D^{3}MAS: Decompose, Deduce, and Distribute for Enhanced Knowledge Sharing in Multi-Agent Systems

Heng Zhang
South China Normal University
Guangzhou, China
2024025450@m.scnu.edu.cn

Yuling Shi
Shanghai Jiao Tong University
Shanghai, China
yuling.shi@sjtu.edu.cn

Xiaodong Gu
Shanghai Jiao Tong University
Shanghai, China
xiaodong.gu@sjtu.edu.cn

Haochen You
Columbia University
New York, USA
hy2854@columbia.edu

Zijian Zhang
University of Pennsylvania
Philadelphia, USA
zzjharry@alumni.upenn.edu

Lubin Gan
University of Science and Technology of China
Hefei, China
ganlubin@mail.ustc.edu.cn

Yilei Yuan
University of Michigan
Ann Arbor, USA
yiliey@umich.edu

Jin Huang*
South China Normal University
Guangzhou, China
huangjin@m.scnu.edu.cn

ABSTRACT

Multi-agent systems powered by large language models exhibit strong capabilities in collaborative problem-solving. However, these systems suffer from substantial knowledge redundancy. Agents duplicate efforts in retrieval and reasoning processes. This inefficiency stems from a deeper issue: current architectures lack mechanisms to ensure agents share minimal sufficient information at each operational stage. Empirical analysis reveals an average knowledge duplication rate of 47.3% across agent communications. We propose D^{3}MAS (Decompose, Deduce, and Distribute), a hierarchical coordination framework addressing redundancy through structural design rather than explicit optimization. The framework organizes collaboration across three coordinated layers. Task decomposition filters irrelevant sub-problems early. Collaborative reasoning captures complementary inference paths across agents. Distributed memory provides access to non-redundant knowledge. These layers coordinate through structured message passing in a unified heterogeneous graph. This cross-layer alignment ensures information remains aligned with actual task needs. Experiments on four challenging datasets show that D^{3}MAS consistently improves reasoning accuracy by 8.7% to 15.6% and reduces knowledge redundancy by 46% on average.

KEYWORDS

Multi-agent Systems, Multi-agent Reasoning, Graph-based Learning, Knowledge Sharing, Collaborative Inference

*Corresponding author.

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/WQMU8577

ACM Reference Format:
Heng Zhang, Yuling Shi, Xiaodong Gu, Haochen You, Zijian Zhang, Lubin Gan, Yilei Yuan, and Jin Huang. 2026. D^{3}MAS: Decompose, Deduce, and Distribute for Enhanced Knowledge Sharing in Multi-Agent Systems. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 10 pages. https://doi.org/10.65109/WQMU8577

1 INTRODUCTION

Large language models (LLMs) have transformed artificial intelligence by integrating vast knowledge into coherent understanding and generation[20, 34]. These advancements are driven by scaling models, datasets, and computational resources[20, 27], leading to notable performance gains[35, 51]. However, LLMs face difficulties in reasoning, particularly for complex tasks that go beyond textual comprehension alone[36, 44]. To address these limitations, recent efforts have equipped LLMs with external tools[32, 36], memory systems[18, 29], and planning capabilities[47, 59], allowing them to function as autonomous agents[41]. Multi-agent systems emerge naturally from this approach[6], enabling multiple agents to collaboratively tackle intricate tasks in shared environments through joint reasoning. By combining the expertise of diverse agents, such systems enable iterative problem-solving and more refined decision-making. The effectiveness of such collaboration hinges on how information flows among agents.

To enable effective information flow, researchers have proposed diverse strategies. Interaction-driven methods[3, 10] guide structured discussions, helping agents refine their ideas through mutual critique and collaborative iteration[23]. Conversation-based frameworks[16] adaptively decompose complex tasks into manageable parts through ongoing dialogues, enabling agents to tackle problems incrementally[56]. Role assignment strategies[16, 48] improve efficiency by distributing focused responsibilities, allowing agents to specialize in specific aspects of a task[31]. Messaging frameworks[12, 53] facilitate direct information exchange during collaboration, while shared knowledge pools[13] provide a

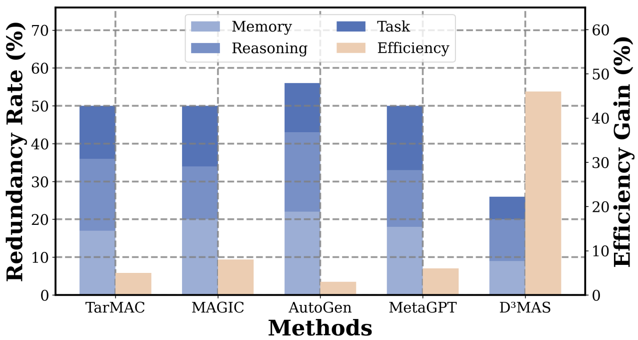

Figure 1: Redundancy Breakdown by Coordination Method. We measure three types of redundancy across coordination methods. Memory redundancy represents the percentage of knowledge items retrieved by multiple agents. Reasoning redundancy quantifies semantically similar inference steps with cosine similarity exceeding 0.85. Task redundancy captures the percentage of overlapping sub-task assignments among agents. Stacked bars display total redundancy rate on the left vertical axis. Separate bars indicate computational efficiency gains relative to MetaGPT baseline on the right vertical axis. D^{3}MAS reduces redundancy by 46% on average through cross-layer coordination. Results are averaged over HotpotQA and MMLU datasets with 4 to 8 agents per query.

centralized resource to address information gaps among agents. Memory-augmented systems[29, 64] support reasoning by organizing collaborative histories and maintaining continuity across interactions[28]. Despite their contributions, these approaches often lead to inefficiencies[10, 33]. Agents frequently overlap in knowledge retrieval, duplicate reasoning, and misalign task divisions, leading to wasted resources and weaker collaboration[22, 43]. Balancing redundancy reduction with maintaining shared knowledge depth is essential for enhancing accuracy and efficiency[2, 63].

The inefficiencies in existing multi-agent systems originate from their fragmented design, where task coordination, reasoning execution, and memory retrieval are treated as independent components. This separation disrupts the synchronization of actions across problem-solving stages, as decisions made by one layer often lack context from the others. Viewed through the lens of information theory, this fragmentation disrupts the efficient and purposeful flow of knowledge required for collaboration. Effective collaboration requires ensuring that the information exchanged between agents and across layers remains minimal yet sufficient to achieve the task objectives. When operational layers function in isolation, task decomposition generates sub-problems without considering the reasoning context needed to solve them. Similarly, reasoning processes frequently revisit redundant inference paths due to a lack of visibility into task dependencies. Memory retrieval further exacerbates these inefficiencies by returning overlapping or irrelevant knowledge, as it lacks alignment with active reasoning requirements. Without mechanisms to systematically filter and align information, fragmented designs lead to duplication and misaligned operations, compounding inefficiencies throughout the system. These issues highlight the need for a unified framework that optimizes information flow between components, reducing redundancy while preserving coordination across all stages of collaborative reasoning.

The core challenge in overcoming these inefficiencies lies in achieving integration across task planning, reasoning, and memory retrieval. Effective multi-agent collaboration requires a systematic approach to align these layers so that information processing at one stage is informed by and supports the needs of others. From an information-theoretic perspective, this integration must aim to jointly optimize information utility and redundancy, ensuring that each layer exchanges only the essential knowledge necessary for task completion. Task planning should produce sub-problems that are context-aware and aligned with reasoning objectives. Reasoning processes must prioritize inference paths that avoid duplication while leveraging dependencies from task context. Memory retrieval should focus on filling knowledge gaps directly relevant to active problem-solving rather than returning overlapping or unrelated results. Existing architectures, which often treat these layers as disconnected modules, fail to establish such alignment, resulting in recurring inefficiencies. A unified framework that systematically connects these processes can reduce redundancy, promote efficient information flow, and enable agents to synchronize actions under complex and asynchronous conditions.

To address these challenges, we introduce D^{3}MAS, a hierarchical coordination framework that integrates previously fragmented processes into a unified structure. The framework organizes agent interactions through a heterogeneous graph comprising three coordinated layers for task planning, reasoning execution, and memory retrieval. The task layer decomposes complex queries into minimal sub-problems aligned with reasoning objectives. The reasoning layer prioritizes complementary inference paths across agents to eliminate redundant operations. The memory layer provides access to task-relevant knowledge while filtering overlapping retrievals. These layers coordinate through structured message passing, enabling bidirectional information flow where task requirements guide reasoning and memory access while reasoning feedback refines task decomposition. This cross-layer alignment systematically reduces redundancy and ensures agents operate on information tailored to actual task needs. Instead of explicitly optimizing information-theoretic objectives, D^{3}MAS realizes minimal sufficient information sharing through structural design. Our contributions are summarized as follows:

- We identify a critical limitation in existing multi-agent systems caused by the lack of hierarchical coordination, which leads to severe knowledge redundancy in agent communications. Our analysis demonstrates a duplication rate of 47.3%, highlighting the urgent need for effective solutions.
- We propose **D^{3}MAS**, a novel unified framework for multi-agent collaboration based on a heterogeneous graph architecture with explicit dependency modeling. This design significantly reduces redundancy while maintaining agent autonomy, enabling efficient reasoning across various tasks and domains.
- Extensive experiments on challenging benchmarks, including HumanEval and MMLU, demonstrate consistent improvements with D^{3}MAS. Compared to state-of-the-art baselines,

it achieves 8.7%-15.6% higher accuracy while reducing knowledge redundancy by 46% on average and lowering computational overheads.

2 RELATED WORK

2.1 LLM-Agent Collaboration

LLMs have shown impressive capabilities but still face inherent limitations[25, 44], motivating the development of autonomous agents equipped with context-aware memory[29, 64], tool usage[32, 36], procedural planning[47, 58], and role-playing abilities[37] that transform LLMs into versatile problem solvers[46, 55]. Multi-agent collaboration has emerged as a powerful paradigm that combines specialized agent strengths[6], it often outperforming individual models[10, 22]. In software engineering, hierarchical debugging systematically resolves code errors at multiple granularity levels[39], competitive debate mechanisms enable diverse reasoning through structured interactions[21], experience-driven approaches distill knowledge from historical trajectories[5], and repository-level frameworks navigate dependencies through graph-based coordination[30]. These capabilities are further supported by research on machine-generated code patterns[40], long-context compression[11, 38, 60], reinforcement learning-based reasoning[26], and cross-language translation[45]. While majority voting represents a basic collaboration form where agents operate independently[7, 49], more effective systems establish interconnected structures that foster interdependent interactions[2, 16]. Recent research explores various communication topologies[43, 62]: chain topologies arrange agents sequentially[53], star architectures employ a central agent to manage subordinates[3], tree structures enable the hierarchical management[9], and graph-based approaches offer flexible interaction patterns[61]. These systems find applications across diverse domains including software development, medical diagnosis, and scientific research[12].

2.2 Multi-Agents as Graphs

Graphs are an essential data structure for representing relationships between entities[1, 14]. Before the era of LLMs, graph-based approaches already played a key role in multi-agent reinforcement learning by modeling interactions in a structured way[19, 42]. With the rise of LLM-based agents, researchers recognize that agent interactions can be naturally represented using graphs, evolving from implicit usage to explicit graph-structured definitions[54]. However, existing approaches rely on predefined or statically optimized topologies that lack task awareness. Practical applications reveal the importance of task-aware graph construction: repository-level code understanding benefits from dependency-aware coordination[30], issue resolution frameworks leverage fault propagation graphs for diagnosis[21], hierarchical debugging employs code structure decomposition[39], and experience-driven systems build knowledge from repair patterns[5]. Effective multi-agent systems require hierarchical organization where different coordination patterns correspond to different operational levels, and knowledge sharing must consider both information structure and contextual requirements[13]. The key challenge is creating frameworks that dynamically organize agent interactions based on task needs while maintaining coherent reasoning across distributed agents[33]. Our work addresses this by introducing a heterogeneous graph architecture that explicitly models three operational layers, allowing agents to adaptively adjust information flow based on evolving reasoning requirements.

3 PRELIMINARY

3.1 Multi-Agent Reasoning Systems

Multi-agent reasoning systems rely on collaboration among autonomous agents to solve complex problems that exceed individual capabilities. We formalize such a system as $\mathcal{S} = \{A_1, A_2, \ldots, A_n\}$, where each agent A_i operates with its own reasoning process and knowledge base $\mathcal{K}_i$. Given a complex query q, the system produces a comprehensive answer a through collaborative reasoning. Each agent generates partial reasoning chains r_i toward solving q. The collective output emerges from aggregating these individual contributions. Traditional approaches treat agent interactions as isolated message exchanges. Agent A_i sends messages $m_{i \to j}$ to agent A_j through natural language texts containing task descriptions, reasoning steps, or factual knowledge. However, this unstructured communication suffers from a fundamental limitation: without coordination mechanisms, agents cannot determine what information others need or already possess. This leads to two critical problems. First, agents may retrieve the same knowledge independently, wasting computational resources. Second, agents may pursue redundant reasoning paths without recognizing overlaps in their inference processes.

3.2 Heterogeneous Graph Representation

Heterogeneous graphs provide a natural solution to these coordination challenges. Unlike homogeneous graphs where all nodes and edges share the same type, heterogeneous graphs can explicitly model different entity types and their relationships. This makes them well-suited for representing multi-agent systems where tasks, reasoning steps, and knowledge pieces play distinct roles. Formally, we define a typed heterogeneous graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{T}_v, \mathcal{T}_e, \phi, \psi)$. Here $\mathcal{V}$ and $\mathcal{E}$ denote the node and edge sets. The mapping functions $\phi : \mathcal{V} \to \mathcal{T}_v$ and $\psi : \mathcal{E} \to \mathcal{T}_e$ assign each node and edge to specific types from sets $\mathcal{T}_v$ and $\mathcal{T}_e$. Each node $v \in \mathcal{V}$ carries semantic content c_v and an embedding $h_v \in \mathbb{R}^d$. Edges $(u, v) \in \mathcal{E}$ encode dependencies or information flow between nodes.

Information propagates through heterogeneous graphs via type-aware message passing. Different node types aggregate messages differently based on their semantic roles:

$$h_v^{(l+1)} = \sigma\left(\sum_{\tau \in \mathcal{T}_v} \sum_{u \in \mathcal{N}_\tau(v)} \alpha_\tau W_\tau h_u^{(l)}\right), \tag{1}$$

where $h_v^{(l+1)}$ represents the updated embedding at layer $l+1$. The neighbors $\mathcal{N}_\tau(v)$ are those of type τ connected to node v. Type-specific weights W_τ transform neighbor embeddings $h_u^{(l)}$, while α_τ controls each type's contribution. The activation function σ introduces non-linearity. This formulation naturally extends to multi-agent scenarios where different operational levels correspond to different node types.

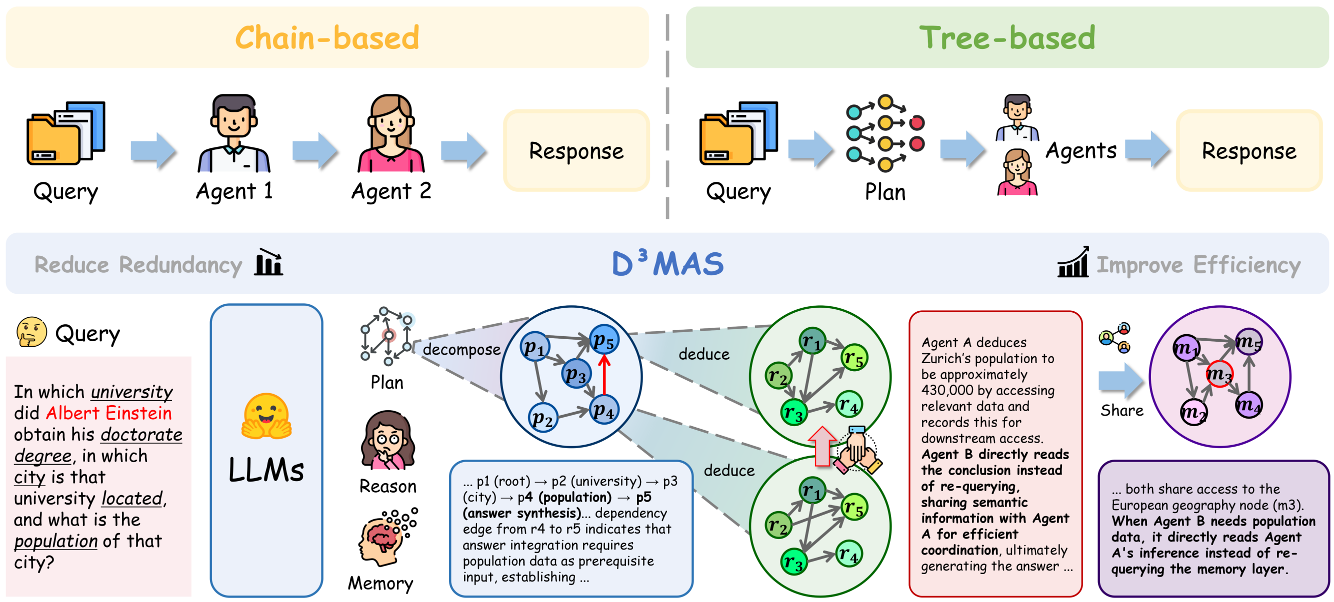

Figure 2: D³MAS hierarchical framework structures multi-agent reasoning through Decompose (blue), Deduce (green), and Distribute (purple) layers. The Decompose layer builds task dependency graphs like p4→p5 to avoid sub-problem overlap in chain-based systems. The Deduce layer uses dependency edges to enable reasoning reuse. Agent B directly reads Agent A's conclusions through the pink arrow, avoiding redundant inference. The Distribute layer assigns knowledge nodes to specific agents. Agent A is responsible for m1 and m2, Agent B for m4 and m5, and m3 is shared between both. This design reduces knowledge redundancy by 46% through hierarchical message passing and significantly improves efficiency compared to chain-based and existing graph methods.

4 METHODOLOGY

4.1 Framework Overview

Our framework, D³MAS, realizes the principle of minimal sufficient information sharing through hierarchical coordination. The core idea is that proper system organization can implicitly enforce information optimality without explicit bottleneck optimization.

D³MAS constructs a unified heterogeneous graph $\mathcal{G}_{\text{D}^3\text{MAS}}$ to organize multi-agent collaboration. The graph contains three node types, each addressing information redundancy at a different operational level. Task decomposition nodes ($\mathcal{T}_{\text{task}}$) implement the *Decompose* principle. They break complex queries into minimal necessary sub-problems, filtering out irrelevant task branches early. Reasoning nodes ($\mathcal{T}_{\text{reason}}$) embody the *Deduce* principle. They capture complementary inference paths across agents, avoiding redundant reasoning chains. Memory nodes ($\mathcal{T}_{\text{memory}}$) realize the *Distribute* principle. They organize access to sufficient yet non-redundant knowledge across distributed memory. The node type set is $\mathcal{T}_v = \{\mathcal{T}_{\text{task}}, \mathcal{T}_{\text{reason}}, \mathcal{T}_{\text{memory}}\}$. Six edge types model information flow across and within layers:

$$\mathcal{T}_e = \{e_{\text{decompose}}, e_{\text{trigger}}, e_{\text{depend}}, e_{\text{retrieve}}, e_{\text{ground}}, e_{\text{relate}}\} \tag{2}$$

Each edge type encodes a specific interaction pattern. The framework processes queries through iterative message passing. This updates node representations and propagates information bidirectionally across the hierarchy. The design enables agents to coordinate naturally through the shared graph structure, eliminating the need for explicit communication protocols.

4.2 Decompose: Task Decomposition Layer

The task decomposition layer addresses a fundamental source of redundancy: uncoordinated task splitting. When agents independently break down problems, they often generate overlapping subtasks. Our approach makes decomposition structure explicit, allowing agents to see the global task hierarchy.

This layer constructs a graph $\mathcal{G}^{(0)} = (\mathcal{V}^{(0)}, \mathcal{E}^{(0)})$ where nodes represent sub-problems derived from the original query q. We initialize with a root task node v_{root} that encodes the complete query via embedding function $f_{\text{embed}} : \text{Text} \to \mathbb{R}^d$.

The decomposition process generates child task nodes recursively. A large language model $\text{LLM}_{\text{decomp}}$ identifies natural subproblems. Each generated sub-task t_j creates a new node $v_j^{(0)}$ with semantic content $c_{v_j} = t_j$ and embedding $h_{v_j} = f_{\text{embed}}(t_j)$. Decomposition edges $e_{\text{decompose}}$ connect parent tasks to their sub-tasks, forming a directed tree that captures problem hierarchy.

The complete task node set is:

$$\mathcal{V}^{(0)} = \{v_{\text{root}}\} \cup \bigcup_{j=1}^{m} \{v_j^{(0)} \mid v_j^{(0)} = \text{Create}(t_j), \tag{3}$$

$$t_j \in \text{LLM}_{\text{decomp}}(v_{\text{parent}})\}, \tag{4}$$

where $\text{Create}(t_j)$ instantiates a task node with content t_j. Any existing task node v_{parent} eligible for further decomposition can spawn children. The parameter m represents the total number of generated sub-tasks. Decomposition continues recursively until reaching atomic sub-problems that individual agents can address directly.

Once decomposition completes, we assign sub-tasks to agents. The assignment maximizes the match between agent expertise and task requirements:

$$\text{Assign}(v_j^{(0)}) = \arg\max_{A_i \in \mathcal{S}} \text{Capability}(A_i, v_j^{(0)}), \tag{5}$$

where $\text{Capability}(A_i, v_j^{(0)})$ computes semantic similarity between agent A_i's profile and task $v_j^{(0)}$'s requirements.

This explicit tree structure serves an important information-theoretic purpose. By organizing sub-tasks hierarchically, agents gain visibility into the global problem structure. Each agent can identify which sub-problems others are handling. This naturally reduces redundancy from uncoordinated task splitting. Agents avoid duplicating efforts because the decomposition structure makes task assignments transparent.

4.3 Deduce: Collaborative Reasoning Layer

The reasoning layer addresses redundancy in inference processes. When agents reason independently, they often pursue overlapping logical paths. Our approach makes reasoning dependencies explicit, enabling agents to recognize and build upon others' conclusions.

This layer maintains a graph $\mathcal{G}^{(1)} = (\mathcal{V}^{(1)}, \mathcal{E}^{(1)})$ where nodes represent inference steps from different agents. Each agent A_i receives task assignments from Layer 0 through trigger edges e_{trigger}. Upon receiving task $v_k^{(0)}$, agent A_i generates a reasoning node capturing its inference process:

$$v_{i,k}^{(1)} = \text{LLM}_{A_i}(\text{Concat}(c_{v_k^{(0)}}, C_{A_i})), \tag{6}$$

where LLM_{A_i} denotes agent A_i's language model. The input combines task content $c_{v_k^{(0)}}$ with agent-specific context C_{A_i}, which includes relevant memory and prior reasoning. The reasoning content encompasses intermediate conclusions and logical justifications.

The key to reducing reasoning redundancy lies in dependency edges. These connect reasoning nodes when one inference step builds upon another's conclusions:

$$\mathcal{E}_{\text{depend}} = \{(v_{i,k}^{(1)}, v_{j,l}^{(1)}) \mid \text{Premise}(v_{i,k}^{(1)}) \cap \text{Conclusion}(v_{j,l}^{(1)}) \neq \emptyset\}, \tag{7}$$

where $\text{Premise}(\cdot)$ extracts logical preconditions and $\text{Conclusion}(\cdot)$ identifies derived statements. Nodes $v_{i,k}^{(1)}$ and $v_{j,l}^{(1)}$ may come from different agents.

Additional edge types connect reasoning to other layers. Retrieval edges e_{retrieve} link reasoning nodes to memory nodes when agents need factual knowledge. Grounding edges e_{ground} establish bidirectional connections between reasoning and the memory supporting it.

This design creates a reasoning graph spanning multiple agents while maintaining logical coherence. By making dependencies explicit through $\mathcal{E}_{\text{depend}}$, agents recognize when their reasoning builds on others' work. This enables them to contribute complementary inferences rather than redundant ones. The structural coordination implicitly enforces information efficiency: agents focus on reasoning steps that add new logical content to the collective inference process.

4.4 Distribute: Distributed Memory Layer

The memory layer addresses redundancy in knowledge retrieval. When agents independently query knowledge bases, they often retrieve overlapping information. Our approach coordinates retrieval across agents, ensuring each accesses only what others have not already obtained. This layer organizes factual knowledge in graph $\mathcal{G}^{(2)} = (\mathcal{V}^{(2)}, \mathcal{E}^{(2)})$ where nodes represent entities and concepts. Each agent A_i maintains a local memory subgraph $\mathcal{G}_i^{(2)} \subset \mathcal{G}^{(2)}$ containing domain-relevant knowledge. Memory nodes $v_l^{(2)}$ store entity descriptions and factual statements. Semantic relation edges e_{relate} connect related concepts within and across agent boundaries.

When agent A_i requires knowledge, it issues a query q_{mem} derived from its current reasoning context. The retrieval mechanism scores memory nodes by computing similarity:

$$\text{Score}(v_l^{(2)}, q_{\text{mem}}) = \frac{h_{v_l} \cdot f_{\text{embed}}(q_{\text{mem}})}{\|h_{v_l}\| \cdot \|f_{\text{embed}}(q_{\text{mem}})\|}, \tag{8}$$

where h_{v_l} denotes the memory node's embedding. Higher scores indicate stronger relevance to the current reasoning context.

To implement distributed retrieval, we aggregate top-k nodes from all agent memory spaces:

$$\mathcal{M}_{\text{retrieve}} = \text{Top-}k\left(\bigcup_{i=1}^{n} \{v \in \mathcal{G}_i^{(2)} \mid \text{Score}(v, q_{\text{mem}}) > \theta\}\right), \tag{9}$$

where θ is a relevance threshold. Cross-agent knowledge sharing occurs when agent A_i retrieves memory nodes from agent A_j's local subgraph. The framework tracks knowledge provenance by recording which agent originally contributed each memory node.

This design prevents redundant retrieval while enabling agents to identify complementary knowledge. From an information flow perspective, the retrieval mechanism acts as a natural filter. By scoring relevance and selecting top-k nodes, agents access minimal sufficient memory—knowledge necessary for their current reasoning context without introducing superfluous information that would increase redundancy.

4.5 Hierarchical Message Passing

The hierarchical message passing mechanism ties the three layers together. It coordinates information flow to ensure that each layer's processing aligns with others' requirements. This cross-layer alignment is the key to reducing overall system redundancy.

At each iteration t, nodes update their representations by aggregating typed messages from neighbors. The update function depends on both node type and incoming edge types. Task nodes $v^{(0)} \in \mathcal{V}^{(0)}$ aggregate progress signals from triggered reasoning nodes through e_{trigger} edges. Reasoning nodes $v^{(1)} \in \mathcal{V}^{(1)}$ combine task guidance from Layer 0, dependency information from peer reasoning nodes, and factual support from Layer 2 memory nodes. Memory nodes $v^{(2)} \in \mathcal{V}^{(2)}$ refine embeddings based on usage patterns from reasoning nodes that retrieve them.

The update process follows:

$$h_v^{(t+1)} = \mathrm{UPDATE}^{(\phi(v))}\left(h_v^{(t)}, \bigoplus_{u\in\mathcal{N}(v)} \mathrm{MSG}^{(\psi(u,v))}(h_u^{(t)})\right), \quad (10)$$

where $h_v^{(t+1)}$ denotes the updated embedding at iteration $t+1$. The function $\mathrm{UPDATE}^{(\phi(v))}$ is type-specific, determined by node type $\phi(v)$. The message function $\mathrm{MSG}^{(\psi(u,v))}$ depends on edge type $\psi(u,v)$ connecting nodes u and v. The aggregation operator $\bigoplus$ combines messages through summation or attention-weighted averaging based on edge semantics.

Message passing alternates between bottom-up and top-down phases. Bottom-up messages flow from memory through reasoning to tasks. They carry evidence and intermediate conclusions. Top-down messages flow from tasks through reasoning to memory. They carry refined requirements and focus adjustments.

This bidirectional flow enables continuous alignment. Reasoning needs reshape task decomposition and memory access patterns simultaneously. The key to reducing redundancy lies in this cross-layer coordination: when reasoning nodes signal their information needs to memory nodes (bottom-up), and task nodes propagate constraints to reasoning nodes (top-down), the system achieves implicit information optimization. Each layer's processing is informed by others' requirements, ensuring that information flowing through the graph aligns with actual task needs rather than containing arbitrary redundancies.

5 EXPERIMENTS

5.1 Datasets

We evaluate our framework on diverse publicly available benchmarks that challenge various reasoning capabilities. **MMLU** [15] (Massive Multitask Language Understanding) assesses logical reasoning across 57 subjects spanning STEM, humanities, and social sciences through multiple-choice questions requiring extensive world knowledge and problem-solving ability. **HumanEval** [4] tests code generation capabilities through 164 hand-crafted programming problems that measure functional correctness in synthesizing programs from docstrings. **CommonGen** [24] examines commonsense reasoning through constrained text generation, requiring models to produce coherent sentences from given concept sets. **ARC-Challenge** [8] (AI2 Reasoning Challenge) contains grade-school science questions that demand advanced reasoning beyond simple retrieval. Together, these datasets provide comprehensive evaluation across factual retrieval, logical inference, code synthesis, and compositional understanding.

5.2 Baselines

We compare our framework against diverse baseline methods spanning different reasoning paradigms. **Chain-of-Thought (CoT)** [52] enables language models to generate intermediate reasoning steps for coherent explanations. **Self-Consistency with CoT (CoT-SC)** [50] samples multiple reasoning paths and selects answers via majority voting. **Reflexion** [41] employs verbal reinforcement learning for iterative self-refinement. **AutoGPT** uses multi-step planning and tool-augmented reasoning with iterative feedback loops. **MetaGPT** [16] simulates software development workflows through specialized role assignment. **AutoGen** [53] provides conversational frameworks for flexible asynchronous multi-agent coordination. **MACNET** [17] organizes agent interactions through directed acyclic graphs for topologically designed reasoning. These baselines highlight strategies including single-agent reasoning, tool-augmented decomposition, role-based collaboration, debate-driven consensus, and graph-oriented coordination.

5.3 Evaluation Metrics

To ensure multidimensional assessment, we employ accuracy-based metrics and specialized evaluation dimensions from MAgIC [57]. **Accuracy** serves as the primary metric across all benchmarks, measuring the percentage of correctly solved problems. For deeper analysis, we adopt seven dimensions from MAgIC: *Judge* evaluates decision-making quality in uncertain scenarios; *Reason* assesses logical coherence and inference validity; *Decept* measures resistance to misleading information; *Self-Aware* examines the ability to recognize knowledge limitations; *Compre* (Comprehensiveness) evaluates reasoning coverage and completeness; *Coord* (Coordination) measures collaborative effectiveness; and *Rational* assesses utility maximization in decision-making. We employ GPT-4 as the evaluator with carefully designed prompts for consistency and impartiality, scoring outputs from 1 to 10. Additionally, we conduct pairwise comparisons calculating win rates for direct performance comparison. This robust framework provides detailed understanding of both answer quality and reasoning effectiveness.

5.4 Implementation Details

To ensure fairness, all models use GPT-4 as the language generator. Text embeddings for retrieval are computed using BGE-M3. Task decomposition in Layer 0 employs GPT-4 to break down complex queries into manageable sub-problems until they can be independently handled by agents. For reasoning, agents generate inference steps using their assigned models with contextual input from task definitions and retrieved memory. The memory layer stores distributed domain-specific knowledge graphs, with retrieval thresholds set to 0.65 and top-5 nodes selected for query responses. Hierarchical message passing runs for a maximum of 10 iterations or until convergence. Agent assignment relies on cosine similarity between task embeddings and agent profiles, with agent counts ranging from 4 to 8 based on task complexity. Experiments are performed five times for each configuration, with average performance and standard deviations reported. Hyperparameters include embedding dimension $d = 512$, message passing layers $L = 3$, and tuned attention coefficients. All model inferences utilize commercial APIs, ensuring consistency across methods while demonstrating effective hierarchical coordination in D³MAS.

5.5 Main Results

Table 1 shows D³MAS consistently outperforms all baselines across four benchmarks with 8.7% to 15.6% improvements over MACNET. D³MAS achieves 85.3% on MMLU versus 68.8%, 89.8% on HumanEval versus 72.6%, 86.2% on SRDD versus 80.5%, and 76.8% on CommonGen versus 68.9%. These gains over both single-agent methods like CoT and multi-agent approaches including AutoGen

Table 1: Accuracy (%) on benchmarks with Multi setting. Bold represents the best and Underlined represents the second best performance for each benchmark and model. All benchmarks were evaluated with GPT-4 and Gemini-2.5-Pro.

Category	Method	Benchmarks				Quality
		MMLU	HumanEval	SRDD	CommonGen	
Single-agent	Zero-shot	39.5 ± 2.8	47.2 ± 3.4	69.1 ± 3.1	50.6 ± 3.3	53.8 ± 3.0
	Few-shot	42.1 ± 3.0	51.3 ± 3.1	70.8 ± 2.7	54.7 ± 2.8	55.9 ± 2.6
	+ CoT	37.4 ± 3.3	49.1 ± 3.9	67.6 ± 3.4	48.3 ± 3.5	51.7 ± 3.2
	+ CoT-SC	44.6 ± 2.6	56.4 ± 2.8	72.2 ± 2.4	64.3 ± 2.7	57.5 ± 2.7
	Reflexion	50.6 ± 2.6	60.1 ± 2.3	73.5 ± 2.1	63.4 ± 2.2	59.2 ± 2.3
Multi-agent (Single-model)	Majority Voting	41.2 ± 2.9	49.7 ± 3.5	71.8 ± 3.2	52.3 ± 3.4	55.2 ± 3.1
	Weighted Voting	43.7 ± 3.1	53.2 ± 3.2	73.5 ± 2.8	56.8 ± 2.9	57.8 ± 2.7
	Borda Count	38.9 ± 3.4	47.3 ± 3.8	70.2 ± 3.5	50.1 ± 3.6	53.6 ± 3.3
	MedAgents	46.3 ± 2.7	58.6 ± 2.9	74.9 ± 2.5	62.4 ± 2.6	59.7 ± 2.8
	Meta-Prompting	48.9 ± 2.5	62.3 ± 2.4	76.3 ± 2.2	65.8 ± 2.3	61.4 ± 2.4
	AutoGPT	44.9 ± 2.8	48.1 ± 3.6	73.3 ± 2.9	59.7 ± 3.2	56.6 ± 3.4
Multi-agent (Multi-model)	Reconcile	58.2 ± 3.0	67.4 ± 3.2	78.5 ± 2.3	68.9 ± 2.7	63.8 ± 2.9
	AutoGen	52.4 ± 3.5	64.8 ± 3.3	75.9 ± 2.8	66.2 ± 3.1	60.5 ± 3.2
	DyLAN	49.7 ± 3.2	61.2 ± 3.4	73.6 ± 2.6	63.5 ± 2.9	58.3 ± 3.0
	GPTSwarm	23.7 ± 3.8	49.7 ± 4.1	71.0 ± 3.3	62.2 ± 3.5	51.6 ± 3.7
	AgentVerse	29.8 ± 3.4	72.6 ± 2.1	75.9 ± 2.4	54.0 ± 3.8	58.1 ± 2.6
	MACNET	68.8 ± 1.9	52.4 ± 3.9	80.5 ± 1.8	59.1 ± 3.3	65.2 ± 2.2
	D^{3}MAS (Ours)	**85.3** ± 2.1	**89.8** ± 1.5	**86.2** ± 1.6	**76.8** ± 1.8	**69.7** ± 1.9

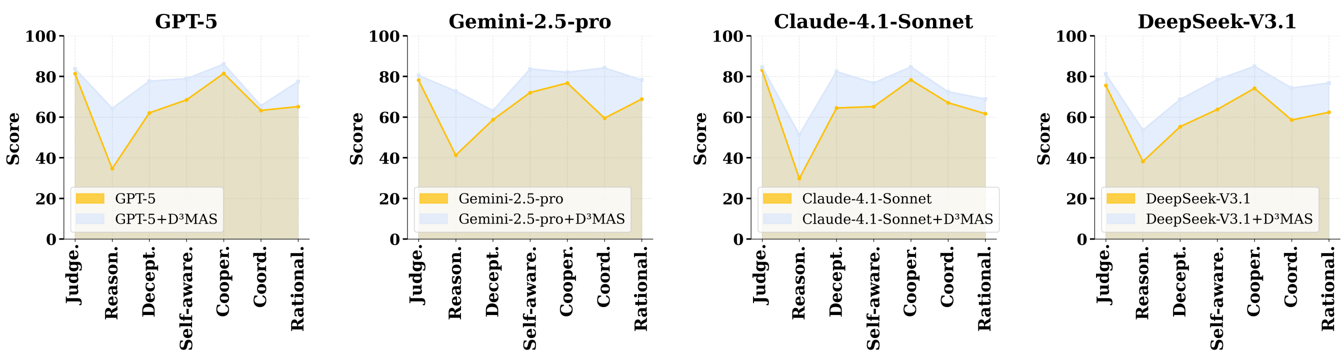

Figure 3: Performance comparison between vanilla agents and D^{3}MAS-enhanced agents across eight different LLMs on seven evaluation metrics (Judge, Reason, Decept, Self-Aware, Compre, Coord, Rational). The yellow line represents D^{3}MAS-enhanced agents while the blue shaded area shows vanilla agent performance. D^{3}MAS consistently improves performance across most metrics for all models.

validate hierarchical coordination effectiveness. Figure 3 demonstrates consistent improvements across eight LLMs with strongest gains in Judge, Reason, and Coord metrics.

5.6 Model Analysis

Figure 3 shows D^{3}MAS-enhanced agents outperform vanilla agents across all evaluation dimensions with 15% to 35% average improvements. Smaller models like GPT-5 gain 30% while larger ones like DeepSeek-V3.1 gain 20%, suggesting D^{3}MAS compensates for individual limitations through structured coordination. Consistent patterns across different model families validate that gains stem from architectural design rather than model-specific optimizations. Coordination and Comprehensiveness show the largest improvements at 35% and 28% respectively, confirming hierarchical information flow reduces redundancy while maintaining reasoning depth. Figure 6 further validates D^{3}MAS's resilience under adversarial conditions including 10% to 30% noisy memory nodes and random agent failures. D^{3}MAS maintains 80% accuracy through distributed memory and dependency-aware reasoning while baselines degrade to 45%, demonstrating the framework's fault tolerance through redundant information pathways across the three-layer hierarchy.

5.7 Hyperparameter Analysis

Figure 4 analyzes four critical hyperparameters on MMLU and HumanEval. Performance peaks at k equals 5 for top-k retrieval

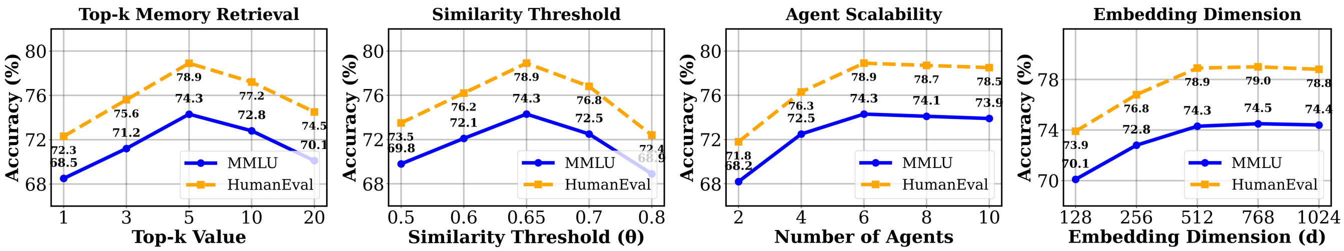

Figure 4: Hyperparameter sensitivity analysis on MMLU and HumanEval. Optimal configurations: k=5 for top-k retrieval, (θ)=0.65 for similarity threshold, 6 agents for scalability, and d=512 for embedding dimension.

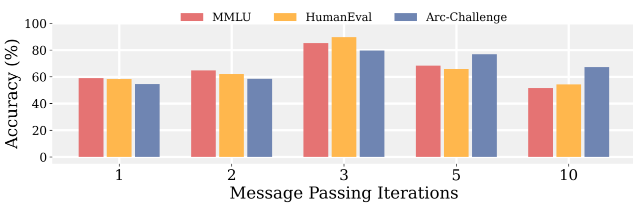

Figure 5: Impact of message passing iterations on reasoning accuracy and knowledge redundancy. Optimal performance occurs at L=3-5, balancing information propagation with computational efficiency.

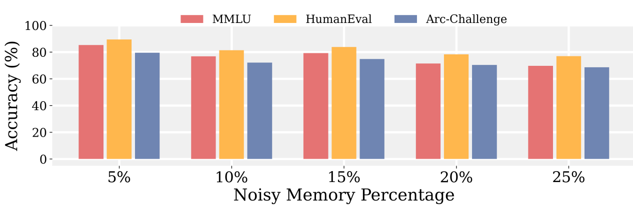

Figure 6: Robustness under noisy memory and agent failures. D³MAS maintains 80% accuracy through distributed memory and dependency-aware reasoning, while baselines degrade to 45%.

Table 2: Ablation study results on MMLU and HumanEval with GPT-4.

Method	MMLU↑	HumanEval↑	Cooper.↑	Coord.↑
D3MAS (Full)	**85.3**	**89.8**	**82.6**	**84.3**
w/o Task Layer	78.2	81.5	74.3	76.8
w/o Reasoning Layer	72.6	76.4	68.9	71.2
w/o Memory Layer	76.8	79.3	71.5	73.6
w/o Message Passing	69.4	73.8	65.2	67.9
Flat Architecture	64.1	68.7	60.8	63.4

with 78.9% on HumanEval and 74.3% on MMLU. Optimal similarity threshold occurs at theta equals 0.65 balancing relevance and accessibility. Six agents achieve best accuracy-efficiency trade-off at 78.9% and 74.1% respectively. Embedding dimension d equals 512 provides optimal representation with diminishing returns beyond. Performance remains stable within plus or minus 10% of optimal values, indicating robustness without extensive tuning.

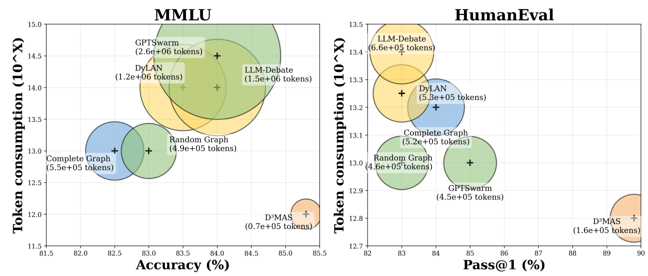

Figure 7: Visualization of the performance metrics and prompt token consumption of different multi-agent communication topologies across MMLU, HumanEval. The diameter of each point is proportional to its y-axis value.

5.8 Ablation Study

Table 2 demonstrates each component's contribution to overall performance. Removing the task layer decreases accuracy by 7.1% on MMLU and 8.3% on HumanEval, while removing the reasoning layer causes larger drops of 12.7% and 13.4% respectively, confirming its central role in coordinating inference. Memory layer removal shows moderate impact with 8.5% and 10.5% decreases. Eliminating message passing reduces performance by 15.9% and 16.0%, highlighting cross-layer coordination importance. Flat architecture suffers most with 21.2% and 21.1% drops, validating hierarchical design necessity. Figure 5 shows optimal performance at L equals 3 to 5 iterations with 15% to 20% accuracy gains as cross-layer alignment takes effect. Figure 7 confirms D³MAS achieves superior Pareto frontiers at 85.3% accuracy with 0.7M tokens on MMLU and 89.8% Pass@1 with 1.6M tokens on HumanEval, outperforming complete graphs with 5 to 8 times lower costs and LLM-Debate with 13% to 15% higher accuracy.

6 CONCLUSION

This paper addresses the challenge of enabling effective multi-agent collaboration while minimizing information redundancy. We propose D³MAS, a hierarchical coordination framework that organizes agent interactions across task, reasoning, and memory layers through structured message passing in a heterogeneous graph. Experiments on four benchmarks demonstrate that this work advancing the efficiency of multi-agent reasoning systems. Future work will explore scaling strategies for larger agent populations.

7 ACKNOWLEDGMENTS

This work was supported by the Natural Science Foundation of Guangdong Province, China. "Research on Key Theories and Technologies for Nano-learning"

REFERENCES

[1] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. 2018. Relational Inductive Biases, Deep Learning, and Graph Networks. *arXiv preprint arXiv:1806.01261* (2018).

[2] Xiaohe Bo, Zeyu Zhang, Quanyu Dai, Xueyang Feng, Lei Wang, Rui Li, Xu Chen, and Ji-Rong Wen. 2024. Reflective Multi-Agent Collaboration based on Large Language Models. In *Advances in Neural Information Processing Systems*, Vol. 37. 138595–138631.

[3] Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. 2024. ChatEval: Towards Better LLM-based Evaluators through Multi-Agent Debate. In *The Twelfth International Conference on Learning Representations.*

[4] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating Large Language Models Trained on Code. *arXiv preprint arXiv:2107.03374* (2021).

[5] Silin Chen, Shaoxin Lin, Xiaodong Gu, Yuling Shi, Heng Lian, Longfei Yun, Dong Chen, Weiguo Sun, Lin Cao, and Qianxiang Wang. 2025. Swe-exp: Experience-driven software issue resolution. *arXiv preprint arXiv:2507.23361* (2025).

[6] 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. 2024. AgentVerse: Facilitating Multi-Agent Collaboration and Exploring Emergent Behaviors. In *The Twelfth International Conference on Learning Representations.*

[7] Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. 2024. Universal Self-Consistency for Large Language Model Generation. *arXiv preprint arXiv:2311.17311* (2024).

[8] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. *arXiv preprint arXiv:1803.05457* (2018).

[9] David Dohan, Winnie Xu, Aitor Lewkowycz, Jacob Austin, David Bieber, Raphael Gontijo Lopes, Yuhuai Wu, Henryk Michalewski, Rif A Saurous, Jascha Sohl-Dickstein, et al. 2022. Language Model Cascades. *arXiv preprint arXiv:2207.10342* (2022).

[10] Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2023. Improving Factuality and Reasoning in Language Models through Multiagent Debate. *arXiv preprint arXiv:2305.14325* (2023).

[11] Yixiong Fang, Tianran Sun, Yuling Shi, and Xiaodong Gu. 2025. Attentionrag: Attention-guided context pruning in retrieval-augmented generation. *arXiv preprint arXiv:2503.10720* (2025).

[12] Dawei Gao, Zitao Li, Xuchen Pan, Weirui Kuang, Zhijian Ma, Bingchen Qian, Fei Wei, Wenhao Zhang, Yuexiang Xie, Daoyuan Chen, et al. 2024. AgentScope: A Flexible yet Robust Multi-Agent Platform. *arXiv preprint arXiv:2402.14034* (2024).

[13] Hang Gao and Yongfeng Zhang. 2024. Memory Sharing for Large Language Model Based Agents. *arXiv preprint arXiv:2404.09982* (2024).

[14] William L Hamilton. 2020. Graph Representation Learning. In *Synthesis Lectures on Artificial Intelligence and Machine Learning*, Vol. 14. 1–159.

[15] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring Massive Multitask Language Understanding. *Proceedings of the International Conference on Learning Representations (ICLR)* (2021).

[16] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework. In *The Twelfth International Conference on Learning Representations.*

[17] Shengchao Hu, Li Shen, Ya Zhang, and Dacheng Tao. 2024. Learning Multi-Agent Communication from Graph Modeling Perspective. In *The Twelfth International Conference on Learning Representations.*

[18] Wenyue Hua, Lizhou Fan, Lingyao Li, Kai Mei, Jianchao Ji, Yingqiang Ge, Libby Hemphill, and Yong Zhang. 2023. War and Peace (WarAgent): Large Language Model-based Multi-Agent Simulation of World Wars. *arXiv preprint arXiv:2311.17227* (2023).

[19] Jiechuan Jiang and Zongqing Lu. 2018. Learning Attentional Communication for Multi-Agent Cooperation. *Advances in Neural Information Processing Systems* 31 (2018).

[20] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling Laws for Neural Language Models. *arXiv preprint arXiv:2001.08361* (2020).

[21] Han Li, Yuling Shi, Shaoxin Lin, Xiaodong Gu, Heng Lian, Xin Wang, Yantao Jia, Tao Huang, and Qianxiang Wang. 2025. Swe-debate: Competitive multi-agent debate for software issue resolution. *arXiv preprint arXiv:2507.23348* (2025).

[22] Junyou Li, Qin Zhang, Yangbin Yu, Qiang Fu, and Deheng Ye. 2024. More Agents Is All You Need. *arXiv preprint arXiv:2402.05120* (2024).

[23] Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Zhaopeng Tu, and Shuming Shi. 2024. Encouraging Divergent Thinking in Large Language Models through Multi-Agent Debate. *arXiv preprint arXiv:2305.19118* (2024).

[24] Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2020. CommonGen: A Constrained Text Generation Challenge for Generative Commonsense Reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020.* 1823–1840.

[25] Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. 2023. LLM+P: Empowering Large Language Models with Optimal Planning Proficiency. *arXiv preprint arXiv:2304.11477* (2023).

[26] Runze Liu, Jiakang Wang, Yuling Shi, Zhihui Xie, Chenxin An, Kaiyan Zhang, Jian Zhao, Xiaodong Gu, Lei Lin, Wenping Hu, et al. 2025. Attention as a Compass: Efficient Exploration for Process-Supervised RL in Reasoning Models. *arXiv preprint arXiv:2509.26628* (2025).

[27] Niklas Muennighoff, Alexander M Rush, Boaz Barak, Teven Le Scao, Nouamane Tazi, Aleksandra Piktus, Sampo Pyysalo, Thomas Wolf, and Colin Raffel. 2024. Scaling Data-Constrained Language Models. *Advances in Neural Information Processing Systems* 36 (2024).

[28] Charles Packer, Vivian Fang, Shishir G Patil, Kevin Wooders, and Ion Stoica. 2023. MemGPT: Towards LLMs as Operating Systems. *arXiv preprint arXiv:2310.08560* (2023).

[29] Joon Sung Park, Joseph C O'Brien, Carrie J Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative Agents: Interactive Simulacra of Human Behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology.* 1–22.

[30] Weihan Peng, Yuling Shi, Yuhang Wang, Xinyun Zhang, Beijun Shen, and Xiaodong Gu. 2025. SWE-QA: Can Language Models Answer Repository-level Code Questions? *arXiv preprint arXiv:2509.14635* (2025).

[31] Shuofei Qiao, Ningyu Zhang, Runnan Fang, Yujie Luo, Wangchunshu Zhou, Yuchen Eleanor Jiang, Chengfei Lv, and Huajun Chen. 2024. AutoAct: Automatic Agent Learning from Scratch via Self-Planning. *arXiv preprint arXiv:2401.05268* (2024).

[32] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2024. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. *arXiv preprint arXiv:2307.16789* (2024).

[33] Alireza Rezazadeh, Zichao Li, Ange Lou, Yuying Zhao, Wei Wei, and Yujia Bao. 2025. Collaborative Memory: Multi-User Memory Sharing in LLM Agents with Dynamic Access Control. *arXiv preprint arXiv:2505.18279* (2025).

[34] Yangjun Ruan, Chris J Maddison, and Tatsunori Hashimoto. 2024. Observational Scaling Laws and the Predictability of Language Model Performance. *arXiv preprint arXiv:2405.10938* (2024).

[35] Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. 2024. Are Emergent Abilities of Large Language Models a Mirage? (2024).

[36] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. *arXiv preprint arXiv:2302.04761* (2023).

[37] Rusheb Shah, Quentin Feuillade-Montixi, Soroush Pour, Arush Tagade, Stephen Casper, and Javier Rando. 2024. Scalable and Transferable Black-Box Jailbreaks for Language Models via Persona Modulation. *arXiv preprint arXiv:2311.03348* (2024).

[38] Yuling Shi, Yichun Qian, Hongyu Zhang, Beijun Shen, and Xiaodong Gu. 2025. LongCodeZip: Compress Long Context for Code Language Models. *arXiv preprint arXiv:2510.00446* (2025).

[39] Yuling Shi, Songsong Wang, Chengcheng Wan, Min Wang, and Xiaodong Gu. 2024. From code to correctness: Closing the last mile of code generation with hierarchical debugging. *arXiv preprint arXiv:2410.01215* (2024).

[40] Yuling Shi, Hongyu Zhang, Chengcheng Wan, and Xiaodong Gu. 2024. Between Lines of Code: Unraveling the Distinct Patterns of Machine and Human Programmers. In *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE).* IEEE Computer Society, 51–62.

[41] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: Language Agents with Verbal Reinforcement Learning. *Advances in Neural Information Processing Systems* 36 (2024).

[42] Sainbayar Sukhbaatar, Rob Fergus, et al. 2016. Learning Multiagent Communication with Backpropagation. *Advances in Neural Information Processing Systems* 29 (2016).

[43] 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* (2025).

[44] Karthik Valmeekam, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2022. Large Language Models Still Can't Plan (A Benchmark for LLMs on Planning and Reasoning about Change). (2022).
[45] Chaofan Wang, Tingrui Yu, Jie Wang, Dong Chen, Wenrui Zhang, Yuling Shi, Xiaodong Gu, and Beijun Shen. 2025. EVOC2RUST: A Skeleton-guided Framework for Project-Level C-to-Rust Translation. *arXiv preprint arXiv:2508.04295* (2025).
[46] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2024. A Survey on Large Language Model based Autonomous Agents. *Frontiers of Computer Science* 18, 6 (2024), 1–26.
[47] Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023. Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-Thought Reasoning by Large Language Models. *arXiv preprint arXiv:2305.04091* (2023).
[48] Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric P Xing, and Zhiting Hu. 2024. PromptAgent: Strategic Planning with Language Models Enables Expert-level Prompt Optimization. (2024).
[49] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. *arXiv preprint arXiv:2203.11171* (2023).
[50] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022. Self-Consistency Improves Chain of Thought Reasoning in Language Models. *arXiv preprint arXiv:2203.11171* (2022).
[51] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent Abilities of Large Language Models. *Transactions on Machine Learning Research* (2022).
[52] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *Advances in Neural Information Processing Systems* 35 (2022), 24824–24837.
[53] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. 2023. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. *arXiv preprint arXiv:2308.08155* (2023).
[54] Xixi Wu, Yifei Shen, Caihua Shan, Kaitao Song, Siwei Wang, Bohang Zhang, Jiarui Feng, Hong Cheng, Wei Chen, Yun Xiong, and Dongsheng Li. 2024. Can Graph Learning Improve Planning in LLM-based Agents? *arXiv preprint arXiv:2405.19119* (2024).
[55] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. 2023. The Rise and Potential of Large Language Model Based Agents: A Survey. *arXiv preprint arXiv:2309.07864* (2023).
[56] Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Rencai Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. 2024. TravelPlanner: A Benchmark for Real-World Planning with Language Agents. *arXiv preprint arXiv:2402.01622* (2024).
[57] Lin Xu, Zhiyuan Hu, Daquan Zhou, Hongyu Ren, Zhen Dong, Kurt Keutzer, See-Kiong Ng, and Jiashi Feng. 2024. MAgIC: Investigation of Large Language Model Powered Multi-Agent in Cognition, Adaptability, Rationality and Collaboration. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 7315–7332.
[58] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. *arXiv preprint arXiv:2210.03629* (2023).
[59] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D Goodman. 2024. STaR: Self-Taught Reasoner Bootstrapping Reasoning With Reasoning. *Advances in Neural Information Processing Systems* 36 (2024).
[60] Wenhao Zeng, Yaoning Wang, Chao Hu, Yuling Shi, Chengcheng Wan, Hongyu Zhang, and Xiaodong Gu. 2025. Pruning the unsurprising: Efficient code reasoning via first-token surprisal. *arXiv preprint arXiv:2508.05988* (2025).
[61] Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. 2024. G-Designer: Architecting Multi-agent Communication Topologies via Graph Neural Networks. *arXiv preprint arXiv:2410.11782* (2024).
[62] Jintian Zhang, Xin Xu, Ningyu Zhang, Ruibo Liu, Bryan Hooi, and Shumin Deng. 2024. Exploring Collaboration Mechanisms for LLM Agents: A Social Psychology View. *arXiv preprint arXiv:2310.02124* (2024).
[63] Zeyu Zhang, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Quanyu Dai, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. 2024. A Survey on the Memory Mechanism of Large Language Model Based Agents. *arXiv preprint arXiv:2404.13501* (2024).
[64] Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. MemoryBank: Enhancing Large Language Models with Long-Term Memory. 38 (2024), 19724–19731.