, changing tax/benefit/employment laws, international/GDPR exposure, and public-company controls.
- Both other agents rank the proposal first.
- The single proposal receives Borda score 3 and is submitted.

✓ `submit_answer(complete regulatory-risk summary)`

The 4-agent run finds more evidence but submits prematurely.
- `agent_2` shares a detailed extraction including FTC, HIPAA, state breach laws, CCPA/CPRA, BIPA, GDPR, money transmitter/MSB licensing, third-party/vendor compliance, and cybersecurity oversight.
- `agent_3` independently extracts the exact named-laws checklist: FTC authority, HIPAA, state breach-notification laws, CCPA/CPRA, BIPA, GDPR, and money transmitter/MSB statutes.
- `agent_1` proposes submitting an incomplete answer.

× `submit_answer(incomplete regulatory-risk summary)`

Later evidence exposes omissions, but no corrected proposal is made.
- After `agent_1`'s proposal, `agent_0` shares an addendum explicitly naming FTC-related rules, state breach-notification laws, Illinois BIPA, and third-party banking/ACH dependencies.
- `agent_2` also shares an additional tax-law risk about sales/use taxes on electronically delivered services.
- However, the group treats the existing proposal as sufficient: `agent_0` says there is no need to submit a competing answer. Because of the extra noise / additional sources of information, the agents do not successfully course-correct.

Ranked voting has no alternative to rescue.
- The orchestrator collects only one intent: `submit_answer` from `agent_1`.
- All four agents rank the only proposal first by default.
- The proposal receives Borda score 4 and is submitted.

× `ranked_vote(single incomplete proposal)`

What the 4-agent answer missed.
- The selected 4-agent answer mentions privacy and cybersecurity generally, but does not preserve the full rubric-critical named-laws bundle: FTC rules, HIPAA, CCPA/CPRA, BIPA, and state breach-notification laws.
- It also fails to separately capture the rubric's third-party/supplier risk: breaches, supply-chain attacks, or system failures at partners and suppliers causing operational harm and liabilities.
- These facts were present in the channel, but were not incorporated into the only submitted proposal.

Step 8: Validation.

Required rubric item	3-agent CoAct	4-agent CoAct
Evolving privacy regulation for HCM/payroll	1 ✓	1 ✓
FTC/HIPAA/CCPA/CPRA/BIPA/state breach laws	1 ✓	0 ×
GDPR and foreign privacy laws	1 ✓	1 ✓
Changing tax/benefit/privacy laws affect products	1 ✓	1 ✓
Money transmitter licensing risk	1 ✓	1 ✓
Third-party/supplier breach or supply-chain risk	1 ✓	0 ×
Security incidents damage client trust	1 ✓	1 ✓
Increasing cybersecurity scrutiny requires investment	1 ✓	1 ✓

Result: 3-agent CoAct ✓ PASS (9/9), 4-agent CoAct × $FAIL(7/9)$.