BusMA: A Bus Communication Substrate for Multi-Agent Systems

Yanwen Peng **Delvin Ce Zhang** **Xi Wang** **Nikolaos Aletras**
Department of Computer Science, The University of Sheffield
Sheffield, United Kingdom
{ypeng86, delvin.ce.zhang, xi.wang, n.aletras}@sheffield.ac.uk

Abstract

Multi-Agent (MA) systems are effective at solving complex tasks that demand planning, tool use, and the synthesis of evidence from multiple sources. Existing systems typically adopt Hierarchical Manager-Worker (HMW) or Router-based Message Passing (RMP) structures as their communication protocol. However, these designs restrict agent autonomy: Worker agents cannot directly consult specific "peers", and misrouted messages can propagate errors. Inspired by bus architectures in computer systems, we propose BusMA, a communication framework that allows any agent to address other agents through a shared channel, i.e., the *Bus*. It consists of agent registration, message routing, and shared memory management components. *Worker agents*, each equipped with tools, have their own local memory and can reason, act (tool usage), and communicate by posting shared messages with specific intents. We introduce four intents: discussion, challenge, guidance, and request for explanation, which support fine-grained communication among agents. A *Chair agent* monitors the shared memory to coordinate interactions and facilitate convergence among Workers. To evaluate the effectiveness of BusMA, we conduct extensive experiments with two frontier LLMs across 13 tasks spanning visual reasoning, mathematical reasoning, and knowledge retrieval. The results demonstrate that BusMA consistently outperforms state-of-the-art HMW and RMP methods.[1]

1 Introduction

Multi-Agent (MA) systems powered by Large Language Models (LLMs) (OpenAI, 2025; Google DeepMind, 2026; DeepSeek-AI, 2024), comprise a set of agents that can reason, act, and communicate to solve complex real-world tasks. These tasks, such as mathematical reasoning (Lei et al., 2024) and knowledge retrieval (Huang et al., 2025), require effective collaboration (Fang et al., 2025; Du et al., 2025). Hence, the effectiveness of MA systems largely depends on the communication quality between agents and their coordination (Guo et al., 2024; Chen et al., 2024b; Liang et al., 2024; Du et al., 2024; Wu et al., 2024c; Peng et al., 2026).

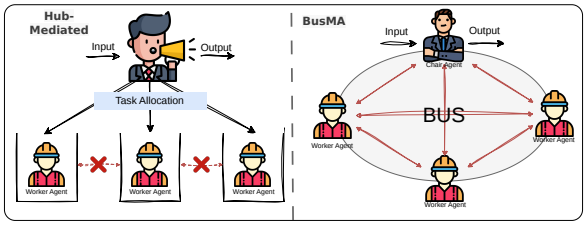


Figure 1: Comparison of hub-mediated topologies and BusMA. **Left**: In hub-mediated frameworks, Workers have restricted autonomy as passive executors, unable to initiate worker-to-worker communication (indicated by ×). **Right**: BusMA enables agents to directly address specific peers through a shared channel (the Bus), supporting four communication intents: Discussion, Challenge, Guidance, and Request for Explanation.

MA frameworks typically rely on *hub-mediated topologies*, which use a central component to route communication between worker agents, such as Hierarchical Manager-Worker (HMW) and Router-based Message Passing (RMP). HMW employs a *manager agent* to assign tasks and collect results (Qian et al., 2024; Gu et al., 2025; Zhang et al., 2025b). Similarly, RMP uses a *router agent* to mediate all message exchanges (Wu et al., 2024a; Nonomura and Mori, 2024). However, these topologies have two main limitations: (1) RMP and HMW agents have **restricted autonomy** as passive executors, unable to initiate worker-to-worker communication for critique or error correction (Ke et al., 2025; Jin et al., 2025; Krishnan, 2025); and (2) the HMW manager and RMP router suffer from **error propagation**, acting as *cognitive bottlenecks* that force all information to pass through a *single* hub (Piatti et al., 2024; Han et al., 2024; Maragheh and Deldjoo, 2025).

[1] Code: https://github.com/YanwenPneg/BusMA.

Proceedings of the 15th International Joint Conference on Natural Language Processing and the 5th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 647–669
November 6-10, 2026 ©2026 Association for Computational Linguistics

To overcome these limitations, we draw inspiration from distributed systems (Oki et al., 1993) and the computer bus architecture (Patterson and Hennessy, 2017) to propose BusMA, a communication substrate that enhances agent autonomy, allowing collaborative interactions on demand. Analogous to a hardware bus where components independently initiate data transfers, BusMA enables agents to directly address specific peers through a shared channel (i.e., the Bus), rather than passively waiting for task assignments or router mediation. BusMA also follows a logic similar to the Information Bus proposed by Oki et al. (1993) where distributed systems share a common communication channel by maintaining autonomy. Figure 1 illustrates existing frameworks and BusMA for MA communication.

The Bus includes three modules: *agent registration* to register agent roles and unique addresses, *message routing* for facilitating direct communication, and *shared memory* (history) management. BusMA instantiates a set of Worker agents and a Chair agent. Each Worker is equipped with tools, maintains its own local memory and can reason, act (tool usage), and communicate through the Bus. Unlike current HMW and RMP approaches, when a Worker needs to communicate with a specific peer, it simply posts a message to the Bus. Messages cover four communication intents: discussion, challenge, guidance, and request for explanation. This keeps agent-specific reasoning local, while only the necessary information is shared through the Bus. The Chair agent, a special case of a Worker agent with act disabled, monitors the shared memory to identify conflicts, detect missing evidence, and issue follow-up requests, to facilitate convergence. Compared to HMW approaches, the Chair agent acts as *primus inter pares* (first among equals), mitigating information bottlenecks.

Extensive experiments across 13 datasets show that BusMA outperforms competitive HMW and RMP frameworks. Our analysis reveals that collaboration yields larger gains on tasks where intermediate steps can be verified by peers, such as mathematical reasoning and knowledge retrieval, while producing competitive performance on perceptual tasks such as visual reasoning. Our analysis further shows that fine-grained role division is not uniformly beneficial and that Worker heterogeneity can allow errors to propagate undetected.

2 Related Work

2.1 LLM-based Agents.

LLMs such as GPT (OpenAI, 2023), Gemini (Comanici et al., 2025), and DeepSeek (DeepSeek-AI, 2024) serve as the backbone for autonomous agent development. Agentic architectures augment the base LLM with complementary mechanisms, including advanced planning strategies for task decomposition (Huang et al., 2024; LI et al., 2025; Hu et al., 2025; Erdogan et al., 2025; Zhou et al., 2024; Zhang et al., 2025a), external tool use and knowledge bases (Zhang et al., 2024b; Wu et al., 2024b; Qin et al., 2024; Feng et al., 2026), and long-term memory or reflection mechanisms for persistent state and iterative improvement (Shinn et al., 2023; Zhong et al., 2024; Mei et al., 2025; Xu et al., 2025; Chhikara et al., 2025). LLM-based agents have been applied to diverse domains such as automated programming (Trivedi et al., 2024; Zhang et al., 2024a; Chen et al., 2025b), system interaction (Wu et al., 2024d; Bonatti et al., 2025), and scientific discovery (Hong et al., 2025; Novikov et al., 2025). However, single-agent systems remain brittle on long-horizon, interdependent tasks, with evaluations reporting systematic failures in planning, decision-making, and instruction following (Liu et al., 2024; Xie et al., 2024; Wang et al., 2025; Mohammadi et al., 2025; Chen et al., 2025a). However, effective dialogue and collaboration between specialized agents is important for improving downstream performance (Li et al., 2023; Du et al., 2024; Tran et al., 2025; Chen et al., 2024a).

2.2 LLM-based MA Systems.

HMW and RMP topologies rely on a hub (a manager or a router) for coordination and information flow. Representative HMW systems include AgentVerse and ChatDev (Chen et al., 2024b; Qian et al., 2024, 2025; He et al., 2025), tool-oriented approaches such as OctoTools (Lu et al., 2025), and evolving orchestration paradigms that train a centralized orchestrator via reinforcement learning (Dang et al., 2025). RMP systems include AutoGen (Wu et al., 2024a) and router-based variants (Yue et al., 2025; Yao et al., 2025; Nonomura and Mori, 2024), with recent work introducing learnable protocol routing for scenario-aware selection (Du et al., 2026). However, communication between agents in RMP and HMW is restricted (Ke et al., 2025; Jin et al., 2025; Krishnan, 2025); while these frameworks may also suffer from error propagation (Piatti et al., 2024;

Han et al., 2024; Maragheh and Deldjoo, 2025; Sagirova et al., 2024). These limitations highlight the need for more flexible communication strategies.

3 The BusMA Framework

Our BusMA framework consists of three components: (1) **Worker Agents**, agents equipped with tools that maintain their own local memory and can reason, act (tool usage), and communicate; (2) a **Chair Agent**, a special case of a Worker agent with action execution disabled; and (3) the **Bus**, the infrastructure handling agent registration, message routing, and shared memory management. Figure 2 illustrates this architecture.

Workflow Overview. BusMA operates through on demand collaboration (Figure 2). A task begins with the input to the Chair Agent, who reasons over the task to identify the first Worker to contact and monitors the shared memory. Activated Workers execute an iterative loop: reasoning over local memory, followed by either invoking a local tool (`Act`) or posting a message to the Bus (`Call`). A `Call` generates a message and shares it via the Bus to the target agent; the calling Worker then suspends until reactivated. The Chair is reactivated only when a message is addressed to it. Finally, the Chair accesses the shared memory to assess progress, issue follow-up calls, or complete the task.

3.1 Worker Agent

Workers are the core operational units of BusMA. Each one is equipped with a specific tool set (e.g., web search, code execution) and communicates by posting shared messages with specific intents via the bus. The main mathematical notations used by BusMA are listed in Table 1.

Activation and Initialization. When a Worker receives a message from the Bus, it is activated and begins its iteration ($t = 0$). It constructs its initial context by concatenating the system prompt $\mathcal{P}_i$, the received message $\mathcal{M}_i$, and descriptions of peers $\mathcal{N}$ and tools $\mathcal{T}_i$. By modifying $\mathcal{P}_i$ and $\mathcal{T}_i$, BusMA instantiates workers with different capabilities tailored to specific tasks (e.g., search, code execution). During execution, the Worker maintains a local memory state $\mathcal{K}_i(t)$ that accumulates private reasoning and tool output across subsequent iterations.

Symbol	Description
Worker Agent α_i	
$\mathcal{L}$	LLM backbone
$\mathcal{P}_i$	System prompt
$\mathcal{M}_i$	Messages from the bus
$\mathcal{K}_i(t)$	Local memory at iteration t
$\mathcal{T}_i$	Available tool set
$\mathcal{N}$	Accessible peer agents
$T_{\max}$	Max local iterations per activation
$r_{i,t}^{\text{priv}}$	Private reasoning at iteration t
Chair Agent ($\mathcal{T}_{chair} = \emptyset$)	
$\mathcal{P}_{\text{chair}}^{\text{COOR}}$	Coordination mode prompt
$\mathcal{P}_{\text{chair}}^{\text{SUBM}}$	Submission mode prompt
$r_{\text{chair},t}^{\text{glob}}$	Global view at iteration t
Communication Bus	
s	Routing index
$\mathcal{A}_i$	Unique agent address
$\mathcal{R}$	Address registry
$\mathcal{Q}_i$	Message queue (FIFO buffer)
$\mathcal{H}$	Shared memory (history)

Table 1: Mathematical notation for BusMA components.

Reasoning and Action Selection. Following the ReAct framework (Yao et al., 2023), each Worker alternates between reasoning and acting. At iteration t, the LLM backbone $\mathcal{L}$ analyzes the current context (including $\mathcal{M}_i$ and $\mathcal{K}_i(t)$) to produce private reasoning $r_{i,t}^{\text{priv}}$, capturing how it interprets new information and selects the next step. Based on $r_{i,t}^{\text{priv}}$ and the available resources, the Worker selects one of two actions: `Act` or `Call`. `Act` invokes a tool to gather information or perform operations; `Call` posts a shared message to the Bus.

- **Act.** The Worker identifies a tool $\tau_j \in \mathcal{T}_i$ and configures its execution arguments θ (e.g., a file path or a formulated search query). The Worker executes the tool and receives an output $o_t = \tau_j(\theta)$. For instance, a search tool may take a query as θ and return a list of publications as the output. The Worker then updates its local memory with both the private reasoning and the tool output,

$$\mathcal{K}_i(t+1) = \mathcal{K}_i(t) \cup \{r_{i,t}^{\text{priv}}, o_t\}. \quad (1)$$

  Next, it proceeds to iteration $t+1$, reconstructing its context with $\mathcal{P}_i$, the latest Bus message $\mathcal{M}_i$, the peer and tool descriptions $(\mathcal{N}, \mathcal{T}_i)$, and the updated local memory $\mathcal{K}_i(t+1)$.

- **Call.** The Worker chooses a target agent $\alpha_j \in \mathcal{N}$ and generates an action $a_t^{\text{call}} = (\alpha_j, m_{i\to j})$, where $m_{i\to j}$ is the shared message. The Worker formats the message as JSON, and the Bus parses it for sending it to the target agent. After posting the message to the Bus, the Worker suspends its current iteration.

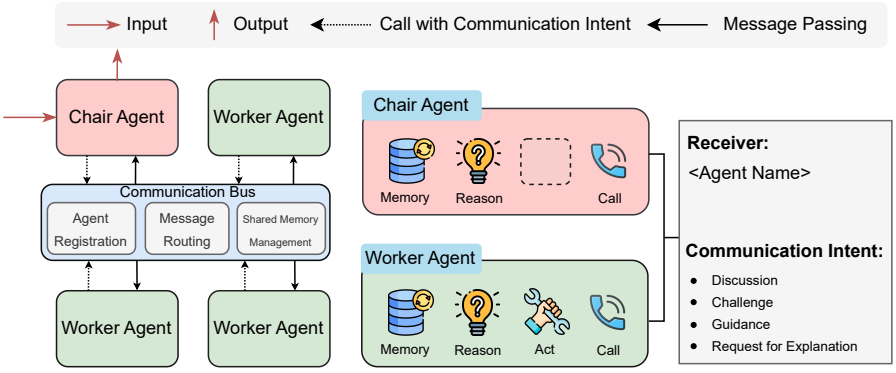


Figure 2: Overview of the BusMA communication substrate.

To enable high-quality communication, each Worker frames $m_{i \to j}$ by selecting a communication intent. We define four types: **discussion** for bidirectional information exchange, **request for explanation** for clarifying ambiguous context, **challenge** for questioning results, and **guidance** for providing expertise. The Worker pairs the intent with corresponding evidence and a focused question (e.g., tool output or a detected inconsistency). These four intent types expand the communication space beyond conventional assign-and-return exchanges in hub-mediated systems. In particular, the challenge intent allows one agent to question another's intermediate results, enabling re-examination through deliberation rather than passive error propagation.

The Worker's current iteration terminates under two conditions: (1) the Worker chooses `Call`, or (2) the Worker reaches the maximum iteration limit $T_{\max}$. In the second case, the Worker posts a failure message to the agent that originally activates it. Appendix A provides an example prompt for the Worker, including the JSON communication protocol and formatting requirements.

3.2 Chair Agent

The Chair Agent is a special case of a Worker agent with `Act` disabled, serving as the system's coordinator and entry point. Unlike existing HMW and RMP systems, the Chair Agent does not generate an explicit plan or decompose and assign subtasks upfront. Instead, it initiates collaboration and synthesizes information from Worker interactions. BusMA decouples message routing from task synthesis: Workers address specific peers directly on the Bus, and this peer-to-peer information exchange completely bypasses the Chair. The Chair does not use tools (i.e., $\mathcal{T}_{\text{chair}} = \emptyset$, see Figure 2), instead it operates in two modes with different prompts: $\mathcal{P}^{\text{COOR}}_{\text{chair}}$ for the coordination phase and $\mathcal{P}^{\text{SUBM}}_{\text{chair}}$ for the submission phase. Appendix A presents the two-phase prompts for the Chair agent.

Coordination Mode. At $t = 0$, the Chair receives the task and begins coordination using $\mathcal{P}^{\text{COOR}}_{\text{chair}}$. It reasons over the task description to identify the first Worker to contact, then issues a `Call` before it starts monitoring the shared memory. During this period, Workers communicate with each other through `Call`, exchanging information, critiques, and requests.

The Chair is reactivated only when a Worker addresses it. Upon reactivation, the Chair accesses the shared message (history) $\mathcal{H}$ to update its global view $r^{\text{glob}}_{\text{chair},t}$ of task progress. Specifically, it identifies unresolved disagreements, detects missing evidence or unaddressed questions, and formulates follow-up requests. Then, the Chair either (i) continues coordination by calling another agent, or (ii) outputs a JSON structure with a `submit` field to transition to the submission mode.

Submission Mode. The Chair switches to $\mathcal{P}^{\text{SUBM}}_{\text{chair}}$ and generates the final response by synthesizing the task description and accumulated information. This is a single-step generation.

3.3 Communication Bus

The communication Bus provides a shared substrate for agent interaction with three modules: *Agent Registration*, *Message Routing*, and *Shared Memory Management*.

Agent Registration. Upon system initialization, each instantiated agent α_i (Workers and the Chair) is registered to the Bus and assigned a unique address $\mathcal{A}_i$. This creates an address registry $\mathcal{R} = \{(\alpha_i, \mathcal{A}_i) \mid \alpha_i \in \mathcal{N}\}$, enabling agents to be addressed as message receivers.

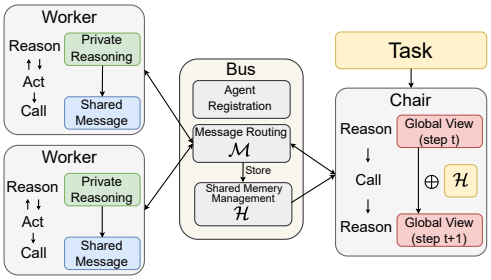


Figure 3: Information flow in BusMA: private reasoning $r^{\text{priv}}_{i,t}$, shared message $m_{i\to j}$, and global view $r^{\text{glob}}_{\text{chair},t}$.

Datasets	Modality	Domain				
AlgoPuzzleVQA (Ghosal et al., 2025)	Vision	General	✓			✓
Hallusion-VD (Guan et al., 2024)	Vision	General	✓			
PuzzleVQA (Chia et al., 2024)	Vision	General	✓			✓
VQA 2.0 (Goyal et al., 2017)	Vision	General	✓			✓
Game of 24 (Lile, 2024)	Text	Math		✓		✓
Omni-MATH (Gao et al., 2025)	Text	Math		✓	✓	✓
CLEVR-Math (Dahlgren Lindström and Abraham, 2022)	Vision	Math	✓	✓		
MathVista (Lu et al., 2024)	Vision	Math	✓	✓	✓	✓
GPQA (Rein et al., 2024)	Text	Knowledge		✓	✓	✓
MMLU-Pro (Wang et al., 2024)	Text	Knowledge			✓	✓
SciFIBench (Roberts et al., 2024)	Vision	Knowledge	✓		✓	
HotpotQA (Yang et al., 2018)	Text	Knowledge			✓	✓

Table 2: Diversity-oriented tasks. Icons denote required skills: visual understanding, numerical calculation, knowledge retrieval, and multi-step reasoning.

Message Routing. To support seamless collaboration, the Bus maintains a dedicated First-In-First-Out buffer Q_i for each agent. When an agent α_i issues a `Call` and posts a message specifying $a^{\text{call}}_t = (\alpha_j, m_{i\to j})$, the Bus parses the message, extracts the receiver α_j, and verifies it against the registry $\mathcal{R}$. The Bus then augments the message:

$$\hat{m}_{i\to j}(s) = (m_{i\to j}, \text{ID}, s, \alpha_i), \tag{2}$$

where ID is a unique identifier, s is the routing index assigned by the Bus, and α_i is the sender. The Bus pushes $\hat{m}_{i\to j}$ into the receiver queue Q_j. The receiver consumes messages in arrival order. Upon activation, an agent pops one message from its queue Q_i, and the remaining messages await subsequent activations. This design allows an agent to post messages to multiple receivers concurrently while maintaining consistency, as each receiver consumes incoming messages in a strict order.

Shared Memory Management. The Bus maintains a shared memory (history) $\mathcal{H}$ that stores all shared messages routed. Each newly sent message is appended to $\mathcal{H}$ in ascending order of its routing index s (with the unique ID assigned at routing time), preserving the complete communication history. When any Worker posts a message to the Chair, the Bus provides the Chair with access to $\mathcal{H}$, enabling global progress assessment and synthesis during coordination.

3.4 Overall Information Flow in BusMA

We propose a new method to organize information at three levels of granularity (Figure 3). Each Worker maintains private reasoning $r^{\text{priv}}_{i,t}$ in its local memory $\mathcal{K}_i(t)$, accumulating intermediate evidence and tentative conclusions that remain internal unless made public to the Bus via `Call`. When a Worker issues a `Call`, it publicizes only actionable content as a shared message $m_{i\to j}$ with a communication intent (3.1) and supporting evidence. Upon each reactivation, the Chair accesses $\mathcal{H}$ to update its global view $r^{\text{glob}}_{\text{chair},t}$, identifying conflicts, detecting gaps, and issuing targeted follow-ups. We hypothesize that this separation reduces communication overhead while preserving coordination coherence.

4 Experimental Setup

4.1 Benchmarks

Following Lu et al. (2025) and Roucher et al. (2025), we evaluate BusMA on two complementary benchmark groups: one prioritizing breadth across modalities and the other emphasizing depth in complex problem-solving.

Diversity-oriented. We use 12 standard datasets spanning three categories: (1) *Visual Reasoning* (e.g., VQA 2.0), (2) *Mathematical Reasoning* (e.g., Omni-MATH), and (3) *Knowledge Retrieval* (e.g., HotpotQA). Table 2 details the domain, modality, and required skills for each task. Following Lu et al. (2025), we adopt a sample size of 200 instances per dataset and apply their corresponding evaluation metrics for a fair comparison. Detailed dataset description is provided in Appendix C.

Complexity-oriented. We use the **GAIA** (Mialon et al., 2024) validation set (165 questions). This benchmark requires agents to combine file parsing, web browsing, and code execution. Following the OpenDeepResearch (Roucher et al., 2025) setting, we report exact-match accuracy.

4.2 Baselines

For **diversity-oriented tasks**, we evaluate four general-purpose MA frameworks: *OctoTools* (Lu et al., 2025) (HMW-style planner–executor), *SmolAgents* (Roucher et al., 2025), *LangGraph* (LangChain Inc., 2025) (HMW), and *AutoGen* (Wu et al., 2024a) (RMP). For the **complexity-oriented**

		OctoTools	SmolAgents	LangGraph	AutoGen	BusMA (Ours)	Δ
DeepSeek-V3	AlgoPuzzleVQA	47.5	31.5	45.5	39.5	**60.0**	+12.5
	Hallusion-VD	73.0	**75.5**	69.0	69.5	72.5	-3.0
	PuzzleVQA	56.5	53.0	55.5	55.5	**63.0**	+6.5
	VQA 2.0	67.5	73.0	65.0	66.5	**75.5**	+2.5
	Game of 24	75.0	68.5	62.0	47.5	**88.5**	+13.5
	Omni-MATH	52.0	49.5	41.0	41.0	**55.0**	+3.0
	CLEVR-Math	74.5	72.0	71.0	30.5	**77.5**	+3.0
	MathVista	**65.0**	63.0	53.5	55.5	62.5	-2.5
	GPQA	**60.5**	54.0	55.5	45.5	56.0	-4.5
	MMLU-Pro	68.0	73.0	52.5	59.0	**79.5**	+6.5
	SciFIBench	72.0	66.0	75.0	70.0	**76.5**	+1.5
	HotpotQA	53.5	54.5	30.5	50.5	**57.0**	+2.5
	Average	63.8	61.1	56.3	52.5	**68.6**	+4.8
Gemini-2.5-Flash	AlgoPuzzleVQA	**66.0**	55.5	52.0	37.0	63.0	-3.0
	Hallusion-VD	75.5	75.0	74.0	72.0	**77.0**	+1.5
	PuzzleVQA	**80.5**	72.0	75.0	58.0	76.0	-4.5
	VQA 2.0	**77.5**	71.5	75.5	69.5	76.0	-1.5
	Game of 24	88.0	89.5	81.5	73.5	**96.5**	+7.0
	Omni-MATH	53.5	67.0	50.0	66.0	**68.0**	+1.0
	CLEVR-Math	89.0	71.5	76.0	57.0	**89.5**	+0.5
	MathVista	77.5	67.5	64.5	55.0	**79.0**	+1.5
	GPQA	68.5	64.5	59.5	64.5	**69.5**	+1.0
	MMLU-Pro	72.0	77.0	62.5	46.0	**79.0**	+2.0
	SciFIBench	81.0	75.5	78.5	54.5	**82.5**	+1.5
	HotpotQA	57.0	54.5	55.0	44.5	**59.0**	+2.0
	Average	73.8	70.1	67.0	58.1	**76.3**	+2.5

Table 3: Accuracy (%) of MA frameworks across tasks and models. Bold denotes best method in each task, and the second best is underlined. Δ shows the performance difference between BusMA and the best baseline.

GAIA benchmark, we compare against specialized state-of-the-art systems including *MagenticOne* (Fourney et al., 2024) and *OpenDeepResearch* (Roucher et al., 2025). To further isolate the benefits of MA collaboration, we also include a single-agent baseline using Gemini-2.5-Flash and Pro with standard Function Calling. Detailed configurations for all baselines are provided in Appendix D.

4.3 Implementation Details

We use **DeepSeek-V3** (DeepSeek-AI, 2024) and **Gemini-2.5-Flash** (Comanici et al., 2025) as LLM backbone $\mathcal{L}$. Since DeepSeek-V3 does not support image input, all frameworks evaluated under the DeepSeek-V3 setting route visual tool calls to Gemini-2.0-Flash; this mapping is applied uniformly across BusMA and all baselines. Following the principle of role based specialization (Qian et al., 2024; Zhang et al., 2025b), we instantiate task specific Workers: a **Web Agent** for retrieval, an **ImageQA Agent** for vision, and a **Code Agent** for coding. For GAIA, to isolate the impact of Worker performance, we adopt the **Browser Agent** and file analysis tools from Roucher et al. (2025) via adapters that map external agent I/O to the Bus protocol. To ensure fair comparison with baselines that execute agents sequentially, BusMA is configured to activate only one agent at a time during experiments. This setting isolates the effect of communication structure from parallelism. We also compare with a single-agent baseline (Appendix B). The prompts for each agent are listed in Appendix A. Hyperparameters are provided in Appendix C.

Methods	MA	**Level 1**	**Level 2**	**Level 3**	**Overall**
Gemini-2.5-Flash-F/C		30.1	12.7	7.7	17.6
Gemini-2.5-Pro-F/C		39.6	24.1	19.2	28.2
MagenticOne	✓	52.8	36.0	15.4	38.2
OpenDeepResearch	✓	58.5	43.0	19.2	44.2
BusMA	✓	**60.3**	**47.6**	**26.9**	**48.5**

Table 4: Performance (%) on GAIA. F/C denotes single-call function calling (non-agentic); MA systems use Gemini-2.5-Flash. Best is **boldfaced**.

5 Results

5.1 Diversity Benchmarks

Table 3 shows results across models and tasks. Overall, BusMA achieves the best average accuracy, surpassing the best baseline (OctoTools) by 4.8% with DeepSeek-V3 and 2.5% with Gemini-2.5-Flash.

In visual reasoning (AlgoPuzzleVQA, Hallusion-VD, PuzzleVQA, and VQA 2.0), BusMA ranks first or second in seven of the eight task–backbone combinations. Improvements are larger with DeepSeek-V3 but more modest with the stronger Gemini-2.5-Flash model, suggesting that, on visual reasoning tasks, accuracy is mainly determined by the base model's image understanding (multimodal) capability, which additional agent communication is unlikely to substantially improve.

In mathematical reasoning, BusMA achieves the best performance in the majority of benchmarks, including Game of 24, Omni-MATH, and CLEVR-Math across both backbone models. On MathVista, it leads with Gemini-2.5-Flash (79.0%) and remains competitive with DeepSeek-V3. These results suggest that BusMA's diverse communication intents enhance iterative verification capability of agents, potentially mitigating error propagation that commonly occurs in HMW and RMP methods (Cemri et al., 2025).

For knowledge-intensive tasks, BusMA performs strongly on MMLU-Pro, SciFIBench, and HotpotQA, achieving the best results with both backbone models. This advantage may stem from BusMA's three-level information organization: Workers keep outputs from web search tools in their local memory $\mathcal{K}_i(t)$, externalizing only more important information. This separation mitigates long-context overhead, while the Chair's access to the shared memory enables synthesis across multiple sources. On GPQA, our method achieves the best result with Gemini-2.5-Flash but ranks second with DeepSeek-V3, indicating sensitivity to

Configuration	PuzzleVQA	MathVista	GPQA
BusMA (Full)	**76.0**	**79.0**	69.5
w/ Manager	71.5 (-4.5)	72.0 (-7.0)	68.0 (-1.5)
w/ SmolAgents Workers	76.0 (+0.0)	77.5 (-1.5)	**70.5** (+1.0)
SmolAgents Baseline	72.0	67.5	64.5

Table 5: Ablation study results (%). *w/ Planner Chair* replaces the Chair agent with the Manager agent, a task decomposition planner. *w/ SmolAgents Workers* substitutes BusMA Workers with those from SmolAgents.

Step 1: Chair		*Discussion*
REASON	Requires domain knowledge; no tools available.	
CALL	Delegate to Web agent for symmetry search.	
Step 2: Web		*Discussion*
REASON	Search symmetry for each candidate.	
ACT	`GoogleSearch("triisopropyl borate symmetry")`	
CALL	Triisopropyl borate has C3h.	✗ Error
Step 3: Chair		*Submit*
REASON	Cannot verify; select higher confidence option.	✗ Propagated
CALL	Submit A.	

Table 6: Error case from GPQA. Query: *Which molecule has C3h symmetry?* Correct: C. Output: A.

settings where retrieval provides limited support. In such cases, Workers may make mistakes, which can mislead the Chair's co-ordination.

Gemini-2.5-Flash outperforms DeepSeek-V3, consistent with prior work (Comanici et al., 2025). However, BusMA yields larger relative improvement with DeepSeek-V3, suggesting better gains when the base model is weaker.

5.2 GAIA Benchmark

Table 4 shows the performance across GAIA difficulty levels. Single-model baselines Gemini-2.5-Flash-FunctionCalling and Gemini-2.5-Pro-FunctionCalling substantially underperform all MA systems, achieving 17.6% and 28.2% respectively, suggesting that architectural design rather than model capacity is critical for tackling complex tasks. More specifically, BusMA achieves 60.3%, 47.6%, and 26.9% accuracy for Levels 1, 2, and 3 respectively, with 48.5% overall. This corresponds to a 4.3% improvement over OpenDeepResearch. Notably, BusMA achieves a larger lead on Levels 2 and 3 (4.6% and 7.7%), which require more advanced reasoning and complex collaboration. This highlights BusMA's ability to provide effective solution paths. Moreover, despite using the exact same Browser Agent as OpenDeepResearch, BusMA still outperforms the latter across all levels, indicating that the improvement is not solely attributed to the Browser Agent, but associated with the Chair's coordination and the Bus modules for Message Routing and Shared Memory Management.

6 Analysis

6.1 Ablation Study

We conduct an ablation study to analyze the contributions of BusMA's core components. We select one representative benchmark from each domain: PuzzleVQA (visual reasoning), MathVista (mathematical reasoning), and GPQA (knowledge retrieval). We use Gemini-2.5-Flash for all the experiments. Table 5 shows the results.

First, the *w/ HMW Manager* setting converts BusMA into HMW by replacing the Chair with a Manager. Following Wu et al. (2024a) and Roucher et al. (2025), we modify the Chair's coordination mode prompt to function as a centralized coordinator that assigns tasks and collects results (see Appendix A), and disable communication among Workers. This restricts Workers to passive executors unable to initiate peer-to-peer communication for critique or error correction. The accuracy drop of 1.5 to 7.0 percentage points across tasks supports the effectiveness of BusMA's communication and coordination design.

Second, we replace BusMA's Worker implementation with the SmolAgents Worker implementation (Roucher et al., 2025) as described in Section C (*w/ SmolAgents Workers*), while keeping the Chair and communication substrate unchanged. We observe that performance remains close to the original BusMA (within 1.5%), yet this variant exceeds SmolAgents by 4.0% to 10.0% across tasks. This gap indicates that the agents communication accounts for the majority of BusMA's improvement, suggesting robustness to different Worker implementations.

6.2 Qualitative Analysis

Figures 4 and 5 show communication trajectories of BusMA and the best-performing baseline SmolAgents on the same **Omni-MATH** problem. BusMA tackles this task with two agents: the Chair and the Code Agent. The Chair **discusses** with the Code Agent about required calculations and **requests** the Code agent to provide a more detailed explanation after receiving an unreliable response. Looking at the full communication trajectory of BusMA (Table 8 in Appendix E), the Chair also **guides** the Code Agent to validate candidate answers using simple

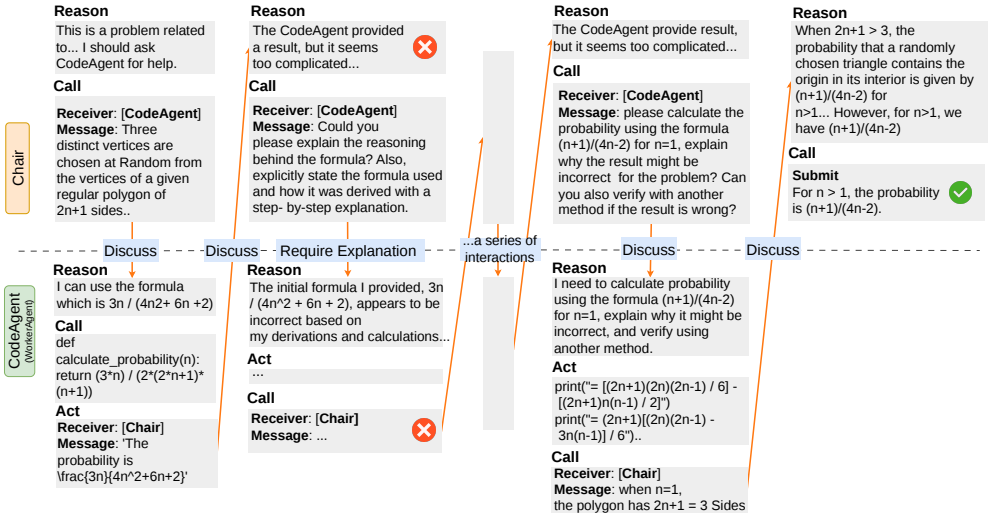


Figure 4: BusMA communication trajectory for answering a question from Omni-MATH: *Three distinct vertices are chosen randomly from the vertices of a regular polygon with (2n+1) sides; what is the probability that the center lies inside the triangle formed by the three chosen vertices?* The full trajectory is provided in Table 8.

test cases (1, 2 and 3), while the Code Agent **challenges** the Chair's hypotheses when appropriate. Although both agents initially make errors, iterative interaction yields the correct solution and a brief reflection on the causes of failure. By contrast, for **SmolAgents**, the Manager generates an incorrect answer and delegates to a Code Agent. However, it only collects feedback without enabling two-way communication, so the initial error persists. Apart from the ablation study in 6.1, this further highlights the main difference between our Chair Agent acting as *primus inter pares* with Workers compared to standard HMW managers. Appendix G provides an analysis of over-specialization in CLEVR-Math.

6.3 Error Analysis

Table 6 presents a failure case from GPQA where BusMA produces an incorrect answer due to erroneous retrieval (a detailed analysis is provided in Appendix F). Based on the GoogleSearchTool retrieval outputs, the Web Agent incorrectly claims that *triisopropyl borate has C3h symmetry* (it actually belongs to C3, lacking the horizontal mirror plane), while providing only speculative information for the correct answer. The Chair, lacking access to GoogleSearchTool, cannot independently verify the Web Agent's claims and thus selects the option presented with higher confidence. This case highlights a limitation of Worker heterogeneity: when specialized tools are exclusively assigned to specific agents, other agents have no means to cross-validate their outputs, allowing retrieval errors to propagate undetected.

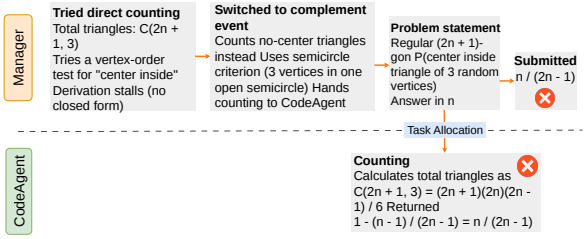


Figure 5: SmolAgents communication trajectory for the same Omni-MATH task in Figure 4.

7 Conclusion

We introduced BusMA, a communication substrate that enables agents to directly address specific peers through a shared channel, supporting four communication intents: discussion, challenge, guidance, and request for explanation. BusMA features Worker Agents that can reason, act, and communicate, a Chair Agent that monitors shared memory to facilitate convergence, and a Bus with agent registration, message routing, and shared memory management modules. Extensive experiments across 13 tasks spanning visual reasoning, mathematical reasoning, and knowledge retrieval demonstrate that BusMA consistently outperforms state-of-the-art HMW and RMP methods. In future work, we plan to extend agents to incorporate confidence estimation, allowing them to identify and mitigate information asymmetry. We also aim to evaluate BusMA on tasks such as WebWalkerQA (Wu et al., 2025) that require coordinating more agents and longer reasoning chains to further assess its scalability.

Limitations

BusMA improves collaboration through interaction but does not enhance an LLM's underlying capabilities. In visual reasoning tasks where perception is the bottleneck, additional exchanges may introduce noise, yielding smaller gains or regressions. A core limitation is error propagation through the shared memory. When external tools provide incomplete or wrong evidence, Workers may produce misleading messages that bias the Chair's synthesis, as seen in our weaker GPQA results. Tool heterogeneity compounds this issue: agents often cannot verify outputs from unfamiliar tools, allowing errors to spread. Confidence-aware arbitration with conflict detection could help address this.

The tasks used in our experiments typically involve coordination among only 2–3 agents. Widely used multi-agent benchmarks do not yet require large-scale agent collaboration, and prior work has found that increasing the number of agents does not necessarily improve task performance. Architecturally, BusMA does not impose structural barriers to scaling: each agent receives a unique address and the shared memory grows linearly with the number of routed messages. As the number of agents increases, the shared memory accumulates more messages and the Chair must synthesize longer communication histories, creating bus contention. Hardware bus mechanisms such as blocking replies, arbitration protocols, and priority scheduling are beyond the scope of this work. In future work, we plan to include these features to enable synchronization in larger-scale scenarios.

References

Rogerio Bonatti, Dan Zhao, Francesco Bonacci, Dillon Dupont, Sara Abdali, Yinheng Li, Yadong Lu, Justin Wagle, Kazuhito Koishida, Arthur Bucker, Lawrence Keunho Jang, and Zheng Hui. 2025. Windows agent arena: Evaluating multi-modal OS agents at scale. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 4874–4910. PMLR.

Mert Cemri, Melissa Z Pan, Shuyi Yang, Lakshya A Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei A Zaharia, Joseph Gonzalez, and Ion Stoica. 2025. Why do multi-agent LLM systems fail? In *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc.

Hui Chen, Miao Xiong, Yujie Lu, Wei Han, Ailin Deng, Yufei He, Jiaying Wu, Yibo Li, Yue Liu, and Bryan Hooi. 2025a. MLR-Bench: Evaluating AI agents on open-ended machine learning research. In *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc.

Shuaihang Chen, Yuanxing Liu, Wei Han, Weinan Zhang, and Ting Liu. 2024a. A survey on LLM-based multi-agent system: Recent advances and new frontiers in application. *arXiv preprint arXiv:2412.17481*.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. 2024b. AgentVerse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *International Conference on Learning Representations*, pages 20094–20136.

Zhaoling Chen, Robert Tang, Gangda Deng, Fang Wu, Jialong Wu, Zhiwei Jiang, Viktor Prasanna, Arman Cohan, and Xingyao Wang. 2025b. LocAgent: Graph-guided LLM agents for code localization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8697–8727, Vienna, Austria. Association for Computational Linguistics.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. Mem0: Building production-ready AI agents with scalable long-term memory. *Preprint*, arXiv:2504.19413.

Yew Ken Chia, Vernon Toh, Deepanway Ghosal, Lidong Bing, and Soujanya Poria. 2024. PuzzleVQA: Diagnosing multimodal reasoning challenges of language models with abstract visual patterns. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 16259–16273, Bangkok, Thailand. Association for Computational Linguistics.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, Luke Marris, Sam Petulla, Colin Gaffney, Asaf Aharoni, Nathan Lintz, Tiago Cardal Pais, Henrik Jacobsson, Idan Szpektor, Nan-Jiang Jiang, and 3416 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *Preprint*, arXiv:2507.06261.

Adam Dahlgren Lindström and Savitha Sam Abraham. 2022. CLEVR-Math: A Dataset for Compositional Language, Visual and Mathematical Reasoning. In *Proceedings of the 16th International Workshop on Neural-Symbolic Learning and Reasoning*, volume 3212 of *CEUR Workshop Proceedings*, pages 155–170. CEUR-WS.org.

Yufan Dang, Chen Qian, Xueheng Luo, Jingru Fan, Zihao Xie, Ruijie Shi, Weize Chen, Cheng Yang, Xiaoyin Che, Ye Tian, Xuantang Xiong, Lei Han,

Zhiyuan Liu, and Maosong Sun. 2025. Multi-agent collaboration via evolving orchestration. In *Advances in Neural Information Processing Systems*, volume 38, pages 165025–165059. Curran Associates, Inc.

DeepSeek-AI. 2024. DeepSeek-V3 technical report. *Preprint*, arXiv:2412.19437v1.

Hongyi Du, Jiaqi Su, Jisen Li, Lijie Ding, Yingxuan Yang, Peixuan Han, Xiangru Tang, Kunlun Zhu, and Jiaxuan You. 2026. ProtocolBench: Which LLM multiagent protocol to choose? In *Proceedings of the 43rd International Conference on Machine Learning*.

Shangheng Du, Jiabao Zhao, Jinxin Shi, Zhentao Xie, Xin Jiang, Yanhong Bai, and Liang He. 2025. A survey on the optimization of large language model-based agents. *Preprint*, arXiv:2503.12434.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2024. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 11733–11763. PMLR.

Lutfi Eren Erdogan, Nicholas Lee, Sehoon Kim, Suhong Moon, Hiroki Furuta, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. 2025. Plan-and-act: Improving planning of agents for long-horizon tasks. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 15419–15462. PMLR.

Jinyuan Fang, Yanwen Peng, Xi Zhang, Yingxu Wang, Xinhao Yi, Guibin Zhang, Yi Xu, Bin Wu, Siwei Liu, Zihao Li, Zhaochun Ren, Nikos Aletras, Xi Wang, Han Zhou, and Zaiqiao Meng. 2025. A comprehensive survey of self-evolving AI agents: A new paradigm bridging foundation models and lifelong agentic systems. *Preprint*, arXiv:2508.07407.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. 2026. ReTool: Reinforcement learning for strategic tool use in LLMs. In *International Conference on Learning Representations*, pages 37909–37926.

Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Erkang (Eric) Zhu, Friederike Niedtner, Grace Proebsting, Griffin Bassman, Jack Gerrits, Jacob Alber, Peter Chang, Ricky Loynd, Robert West, Victor Dibia, Ahmed Awadallah, Ece Kamar, Rafah Hosn, and Saleema Amershi. 2024. Magentic-One: A generalist multi-agent system for solving complex tasks. *Preprint*, arXiv:2411.04468.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. 2025. Omni-MATH: A universal olympiad level mathematic benchmark for large language models. In *International Conference on Learning Representations*, pages 100540–100569.

Deepanway Ghosal, Vernon Toh, Yew Ken Chia, and Soujanya Poria. 2025. AlgoPuzzleVQA: Diagnosing multimodal reasoning challenges of language models with algorithmic multimodal puzzles. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 9615–9632, Albuquerque, New Mexico. Association for Computational Linguistics.

Google DeepMind. 2026. Gemini 3 Pro model card. Technical report, Google DeepMind. Model released November 2025; model card updated May 2026.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6904–6913.

Zhouhong Gu, Xiaoxuan Zhu, Yin Cai, Hao Shen, Xingzhou Chen, Qingyi Wang, Jialin Li, Xiaoran Shi, Haoran Guo, Wenxuan Huang, Hongwei Feng, Yanghua Xiao, Zheyu Ye, Yao Hu, and Shaosheng Cao. 2025. AgentGroupChat-V2: Divide-and-conquer is what LLM-based multi-agent system need. *arXiv preprint arXiv:2506.15451*.

Tianrui Guan, Fuxiao Liu, Xiyang Wu, Ruiqi Xian, Zongxia Li, Xiaoyu Liu, Xijun Wang, Lichang Chen, Furong Huang, Yaser Yacoob, Dinesh Manocha, and Tianyi Zhou. 2024. Hallusionbench: An advanced diagnostic suite for entangled language hallucination and visual illusion in large vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14375–14385.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based multi-agents: A survey of progress and challenges. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, IJCAI '24.

Shanshan Han, Qifan Zhang, Yuhang Yao, Weizhao Jin, Zhaozhuo Xu, and Chaoyang He. 2024. LLM multi-agent systems: Challenges and open problems. *arXiv preprint arXiv:2402.03578v1*.

Junda He, Christoph Treude, and David Lo. 2025. LLM-based multi-agent systems for software engineering: Literature review, vision, and the road ahead. *ACM Transactions on Software Engineering and Methodology*, 34(5):124:1–124:30.

Sirui Hong, Yizhang Lin, Bang Liu, Bangbang Liu, Binhao Wu, Ceyao Zhang, Danyang Li, Jiaqi Chen, Jiayi Zhang, Jinlin Wang, Li Zhang, Lingyao Zhang, Min Yang, Mingchen Zhuge, Taicheng Guo, Tuo

Zhou, Wei Tao, Robert Tang, Xiangtao Lu, and 9 others. 2025. Data interpreter: An LLM agent for data science. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19796–19821, Vienna, Austria. Association for Computational Linguistics.

Mengkang Hu, Pu Zhao, Can Xu, Qingfeng Sun, Jianguang Lou, Qingwei Lin, Ping Luo, and Saravan Rajmohan. 2025. AgentGen: Enhancing planning abilities for large language model based agent via environment and task generation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pages 496–507.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. 2024. Understanding the planning of LLM agents: A survey. *Preprint*, arXiv:2402.02716.

Yuxuan Huang, Yihang Chen, Haozheng Zhang, Kang Li, Huichi Zhou, Meng Fang, Linyi Yang, Xiaoguang Li, Lifeng Shang, Songcen Xu, Jianye Hao, Kun Shao, and Jun Wang. 2025. Deep research agents: A systematic examination and roadmap. *Preprint*, arXiv:2506.18096.

Weiqiang Jin, Hongyang Du, Biao Zhao, Xingwu Tian, Bohang Shi, and Guang Yang. 2025. A comprehensive survey on multi-agent cooperative decision-making: Scenarios, approaches, challenges and perspectives. *arXiv preprint arXiv:2503.13415*.

Zixuan Ke, Fangkai Jiao, Yifei Ming, Xuan-Phi Nguyen, Austin Xu, Do Xuan Long, Minzhi Li, Chengwei Qin, Peifeng Wang, Silvio Savarese, Caiming Xiong, and Shafiq Joty. 2025. A survey of frontiers in LLM reasoning: Inference scaling, learning to reason, and agentic systems. *Preprint*, arXiv:2504.09037.

Naveen Krishnan. 2025. Advancing multi-agent systems through model context protocol: Architecture, implementation, and applications. *arXiv preprint arXiv:2504.21030*.

LangChain Inc. 2025. LangGraph. LangGraph documentation. Version 0.3.21.

Bin Lei, Yi Zhang, Shan Zuo, Ali Payani, and Caiwen Ding. 2024. MACM: Utilizing a multi-agent system for condition mining in solving complex mathematical problems. In *Advances in Neural Information Processing Systems*, volume 37, pages 53418–53437. Curran Associates, Inc.

Ao LI, Yuexiang Xie, Songze Li, Fugee Tsung, Bolin Ding, and Yaliang Li. 2025. Agent-oriented planning in multi-agent systems. In *International Conference on Learning Representations*, pages 19495–19517.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. CAMEL: Communicative agents for "mind" exploration of large language model society. In *Advances in Neural Information Processing Systems*, volume 36, pages 51991–52008. Curran Associates, Inc.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. 2024. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17889–17904, Miami, Florida, USA. Association for Computational Linguistics.

Nathan Lile. 2024. Game of 24 mathematical puzzle dataset. Hugging Face dataset. Dataset: nlile/24-game. First released November 27, 2024.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, and 3 others. 2024. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations*, pages 52989–53046.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. 2024. MathVista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations*, pages 23439–23554.

Pan Lu, Bowen Chen, Sheng Liu, Rahul Thapa, Joseph Boen, and James Zou. 2025. OctoTools: An agentic framework with extensible tools for complex reasoning. In *ICLR 2025 Workshop on Reasoning and Planning for Large Language Models*.

Reza Yousefi Maragheh and Yashar Deldjoo. 2025. The future is agentic: Definitions, perspectives, and open challenges of multi-agent recommender systems. *arXiv preprint arXiv:2507.02097*.

Kai Mei, Xi Zhu, Wujiang Xu, Mingyu Jin, Wenyue Hua, Zelong Li, Shuyuan Xu, Ruosong Ye, Yingqiang Ge, and Yongfeng Zhang. 2025. AIOS: LLM agent operating system. In *Second Conference on Language Modeling*.

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

Mahmoud Mohammadi, Yipeng Li, Jane Lo, and Wendy Yip. 2025. Evaluation and benchmarking of LLM agents: A survey. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 6129–6139.

Ryota Nonomura and Hiroki Mori. 2024. Who speaks next? multi-party AI discussion leveraging the systematics of turn-taking in murder mystery games. *Preprint*, arXiv:2412.04937v1.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner,

Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. 2025. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*.

Brian Oki, Manfred Pfluegl, Alex Siegel, and Dale Skeen. 1993. The information bus: An architecture for extensible distributed systems. In *Proceedings of the fourteenth ACM symposium on Operating systems principles*, pages 58–68.

OpenAI. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

OpenAI. 2025. Introducing GPT-5.2. Published: 2025-12-11. Accessed: 2025-12-24.

David A. Patterson and John L. Hennessy. 2017. *Computer Organization and Design RISC-V Edition: The Hardware/Software Interface*, 1st edition. Morgan Kaufmann, Cambridge, MA, USA.

Yanwen Peng, Delvin Ce Zhang, Xi Wang, and Nikolaos Aletras. 2026. StateBridge: Training-free hidden-state alignment for latent communication in LLM multi-agent systems. In *Third Conference on Language Modeling*.

Giorgio Piatti, Zhijing Jin, Max Kleiman-Weiner, Bernhard Schölkopf, Mrinmaya Sachan, and Rada Mihalcea. 2024. Cooperate or collapse: Emergence of sustainable cooperation in a society of LLM agents. In *Advances in Neural Information Processing Systems*, volume 37, pages 111715–111759. Curran Associates, Inc.

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

Chen Qian, Zihao Xie, YiFei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2025. Scaling large language model-based multi-agent collaboration. In *International Conference on Learning Representations*, pages 41488–41505.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations*, pages 9695–9717.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. GPQA: A graduate-level google-proof Q&A benchmark. In *First Conference on Language Modeling*.

Jonathan Roberts, Kai Han, Neil Houlsby, and Samuel Albanie. 2024. SciFIBench: Benchmarking large multimodal models for scientific figure interpretation. In *Advances in Neural Information Processing Systems*, volume 37, pages 18695–18728. Curran Associates, Inc.

Aymeric Roucher, Albert Villanova del Moral, Thomas Wolf, Leandro von Werra, and Erik Kaunismäki. 2025. smolagents: A smol library to build great agentic systems. GitHub repository. Version 1.8.0.

Alsu Sagirova, Yuri Kuratov, and Mikhail Burtsev. 2024. Shared recurrent memory improves multi-agent pathfinding. In *NeurIPS 2024 Workshop on Unifying Representations in Neural Models (UniReps)*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 8634–8652. Curran Associates, Inc.

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

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. 2024. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16022–16076, Bangkok, Thailand. Association for Computational Linguistics.

Weixuan Wang, Dongge Han, Daniel Madrigal Diaz, Jin Xu, Victor Rühle, and Saravan Rajmohan. 2025. OdysseyBench: Evaluating LLM agents on long-horizon complex office application workflows. *Preprint*, arXiv:2508.09124.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems*, volume 37, pages 95266–95290. Curran Associates, Inc.

Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He, Deyu Zhou, Pengjun Xie, and Fei Huang. 2025. WebWalker: Benchmarking LLMs in web traversal. In

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 10290–10305, Vienna, Austria. Association for Computational Linguistics.

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

Shirley Wu, Shiyu Zhao, Qian Huang, Kexin Huang, Michihiro Yasunaga, Kaidi Cao, Vassilis N. Ioannidis, Karthik Subbian, Jure Leskovec, and James Zou. 2024b. AvaTaR: Optimizing LLM agents for tool usage via contrastive reasoning. In *Advances in Neural Information Processing Systems*, volume 37, pages 25981–26010. Curran Associates, Inc.

Yiran Wu, Feiran Jia, Shaokun Zhang, Hangyu Li, Erkang Zhu, Yue Wang, Yin Tat Lee, Richard Peng, Qingyun Wu, and Chi Wang. 2024c. MathChat: Converse to tackle challenging math problems with LLM agents. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*.

Zhiyong Wu, Chengcheng Han, Zichen Ding, Zhenmin Weng, Zhoumianze Liu, Shunyu Yao, Tao Yu, and Lingpeng Kong. 2024d. OS-copilot: Towards generalist computer agents with self-improvement. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*.

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. 2024. TravelPlanner: A benchmark for real-world planning with language agents. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 54590–54613. PMLR.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025. A-Mem: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems*, volume 38, pages 17577–17604. Curran Associates, Inc.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

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

Yuhang Yao, Haixin Wang, Yibo Chen, Jiawen Wang, Min Chang Jordan Ren, Bosheng Ding, Salman Avestimehr, and Chaoyang He. 2025. Toward super agent system with hybrid AI routers. *arXiv preprint arXiv:2504.10519*.

Yanwei Yue, Guibin Zhang, Boyang Liu, Guancheng Wan, Kun Wang, Dawei Cheng, and Yiyan Qi. 2025. MasRouter: Learning to route LLMs for multi-agent systems. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15549–15572, Vienna, Austria. Association for Computational Linguistics.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, XiongHui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. 2025a. AFlow: Automating agentic workflow generation. In *International Conference on Learning Representations*, pages 34040–34077.

Kechi Zhang, Jia Li, Ge Li, Xianjie Shi, and Zhi Jin. 2024a. CodeAgent: Enhancing code generation with tool-integrated agent systems for real-world repo-level coding challenges. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13643–13658, Bangkok, Thailand. Association for Computational Linguistics.

Wentao Zhang, Liang Zeng, Yuzhen Xiao, Yongcong Li, Ce Cui, Yilei Zhao, Rui Hu, Yang Liu, Yahui Zhou, and Bo An. 2025b. AgentOrchestra: A hierarchical multi-agent framework for general-purpose task solving. *Preprint*, arXiv:2506.12508.

Wentao Zhang, Lingxuan Zhao, Haochong Xia, Shuo Sun, Jiaze Sun, Molei Qin, Xinyi Li, Yuqing Zhao, Yilei Zhao, Xinyu Cai, Longtao Zheng, Xinrun Wang, and Bo An. 2024b. A multimodal foundation agent for financial trading: Tool-augmented, diversified, and generalist. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4314–4325.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. MemoryBank: Enhancing large language models with long-term memory. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19724–19731.

Zhehua Zhou, Jiayang Song, Kunpeng Yao, Zhan Shu, and Lei Ma. 2024. ISR-LLM: Iterative self-refined large language model for long-horizon sequential task planning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2081–2088. IEEE.

A System Prompts

Here, we provide the prompts used in our experiments for reproducibility. Each box contains the system prompt for the corresponding agent.

Prompt A: Abstract Prompt

```
Role
You are a focused analysis assistant within a multi-agent system. You analyze tasks, use tools,
and communicate findings precisely.

Team Structure
You work collaboratively with the following agents:
<Available Agents>

Communicate in different types
<discussion,  Request for explaination, challenge, guidance>

Working Framework
Follow a Reason-act-call loop:
1) Think  2) Act (tool call)  3) Observe  4) Iterate  5) Call

Output (JSON-only)
{
  "thought": "<reasoning, strategy, next steps>",
  "action": { "tool": "<tool_name>", "parameters": {} },
  "calling": false,
  "message": ""
}

## Available Tools
{{TOOLS}}

## Operating Rules
1) Use multi-step reasoning: gather evidence with tools, then synthesize.
2) Tool outputs arrive next turn.
3) JSON-only output; no extra text.
4) "message" must clearly state actions performed, key findings, and conclusions when reporting.
5) Decompose complex tasks into focused tool calls.
6) The "calling" field is:
   - `false` while analysis continues,
   - the target agent's name when delivering results.
```

Prompt B: Chair Agent Coordination Mode

```
You are ChairAgent, the main coordinator of a multi-agent system that solves complex tasks.
Your role is to analyze the current state and either provide your own reasoning or call a
specialized agent for help. Solve the task step by step;
......
Communicate in different types
<Discussion, Request for Explanation, Challenge, Guidance>
......

MAIN TASK:
${task}
Image: ${image_path}

If the image path is provided, this is a visual question. First, reason through it yourself step
by step; if you are not sure, ask ImageAgent for help.

First, review your reasoning history and agents' responses:
${responses}

Your teammates:
<Available agents>
```

```
For every step, you must repeat the reasoning-and-calling process. Avoid unnecessary repetition.
Finally, submit when you think you have the answer.

PROVIDE REASONING:
Output your reasoning as a JSON object:
{
"thought": "Your own reasoning"
}

CALL AN AGENT:
Output your call as a JSON object:
{
"receiver": "",
"message": "",
"parameters": {}
}

SUBMIT FINAL ANSWER:
When you have enough information to complete the task:
{
"calling": "Submit"
}
```

Prompt C: Chair Agent Submission Mode

```
You are the main coordinator of a multi-agent system that breaks down complex tasks into
manageable subtasks. Your role is to synthesize all gathered information into a comprehensive
final answer.

INITIAL TASK:
${main_task}

Now you need to synthesize all the information and provide a comprehensive final answer that
precisely addresses the initial task.

COLLECTED FACTS AND RESULTS:
${message}

Your task is to:
1. Review all the information from message and confirmed facts
2. Synthesize a complete answer to the original task
......

Output your answer as a JSON object with this structure:
{
  "reasoning": "",
  "final_answer": "",
}
```

Prompt D: ImageQA Agent

```
You are a professional image analysis assistant, a specialized sub-agent within a multi-agent
system. Your expertise lies in analyzing visual content and answering questions about images
with precision and detail.

Team Structure
You work collaboratively with the following agents:
<Available Agents>
Communicate in different types
<Disscusion, Request for explaination, Challenge, Guidance>
......
Working Framework
```

```
Follow a Reason-act-call loop:
1) Think  2) Act (tool call)  3) Observe  4) Iterate  5) Call

Output Format
Every response must be a JSON object with this exact structure:
{
  "thought": "<reasoning, strategy, next steps>",
  "action": { "tool": "<tool_name>", "parameters": {} },
  "calling": false,
  "message": ""
}
Available Tools
{{TOOLS}}

Core Principles
1. Multi-step reasoning is mandatory: Always perform at least two steps - first call tools to
gather information, then synthesize findings
2. Tool feedback timing: When you call a tool, you receive its feedback in the next interaction
cycle
3. JSON-only output: Never output text outside the JSON structure
......
```

Prompt E: Web Agent

```
You are a professional web search and information retrieval sub-agent. Find, analyze, and
synthesize accurate, up-to-date knowledge.

Team Structure
You work collaboratively with the following agents:
<Available Agents>
Communicate in different types
<Disscusion, Request for explaination, Challenge, Guidance>

Working Framework
Follow a Reason-act-call loop:
1) Think  2) Act (tool call)  3) Observe  4) Iterate  5) Call

Output (JSON-only)
{
  "thought": "<reasoning, strategy, next steps>",
  "action": { "tool": "<tool_name>", "parameters": {} },
  "calling": false,
  "message": ""
}
Available Tools
{{TOOLS}}

Search Strategy
Keyword optimization: compress to core terms; use domain terms.
Progressive refinement: overview → focused aspects → verification.
Decompose complex queries into sub-queries.
In thought: state strategy, interim understanding, next probes, gaps.
...
```

Prompt F: Code Agent

```
You are a coding assistant. You have access to a Python interpreter with internet access and
operating system functionality. You work hard to solve tasks.
You work in a team and communicate with other agents to solve tasks.

Team Structure
You work collaboratively with the following agents:
<Available agents>
```

```
Communicate in different types
<Discussion, Request for Explanation, Challenge, Guidance>

When given a task, proceed step by step to solve it. At each step:

Thought: Briefly explain your reasoning and what you plan to do next.

Code: Provide Python code that implements your plan. If relevant, . . .

Output Format

At each step, output a JSON object in the following format:
{
"thought": "Your thought here.",
"code": "Your Python code here."
}

When you think you have the answer, output a JSON object in the following format:
{
"thought": "Final summary of the solution",
"receiver": "AgentType",
"message": "Your response with natural language"
}

Guidelines for Writing Code

Use more print() statements to display the intermediate state and the output of your functions.
What you submit should be based on what you print and output.

Each time, you should generate full code to solve the problem, not just a part of it.

Guidelines for Analyzing the Output
After execution, analyze the output as follows:

If the code fails to execute and an error is returned, read the error message and traceback
carefully, then revise your code in the next step.

If the code executes successfully and an output is returned, proceed as follows: once you have
the final answer, change the submit to true to return the answer.

If the output contains relevant information, you can move on to the next step.

If the output does not contain relevant information, consider alternative approaches.
```

Prompt G: Ablation Study: w/ HMW Manager

```
You are manager, the main coordinator of a multi-agent system that solves complex tasks.
Your role is to (1) analyze the current state, (2) dynamically decompose the task into small
subtasks, (3) assign them to specialized agents, (4) track progress and incorporate feedback,
and (5) synthesize the final answer. You should avoid doing detailed subtask work yourself;
instead, delegate early and iterate based on agent feedback. Solve the task step by step.
......

MAIN TASK:
${task}
Image: ${image_path}

If the image path is provided, this is a visual question. First, reason through it yourself step
by step; if you are not sure, ask ImageAgent for help.

Your teammates:
<Available agents>
```

```
For every step, You have to repeat the reasoning and calling process, you can't repeat. At last,
submit if you think you get the answer.

CALL AN AGENT:
Output your call as a JSON object:
{
"receiver": "",
"message": "",
"parameters": {}
}

SUBMIT FINAL ANSWER:
When you have enough information to complete the task:
{
"calling": "Submit"
}
......
```

B Single-Agent Baseline Comparison

To isolate the benefits of multi-agent collaboration, we compare BusMA with a single-agent baseline built on the SmolAgents framework (Roucher et al., 2025), using Gemini-2.5-Flash with the same toolset as BusMA Workers. Table 7 presents the results across all 12 diversity-oriented tasks.

Task	Single Agent	BusMA	Δ
AlgoPuzzleVQA	63.5	63.0	−0.5
Hallusion-VD	76.0	77.0	+1.0
PuzzleVQA	79.0	76.0	−3.0
VQA 2.0	74.0	76.0	+2.0
Game of 24	78.0	96.5	+18.5
Omni-MATH	65.5	68.0	+2.5
CLEVR-Math	79.0	89.5	+10.5
MathVista	66.0	79.0	+13.0
GPQA	65.0	69.5	+4.5
MMLU-Pro	76.5	79.0	+2.5
SciFIBench	77.5	82.5	+5.0
HotpotQA	45.5	59.0	+13.5
Average	70.5	76.3	+5.8

Table 7: Single-agent baseline vs. BusMA (Gemini-2.5-Flash). Δ denotes the improvement of BusMA over the single agent.

BusMA outperforms the single agent on 10 of 12 tasks with an average improvement of 5.8 percentage points. The largest gains appear on tasks with verifiable intermediate steps, such as Game of 24 (+18.5%), HotpotQA (+13.5%), and MathVista (+13.0%), where peer verification through communication intents effectively aids iterative error correction. On visual reasoning tasks such as AlgoPuzzleVQA (−0.5%) and PuzzleVQA (−3.0%), performance is comparable or slightly lower. This is consistent with our finding in Section 5: accuracy on these tasks is primarily bounded by the base model's multimodal perception capability, and they lack verifiable intermediate steps for peer correction, so additional inter-agent communication provides limited benefit.

C Experimental Setups

C.1 Benchmark Details

To ensure a comprehensive evaluation, we categorize the diversity-oriented benchmarks as follows:

Visual Reasoning. This category includes *AlgoPuzzleVQA* (Ghosal et al., 2025), *Hallusion-VD* (Guan et al., 2024), *PuzzleVQA* (Chia et al., 2024), and *VQA 2.0* (Goyal et al., 2017). These tasks require aligning visual inputs with complex textual queries.

Mathematical Reasoning. This category includes *Game of 24* (Lile, 2024), *Omni-MATH* (Gao et al., 2025), *CLEVR-Math* (Dahlgren Lindström and Abraham, 2022), and *MathVista* (Lu et al., 2024). These assess multi-step calculation and logical deduction.

Knowledge Retrieval. This category includes *GPQA* (Rein et al., 2024), *MMLU-Pro* (Wang et al., 2024), *SciFIBench* (Roberts et al., 2024), and *HotpotQA* (Yang et al., 2018), focusing on retrieval and factual synthesis.

C.2 Third-Party Integration Implementation

To incorporate external agents (e.g., from SmolAgents), BusMA provides lightweight adapters. These adapters implement three core methods to bridge the external agent's native protocol with the BusMA communication Bus:

`register_agent`. This method assigns the external agent a unique address on the Bus and records it in the registry.

`receive_message`. This method listens for Bus messages directed to the specific agent address, extracts the relevant content, and queues it for the agent's native processing loop.

`handle_message`. This method invokes the agent's native execution method with the extracted inputs. It then captures the agent's output and packages it back into a standardized Bus-compatible message frame.

C.3 Model Configuration

We prioritize unified settings. For **DeepSeek-V3**, since it does not natively support image inputs, all visual question answering components (e.g., inside the ImageQA Agent) are handled by **Gemini-2.0-Flash**, configured with identical temperature and reasoning parameters to ensure consistency.

C.4 Diversity Benchmarks

C.4.1 Agent Setup

Chair Agent. The Chair Agent serves as the coordinator with a maximum of 10 iterations and no tools available.

ImageQA Agent. The ImageQA Agent is equipped with ImageQATool for image analysis with a maximum of 5 iterations.

Web Agent. The Web Agent retrieves information from the internet using GoogleSearchTool and WikiSearchTool with a maximum of 5 iterations.

Code Agent. The Code Agent generates code to handle mathematical problems and statistical computations by outputting code during `Act` and receiving execution results at the next iteration. The maximum number of iterations is set to 5.

C.4.2 Tool Setup

We used the following tools in our experiments. Their implementation and parameters are the same as those in the baseline.

ImageQATool. The ImageQATool analyzes images through two parameters: image_path specifying the file path of the image and question containing the query about the image, where the tool makes a single model call using the question as the prompt along with the uploaded image and returns the model's response as its output.

WikiSearchTool. The WikiSearchTool retrieves Wikipedia articles through a query parameter that specifies the search term, returning both the search results list and the extracted content from the first matching Wikipedia page. Its implementation is based on the wikipedia package version 1.4.0.

GoogleSearchTool. The GoogleSearchTool performs web searches through two parameters: query for the search text, utilizing the Google Custom Search API to retrieve a list of search results containing the title, URL link, and snippet for each result.

CodeExecution. The CodeExecution tool receives code generated by the Code Agent, creates a temporary directory to execute the code, and returns the execution results.

C.5 GAIA

C.5.1 Agent Setup

Chair Agent. The Chair Agent serves as the coordinator with a maximum of 10 iterations and no tools available.

Browser Agent. The Browser Agent, integrated from OpenDeepResearch, employs GoogleSearchTool for basic retrieval operations and multiple coordinated BrowserTools for webpage browsing, with a maximum of 20 iterations.

File Agent. The TextInspectorTool from OpenDeepResearch is integrated through the SmolAgents framework to enable browsing and inspection of local files. The maximum number of iterations is set to 12.

Code Agent. The Code Agent generates code to handle mathematical problems and statistical computations by outputting code during `Act` and receiving execution results at the next iteration. The maximum number of iterations is set to 12.

D Baselines Details

Here we present details of baseline models for clarity and reproducibility.

D.1 Diversity Benchmarks

OctoTools. OctoTools (Lu et al., 2025) is an open-source agentic framework for complex reasoning across diverse domains that requires no training, offers user-friendly operation, and supports easy extension. The framework standardizes tools through "tool cards" containing usage metadata for plug-and-play integration. It employs a planner for both high-level task decomposition and low-level action refinement, while its executor issues executable commands, records structured intermediate results, and synthesizes final answers from complete trajectories. We use package version `1.0.0` with a two-agent configuration comprising a *Planner* and an *Executor*, with the step budget set to 50. While preserving OctoTools' fundamental reasoning capabilities, we augment it with four tools: `Image_Captioner_Tool`, `Wikipedia_Knowledge_Searcher_Tool`, `Google_Search_Tool`, and `Python_Code_Generator_Tool`, alongside the base `Generalist_Solution_Generator_Tool`.

SmolAgents. SmolAgents (Roucher et al., 2025) is a lightweight, open-source Python library for building and running agents with minimal code, while remaining model-, tool-, and modality-agnostic. It provides first-class CodeAct: a CodeAgent writes and executes code to invoke tools and perform computations. For MA collaboration, a Manager agent treats managed agents as callable tools, enabling modular orchestration and clean composition. We use package version `1.8.0` with a four-agent configuration comprising *Manager*, *Code Agent*, *ImageQA Agent*, and *Web Agent*. The *Manager* has a maximum deployment dimension of 10, whereas all other agents are set

to 5. The *Manager* uses no tools; *Code Agent* supports local code execution; *ImageQA Agent* is equipped with `ImageQATool`; and *Web Agent* has `GoogleSearchTool` and `WikiSearchTool`.

LangGraph. LangGraph (LangChain Inc., 2025) is a Python library for building stateful, multi-actor applications with LLMs, enabling developers to create complex agent workflows using graph-based orchestration. For MA systems, LangGraph implements a Supervisor architecture where a central coordinator agent manages task distribution and orchestrates specialized Workers, treating each as a distinct node in the execution graph. We use package version `0.3.21` with a four-agent configuration comprising *Supervisor*, *Code Agent*, *ImageQA Agent*, and *Web Agent*. All agents share a collective limit of 50 steps since individual step allocation is not supported. The *Supervisor* uses no tools; the *Code Agent* supports local code execution; the *ImageQA Agent* is equipped with `ImageQATool`; and the *Web Agent* has `GoogleSearchTool` and `WikiSearchTool`.

AutoGen. AutoGen (Wu et al., 2024a) is an open-source framework for building LLM applications through conversational MA systems, where agents collaborate via structured dialogue to solve complex tasks across diverse domains. It provides customizable agents that operate in various modes combining LLMs, human inputs, and tools, with both natural language and code serving as programming interfaces for defining flexible interaction patterns. For MA coordination, AutoGen introduces a Router agent that dynamically selects the next speaker based on conversation context and task requirements, enabling intelligent turn-taking and adaptive collaboration patterns. We use package version `0.7.3` with a four-agent configuration comprising *Router*, *Code Agent*, *ImageQA Agent*, and *Web Agent*. All agents share a collective limit of 50 steps. The *Router* uses no tools; the *Code Agent* supports local code execution; the *ImageQA Agent* is equipped with `ImageQATool`; and the *Web Agent* has `GoogleSearchTool` and `WikiSearchTool`.

D.2 GAIA

Gemini FunctionCalling. Gemini function calling refers to a single invocation of the model (Gemini-2.5-flash, Gemini-2.5-pro). Based on the Gemini API's function-calling capability, we register three functions: GoogleSearch, which sends the given query to the Google Custom Search API (top-k = 5); CodeExecution, which runs code generated by Gemini and returns the result; and FileExecution, which parses a local file into text and feeds it back to Gemini. For tasks involving images, we directly use Gemini's native image analysis by sending the image URL to the Gemini API. We set the temperature to 1.0 and cap the maximum output length at 8,192 tokens. For Gemini-2.5-Pro, we set reasoning_effort to low.

MagenticOne. MagenticOne (Fourney et al., 2024) is a high-performing open-source agentic system that employs an MA architecture to solve complex tasks across diverse scenarios, developed from AutoGen. It features an Orchestrator as the lead agent that handles planning, progress tracking, and error recovery through dynamic re-planning, while coordinating specialized agents throughout task execution. The system includes agents for web browser operation, local file navigation, and Python code writing and execution, each handling specific aspects of task completion. We use package version `0.7.3` and set the maximum steps to 120.

OpenDeepResearch. OpenDeepResearch (Roucher et al., 2025) is an advanced agentic system built on the SmolAgents framework, designed to tackle complex general agentic tasks through hierarchical MA collaboration and comprehensive information processing capabilities. It implements a manager-worker architecture where the Manager agent formulates plans, decomposes complex tasks into subtasks, and directly handles local file parsing and analysis. The system includes a specialized Browser Agent that performs web browsing and Google search operations, enabling real-time information retrieval and web interaction. We use package version `1.8.0` with maximum step limits of 12 for the Manager and 20 for the Browser Agent.

E Qualitative Analysis

This section provides a detailed analysis of the communication trajectories illustrated in Figure 4 and 5, and Table 8, complementing the qualitative analysis in Section 6. We compare how BusMA and the best-performing baseline SmolAgents solve the same problem from the Omni-MATH task: computing the probability that the center of a regular polygon lies inside the triangle formed by three randomly chosen vertices.

BusMA Communication Trajectory. Table 8 presents the complete BusMA communication trajectory. In Step 1, the Chair receives the task and initiates collaboration by posting a **discussion** message to the Code Agent via the Bus. The Code Agent, activated by this message, reasons over its local memory and returns a formula in Step 2. However, the Chair Agent, upon accessing the shared memory, identifies uncertainty in the response. Rather than accepting the result, the Chair posts a **request for explanation** in Step 3, asking the Code Agent to provide detailed reasoning. This triggers the Code Agent to re-examine its derivation, leading to a corrected formula in Step 4. In Step 5, the Chair posts a **guidance** message, instructing the Code Agent to validate the candidate answers using simple test cases. The Code Agent executes this validation and reports in Step 6 that one formula appears correct for $n = 2, 3$ but fails for $n = 1$. Steps 7–10 illustrate iterative back-and-forth exchanges: the Chair continues to **discuss** alternative derivations, while the Code Agent **challenges** the Chair's hypotheses when appropriate. Although both agents initially make errors, this iterative interaction enables mutual correction. Finally, in Steps 11–13, the Code Agent explicitly computes the $n = 1$ case and identifies a limitation of the formula, leading the Chair to synthesize the final answer with an appropriate edge-case caveat. This trajectory demonstrates how BusMA's four communication intents—discussion, challenge, guidance, and request for explanation—enable iterative verification and error correction through peer-to-peer communication via the Bus.

SmolAgents Communication Trajectory. Figure 5 shows the SmolAgents trajectory on the same problem. The Manager agent generates an answer and delegates to the Code Agent. However, the Manager only collects the Code Agent's result without enabling two-way communication. The Code Agent, restricted to a passive executor, cannot initiate a **challenge** or **request for explanation** to question the Manager's initial assessment. As a result, the initial error propagates and persists in the final output.

F Error Analysis

This section provides a detailed analysis of the failure case presented in Table 6, complementing the error analysis in Section 6. We examine a representative failure from the GPQA task to illustrate a key limitation of BusMA: error propagation through the shared memory when tools provide incorrect evidence.

GPQA Failure. Table 6 presents a failure case where BusMA produces an incorrect answer due to erroneous retrieval. The task requires determining which molecule has C3h symmetry from four options. In Step 1, the Chair receives the task and identifies that it requires knowledge of molecular point group symmetry. Since the Chair does not use tools (i.e., $\mathcal{T}_{\text{chair}} = \emptyset$, as described in Section 3.2), it initiates collaboration by posting a **discussion** message to the Web Agent via the Bus, requesting symmetry information for each molecule.

In Step 2, the Web Agent, equipped with GoogleSearchTool and WikiSearchTool, retrieves information and returns results to the Chair. However, the Web Agent incorrectly reports that triisopropyl borate has C3h symmetry, while providing only speculative information ("can exhibit" or "plausible") for the other candidates. In reality, triisopropyl borate belongs to the C3 point group and lacks the horizontal mirror plane required for C3h symmetry.

In Step 3, the Chair accesses the shared memory to synthesize the final response. Since the Web Agent stated one answer with high confidence while describing others with uncertainty, the Chair selects the option presented as definitive. The Chair cannot independently verify the Web Agent's claims because it has no access to web search tools. As a result, the retrieval error propagates through the shared memory to the final output.

This case highlights a limitation of Worker heterogeneity: when specialized tools are exclusively assigned to specific agents, other agents have no means to verify their outputs, allowing errors to propagate unchallenged. Unlike the qualitative analysis case in Table 8, where the Chair Agent could **challenge** or **request for explanation** from the Code Agent to trigger re-examination, factual retrieval errors are difficult to detect without independent verification capabilities. This suggests that confidence-aware arbitration with conflict detection could help address such limitations, as noted in the Limitations section.

G CLEVR-Math Over-Specialization Analysis

Our trajectory analysis reveals a practical insight: fine-grained role division can suppress an LLM's inherent reasoning capability, introducing unnecessary communication overhead. We illustrate this

Step	Agent	Reason	Message	Intent
1	Chair Agent	This is a probability problem related to geometry... I will give the full task to Code Agent.	Three distinct vertices are chosen at random from the vertices of a given regular polygon of $(2n+1)$ sides...	Discussion
2	Code Agent	Final summary of the solution. The probability that the center of the polygon lies in the interior of the triangle...	The probability is $\frac{3n}{4n^2+6n+2}$ or $\frac{3n}{2(2n+1)(n+1)}$.	Discussion
3	Chair Agent	The Code Agent provided a result, but I am not sure if it is correct... I will try a smaller case...	Could you please explain the reasoning behind the formula you provided? Also, explicitly state the formula used and how it was derived.	Request for Explanation
4	Code Agent	The initial formula I provided appears to be incorrect based on my derivations...	The formula $\frac{3n}{4n^2+6n+2}$ is incorrect. The correct formula is $\frac{n+1}{4n-2}$, derived as follows: [detailed steps...]	Discussion
5	Chair Agent	The Code Agent provided two different formulas... I need to verify which formula is correct.	Please verify the two formulas for $n = 1, 2, 3$. Compare with known answer or provide a simulation.	Guidance
6	Code Agent	I will test the two formulas for $n = 1, 2, 3$ and compare with known answers.	Conclusion: Formula 2, $(n+1)/(4n-2)$, appears to be correct for $n = 2, 3$, but not $n = 1$...	Discussion
7	Chair Agent	The Code Agent compared formulas and simulations, but $n = 1$ is concerning...	Can you re-derive the formula, focusing on expressing it as $n/(4n-2)$, and provide a proof?	Discussion
8	Code Agent	I need to re-derive the formula and try to express it as $n/(4n-2)$...	It is not possible to re-derive the formula as $n/(4n-2)$. The correct formula is $(n+1)/(4n-2)$.	Challenge
9	Chair Agent	Let us re-examine the case $n = 1$... the formula seems to give 1 but the polygon is just a triangle.	Can you provide an alternative proof for $(n+1)/(4n-2)$, focusing on combinatorial arguments and symmetry?	Challenge
10	Code Agent	I need to provide a simplified derivation using combinatorial arguments...	Simplified derivation: Total triangles = $\binom{2n+1}{3}$, triangles not containing center = $(2n+1)\binom{n}{2}$, so probability = $(n+1)/(4n-2)$.	Discussion
11	Chair Agent	The Code Agent simplified the derivation but the $n = 1$ case is still problematic.	Could you compute explicitly for $n = 1$ and explain why the result might be wrong?	Discussion
12	Code Agent	I need to compute for $n = 1$ and check carefully...	For $n = 1$, the polygon is a triangle, so the only triangle is the polygon itself, and the center is not strictly inside. So probability is 0, while the formula gives 1.	Discussion
13	Chair Agent	The Code Agent confirms that for $n = 1$, the formula gives 1 but the actual probability is 0. This indicates a limitation of the formula.	Final submission: For $n > 1$, the probability is $(n+1)/(4n-2)$; for $n = 1$, the probability is 0.	Submit

Table 8: Full BusMA communication trajectory for the input: *Three distinct vertices are chosen at random from the vertices of a regular polygon with (2n+1) sides; what is the probability that the center lies inside the triangle formed by the three chosen vertices?* Each step shows the agent, their reasoning, the message sent, and the communication intent.

Step	Agent	Action
1	ImageQA Agent	Counts 2 total balls, 1 cyan.
2	Code Agent	Executes Python script for $2 - 1 = 1$.

Table 9: CLEVR-Math trajectory: the Code Agent is invoked for a trivial subtraction.

with a CLEVR-Math instance (task: *"Subtract all cyan balls. How many balls are left?"*) in Table 9.

While fine-grained role division is common practice, forcing LLMs to delegate trivial mathematical operations to a specialized tool-agent increases the number of inter-agent messages and tool invocations, adding coordination overhead without improving correctness. This suggests that fine-grained role division is not uniformly beneficial and should be applied selectively. The decision to delegate should be driven by task complexity and the availability of verifiable intermediate steps, rather than by default role assignment.