Preprints.org

Article

Not peer-reviewed version

Efficiency in LLM Multi-Agent Systems: A Survey

Qi Li , Lexiang Xiong , Haiquan Lu , Wenjie Qu , Xingyi Yang , Jiaheng Zhang , Xinchao Wang *

Posted Date: 20 September 2026

Keywords: multi-agent systems; large language models; efficient deep learning

Preprints.org is a free multidisciplinary platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC, OpenAlex.

Copyright: This open access article is published under a Creative Commons CC BY 4.0 license, which permit the free download, distribution, and reuse, provided that the author and preprint are cited in any reuse.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

Disclaimer/Publisher's Note: The statements, opinions, and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions, or products referred to in the content.

Article

Efficiency in LLM Multi-Agent Systems: A Survey

Qi Li [1,†], Lexiang Xiong [1,†], Haiquan Lu [1,†], Wenjie Qu [2], Xingyi Yang [3], Jiaheng Zhang [1] and Xinchao Wang [1,*]

[1] National University of Singapore
[2] University of California, Berkeley
[3] The Hong Kong Polytechnic University
* Correspondence: xinchao@nus.edu.sg
† Equal contribution.

Abstract

Large language model (LLM)-based multi-agent systems (MAS) have moved beyond small debate ensembles toward workflows in which specialized agents operate across tools, memory, and external environments. This shift expands what a system can search and verify, but it also makes collaboration expensive: repeated inference and stateful coordination can outweigh the performance gain. The central efficiency question is therefore not simply whether multiple agents outperform a single agent under a fixed budget, but when the benefit of collaboration justifies its end-to-end resource cost. Existing work addresses this question from different aspects: some studies decide whether collaboration should be invoked, while others assume a multi-agent workflow is already in place and seek to reduce its overhead. Since these studies define baselines and costs differently, their reported efficiency gains are difficult to compare. This motivates a unified accounting view that relates each performance gain to the collaboration overhead required to obtain it. This survey organizes these efforts around a common view of the problem. We first characterize the **Collaboration Boundary**, which reviews previous works and determines when collaboration is worth invoking. We then organize existing efficiency methods by three primary control points: **Topology**, which determines the participating agents and communication structure; **Runtime**, which controls the information, models, and state activated during execution; and **Optimization**, which learns reusable prompts, workflows, and policies through search or training. In **Evaluation**, we further examine how these methods should be evaluated through matched baselines, explicit resource accounting, and lifecycle cost. We further provide a **Synthesis** that compares the resulting evidence, clarifies where reported savings are well supported, and explains why some efficiency gains may not transfer across settings. We conclude with **Open Problems** that identify the remaining gaps in- efficient and scalable MAS design. Across the literature, a consistent picture emerges. MAS efficiency is a quality–resource trade-off, not a single measure of agent count, tokens, or cost. The best design depends on the task, the system bottleneck, and the stage at which resources are spent.

Keywords: multi-agent systems; large language models; efficient deep learning

---

Reading guide. Figure 1 summarizes the organization of this survey. Readers primarily seeking an overview of efficient MAS methods may proceed from Sections 1 and 2 directly to Section 5, whereas those interested in systematic MAS evaluation and comparable evidence across methods are encouraged to read the survey in full.

We will continue to update this survey and its accompanying GitHub repository, and we welcome contributions from the community. If you are aware of relevant work that is not yet included, please contact us via GitHub or at any of the following email addresses: liqi@u.nus.edu · lexiang@u.nus.edu · haiquanlu@u.nus.edu. The experimental results presented in this survey are mainly compiled from those reported in the cited papers for comparative analysis. We welcome corrections or update requests from the original authors.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

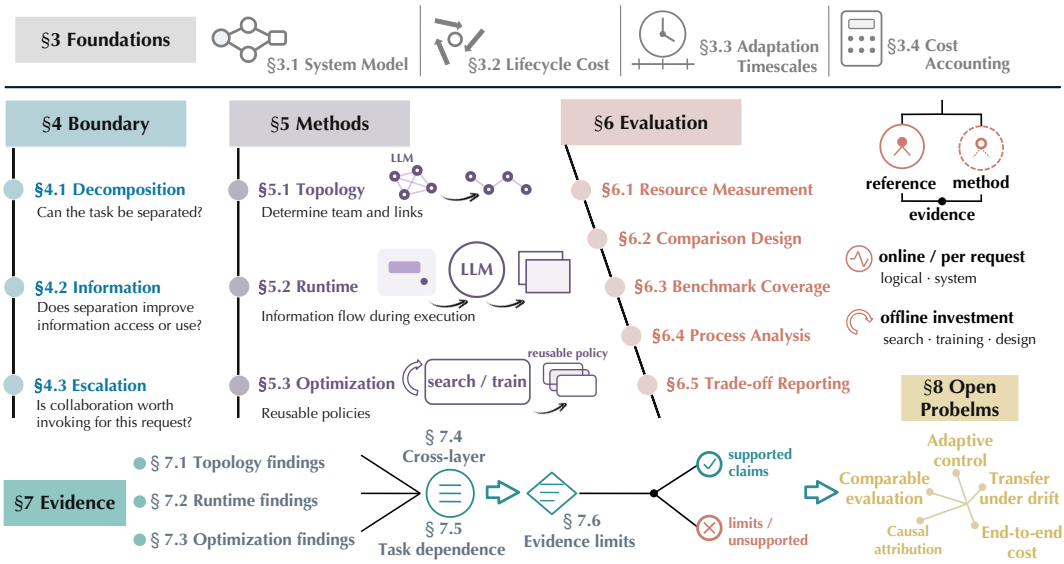


Figure 1. Survey structure. Foundations reviews the common system and resource concepts, and the Boundary identifies the relevant comparison. Existing efficiency methods are then grouped by their primary control point: Topology, Runtime, or Optimization. Evaluation and Synthesis organize the available evidence, from which the Open Problems are derived.

1. Introduction

Large language models have evolved from passive text generators into agents that pursue goals through repeated interaction with external environments. Foundational work developed iterative agent loops and extended them to persistent skills and tool use [93,94,117,138,142,157,161,175,210]. Building on this progress, large language model (LLM)-based multi-agent systems (MAS) have expanded from small debate ensembles into larger workflows that coordinate specialized models, tools, memory, and environmental interaction. They now appear in deliberative reasoning, long-context processing, software and tool workflows, and embodied cooperation [33,156,230,238,240]. As these systems grow in scope, organization becomes a first-class systems problem: the challenge is not only to obtain the benefits of collaboration, but also to avoid making every request pay for every agent, message, model, and state that a team can activate.

Understanding this organizational problem requires looking beyond agent count. The value of a team depends on which participants, information paths, and capabilities are activated for a given task. These choices enable several forms of factorization. Multiple agents can explore different parts of a search space [91], preserve diverse hypotheses [18,176], and cross-check one another [33]. Specialized participants can use different models, instructions, tools, memories, or information sources [176,224,230]. Distributed workers can partition long inputs [240] or execute independent actions in parallel [119,230], while modular workflows can support local verification, selective retry, and targeted recovery [66,204]. In embodied or distributed settings, separate agents may possess local observations or physically distinct action capabilities [119,230]. Empirical studies have shown that interaction, debate, and aggregation can improve model outputs [18,33] and evaluation reliability [13], and that the resulting quality depends strongly on how candidate solutions are generated and combined [79,91,176].

The same mechanisms that create this value also introduce substantial overhead. Broader search requires additional model calls, while specialization can duplicate instructions and shared state. Communication expands model contexts, and coordination adds orchestration overhead. Parallel execution may shorten the critical path at the cost of greater total work and peak demand. Persistent memory can reduce repeated reasoning but consumes resources to build, access, and maintain. Likewise, offline optimization becomes beneficial only when its search or training cost is amortized over subsequent use. Savings also do not necessarily transfer across system layers: reducing message tokens may leave cache behavior unchanged, while cache sharing or model routing can improve one resource dimension

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

without reducing logical work. MAS efficiency is therefore multidimensional and lifecycle-dependent rather than a synonym for fewer agents or fewer visible tokens.

These observations show that capability alone is insufficient for assessing multi-agent systems, bringing efficiency into focus. Besides, existing surveys have not treated efficiency as a central organizing problem, while relevant studies remain scattered across several lines of research. A sufficiently comprehensive and clearly delineated taxonomy is therefore essential. Such a taxonomy must do more than catalogue existing methods: it should identify the system object each method modifies, the timescale at which it operates, the resource dimensions it targets, and whether its apparent savings are shifted to another stage. Constructing such a taxonomy is challenging because similar mechanisms are often described using different terminology, while a single method may span communication, routing, scheduling, and learning. Moreover, logical organization, model execution, and system serving are tightly coupled, so a local reduction in one resource does not necessarily translate into an end-to-end efficiency improvement.

Accordingly, this survey organizes existing methods based on *where they primarily intervene*. After defining the collaboration boundary, we distinguish three method families: *Topology*, which modifies team composition and communication structure; *Runtime*, which governs the information, models, and system state activated during execution; and *Optimization*, which learns reusable configurations and policies through search or training. This taxonomy separates methods according to what they principally change, while retaining the interactions among system structure, execution, and learning.

The survey question.

These considerations lead to the central question of this survey: *when does multi-agent collaboration justify its resource cost, and how can it be made more efficient?*

Positioning.

Prior surveys have catalogued LLM-based multi-agent architectures and collaboration mechanisms [46,50,170], benchmarked agent capability [111,253], or reviewed focused subsets such as agentic RAG [159], agent memory [194], and agent interoperability [34]. This survey differs in three respects: (i) it places efficiency at the center and considers collaboration across its full lifecycle; (ii) it classifies methods by their primary control point, providing a consistent basis for organizing future systems; and (iii) it incorporates matched baselines, resource accounting, and offline-cost amortization into the interpretation of reported gains.

Organization.

The remainder of the paper follows this structure. Section 2 provides background on LLM-based agents and multi-agent systems. Section 3 establishes the common system and resource foundations, and Section 4 examines the collaboration boundary. Sections 5.1–5.3 then review topology, runtime, and optimization methods, respectively. Section 6 presents the evaluation framework, Section 7 consolidates the empirical evidence and deployment implications, and Section 8 develops the open research agenda. Empirical results are discussed alongside the mechanisms they support, so that their resource definitions and evidence boundaries remain explicit.

2. Background

LLM-based multi-agent systems build on two lines of research: LLM-based agents and traditional multi-agent systems. The former place a language model within an iterative decision loop, while the latter provide formalisms for interaction among multiple autonomous decision makers. This section introduces both foundations and explains how they converge in contemporary LLM-MAS, providing the technical context for the resource-aware system model developed in the next section.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

2.1. LLM-Based Agents

An LLM-based agent places a language model within an iterative decision process [210]. Unlike a standalone completion, it maintains state and uses environmental feedback to guide subsequent actions, often through external tools.

Let h_t denote the interaction history available at step t, M_t the agent's persistent memory, and $\mathcal{U}$ its tool interface. A generic agent loop can be written as

$$a_t \sim \varphi_\theta(\cdot \mid h_t, M_t, \mathcal{U}), \qquad o_{t+1} \sim \mathcal{E}(\cdot \mid a_t), \tag{1}$$

followed by

$$(h_{t+1}, M_{t+1}) = U(h_t, M_t, a_t, o_{t+1}), \tag{2}$$

where φ_θ is the LLM-conditioned action policy, $\mathcal{E}$ is the environment, and U updates the working context and memory. Planning, reflection, retrieval, and tool selection can be implemented inside φ_θ or delegated to auxiliary modules.

Early work established the main components of this loop. ReAct coupled language reasoning with environment actions and observations [210]. Reflexion showed how feedback from previous attempts could be stored and used to guide later decisions [157]. Tool-oriented systems such as ToolLLM expanded the action space from text generation to large collections of APIs [142], while Voyager combined iterative prompting, environment feedback, and a reusable skill library in a long-horizon embodied setting [175].

Together, these systems show that a single agent can already complete multi-step tasks through external interaction and iterative refinement. Multi-agent systems extend this setting by coordinating several such processes, creating additional capabilities as well as overhead.

2.2. Traditional Multi-Agent Systems

A traditional MAS contains multiple autonomous agents whose decisions affect one another through a shared environment or an explicit communication mechanism. Each agent acts from local information, while the system outcome depends on their joint behavior. The agents may share a goal, pursue different rewards, or operate under mixed incentives.

Symbolic MAS described agents through explicit internal states and rule-based interaction. The belief–desire–intention model organized decision making around beliefs, goals, and intentions [143]. Coordination was handled through predefined protocols; the Contract Net Protocol, for example, distributed tasks through structured bidding [160]. These designs were interpretable but depended on fixed action spaces and message semantics.

Learning-based MAS replaced part of this symbolic machinery with policies learned from interaction. A common representation is a partially observable stochastic game

$$\mathcal{M}_{\text{MAS}} = \langle \mathcal{I}, \mathcal{S}, \{\mathcal{A}_i\}_{i\in\mathcal{I}}, P, \{\Omega_i\}_{i\in\mathcal{I}}, O, \{R_i\}_{i\in\mathcal{I}}, \gamma \rangle, \tag{3}$$

where $\mathcal{I}$ is the agent set, $\mathcal{S}$ the environment state, $\mathcal{A}_i$ and Ω_i the action and observation spaces of agent i, P the transition function, O the observation model, R_i the reward, and γ the discount factor. At step t, each agent selects

$$a_t^i \sim \varphi_i(\cdot \mid z_t^i), \qquad s_{t+1} \sim P(\cdot \mid s_t, \mathbf{a}_t), \quad \text{where} \quad \mathbf{a}_t = (a_t^1, \ldots, a_t^{|\mathcal{I}|}), \tag{4}$$

using only its local action–observation history z_t^i.

Markov games provided an early framework for learning under interacting orcompeting policies [105]. In cooperative settings,the Dec-POMDP formalism captured agents that share a reward but act fromdifferent observations, while also exposing the computational difficulty ofdecentralized planning [7]. Later methods developedcentralized-training-with-decentralized-execution: MADDPG

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

usedcentralized critics for decentralized policies[115], and QMIX factorized a joint action-valuefunction into locally executable agent values [144].

Traditional MAS therefore established the foundations for decision making under limited information and interdependent behavior. Its protocols and action spaces, however, were usually tailored to a specific environment. LLM-based agents provide a more general interface in which tasks and coordination can be expressed through language.

2.3. LLM-Based Multi-Agent Systems

An LLM-based multi-agent system consists of multiple LLM-driven agents whose decisions are coupled through communication, a shared environment, or an explicit aggregation mechanism. The agents may use the same underlying model; distinct model parameters are not required. Multiple calls constitute separate agents when they maintain different roles, contexts, tools, or memories and participate in an agent-level coordination process. Parallel inference without such organization is not, by itself, an MAS.

At step t, let V_t denote the active agents and $E_t \subseteq V_t \times V_t$ the directed communication links. Their step-level communication graph is

$$G_t^{\text{comm}} = (V_t, E_t). \tag{5}$$

An edge $(j, i) \in E_t$ allows agent j to send a message $\xi_t^{j \to i}$ to agent i. The messages received by agent i form

$$\mathcal{C}_t^i = \left\{ \xi_t^{j \to i} : (j, i) \in E_t \right\}. \tag{6}$$

The single-agent policy introduced in Eq. 4 is therefore extended to

$$a_t^i \sim \varphi_{\theta_i}\left(\cdot \mid h_t^i, \mathcal{C}_t^i, \mathcal{U}_i \right), \tag{7}$$

where h_t^i is the local interaction history and $\mathcal{U}_i$ is the available tool interface. The communication graph determines which agents can exchange information, while $\mathcal{C}_t^i$ records the information actually received by agent i. Both may change during execution.

This step-level description is summarized over a complete query by the execution object $\mathcal{G}(x)$ defined in Section 3 below. Its components $V(x)$, $E(x)$, $m_i(x)$, and $h_i(x)$ record the realized agents, communication links, model assignments, and local contexts, respectively.

Several early systems established the main forms of LLM-based collaboration. CAMEL used role-conditioned dialogue to study cooperation between communicative agents [90]. Multi-agent debate introduced iterative proposal, critique, and revision as a reasoning mechanism [33]. Generative Agents combined memory, reflection, planning, and natural-language interaction in a simulated society [135].

Subsequent frameworks turned these interactions into reusable workflows. ChatDev organized software development around communicating professional roles [140], while MetaGPT used structured intermediate artifacts and standard operating procedures to coordinate specialized agents [61]. AutoGen provided a general programming framework in which agents, tools, and humans interact through configurable conversations [193]. Together, these systems established debate, role specialization, staged workflows, society simulation, and programmable conversation as recurring forms of LLM-based multi-agent organization.

LLM-MAS also alter how resources are consumed. Messages produced by one model often become inputs to another, increasing both inference work and context length. Role specialization and orchestration introduce further overhead. Design-time optimization may reduce deployment cost, but only by shifting part of the expense offline. Efficiency must therefore be assessed over the complete execution rather than inferred from agent count or task accuracy alone. The next section formalizes this execution and its associated resource dimensions.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

3. Foundations

3.1. System Model

For a query x, an MAS instantiates a set of agents V, communication links E, model assignments m_i, local contexts h_i, and a stopping and aggregation policy π. We represent its execution as

$$\mathcal{G}(x) = \big(V(x), E(x), \{m_i(x)\}, \{h_i(x)\}, \pi(x)\big). \tag{8}$$

This representation covers repeated calls to one shared model, heterogeneous model pools, and agents with private tools or memory. Parallel hardware execution alone is not an MAS unless the system also makes agent-level decomposition, communication, or aggregation decisions.

Existing efficiency objectives can be expressed as maximizing task utility Q under a vector of realized resources:

$$\boldsymbol{c} = \left(\begin{array}{cccc} \underbrace{T_{\text{prompt}}}_{\text{input tokens}}, & \underbrace{T_{\text{reason}}}_{\text{reasoning tokens}}, & \underbrace{T_{\text{out}}}_{\text{output tokens}}, & \underbrace{N_{\text{call}}}_{\text{model calls}} \\ \underbrace{L_{\text{wall}}}_{\text{wall-clock latency}}, & \underbrace{P_{\text{peak}}}_{\text{peak concurrency}}, & \underbrace{M_{\text{KV}}}_{\text{KV-cache memory}} & \\ \underbrace{B_{\text{network}}}_{\text{network traffic}}, & \underbrace{C_{\$}}_{\text{monetary cost}}, & \underbrace{E_{\text{joule}}}_{\text{energy}}, & \underbrace{C_{\text{design}}}_{\substack{\text{search and} \\ \text{training cost}}} \end{array} \right). \tag{9}$$

These coordinates are not interchangeable. Parallel calls, for example, may reduce L_{wall} while increasing generated tokens and P_{peak}.

Let $\mathcal{K}$ index the resource coordinates constrained by a study, with c_k denoting a selected coordinate of $\boldsymbol{c}$ and B_k its budget. The objectives studied in the literature can then be summarized by

$$\max_{\mathcal{G}} \mathbb{E}[Q(\mathcal{G}, x)] \quad \text{s.t.} \quad \mathbb{E}[c_k(\mathcal{G}, x)] \le B_k, \qquad k \in \mathcal{K}. \tag{10}$$

A method changes one or more components of $\mathcal{G}$ while controlling only part of $\boldsymbol{c}$. Uncontrolled coordinates should remain visible rather than being absorbed into a single efficiency ratio.

3.2. Lifecycle Cost

Topology construction, prompt search, and policy learning may incur substantial cost before deployment. The coordinate C_{design} represents this offline expenditure. For amortization, let $C^{(u)}_{\text{design}}$ denote its value in a declared accounting unit u, and let $C^{(u)}_{\text{run}}$ denote the expected per-query cost in the same unit.[1]

After n requests, the total and average lifecycle costs are

$$C^{(u)}_{\text{total}}(n) = C^{(u)}_{\text{design}} + nC^{(u)}_{\text{run}}, \qquad \bar{C}^{(u)}(n) = \frac{C^{(u)}_{\text{design}}}{n} + C^{(u)}_{\text{run}}. \tag{11}$$

Let $C^{(u)}_0$ be the per-query cost of a baseline under the same accounting unit. If its additional design cost is zero or matched, the nominal break-even workload is

$$n^\star = \frac{C^{(u)}_{\text{design}}}{C^{(u)}_0 - C^{(u)}_{\text{run}}}, \qquad C^{(u)}_{\text{run}} < C^{(u)}_0. \tag{12}$$

[1] We use design/run rather than train/test because MAS construction may incur setup cost without updating model weights. For monetary accounting, $C^{(\$)}_{\text{run}} = \mathbb{E}[C_{\$}]$. A normalized-compute measure may combine token usage and model calls using declared model-specific weights. Latency can be analyzed analogously only when setup and per-query expenditure are both expressed in units of time.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

For example, a $2,000 search that reduces per-query cost from $0.020 to $0.014 breaks even after approximately 3.3×10^5 requests. Such reporting remains uncommon; TopoPrior (Section 5.1) is a notable exception.

3.3. Adaptation Timing

The term "dynamic" covers decisions made at different times. We distinguish:

- **Offline configuration**: a workflow or policy is learned once and reused across requests;
- **Input-conditioned adaptation**: the team, graph, context, or model assignment is selected for the incoming request;
- **State-conditioned adaptation**: these decisions are revised from intermediate messages, tool outputs, or failures.

Timing is distinct from the object being adapted. Team formation selects participants, hierarchical delegation creates scoped subagents, and workflow adaptation changes task dependencies or execution order. Each may operate at any of the three timescales. Their overhead should be assigned to the same timescale during evaluation.

3.4. Cost Accounting

For a query x, let τ denote the sequence of model and tool events generated during execution. Resource use has two complementary views: *logical work* requested by the agent graph and its *system realization* under a serving configuration. They describe the same execution and should not be added as independent costs.

Let model call j use model m_j and consume p_j prompt tokens, r_j reasoning tokens, and o_j visible output tokens. The trace recovers the token coordinates in Equation 9:

$$T_{\text{prompt}} = \sum_{j=1}^{N_{\text{call}}} p_j, \qquad T_{\text{reason}} = \sum_{j=1}^{N_{\text{call}}} r_j, \qquad T_{\text{out}} = \sum_{j=1}^{N_{\text{call}}} o_j. \tag{13}$$

For a declared accounting unit u, token-derived logical cost is

$$C^{(u)}_{\text{logical}}(\tau) = \sum_{j=1}^{N_{\text{call}}} \left[\alpha^{(u)}_{m_j} p_j + \beta^{(u)}_{m_j} r_j + \gamma^{(u)}_{m_j} o_j \right] + C^{(u)}_{\text{tool}}(\tau), \tag{14}$$

where the coefficients represent monetary prices or declared compute weights. For these accounting units,

$$C^{(u)}_{\text{run}} = \mathbb{E}_\tau \left[C^{(u)}_{\text{logical}}(\tau) \right]. \tag{15}$$

All terms in Equations 14–15 must use the same unit. Tool use that cannot be converted without an arbitrary assumption should be reported separately. Cached input should also be distinguished when it is billed differently, and unavailable reasoning-token counts should be marked as unobserved.

The system-level coordinates L_{wall}, P_{peak}, M_{KV}, B_{network}, and E_{joule} must be measured from the serving realization. They cannot be inferred directly from $C^{(u)}_{\text{logical}}(\tau)$.

A received message that appears in a later prompt is already included in p_j and should not be added again as a separate token cost. Message tokens or bytes remain useful structural measurements because they reveal the source of context growth, transmission, and storage overhead.

Component attribution.

Each event in τ should be assigned to the component and phase that produced it. Routing, retrieval, runtime verification, aggregation, and recovery belong to deployment cost when they are invoked before the final output; post-hoc evaluators should be reported separately. Throughout this survey, a comparison is called *matched* when it controls model family, tool access, retrieval sources,

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

verification signals, and, where possible, the call budget. Otherwise, an observed gain may reflect additional capabilities rather than multi-agent organization itself.

Table 1 connects the method taxonomy used later in the survey to the most common resource targets. Its five resource columns group related coordinates from Equation 9; they do not replace the full vector.

Table 1. Efficient MAS method families and their typical resource effects.

Subfamily	*Tokens*	*Calls*	*Cost*	*Latency*	*Memory*	Decision	Added overhead	Literature
(T) Topology: participants and communication links								
Pruning	✓	○	○	○	○	remove nodes or edges	selector or search	▮▮▯▯
Construction	○	○	○	○	—	set initial team and links	design or generation	▮▮▯▯
Adaptation	○	○	○	○	—	update graph from prior or feedback	prior learning or monitoring	▮▮▯▯
(R) Runtime: information flow for the current request								
Communication	✓	○	○	○	○	filter or compress exchange	selection or encoding	▮▮▮▯
Routing	○	✓	✓	✓	○	choose mode, role, context, model	controller and errors	▮▮▯▯
State	—	—	○	✓	✓	share memory or KV state	transfer and coherence	▮▮▮▯
Scheduling	—	○	○	✓	○	parallelize, stop, or recover	peak load and monitoring	▮▮▮▯
(O) Optimization: reusable behavior and execution programs								
Prompt	○	—	○	○	—	revise coupled prompts	candidate evaluation	▮▯▯▯
Workflow	○	○	○	○	—	search executable workflow	search and evaluation	▮▮▯▯
Policy	○	✓	✓	○	—	learn sequential control	rollouts and judges	▮▮▯▯
Continual	○	○	○	○	○	update reusable assets	storage and updates	▮▯▯▯

✓≙ primary controlled effect; ○≙ secondary or workload-dependent effect; —≙ not a defining target. Literature density: ▮▯▯▯ 1–6; ▮▮▯▯ 7–10; ▮▮▮▯ 11–15; ▮▮▮▮ 16–30; ▮▮▮▮ 31+.

4. Collaboration Boundary

A fundamental question for multi-agent systems is when collaboration actually provides an advantage. This cannot be assessed against a single universal baseline. When a task can be performed by one controller, a multi-agent system should be compared with a capable single-agent system under comparable model, tool, information, and test-time access. For tasks that inherently require multiple actors or distributed observations, the appropriate baseline is instead a simpler or fixed multi-agent workflow. Thus, the value of collaboration should be judged relative to the strongest relevant alternative, rather than inferred from the number of agents alone.

Prior studies can be organized around three successive questions. This organization is our synthesis rather than a framework taken from any single work: Decomposition draws on decomposition and scaling analyses [81,82], Information on information-bottleneck and task-structure analyses [133,217], and Escalation on hybrid execution and learned collaboration gating [42,219]. *Decomposition* examines whether the work can be separated without excessive dependencies. *Information* asks whether that separation changes the evidence available to the system or helps it use the evidence more effectively. *Escalation* considers whether the expected benefit is large enough to activate additional collaboration for the current request:

$$\underbrace{\text{separable?}}_{\text{Decomposition}} \Longrightarrow \underbrace{\text{informative?}}_{\text{Information}} \Longrightarrow \underbrace{\text{worth invoking?}}_{\text{Escalation}}$$

4.1. Decomposition

Decomposition is useful when subtasks have limited dependencies, compact interfaces, and outputs that can be checked or integrated reliably. Under these conditions, separate workers can explore modules in parallel, retry a failed component, or apply a specialized verifier without repeating the complete workflow. Strong coupling has the opposite effect: agents must repeatedly reconstruct shared state, and local decisions may become invalid after integration.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

Existing evaluations support this conditional view rather than a general benefit from adding agents. A broad study of agent-system scaling finds that the effect of collaboration varies with task decomposability, model capability, tool use, and verification structure [82]. These variables determine whether local work remains useful after the communication and integration needed to combine it.

Decomposition alone is not evidence for a multi-agent advantage. Decomposed Prompting shows that one controller can invoke modular subroutines sequentially without agent-to-agent communication [81]. A study claiming value from collaboration should therefore compare against a single-controller decomposition when that alternative is feasible. Conversely, a pruning or scheduling method may be evaluated against the original multi-agent workflow when the task already requires multiple participants. In both cases, the comparison should include planning, communication, integration, and failed local attempts.

4.2. Information

Separating computation does not necessarily create new information. A controlled study of multi-hop question answering considers a setting in which all agent messages are derived from the same complete context available to a single agent [169]. In this setting, communication may reorganize or compress the context, but it cannot add evidence that was absent from it. Under matched requested thinking-token budgets, the reported strong single-agent configurations generally match or outperform the tested multi-agent designs. The result applies to shared information and effective full-context use; it does not establish universal single-agent dominance or match every realized resource.

Bounded contexts make the comparison less direct. An information-bottleneck analysis shows that collaboration depends on how much task-relevant information survives the relay between agents [217]. Collaboration can help when a relay preserves nearly sufficient evidence while reducing the context each participant must process. Its advantage diminishes when compression removes decisive information or a stronger model can already use the full context effectively.

Long-context and retrieval systems illustrate the two sides of this boundary. Chain of Agents partitions a long source among sequential workers and passes compact intermediate summaries to a manager [240]. MAIN-RAG assigns agents to filter retrieved documents before final generation [14]. Such systems may improve source coverage or reduce the irrelevant context seen by the final model, but their efficiency depends on the worker, filtering, relay, and aggregation calls required to obtain that context.

The relevant test is therefore whether collaboration acquires, preserves, or exposes useful evidence more effectively under the reported budget. Comparisons should match source and tool access whenever possible. When agents possess genuinely private observations or distinct tools, those inputs should be stated explicitly rather than attributed to multi-agent organization itself.

4.3. Escalation

Even when collaboration can help, it need not be activated for every request. A hybrid SAS–MAS study evaluates a router that selects the execution mode directly and a cascade that invokes a multi-agent workflow only after a lower-cost verifier rejects the single-agent result [42]. Across the reported applications, the cascade matches or improves task performance while using fewer normalized tokens than always invoking the multi-agent workflow. The saving is application-dependent: it requires the avoided collaboration cost to exceed the initial attempt and gate overhead.

Routing errors are part of this trade-off. False acceptance preserves a weak single-agent answer, whereas unnecessary escalation removes the intended saving. A deterministic verifier can make this decision inexpensive; an LLM judge or human check may substantially change the operating point. The gate must therefore be evaluated as part of the deployed system rather than treated as a free preprocessing step.

Related methods move the decision inside the multi-agent workflow. MasRouter selects collaboration mode, roles, and model assignments [219], whereas AnyMAC predicts the next role and the context visible to it after execution has begun [181]. These methods are reviewed as runtime routing mechanisms in Section 5.2; here they show how the collaboration boundary can be implemented at request or step level.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

Evidence for conditional collaboration should compare the learned policy with always-single-agent and always-multi-agent execution, as well as an oracle gate when labels permit. Gate calls, decision errors, and downstream retries should be reported with the resources they save. The resulting boundary is a workload-dependent operating policy, not a universal preference for one architecture.

5. Efficiency Methods

Prior work improves MAS efficiency at three main control points. *Topology* shapes the collaboration structure, while *Runtime* limits the work activated for each request. *Optimization* improves persistent agent behavior or compound workflows through search or learning, shifting part of the cost to development time. We classify methods by the primary decision they affect, rather than by when that decision is executed. Methods that alter the persistent composition or connectivity of a team are classified as *Topology*. Those that determine the information, model, state, or schedule used for an individual request are classified as *Runtime*. Methods whose main output is a reusable configuration, such as a prompt, workflow, policy, or experience store, are classified as *Optimization*. Methods that substantively affect multiple classes are discussed in each relevant section. This classification is independent of when a mechanism is constructed: a topology may be learned offline or formed online, whereas a learned router remains a runtime method because it determines the work performed for each request. Tables 1 and 2 summarize the taxonomy and representative systems.

Table 2. Evidence coverage of representative efficient MAS methods. The **Update** column indicates when the primary controlled object is selected or revised. Each method is listed once under the family that owns this object. ★ = explicitly quantified in the cited evaluation; ☆ = not established in the evidence summarized here. *Q*: paired outcome; *R*: measured deployment use; *O*: offline cost included; *X*: cross-setting evaluation.

Subfamily	Method	Update	Reported effect	Q	R	O	X	Main boundary
T Topology								
Pruning	AgentPrune [225]	task	prompt tokens, dollars	★	★	☆	★	offline graph search
Pruning	AgentDropout [188]	task/round	prompt and output tokens	★	★	☆	★	not query-conditioned
Construction	G-Designer [226]	offline + query	prompt tokens	★	★	☆	★	fixed team and anchor
Construction	GTD [72]	offline + query	communication	★	★	☆	★	diffusion and proxy cost
Construction	HiVA [166]	evolution + query	accuracy and dollar efficiency	★	★	☆	★	semantics and topology co-evolve
Construction	MACNET [141]	design	quality across agent scale	★	☆	☆	★	scale and topology co-vary
Adaptation	TopoPrior [237]	prior + query	online tokens	★	★	★	★	reported break-even
Adaptation	MasFACT [184]	continual + query	topology transfer	★	☆	☆	★	drift across task streams
Adaptation	TacoMAS [198]	runtime	quality and calls	★	★	☆	★	additional test-time calls
Adaptation	MetaGen [187]	continual + runtime	roles and topology	★	☆	☆	★	roles and topology co-adapt
R Runtime								
Communication	S2-MAD [222]	runtime	message tokens	★	★	☆	★	evidence retention
Communication	DebateOCR [192]	runtime	input tokens, latency	★	★	☆	★	multimodal channel
Communication	EcoLANG [123]	design + runtime	response and total tokens	★	★	☆	★	induction cost excluded
Routing	CASTER [110]	step	dollars	★	★	☆	★	quality tolerance varies
Routing	RCR-Router [107]	step	exposed context	★	★	☆	★	memory construction cost
Routing	Smurfs [17]	runtime	tokens and pass rate	★	★	☆	★	tool retries remain
State	TokenDance [10]	serving	KV memory, concurrency	☆	★	☆	★	workload and hardware specific
State	KVCOMM [213]	runtime	KV traffic, prefill	★	★	☆	☆	compatibility assumptions
Scheduling	Act-or-Defer [178]	runtime	calls and risk	★	★	☆	☆	calibration required
Scheduling	AgentRadio [146]	runtime/step	task quality and API cost	★	★	☆	★	wall time not reported
O Optimization								
Prompt	MAPRO [243]	design	task quality	★	☆	☆	★	search cost excluded
Prompt	MAPGD [51]	design	search tokens, calls, and F1	★	★	★	★	wall time increases
Workflow	MASS [248]	design	reusable workflow	★	☆	☆	★	coupled search stages
Workflow	AFlow [233]	design	executable workflow	★	☆	☆	★	search amortization
Workflow	MetaAgent [242]	task design	design and deployment tokens	★	★	★	★	human-design cost unmeasured
Policy	OPTIMA [21]	training	message tokens	★	★	☆	★	quality–length trade-off
Policy	CORL [76]	training	calls and dollars	★	★	☆	★	rollout cost
Policy	LEMON [22]	training	orchestration	★	☆	☆	★	counterfactual cost
Continual	MetaTeam [53]	continual	reusable team updates	★	☆	☆	★	evolution cost not isolated

The following subsections share the same structure within each class. We first identify the controlled object, then review the main method families, and finally discuss their resource implications and limitations.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

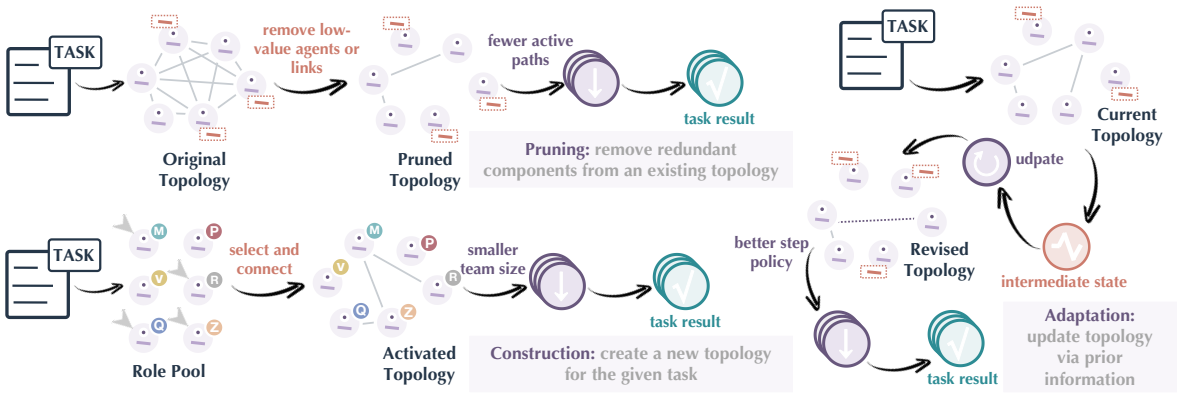


Figure 2. Topology methods for efficient LLM-based MAS. Pruning simplifies an existing collaboration graph, construction creates a task-specific topology, and adaptation updates the topology during execution in response to intermediate feedback.

5.1. Topology

Topology methods change the active team or the communication links among agents. Early multi-agent debate allows multiple model instances to exchange and revise their responses over several rounds [33]. Subsequent work characterizes its dense form as an all-to-all communication graph [100]. Under this setting, if each of N agents sends an L-token message to every other agent in each of R rounds, the delivered message volume is

$$RN(N-1)L = O(RN^2L). \tag{16}$$

This quantity measures replicated communication or prompt input rather than unique generated tokens.

Prior work provides two distinct foundations for topology design. Learned communication in cooperative multi-agent reinforcement learning (MARL) and analyses of relational priors provide task-dependent structural biases [63,153]. By contrast, interoperability protocols standardize how independently implemented agents discover and address one another [15,34]. Such protocols establish feasible connectivity, but do not show that a particular connection is necessary or efficient.

5.1.1. Topology Pruning

Early MAS experiments show that fully connected debate is often unnecessary [100]. AgentPrune represents interaction as a spatial–temporal message-passing graph and performs one-shot edge pruning [225]. Across six benchmarks, it reports 28.1–72.8% prompt-token reductions when integrated with existing systems; an MMLU comparison reports approximately $5.6 rather than $43.7 at comparable performance. The result demonstrates that dense communication can contain substantial redundancy, while remaining specific to the evaluated systems and training procedure.

AgentDropout extends pruning to nodes and round-specific intra- and inter-round edges [188]. Its learned schedule is fixed at the task level after offline training; it is round-conditioned but not conditioned on the content of each test request. Across different benchmarks including MMLU, GSM8K, HumanEval, etc., the comparison with AgentPrune reports mean reductions of 21.6% in prompt tokens and 18.4% in completion tokens, together with a 1.14% average score increase. Figure 3 reports the per-backbone results.

Other pruning methods change the selection signal. Adaptive Graph Pruning performs query-conditioned node and edge selection [89]. SafeSieve combines semantic initialization, accumulated success feedback, and cluster-aware progressive pruning [234]. It reports 12.4–27.8% token reductions, a 1.23% average performances drop under its malicious agent and prompt-injection conditions, and a 13.3% cost reduction in a heterogeneous deployment. These results support experience-driven pruning under the reported verifier and attack settings; they do not establish a general security guarantee.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

Structural sparsity alone does not determine information value. Multiple edges may transmit redundant evidence, while a single short message may change the final decision. Information-propagation analysis finds that moderately sparse topologies can suppress erroneous messages while preserving useful diffusion [154]. Complementary process metrics measure information diversity and unnecessary reasoning paths [87]. These results suggest that pruning should preserve useful information flow rather than minimize edge count alone.

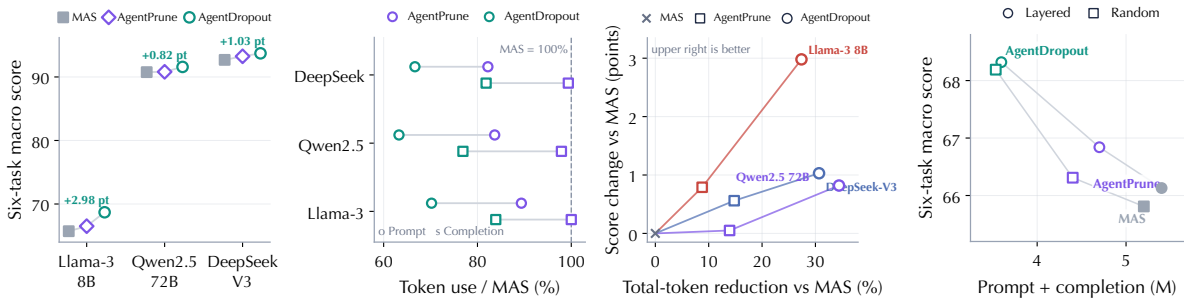


Figure 3. AgentDropout results against unpruned and pruned MAS baselines [188]. **A:** Six-task macro scores for the unpruned $\text{MAS}_{\text{round}=T}$, AgentPrune, and AgentDropout under three backbones. **B:** Prompt and completion tokens normalized by the paired unpruned MAS. **C:** Score change and total-token reduction relative to that MAS; each line follows one backbone from MAS through AgentPrune to AgentDropout. **D:** Matched results under Layered and Random initializations on Llama-3-8B. Deployment totals exclude topology-learning calls.

5.1.2. Topology Construction

Graph construction replaces a fixed dense graph with a task-specific organization. G-Designer marks an early transition from hand-selected motifs to query-conditioned edge design [226]. Starting from a fixed agent–role–tool pool and a chain anchor, it uses query and agent representations to generate communication edges. Its optimizer is trained on 40–80 queries per benchmark and then reused at test time. Across six benchmarks, the study reports task performance and prompt-token use for the constructed graphs. These results characterize anchor-regularized edge construction within a fixed agent pool rather than unconstrained system generation.

MACNET examines topology construction at a larger scale by organizing agents into directed acyclic collaboration networks [141]. Rather than learning a transferable prior or revising the graph during execution, it studies how agent count and graph structure affect collaborative scaling. Its experiments find that irregular topologies can outperform regular structures as the network grows.

ARG-Designer generates agent roles and communication links autoregressively instead of modifying a predefined graph [97]. Its role pool and DAG constraint nevertheless bound the resulting design space. GoAgent moves beyond individual agents by treating candidate groups as the units of graph construction [16]. HieraMAS introduces a different hierarchy: each functional role becomes a supernode containing multiple heterogeneous LLMs, and the communication topology is selected between these supernodes [212]. These methods extend topology construction from edge selection over a fixed team to the joint design of team composition and higher-level collaboration units.

Generative topology methods add an explicit inference or training pipeline. AMAS adapts communication structure at runtime [88]. GTD uses a conditional discrete graph-diffusion generator with a proxy reward model, whereas RADAR conditions a redundancy-aware diffusion process on the request [72,244]. Denoising, candidate scoring, and proxy-model training can be repaid by downstream savings, but they cannot be inferred from the sparsity of the selected graph.

A broader construction strategy appears in HiVA, which represents an agentic workflow as a self-organized graph and uses Semantic–Topological Evolution to update agent behavior and connectivity jointly [166]. Each optimization round executes a dynamically selected subgraph and converts environmental feedback into textual update signals for the next round. Since agent semantics and topology change together, the reported gains support the complete evolutionary workflow rather than an isolated graph effect. The repeated routing, feedback, and evaluation calls also belong to its construction cost.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

In summary, construction methods differ in scope, initialization, and cost timing. Their search space ranges from local edge decisions to higher-level organizational units, while generation may begin from an anchor or an empty graph. The resulting overhead can be amortized across a domain or paid for each query. These methods can therefore differ substantially in both flexibility and lifecycle cost.

Construction also complicates attribution because a learned graph may change both information flow and compute allocation. Comparisons with random or hand-designed graphs under matched call or token budgets are therefore needed to isolate the contribution of the learned structure.

5.1.3. Topology Adaptation

Topology adaptation conditions graph selection on information accumulated before the current structural decision. We distinguish two sources of such information. A *transferred prior* summarizes collaboration patterns learned from earlier tasks and conditions the initial topology for a new request. By contrast, *trajectory feedback* summarizes the current execution and determines which agents or links remain active at the next step. Construction generates an initial graph primarily from the request and design constraints; adaptation additionally uses structural knowledge acquired before or during execution.

Transferred Priors

TopoPrior learns reusable topology priors from reference collaboration graphs across domains [237]. For each request, a query-conditioned latent produces an initial graph that can be refined by a downstream topology optimizer. Figure 4 separates the offline supervision used to learn this prior from the resulting online token reduction. It also reports the workload at which the accumulated deployment savings offset the initial training cost. The result therefore supports an amortized efficiency gain rather than treating prior learning as cost-free. OFA-MAS learns structural priors and task–topology mappings within a shared mixture-of-experts graph generator [98]. It uses these learned regularities to generate query-specific initial graphs across domains. Its adaptation therefore occurs before collaboration begins and remains bounded by the role and graph families represented during training. MasFACT addresses a continual setting in which tasks arrive sequentially [184]. It transfers a posterior over collaboration structures from earlier tasks while limiting the loss of previously useful topologies. Unlike general workflow optimization, its reusable object is specifically a topology prior, so it belongs to topology adaptation despite incurring offline learning cost.

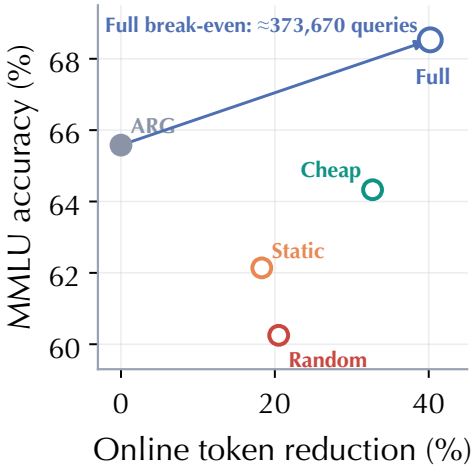


Figure 4. TopoPrior supervision, online token reduction, and the reported training break-even point for its Full configuration [237].

Trajectory Feedback

A second group revises topology during execution. DyLAN evaluates the contribution of agents in a preliminary trial and uses an LLM ranker to deactivate low-performing agents in later layers [114].

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

The responses observed at one stage therefore determine the active team and communication structure at the next. TacoMAS formulates inference as online graph adaptation [198]. Trajectory feedback updates agent capabilities on a fast loop, while a slower topology loop edits edges or changes team membership. The reported results support the value of test-time adaptation, but the accompanying controller calls must be included when assessing efficiency. MetaGen similarly adjusts structural decisions from lightweight runtime feedback [187]. Because it also rewrites role specifications, its results reflect joint semantic and topological adaptation rather than a topology-only intervention.

This classification is determined by the object being adapted, not merely by when learning occurs. Methods that jointly search prompts, operators, and graph structure produce compound workflows and are therefore discussed under Optimization in Section 5.3.

The two forms of adaptation have different failure modes. A transferred prior may not match a new domain or model pool, whereas noisy trajectory feedback can remove a useful specialist or repeatedly destabilize the graph. Evaluation should therefore separate the effect of initial topology transfer from subsequent online revisions and include the cost of the adaptation policy. When roles, prompts, and topology change together, the result supports system-level adaptation rather than a topology-only claim.

5.2. Runtime

Topology defines the available agents and communication links, whereas runtime control decides how this structure is used for the current request. We group runtime methods by their main target: *communication* controls exchanged content, *routing* selects the next computation, *state* manages reusable information and serving artifacts, and *scheduling* controls execution timing. These mechanisms may operate jointly rather than in a fixed sequence.

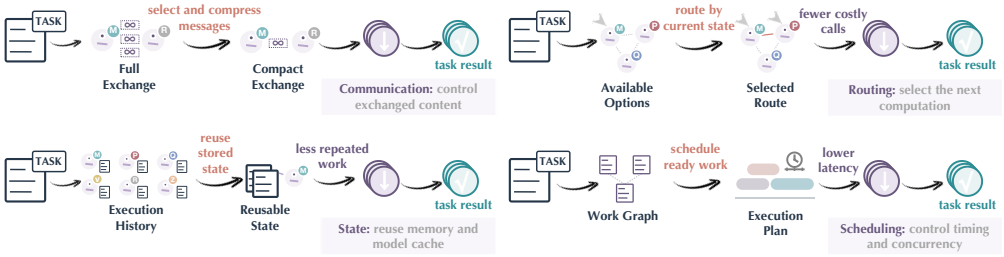


Figure 5. Runtime control points for efficient LLM-based MAS. With the topology fixed, runtime control decides how the structure is used for the current request. **Communication** compresses a full exchange into a compact one, **Routing** selects a route from the available options, **State** condenses the execution history into reusable memory and cache, and **Scheduling** compiles the work graph into an execution plan. In each panel, the intervention transforms the configuration on the left into the one on the right; the paired markers denote the resulting resource cost and task result.

5.2.1. Communication

Communication methods reduce the information exchanged between active agents through selective transmission or compact representations. Their efficiency depends on whether this reduction preserves the evidence needed by the receiver.

Distributed Sources

Multi-agent source processing partitions documents across workers and then aggregates their outputs. LLMxMapReduce [252] makes this structure explicit, while collaborative RAG systems [150] distribute retrieval or source access across agents. RAGentA [9] emphasizes attributed answers, MAIN-RAG [14] filters retrieved evidence with multiple agents, and MA-RAG [125] adds collaborative reasoning over retrieved passages. A survey of agentic RAG [159] places these designs within the broader retrieval pipeline.

Long-context systems use related communication patterns. Chain-of-Agents [240] partitions a document among sequential workers and relays compact intermediate states. LongAgent [246] uses a leader–member design for extended contexts, whereas GraphReader [95] explores facts, chunks,

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

and document-graph neighbors while keeping a small active context. Tree-of-Agents [216] combines hierarchical aggregation, prefix-hash caching, and adaptive pruning. The long-context Graph of Agents [77] instead treats input-specific collaboration as information compression. LSTM-MAS [74] further introduces gated and hierarchical information flow to limit error accumulation.

A noise-decomposition analysis [201] explains why divide-and-conquer can help when local chunks contain useful evidence and cross-chunk dependencies fit into compact relays, but can fail when the relay discards globally coupled information. This boundary is reinforced by information-bottleneck and task-structure analyses[133,217]: under unlimited relays, a distributed workflow can emulate a shared-context system, whereas bounded messages trade upstream filtering against downstream information loss.

Communication accounting should separate source-side processing from the context received by each model. A short receiver prompt may conceal substantial retrieval or preprocessing work. Provenance must also remain visible so that final claims can be traced through the transformations that produced them. These costs and information-preservation properties cannot be represented by a single "communication cost" measure.

Message Representations

Communication cost also depends on how information is represented. Textual summaries can shorten downstream prompts but may weaken provenance, while reusable state can avoid repeated processing without changing the visible response. CodeAgents [203] encodes communication as pseudocode artifacts rather than free-form dialogue. Agent-RADAR [231] retains textual messages but directs attention toward relevant content. LatentMAS [256] instead transfers latent state between agents. Each representation removes a different source of overhead, but may reduce auditability or restrict compatibility across models.

EcoLANG instead optimizes the natural-language channel itself [123]. It induces a compact communication language by removing redundant lexical alternatives and evolving sentence-level expression patterns, then uses the resulting language during social simulation. As shown in the *Compact exchange* panel of Figure 7, EcoLANG reduces response tokens by 20.3% on PHEME and 24.7% on HiSim, while the reductions in accumulated prompts and completions are smaller. The paired simulation scores do not decrease, indicating that the induced language primarily removes redundancy from generated communication rather than uniformly reducing every token source. Since language induction occurs before deployment, its construction cost, reuse frequency, and transfer to new simulation settings should be separated from the resulting runtime token saving.

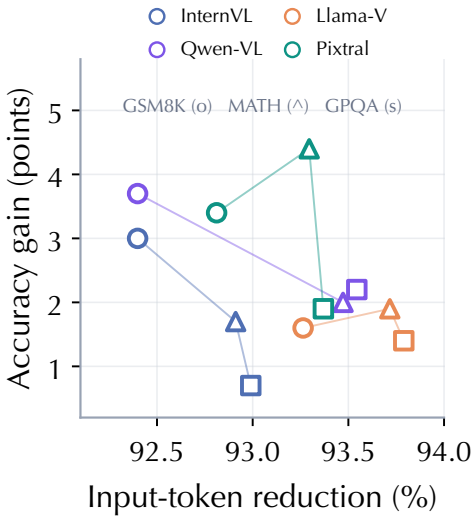


Figure 6. DebateOCR input-token reduction and paired accuracy change across the reported model–dataset settings [192].

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

Long-context communication usually combines three operations. *Partitioning* assigns chunks to workers and risks cross-chunk dependencies. *Retrieval* selects a small document or memory view but adds indexing and search. *Aggregation* bounds fan-in through summaries or trees but can accumulate compression error. Chain-of-Agents [240], GraphReader [95], and Tree-of-Agents [216] instantiate different combinations.

In summary, agents may exchange natural language, structured artifacts, compressed text, latent vectors, or KV state. These representations cannot be ordered by token count alone. Natural language is portable and inspectable, whereas structured formats replace free-form messages with explicit fields. Latent and KV channels may avoid decoding or repeated prefill, but are harder to audit and more dependent on the underlying model. Section 5.2.3 discusses these model-native states.

Evidence Fusion

Multiple agents have the clearest information advantage when their observations are jointly useful but individually incomplete. The aggregator must therefore preserve unique, source-grounded findings rather than equating agreement with correctness. RAGentA [9] and MAIN-RAG [14] illustrate how retrieved evidence can be filtered before synthesis. MARCH [101] separates proposal generation from document-grounded verification. If all agents receive the same incomplete evidence, or if the aggregator removes source provenance, collaboration adds no information and mainly duplicates computation.

Hi-Agreement [202] stress-tests this setting: private observations jointly identify the answer, yet one coherent deceiver can redirect the group. This is not itself an efficiency result, but it shows why communication reduction can be harmful when it removes the only correct evidence path. Evaluation should therefore examine whether source-grounded minority evidence survives, in addition to reporting tokens and latency.

Debate and Selection

Debate belongs to communication when the intervention changes the exchange protocol. Its benefit must be separated from that of generating additional candidates. Self-consistency [183], LLM-Blender [71], and best-of-N selection based on self-certainty [80] improve candidate diversity without inter-agent communication. Early multi-agent debate instead allows candidates to influence one another through critique or cross-model exchange [33,102,214].

Protocol design affects both information flow and cost. Dense debate repeatedly broadcasts the discussion history, whereas selective protocols narrow the exchange or terminate it early. Modular and social-psychology studies show that observed gains depend on both interaction structure and agent behavior [5,232]. Ringelmann-style scaling further suggests that an agent's marginal contribution can decline as the team grows [8]. Moreover, selective exchange reduces the broadcast overhead directly. S2-MAD [222] gates participation using message similarity. TodyComm [35] learns a communication DAG for each round under a degree budget. BANDMAS [31] filters semantic packets at the receiver according to their task relevance. CARA [25] analyzes the information exchanged by agents and provides diagnostics for these selectors. Localized debate and learned interruption either focus communication on disputed content or suppress messages with low expected value [43,174].

DebateOCR [192] compresses a debate history into a visual representation and reports input-token reductions above 92% in its multimodal-model setting, together with substantial latency reductions. The result establishes a within-study trade-off for one communication channel; it does not imply an equivalent reduction in total compute. The figure therefore pairs each token saving with its reported accuracy change instead of ranking models by compression alone.

Communication may correlate errors and suppress correct dissent. MINT [6] shows how misinformation can spread among benign agents, while DelibTrace [173] documents the loss of critical evidence during deliberation. Consensus may also become biased without a malicious participant [129], and an answer change does not necessarily indicate conformity [52]. SIGMA [55] and SEICS [128] model how

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

conflict and unreliable information propagate through the interaction graph. Agreement speed alone is therefore insufficient for evaluating communication efficiency.

Arbitration methods reduce these failures by controlling how conflicting evidence affects the final decision. EQUIMEM [121] calibrates shared debate memory, while ACE-MAD [113] assigns unequal influence to different reasoning states. SIGMA [55] represents support and conflict explicitly in the interaction graph. Budgeted Act-or-Defer [178] intervenes only when its estimated reliability satisfies a specified risk constraint. Evidence from data-cleaning tasks further shows that debate can be harmful when intervention is applied indiscriminately [136]. These methods improve efficiency by selecting useful interventions rather than merely reducing message volume.

Trust Boundaries

Communication and tool access also expose MAS to adversarial inputs. Prompt Infection [86] demonstrates injection that propagates between LLM agents, while InjecAgent [223] and AgentDojo [28] evaluate indirect prompt injection and defensive mechanisms. Tool-enabled agents introduce additional risks when interacting with websites or known vulnerabilities [36,37]; on one zero-day benchmark, a specialized team outperforms a single unguided agent [254]. Although these studies do not provide directly comparable efficiency results, they show that communication policies must be evaluated under consistent tool permissions and threat conditions. Security checks and recovery actions also contribute to runtime cost. SafeSieve [234] provides a narrower example of communication pruning under attack.

5.2.2. Routing

Routing decides where information or computation goes next. The routed object can be a document segment, a message, an agent role, a model endpoint, or the decision to continue. This distinguishes routing from topology: a graph defines possible paths, whereas a router activates one path for the current request or intermediate state.

Context and Intervention

Context routers limit what each active agent sees. RCR-Router [107] selects a role- and stage-relevant memory view under a token budget and reports up to 30% token reduction on multi-hop QA. This control is orthogonal to graph sparsity: the same edge can transmit a full history or a small retrieved state. SupervisorAgent [103] uses an LLM-free filter to invoke observation correction, guidance, or verification only at selected runtime states. On its SmolAgent GAIA evaluation, the paper reports about 29–30% token reduction without a success-rate decrease.

Smurfs [17] provides a tool-planning example of context control. It separates planning and tool execution from answer extraction and verification, allowing each role to use a task-specific view rather than repeatedly processing the complete tool history. An Answer Agent summarizes evidence from intermediate steps, while a rule-based rollback mechanism revises failed tool paths. On StableToolBench [48], Smurfs reports 8,096 tokens per query and a 57.4% pass rate, compared with 20,714 tokens and 55.4% for DFSDT [142]. The *Context control* panel of Figure 7 visualizes this paired result. It supports role-scoped context control under the reported protocol, provided that the additional role and retry overhead remains included in the end-to-end cost.

Context routing and communication compression can look similar at the token level, but they answer different questions. A compressor changes the representation sent over an active link; a router chooses the source, receiver, or subset to expose.

Agents and Models

CASTER [110] performs step-level routing between strong and weak models using semantic and structural signals. The paper compares it with *Force Strong*, a static baseline that assigns every step to the strong model. Relative to this baseline, CASTER lowers mean dollar cost across the four reported domains (i.e., Security, Science, Data, Software), while its domain-level quality change ranges from

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

-2.5 to $+0.7$ points. Thus, "maintains quality" is an aggregate claim whose acceptability depends on the deployment tolerance. The *Model routing* panel of Figure 7 makes this variation explicit, from a 0.7-point gain in Security to a 2.5-point loss in Software.

MasRouter [219] jointly determines the collaboration mode, team roles, and model assignments. It reports up to 52.07% lower overhead on code tasks and reductions of 17–28% when integrated with existing systems. MoMA [47] instead incorporates model prices into intent-conditioned routing governed by a state machine. STRMAC [177] performs sequential routing, selecting the next agent from the interaction history and available agent knowledge. Its reported path count can fall to 9.9% of exhaustive search.

These methods build on model-cascade approaches such as FrugalGPT [19] and preference-trained routing [130]. AnyMAC [181] extends this idea by jointly selecting the next agent and the context available to it. MAS routing can therefore involve several nested decisions. Let $\tau_{<t}$ denote the execution trace available before step t. A runtime routing policy ρ may produce the decision vector

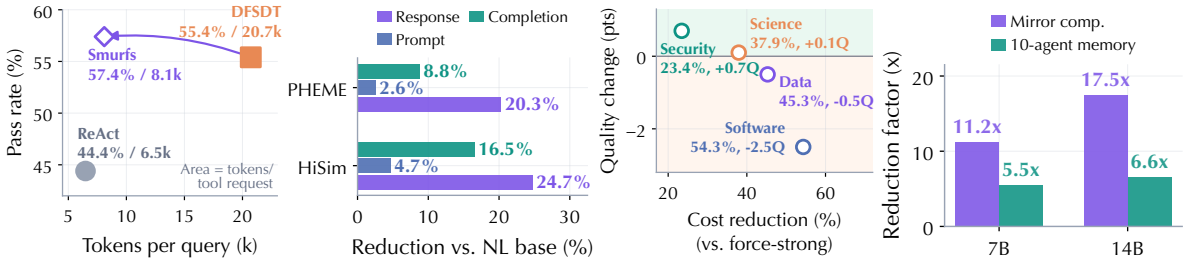
$$\mathbf{d}_t \sim \rho(\cdot \mid x, \tau_{<t}), \qquad \mathbf{d}_t = \left(d_t^{\text{mode}}, d_t^{\text{role}}, d_t^{\text{model}}, d_t^{\text{context}}, d_t^{\text{next}}\right). \tag{17}$$

These decisions respectively determine the collaboration mode, active roles, model assignments, visible context, and next execution step. They need not be made sequentially, and an individual method may control only a subset of them. At the system level, role activation changes the active subset of $V(x)$, model and context routing instantiate $m_i(x)$ and $h_i(x)$, and the next-step decision forms part of the execution policy $\pi(x)$ in Equation 8. Joint routing can exploit dependencies among these choices, but it also adds controller overhead and complicates attribution. Each adaptive component should therefore be compared with a fixed alternative under the same downstream resource budget.

Reliability Under Shift

Routing savings depend on the reliability of the controller. Aggregate routing accuracy can hide selective risk when unfamiliar or minority-domain requests are assigned to an inadequate model. Coverage–risk curves should relate the fraction handled by the cheap path to correctness, cost, and escalation. Model prices, endpoint behavior, context limits, and tool availability also change. A router that encodes a fixed strong–weak ordering may become obsolete even when the task distribution is stable, making model-pool replacement and policy-update cost important transfer tests.

Routing is especially consequential in stateful environments. Benchmarks such as WebArena [249], WorkArena [32], OSWorld [197], and τ-bench [211] contain trajectories in which each action depends on the preceding state. A local error by a cheaper model may therefore trigger an extended recovery process or an irreversible action. Per-step price does not represent episode-level cost, which must include the expected cost of the remaining trajectory and any subsequent recovery.



(a) Context control **(b) Compact exchange** **(c) Model routing** **(d) Shared state**

Figure 7. Representative runtime results under paper-specific baselines. **(a)** Smurfs trades StableToolBench tokens for pass rate; marker area denotes tokens per tool request [17]. **(b)** EcoLANG reports separate reductions in responses, accumulated prompts, and completions [123]. **(c)** CASTER pairs dollar-cost reduction with domain-specific quality change [110]. **(d)** TokenDance reports mirror-cache compression [10]; the 10-agent memory reduction is derived from its Master–Mirror formula. Panels retain their original experimental scopes and are not a cross-paper ranking.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

5.2.3. State

State methods reduce repeated work through two main mechanisms: memory and cache. Memory retains task-relevant information across steps or requests and can therefore affect subsequent decisions. Cache reuses model-internal computation during serving, such as shared prefixes or KV states, without necessarily changing the logical sequence of model calls.

Persistent Memory

Single-agent memory systems provide important mechanism baselines. MemGPT [132] and MemoryBank [247] focus on retaining information beyond the active context, while ExpeL [245] reuses experience from previous tasks and A-MEM [200] dynamically organizes stored memories. Multi-agent systems additionally need to manage how memory is shared. Collaborative Memory [147] controls access to shared state, whereas MIRIX [185] assigns memory functions to specialized agents. G-Memory [228] stores hierarchical team traces, and Agent KB [167] retrieves reusable experience across domains.

Recent systems differ in how memory is represented and assigned within a team. Intrinsic Memory Agents [220] equips each role with a structured memory template that is updated from the agent's own outputs. This design preserves role-specific knowledge and procedural constraints that may be lost in a shared conversation history. LEGOMem [49] decomposes previous task trajectories into reusable procedural units and assigns them to different levels of the workflow. Memories available to the orchestrator support task decomposition and delegation, whereas agent-level memories guide execution and tool use. LatentMem [39] stores lightweight interaction trajectories in an experience bank and uses a learned composer to transform retrieved experience into compact, agent-specific latent memories. Its optimization objective trains these representations using task-level feedback rather than preserving retrieved text directly. A broader survey [194] examines how memory supports coordination and collective adaptation, while an architecture-oriented analysis [218] focuses on ownership, coherence, and storage organization.

Persistent memory moves cost across requests. The relevant comparison is therefore longitudinal: how often a stored item is reused, how much work it saves on each use, and how frequently it must be updated. Section 5.3 treats systems that primarily learn reusable skills, workflows, or organization priors; the present section concerns the runtime storage and retrieval that exposes those assets to agents.

Cache Sharing

Token Coherence [134] applies MESI-inspired coherence states and lazy invalidation to shared multi-agent artifacts. TokenDance [10] instead targets collective KV-cache sharing in all-gather-style multi-agent serving. The *Shared state* panel of Figure 7 presents its reported 11.2× and 17.5× mirror-cache compression, together with the corresponding 5.5× and 6.6× memory reductions for 10 agents derived from the Master–Mirror formula. TokenDance also reports maximum gains of 1.9× in prefill speed, 2.3× in end-to-end throughput, and 2.7× in supported concurrency.

Beyond collective cache sharing, several methods reuse internal state between individual agents. KVCOMM [213] addresses the cache-offset mismatch created when agents process shared content under different prefixes. It estimates the required adjustment from an online anchor pool, allowing overlapping KV states to be reused during prefill without additional training. Q-KVComm [84] instead reduces the state transmitted between agents through sensitivity-aware, layer-wise quantization. It also calibrates the compressed representation for heterogeneous model pairs, making communication bandwidth rather than cache capacity its primary target.

Cross-model methods require stronger compatibility mechanisms. Cache-to-Cache [40] learns to project and fuse a source model's KV state into selected layers of a target model, enabling semantic transfer without generating an intermediate textual message. DroidSpeak [112] focuses on fine-tuned models derived from the same foundation model. It reuses compatible cache layers and recomputes only those identified as sensitive to the difference between model variants. LRAgent [70] specializes

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

this idea for Multi-LoRA agents by separating the cache into a shared backbone component and a low-rank adapter-specific component, thereby avoiding a complete cache copy for every role.

Latent communication is related but conceptually distinct from KV-cache reuse. LatentMAS [256] places agents' last-layer hidden representations in a shared latent workspace so that subsequent agents can consume them without text decoding. LatentMem [39], by contrast, composes persistent latent memories from retrieved experience for later reuse.

To interpret these savings, it is necessary to separate agent-level organization from physical state reuse. The execution graph $\mathcal{G}(x)$ describes the semantic organization of an MAS, whereas its serving realization determines how model state is stored and reused. Let $\mathbf{K}_i$ denote the KV cache required by agent i. Storing each cache independently gives

$$M_{\mathrm{KV}}^{\mathrm{naive}} = \sum_{i \in V(x)} |\mathbf{K}_i|. \tag{18}$$

When the agent caches contain a reusable component, each cache may be reconstructed as

$$\mathbf{K}_i = \mathcal{R}\Big(\mathbf{K}_{\mathrm{shared}}, \mathbf{K}_i^{\mathrm{private}}\Big), \tag{19}$$

where $\mathcal{R}$ denotes the method-specific reconstruction operation. The resulting memory footprint can then approach

$$M_{\mathrm{KV}}^{\mathrm{shared}} = |\mathbf{K}_{\mathrm{shared}}| + \sum_{i \in V(x)} \left|\mathbf{K}_i^{\mathrm{private}}\right| + M_{\mathrm{meta}}, \tag{20}$$

where M_{meta} accounts for indexing and reconstruction metadata. The saving is substantial only when the shared component is large and inexpensive to reuse. It reduces duplicated serving state and may shorten prefill, but does not by itself change logical model calls or generated tokens. Likewise, shorter messages need not improve cache reuse when agents process different prefixes.

Serving results are meaningful only when cache reuse and memory footprint are paired with latency measurements under a stated model–hardware–workload configuration. Maximum agent count or peak speedup is otherwise not comparable across systems.

5.2.4. Scheduling

Scheduling determines the execution order and concurrency of an agent workflow under resource and dependency constraints. It translates the selected workflow into a concrete execution plan, thereby affecting wall-clock latency, peak demand, and redundant computation.

Dependencies

A work graph specifies which operations depend on earlier results; the scheduler realizes those dependencies during execution. L2MAC [60] executes a stored prompt program through instruction-specific agents coordinated by a control unit and shared file store. MapCoder [68] follows a staged schedule in which retrieval and planning precede code generation, with debugging activated after execution feedback. Flow [127] represents a request as an activity-on-vertex graph and revises task allocation from earlier outcomes, allowing the executable portion of the graph to change during a run.

Some systems also modify the work graph itself. AgentNet [207] allows decentralized agents to revise task routes and connectivity, whereas LEMON [22] produces an executable orchestration specification whose dependencies constrain subsequent execution. These methods also involve topology or learned control, but their scheduling effect lies in determining which operations become ready after each completed event.

Earlier collaboration frameworks [164,250] and later studies of agent services and operational infrastructure [29,92,190] provide the substrate for executing such graphs. Their relevance to scheduling concerns queueing, artifact handoff, and failure handling.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

Concurrency

Concurrency reduces latency only when ready operations do not depend on one another. Parallel branches may shorten the critical path while increasing aggregate model time or peak demand. AgentRadio [146] moves incoming messages to an asynchronous background channel, allowing an agent to continue local work instead of blocking on every exchange. MAESTRO [205] instead conditions one broadcast at each round, creating explicit synchronization points while limiting repeated discussion. Both methods alter waiting and overlap rather than the underlying task set.

Capacity

A selected operation may still need to wait when its model, tool, or budget is unavailable. BudgetMLAgent [41] combines specialist execution with bounded access to stronger models, making the admission of expensive calls part of the schedule. CORL [76] learns budget-conditioned delegation over a heterogeneous model pool; routing identifies an appropriate worker, while scheduling determines the order and concurrency of the resulting calls.

Resource constraints also arise from the task environment. Collaborative RAG workflows [150] can process multiple sources concurrently when their retrieval branches are independent, but shared retrievers or dependent evidence may limit this parallelism. In WarAgent [65], role-specific actions interact with an evolving simulated state, so later decisions must wait for the outcomes of preceding events. These examples show that feasible concurrency depends not only on the workflow graph but also on the resources and state transitions exposed by the environment.

Recovery

Scheduling also determines what happens after a failed or uncertain event. Retrying an isolated operation can avoid restarting an entire trajectory, but monitoring and diagnosis add their own calls. MAST [12] provides a taxonomy of failures in multi-agent workflows, including breakdowns in verification and termination. Failure-attribution methods [235] seek to identify the responsible component before recovery is triggered.

Recovery systems differ in how much of the execution they revise. MASC [155] monitors workflow state and applies targeted correction, while AgentRevive [236] repairs a failed trajectory from intermediate evidence rather than repeating it in full. Agent Q-Mix [73] learns agent-level communication actions that alter the active interaction graph from round to round, belonging to policy optimization and is discussed in Section 5.3 . Although its controller is trained offline, its scheduling consequence appears online: the selected actions determine which agents remain active and which operations become executable in the next round. Recovery efficiency therefore depends on avoided recomputation after accounting for monitoring, attribution, and unsuccessful repair attempts.

Accounting.

End-to-end results show why scheduling should not be inferred from workflow structure alone. MapCoder [68] can improve code-generation accuracy while using substantially more calls and tokens than direct prompting; its staged execution trades additional work for a higher probability of successful repair. OneFlow [199] reports lower execution cost for several reconstructed workflows by consolidating their operation into a shared-state single-model process.

Empirical audits of agent frameworks [3,149,215] further show that orchestration calls can account for a substantial part of deployment cost. Scalability analyses [151] examine how this overhead changes with larger teams, while Digital Pantheon [124] illustrates a different constraint: in large social simulations, a faster schedule remains useful only if it preserves the behavior being simulated.

In summary, scheduling does not decide which agents or models should exist. It realizes the selected workflow under dependency, capacity, and failure constraints. Its efficiency is established when the same useful work is completed with a shorter critical path or less duplicated execution, after scheduler and recovery overhead have been included.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

5.3. Optimization

Optimization methods use development-time search or learning to improve an asset that can shape later executions. In this survey, the category covers reusable prompts, decision policies, compound workflows, and experience-derived skills. Learning alone does not determine the category. A learned mechanism whose primary output is a collaboration graph belongs to *Topology*, while one that selects the information or system state used by the current request belongs to *Runtime*.

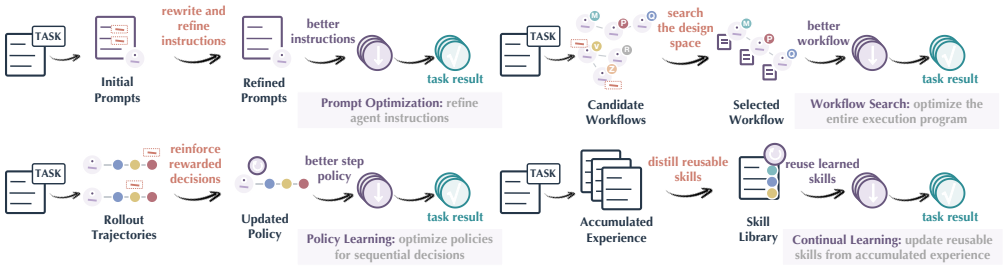


Figure 8. Optimization methods for efficient LLM-based MAS. Prompt optimization refines the instructions of agents, workflow optimize the entire execution program, policy learning improves the policy behind sequential decisions from rewarded rollouts, and continual learning distills accumulated experience into a reusable skill library.

We organize optimization methods by the object they improve. *Prompt Optimization* changes agent instructions within a largely fixed workflow. *Workflow Search* treats the execution program as the search object. *Policy Learning* trains sequential decisions rather than selecting a complete program explicitly. *Continual Learning* updates reusable assets from experience accumulated across requests.

Reuse creates a lifecycle trade-off. Search and training may consume substantial resources before deployment, even when the resulting system is inexpensive to run. The relevant efficiency gain therefore depends on both the deployed operation and the number of subsequent requests over which the development cost is amortized. Re-optimization after task, model, or environment changes shortens this effective reuse horizon.

5.3.1. Prompt Optimization

Multi-agent prompts are coupled through the execution workflow. Changing one role alters the information received by subsequent roles, so an intermediate response that appears locally stronger need not improve the final result. Prompt optimization must therefore evaluate prompts through their contribution to system-level performance.

MAPGD addresses this coupling by assigning prompt-refinement objectives to specialized agents [51]. These agents produce textual pseudo-gradients that are coordinated in a shared semantic space before candidate prompts are generated and selected through a bandit mechanism. The method therefore handles conflicting updates without optimizing each role independently.

The *Prompt optimization* panel of Figure 10 shows the resulting resource trade-off. Under the reported 50-query protocol, MAPGD improves average F1 from .83 to .87 while reducing calls from 962 to 643 and tokens from 256k to 236k relative to ProTeGi, a single-agent automatic prompt optimizer baseline based on textual gradients [139]. Its wall time nevertheless increases from 159.4 to 201.5 seconds. MAPGD therefore improves model usage under this protocol, but not every optimization resource simultaneously.

MAPRO formulates coupled prompt optimization as maximum-a-posteriori inference and applies topology-aware belief propagation [243]. MASPOB instead combines a UCB bandit with a topology-aware GNN surrogate and coordinate ascent [62]. Both methods retain the workflow topology while modeling how that topology couples the prompts being optimized.

Other approaches differ mainly in how they evaluate and explore candidate prompts. MASPO measures each prompt through its downstream contribution and performs an evolutionary beam search [189]. MA-SAPO uses multiple agents to interpret evaluation results and convert them into prompt-revision guidance [152]. These approaches extend prompt optimization from isolated textual quality to the effect of each instruction on the complete team.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

MAS-PromptBench examines whether local prompt improvements transfer to system-level performance [2]. Its results reinforce the need to distinguish prompt quality from workflow quality: the same prompt can behave differently when its downstream recipients or communication paths change.

The main setup cost of this family comes from generating and evaluating candidates. A method may discover prompts that improve quality at unchanged deployment cost, or prompts that preserve quality with shorter exchanges. In either case, the optimization calls remain part of the lifecycle result even though they disappear from the final execution trace.

5.3.2. Workflow Search

Workflow search treats the execution program as a coupled object. The searched program may change several components at once, so its benefit cannot usually be attributed to one prompt or one communication edge.

GPTSwarm represents agent systems as optimizable computational graphs and updates both node prompts and graph connectivity [255]. AFlow searches code-represented workflows through execution feedback [233]. MaAS learns an agentic supernet from which query-dependent systems can be sampled [229]. EvoMAC evolves collaborative networks from textual feedback, while EvoFlow maintains and evolves a population of heterogeneous workflows [64,227]. Their common output is a reusable execution program rather than a decision limited to the current message or cache state.

OMAC broadens this search by defining functional and structural dimensions of an MAS and supporting both individual and joint optimization [96]. Since its search can alter several parts of the system, it is better understood as workflow optimization than prompt optimization. Its dimension-wise search improves tractability, while joint refinement captures interactions that isolated updates may miss.

MetaAgent represents an execution program as a finite-state machine [242]. Its states specify agent operations and its transitions control subsequent execution. The generated object therefore contains state-dependent control flow and tool use, rather than only a communication graph. Its lifecycle cost includes construction and refinement of the finite-state machine before it can be reused.

MASS makes coupling across workflow variables explicit [248]. It first optimizes local prompts, then searches topology and configuration, and finally refines the workflow-level prompts. The *Stage returns* panel of Figure 10 shows that these stages contribute unevenly: block-level prompt optimization produces the largest reported improvement, whereas the final global refinement adds less than one point. This result indicates diminishing returns within the reported MASS search process, although it does not establish the same ordering for other workflow spaces.

Workflow search can improve the final deployment point while consuming many development evaluations. The *Lifecycle trade-off* panel of Figure 10 retains both quantities for MASS: stronger MATH performance is associated with greater optimization expenditure, while marker area records the resulting per-query inference cost. The panel therefore separates the cost of discovering a workflow from the cost of executing it.

The *Design and deployment* panel provides a second lifecycle view. MetaAgent records both construction and execution tokens for its software and machine-learning task batches. It achieves the highest reported task score and the lowest recorded token total in that comparison. However, the human design effort used by MetaGPT is not tokenized, so the panel supports automated construction under the reported accounting boundary rather than a general claim that human design cost has been eliminated.

A reusable workflow can also become obsolete. Changes in the model pool or tool interface may alter the ordering discovered during search. Transfer across tasks and model families is therefore stronger evidence of reuse than repeated gains after benchmark-specific optimization. This largely remains as an open challenge.

5.3.3. Policy Learning

Policy learning trains decisions made along a collaboration trajectory instead of selecting one complete workflow in advance. The learned policy may control communication, delegation, or organizational actions while retaining the surrounding execution framework.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

OPTIMA trains communication behavior with objectives that combine task effectiveness and token efficiency [21]. Its iterative data generation and preference-training process moves cost to development time, while the resulting communication policy changes the length and content of deployed exchanges. The reported Pareto trade-off therefore concerns a learned policy rather than direct prompt compression.

MAGRPO formulates language-model collaboration as cooperative MARL and extends group-relative policy optimization to multi-agent trajectories [108]. CoLLM compares centralized and decentralized actor–critic training [109]. Its centralized critic is more stable in the reported sparse-reward settings, while deployment remains decentralized.

Embodied systems expose resources beyond tokens. ReAd learns a critic-based advantage function for multi-agent planning [241]. It reduces environment steps on its Overcooked-AI and RoCoBench variants and reduces LLM-query rounds on the latter. Dr. MAS addresses instability in multi-agent GRPO through agent-wise advantage normalization [38]. These results concern interaction efficiency and optimizer stability rather than the same resource coordinate.

MARCH uses asymmetric information during training to improve factual grounding [101]. A Proposer converts a Solver response into verifiable propositions, and a Checker compares them with retrieved evidence. Since these roles generate a training signal, the downstream object being evaluated is the learned policy rather than an online three-agent pipeline. MARCH therefore demonstrates information asymmetry as an optimization mechanism, not the deployment efficiency of keeping all three roles active.

Other methods learn decisions about delegation and organizational control. CORL conditions a centralized controller on a deployment budget, while LEMON learns executable orchestration from counterfactual feedback [22,76]. EvolveRouter jointly improves routing and the prompts used by the routed agents [67]. Agent Q-Mix learns team-level action selection [73]. These methods differ in their action spaces, but their common output is a reusable decision rule.

Credit assignment remains a central difficulty. A shared terminal reward does not directly identify which agent action or exchange produced the gain. Centralized critics can use joint information during training, whereas decentralized critics better match the observations available during deployment. The comparisons in CoLLM and Dr. MAS show that training stability and deployment structure must be considered separately [38,109].

Policy-learning efficiency also has three distinct levels. Sample efficiency concerns the experience required to learn the policy. Optimization efficiency concerns the resources consumed by training, while execution efficiency concerns the deployed trajectory. Improvement at one level does not imply improvement at the others.

5.3.4. Continual Learning

Continual learning changes when optimization occurs. Instead of producing one fixed asset before deployment, these methods use experience from completed requests to revise future behavior. The reusable object may be a skill, memory-derived procedure, or broader team strategy, but not merely the topology prior covered in Section 5.

MetaTeam performs post-task collaboration over the agents' execution experience and turns the resulting feedback into reusable system improvements [53]. Its updates can affect agent behavior and team-level coordination, so it is broader than a topology adaptation method.

SkillGraph attaches reusable skills to a multimodal collaboration graph [126]. G-Memory and LEGOMem become continual mechanisms when stored traces or procedures alter later team behavior, rather than only supplying context to the current request [49,228]. Counterfactual planning with agentic memory similarly retains experience that can modify subsequent collaboration [24].

Human-in-the-loop adaptation introduces an external source of persistent feedback [206]. Its efficiency depends on whether delayed corrections improve enough future requests to repay the required human effort. This differs from asking a human to repair only the current answer.

The boundary with Runtime lies in what changes across requests. Retrieving an existing memory item for the current execution is a runtime state decision. Updating a reusable skill, procedure, or

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

policy from accumulated experience is continual optimization. A system may perform both, but their construction and per-request use represent different costs.

Continual efficiency is therefore longitudinal. A reusable asset is beneficial only when later improvements repay its construction and maintenance, and when it remains useful after task or model changes. Transfer without rebuilding the asset provides stronger evidence of continual learning than repeated optimization on the same benchmark.

6. Evaluation

Evaluation applies the concepts defined in Section 3 to the analysis of prior empirical studies. We organize existing evaluation practices along five complementary dimensions: *resource measurement*, *comparison design*, *benchmark coverage*, *process analysis*, and *trade-off reporting*. Together, they connect an intervention in $\mathcal{G}$ to its observed effect on Q and $\boldsymbol{c}$, while clarifying the attribution, scope, and practical significance of that effect.

6.1. Resource Measurement

Resource measurement first distinguishes configured budgets from realized consumption. A reasoning-token limit constrains only one coordinate and does not determine the complete prompt, model calls, or execution time; this distinction is also visible in controlled SAS–MAS comparisons under requested reasoning budgets [169]. Trace-based analyses such as AgentTaxo [179] and token-economics audits [149] further show that resource use can be distributed unevenly across workflow components. We therefore interpret efficiency claims using usage recovered from τ together with direct system measurements.

Recovering these quantities is not sufficient; they must also be aggregated under a consistent counting boundary. A message inserted into a later prompt is already included in T_{prompt} and should not be added again to the token total. Message volume and interaction structure can still be reported separately to explain where resource use originates. Calls introduced by orchestration must likewise be assigned to the component that invokes them rather than attributed only to worker agents.

This accounting boundary also extends across the system lifecycle. TopoPrior [237], for example, reports online token reduction together with the workload required to recover its training expenditure. More generally, search or training contributes to $C^{(u)}_{\text{design}}$, whereas operations performed for a request contribute to $C^{(u)}_{\text{run}}$. Monetary cost is then obtained by applying a stated price schedule to measured usage; it is a valuation of resource consumption rather than an independent physical resource.

6.2. Comparison Design

Once resources are measured consistently, an efficiency claim still requires an appropriate counterfactual. The baseline should follow the claimed contribution. A method that improves an existing workflow can be compared with the corresponding unmodified MAS. By contrast, a claim about collaboration itself requires a single-controller alternative with comparable access to models, tools, information, and test-time computation. A one-shot baseline cannot separate collaboration from the additional capabilities or resources introduced by the evaluated system.

After the baseline is chosen, mechanism attribution requires a controlled change to the execution graph. In the notation of Equation 8, an ablation should modify the relevant component of $\mathcal{G}$—one of V, E, $\{m_i\}$, $\{h_i\}$, or π—while holding the others fixed where possible. Paired traces strengthen this comparison by preserving the query, available evidence, and resource budget while changing only the organizational decision under study.

A controlled ablation at one operating point does not establish general scaling behavior. An evaluation of 260 configurations shows that MAS gains vary across models and task conditions rather than following agent count alone [82]. Effective-team-size analysis further finds diminishing marginal participation as teams grow [8], while information-bottleneck analysis shows that relay loss can offset the benefit of decomposition [217]. OneFlow [199] supplies a shared-state, single-model counterfactual

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

and executes some homogeneous workflows more cheaply in its evaluated settings. These results favor comparisons across multiple operating conditions over a universal scaling curve.

Evaluation platforms provide the configurations needed for such comparisons. MultiAgentBench [253] compares several communication topologies, while MALLM [5] varies both interaction and decision protocols. These platforms support mechanism ablations, but differences in task score constitute efficiency evidence only when the corresponding resource use is also measured.

6.3. Benchmark Coverage

After a comparison isolates the target mechanism, benchmark coverage determines how far the resulting claim can be generalized. Short-form tasks simplify resource accounting but reveal little about sustained coordination. Interactive environments expose this overhead more directly, although their results are more sensitive to environment variability.

Existing evaluation infrastructures differ in the behavior they emphasize. AgentBench [111] spans multiple agent environments, while AgentBoard [116] adds trajectory-level progress and failure analysis. AgentGym [195] provides environments for agent evolution. MAS-focused suites examine interaction more directly: MultiAgentBench [253] and BattleAgentBench [182] evaluate cooperative or competitive behavior, whereas SOTOPIA [251] targets social interaction. AgentCollabBench [120] focuses on cases in which individually capable agents still collaborate poorly, and MAST [12] attributes failures from execution traces.

Tool-mediated suites expose a different operating regime. GAIA [122] evaluates general assistance, while WebArena [249], WorkArena [32], and OSWorld [197] require interaction with web or desktop environments. τ-bench [211] evaluates policy-constrained tool use.

Table 3 summarizes the workloads, quality metrics, and resource reporting in the surveyed studies. Evaluation remains concentrated on static short-form tasks, particularly knowledge MCQA, function-level code, and grade-school arithmetic. These studies commonly report tokens but seldom measure latency or monetary cost. The latter measurements appear more often in long-context or tool-mediated settings. Live web and desktop benchmarks remain rare, leaving much of the operational overhead outside the current evidence base.

6.4. Process Analysis

Once benchmark coverage establishes where a result is observed, process analysis examines how that result is produced. Endpoint quality alone cannot identify the information path responsible for the final answer. GEMMAS [87] represents agent interaction as a DAG and measures both information diversity and unnecessary paths. On GSM8K, systems separated by only 2.1% in task score exhibit substantially different interaction patterns. Information-propagation analysis provides a related view by tracking how topology affects the evidence received and retained by agents [154].

Graph-level evaluation connects these process measurements to system outcomes. Graph-of-Agents [221] combines graph construction with iterative message passing and refinement. On MMLU-Pro, its GoA-Max configuration improves accuracy over MoA while reducing calls, tokens, and execution time; Figure 9 preserves these reported coordinates and will be discussed in detail in Section 7. DySCo [45] instead constructs sparse consensus using reliability and resource signals.

Execution traces provide a finer view of the same process. Participation records show which agents were active, while provenance traces reveal how evidence reached the final decision. Timing and failure labels locate the critical path and quantify the work performed before recovery or termination. When paired with a controlled intervention on $\mathcal{G}$, these measurements connect observed resource changes to specific organizational decisions.

Resource attribution adds a component-level view. AgentTaxo [179] identifies where tokens accumulate within a workflow, while software-agent audits examine the cost of orchestration and tool interaction [3,149]. Token-economics analysis further maps trace-level usage to endpoint prices and workload characteristics [23]. Combining usage records with failure labels distinguishes productive control overhead from work that does not alter the outcome.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

Table 3. Benchmarks in the surveyed corpus by workload family. The *Quality metric* column states the reported score and the quantity it measures. The *Tokens*, *Latency*, and *Cost* columns mark how often the surveyed evaluations on a benchmark family also report that realized resource coordinate, and the *Corpus use* bar indicates how many surveyed works are evaluated on the family.

Benchmark	Quality metric	Tokens	Latency	Cost	Corpus use
Static knowledge and reasoning QA					
MMLU [57]; MMLU-Pro [186]	MCQ accuracy	✓	○	○	▮▮▮▮
GPQA(-Diamond) [145]	MCQ accuracy	○	✓	○	▮▮▮
StrategyQA [44]; CSQA [165]; ARC [26]	accuracy	○	○	○	▮▮▮
BBH [163]; MuSR [162]; LogiQA [106]	accuracy	○	—	○	▮▮
Mathematical reasoning					
GSM8K [27]	exact-match accuracy	✓	○	○	▮▮▮▮
MATH; MATH-500 [58]	exact-match accuracy	✓	○	✓	▮▮▮
MultiArith [148]; SVAMP [137]; AQuA [104]	accuracy	✓	○	○	▮▮▮
AIME; AMC; OlympiadBench [54]	exact match; pass@1	✓	○	✓	▮▮
Code generation and software engineering					
HumanEval [20]	pass@1	✓	○	○	▮▮▮▮
MBPP [1]	pass@1	✓	○	✓	▮▮▮
LiveCodeBench [69]; APPS [56]; CodeContests [99]	pass@1	✓	○	○	▮▮
SWE-bench [75]	% resolved	○	○	✓	▮▮
Retrieval, long context, and memory					
HotpotQA [208]	EM / answer F1	✓	✓	○	▮▮▮
2WikiMultihopQA [59]; MuSiQue [171]	EM / F1	○	✓	○	▮▮
TriviaQA [78]; NQ [85]; FEVER [168]	EM; label accuracy	✓	✓	—	▮▮
InfiniteBench [239]; LongBench [4]; LoCoMo [118]	retrieval acc.; QA F1; ROUGE	✓	✓	✓	▮▮
Interactive, embodied, and social environments					
GAIA [122]	exact-match accuracy	✓	○	✓	▮▮
WebShop [209]; AppWorld [172]; TravelPlanner [196]	reward / success	✓	○	○	▮▮
ALFWorld [158]; VirtualHome; ScienceWorld [180]	success rate	✓	○	○	▮▮
WebArena [249]; WorkArena [32]; OSWorld [197]; τ-bench [211]	programmatic success	—	—	—	▮
Minecraft (Voyager [175]; building [253])	items unlocked; success	✓	○	✓	▮
TDW-MAT; C-WAH [230]; Overcooked-AI [11]	transport rate; steps	✓	○	✓	▮
SOTOPIA [251]	judge scores	—	—	—	
MAS-oriented instruments					
MultiAgentBench [253]; BattleAgentBench [182]	milestone / KPI scores	—	—	—	▮
MALLM [5]	QA accuracy	—	—	—	▮
MAST [12]; Who&When [235]	coverage; attribution acc.	—	—	—	▮
AgentCollabBench [120]; MASPromptBench [2]	task success; robustness delta	—	—	—	▮

✓ = reported by at least half of the surveyed uses on the family; ○ = reported by a minority; — = rarely or never reported. Corpus use: ▮ 1–6; ▮▮ 7–10; ▮▮▮ 11–15; ▮▮▮▮ 16–30; ▮▮▮▮▮ 31+ surveyed systems.

6.5. Trade-Off Reporting

Trade-off reporting examines how observed quality changes with resource availability. For a per-request resource coordinate c_k and evaluation queries $\{x_i\}_{i=1}^{N_{\text{eval}}}$, an evaluated system produces the empirical operating point

$$\widehat{Q}(\mathcal{G}) = \frac{1}{N_{\text{eval}}} \sum_{i=1}^{N_{\text{eval}}} Q(\mathcal{G}(x_i), x_i), \qquad \widehat{c}_k(\mathcal{G}) = \frac{1}{N_{\text{eval}}} \sum_{i=1}^{N_{\text{eval}}} c_k(\mathcal{G}(x_i), x_i). \tag{21}$$

Repeating this calculation for configurations evaluated at different budgets yields a response curve over quality and realized resource use.

A response curve traces one resource coordinate at a time. When several coordinates vary together, configurations can instead be compared through their Pareto relation. A configuration is dominated if another provides no lower quality, consumes no more of every measured resource, and is strictly better on at least one of these dimensions. Frontier membership should account for paired uncertainty when quality differences are small.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

Both response curves and Pareto sets depend on how results are aggregated. Macro averaging weights tasks or benchmarks equally, whereas micro averaging weights individual examples. Resource totals may therefore use a different denominator from the reported quality score. The AgentDropout reconstruction in Figure 3, for example, retains the reported six-task macro score and presents token reductions separately for each backbone.

For methods with offline optimization, the deployment frontier captures only the online portion of the trade-off. Lifecycle reporting relates $C^{(u)}_{\text{design}}$ to $C^{(u)}_{\text{run}}$ through the break-even workload $n^\star$ under a common accounting unit. TopoPrior [237] provides one example by reporting online token reduction together with the workload required to recover its training expenditure. If a task or model change requires re-optimization, the additional design cost changes the corresponding $n^\star$.

7. Evidence Synthesis

Following the taxonomy in Section 5, this section synthesizes how topology, runtime control, and reusable optimization affect the quality–resource trade-off. Quantitative results retain the baseline and denominator used in their original studies.

7.1. Topology Findings

Topology studies show that graph organization can affect both task quality and resource use. G-Designer [226] compares four graph families at matched agent counts in its MMLU scaling experiment, thereby controlling for team size within each comparison. Graph-of-Agents [221] evaluates GoA against MoA across six benchmarks and additionally reports calls, tokens, and latency on MMLU-Pro. Figure 9 presents these as two distinct within-study comparisons: graph-family scaling in G-Designer and system-level operating points relative to MoA.

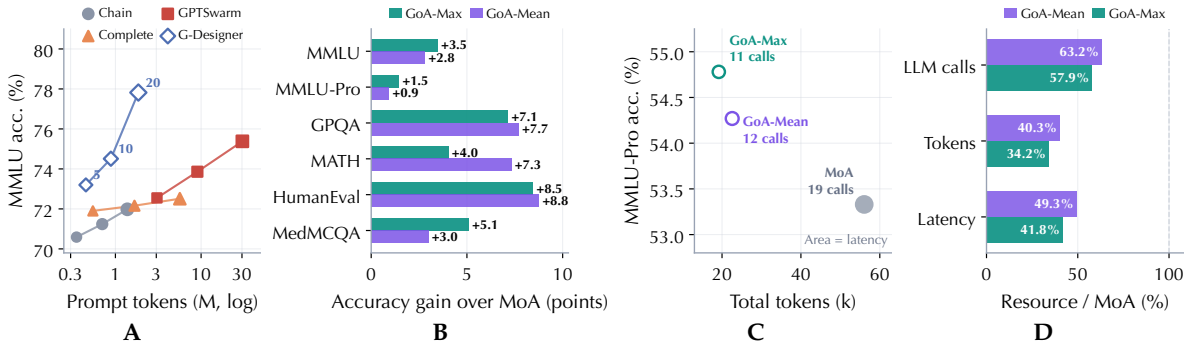


Figure 9. Graph organization within matched studies. A: G-Designer operating points from its GPT-3.5 MMLU scaling experiment; labels denote agent count [226]. **B:** GoA-Max and GoA-Mean score changes relative to the same MoA baseline across six benchmarks. **C:** MMLU-Pro accuracy, calls, tokens, and latency for the three reported systems; marker area denotes latency. **D:** The resource values in C normalized by MoA [221]. Panel D is a normalized view of the same experiment, not additional evidence.

Panel A compares graph families at matched agent counts, showing that G-Designer's advantage is not explained by a smaller team [226]. It achieves the highest MMLU accuracy at every reported scale. At 20 agents, G-Designer reaches 77.82% accuracy with 1.85 million prompt tokens, whereas GPTSwarm reaches 75.38% with 30.32 million. This amounts to a 2.44-point accuracy gain and a 93.9% reduction in prompt tokens. Relative to Chain, G-Designer uses approximately 34% more tokens but improves accuracy by 5.84 points, indicating that graph construction reallocates communication rather than simply minimizing it.

Within the same MMLU experiment, the benefit of increasing team size also varies by graph family. Scaling G-Designer from 5 to 20 agents improves accuracy by 4.62 points, compared with gains of 0.61–2.83 points for Chain, Complete, and GPTSwarm. The constructed graph therefore makes more effective use of additional agents over the reported range.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

Panel B compares the two Graph-of-Agents variants with MoA across six benchmarks [221]. Both variants improve the reported task scores, with gains ranging from approximately 0.9 to 8.8 points. GoA-Mean performs better on GPQA, MATH, and HumanEval, whereas GoA-Max leads on MMLU, MMLU-Pro, and MedMCQA. The preferred aggregation rule consequently varies with the task.

Panels C and D relate these gains to deployment resources. On MMLU-Pro, GoA-Max raises accuracy from 53.33% to 54.78% while reducing calls from 19 to 11, tokens from 56.05k to 19.18k, and latency from 240.26 to 100.43 seconds [221]. The corresponding reductions are approximately 42%, 66%, and 58%. GoA-Mean also improves accuracy, but GoA-Max provides the stronger reported operating point on all four coordinates in this setting. The different reduction rates illustrate that token, call, and latency changes remain distinct effects.

These studies identify two complementary effects of topology optimization. G-Designer shows how the connection pattern changes communication and quality at a fixed team size, while Graph-of-Agents shows how graph organization can shift several deployment coordinates together. The observed gains may arise from improved information paths, fewer model invocations, or changes in coordination; matched graph and resource ablations distinguish these contributions.

7.2. Runtime Findings

Runtime methods control the work realized after an organization has been selected. Their effects occur at different system layers, ranging from the context processed by a model to the state reused by the serving system. Figure 7 separates these effects according to the resource boundary directly changed by each method.

The *Context control* panel compares role-scoped context with alternative tool-planning strategies. Smurfs [17] achieves a 57.4% pass rate on StableToolBench, compared with 55.4% for DFSDT, while reducing tokens per query from 20.7k to 8.1k. Its tokens per tool request also fall from 1,743 to 459. Smurfs therefore improves both reported quality and token use relative to DFSDT. ReAct uses fewer tokens per query at 6.5k, but its pass rate is 13.0 points lower than that of Smurfs. The three operating points illustrate a quality–token trade-off: ReAct minimizes token use, whereas Smurfs obtains higher task success at additional cost.

The *Compact exchange* panel concerns the representation transmitted between agents rather than the context assigned to each role. EcoLANG [123] replaces natural-language interaction with an induced compact communication language. On PHEME, it reduces generated responses by 20.3%, accumulated prompts by 2.6%, and completions by 8.8%. The corresponding reductions on HiSim are 24.7%, 4.7%, and 16.6%. Prompt savings are consistently smaller because compacting a new message does not remove the instructions and earlier history already included in the receiver's context. The comparison with Smurfs thus identifies two different mechanisms: role-scoped context control limits what an agent receives, while compact exchange shortens what agents transmit.

The *Model routing* panel pairs the cost saved by CASTER [110] with its quality change relative to an always-strong-model baseline. On software tasks, CASTER reduces dollar cost by 54.3% but loses 2.5 quality points. Across the other three domains, cost reductions range from 23.4% to 45.3%, while quality changes remain between -0.5 and $+0.7$ points. The largest saving is therefore associated with the largest quality loss, whereas the science and security settings retain modest quality gains at smaller savings. These domain-specific operating points make the value of routing dependent on the quality constraint imposed by the workload.

The *Shared state* panel measures a serving-level intervention. TokenDance [10] reports 11.2× and 17.5× compression of the mirror cache for the two plotted model sizes. Once the shared master cache is included, the corresponding total memory reductions for the illustrated ten-agent deployment are approximately 5.5× and 6.6×. The same study reports system-level maxima of 1.9× faster prefill, 2.3× end-to-end speedup, and 2.7× higher agent concurrency. KVCOMM [213] addresses a related source of overhead by reusing KV state across contexts. These mechanisms reduce duplicated serving work while leaving the logical number of agent decisions unchanged.

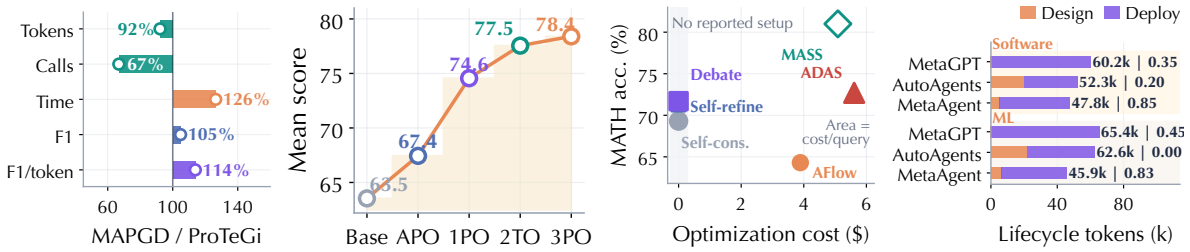
© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

Figure 11 provides two counterfactual views of execution cost. MapCoder [68] serves as a capability-oriented reference: across its ten model–dataset settings, it uses approximately 8.7× to 43.7× as many tokens as direct prompting and makes 8–23 API calls per problem. Its reported accuracy improvement ranges from 2.5% to 135.1%, but does not grow monotonically with token use. Additional inference therefore produces strongly task-dependent returns.

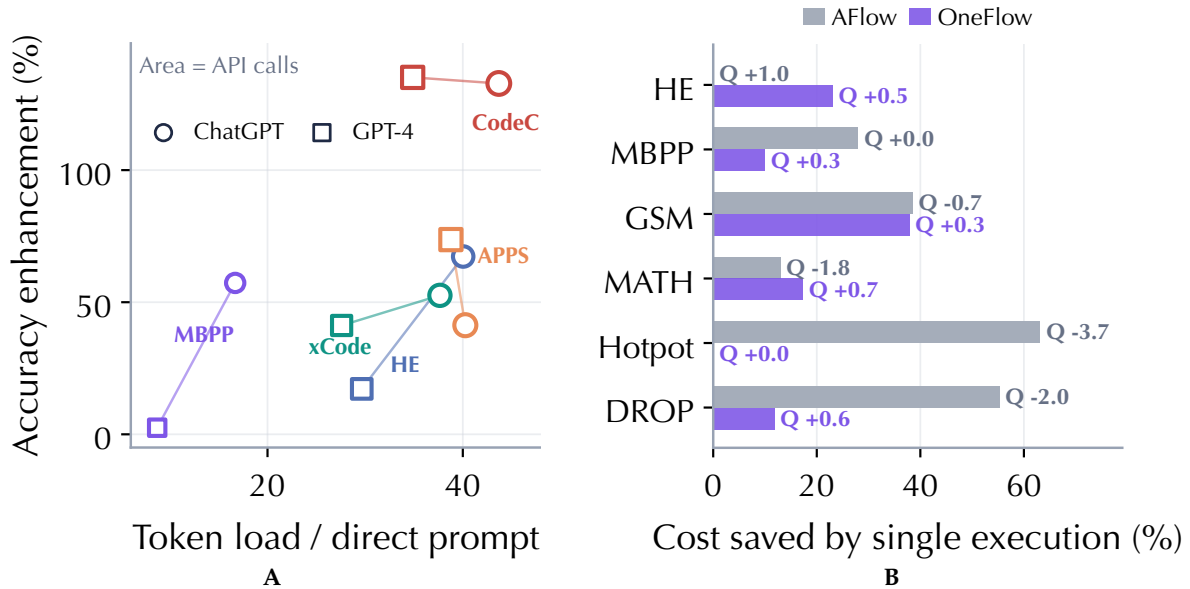
The OneFlow panel [199] provides a single-model, shared-state counterfactual to multi-agent workflow execution. Applying this execution scheme to AFlow-designed workflows saves up to 63.1% in cost, although the largest savings coincide with score losses of 2.0–3.7 points on DROP and HotpotQA. Workflows constructed by OneFlow show a different pattern: all six plotted scores are preserved or improved, with cost savings ranging from 0% to 37.9%. This contrast links the success of shared-state execution to whether the workflow was designed for that execution model.

Across these comparisons, runtime control changes the location and form of computation rather than one common notion of activation. The figures distinguish receiver-side context, routed model expenditure, and reusable serving state, allowing each result to be interpreted on its corresponding quality–resource axis.



(a) Prompt optimization **(b) Stage returns** **(c) Lifecycle trade-off** **(d) Design and deployment**

Figure 10. Optimization gains with setup cost retained. **(a)** MAPGD relative to ProTeGi under the reported 50-query protocol [51]. **(b)** MASS performance after agent-prompt optimization (APO), block-prompt optimization (1PO), topology optimization (2TO), and final workflow-prompt optimization (3PO). **(c)** Reported optimization cost and MATH accuracy for the same study; marker area denotes inference cost per query [248]. **(d)** MetaAgent design and deployment tokens for the reported software and ML task batches; bar-end labels report total tokens and quality [242].



A **B**

Figure 11. Execution cost under paired baselines. **A:** MapCoder results normalized by direct prompting; marker area denotes API calls [68]. **B:** AFlow and OneFlow single-model execution relative to the paired multi-agent workflows; bars show cost savings and Q gives score changes [199].

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

7.3. Optimization Findings

Optimization methods spend resources during development to improve later execution. The resulting deployment trace reflects only part of the cost, since the selected configuration must first be generated and evaluated. Figure 10 presents this trade-off through paper-specific comparisons rather than treating the four panels as a cross-paper ranking.

The *Prompt optimization* panel compares MAPGD [51] with ProTeGi under the reported 50-query protocol. MAPGD reduces token usage from 256k to 236k and model calls from 962 to 643, corresponding to reductions of 7.8% and 33.2%. Average F1 simultaneously rises from 0.83 to 0.87, and the reported normalized F1-per-token measure increases from 3.24 to 3.69. Wall time, however, grows from 159.4 to 201.5 seconds. MAPGD therefore occupies a better quality–token and quality–call operating point, while ProTeGi remains faster in elapsed time.

The *Stage returns* panel follows the ordered optimization pipeline of MASS [248]. Its mean score across eight tasks increases from 63.54 for the initial system to 67.44 after agent-prompt optimization. Block-level prompt optimization produces the largest improvement, raising the score to 74.56. Topology optimization then raises the mean score from 74.56 to 77.55, and final workflow-prompt refinement increases it further to 78.40.

Since these stages are cumulative and executed in a fixed order, the panel measures the marginal return along the reported pipeline. Most of the observed improvement is obtained before the final workflow-level refinement.

The *Lifecycle trade-off* panel relates the final MASS configuration to both development and deployment expenditure. MASS reaches 81.0% MATH accuracy after approximately $5.09 of optimization, with an inference cost of about $0.0014 per query. In the reported comparison, it achieves higher accuracy than ADAS while using slightly less optimization and per-query inference cost. The self-consistency baseline occupies a lower-cost point at approximately $0.0004 per query, but reaches only 71.3% accuracy and has no separately reported optimization expenditure. The panel therefore distinguishes the cost of constructing a workflow from the cost of executing it after construction.

The *Design and deployment* panel provides a token-level view of automatic system construction. On the software workload, MetaAgent [242] uses approximately 5.4k design tokens and 42.4k deployment tokens, reaching a quality score of 0.85. Its combined total of 47.8k tokens is lower than the reported totals for MetaGPT and AutoAgents. On ML Bench, MetaAgent uses 6.2k design tokens and 39.7k deployment tokens, for a total of 45.9k and a quality score of 0.83. Within the paper's accounting, MetaAgent combines higher task quality with lower reported token use than the plotted alternatives, although MetaGPT's human design effort is not represented in token units.

These panels show that optimization changes both the deployed configuration and the stage at which its cost is paid. The central lifecycle variable is reuse: development expenditure can be recovered only across requests for which the learned prompt, workflow, or policy remains applicable. Transfer across tasks, models, and tools therefore determines whether a better deployment operating point becomes an end-to-end efficiency improvement.

7.4. Cross-Layer Effects

The three control points identify a method's primary intervention rather than mutually exclusive system layers. A topology change reshapes the operations available during execution, and the resulting traces may later guide optimization. Optimization may then produce a new graph or runtime controller. We therefore classify each method by what it directly produces, while tracing its effects across the system.

The result figures show this coupling at different stages. Figure 9 demonstrates that graph organization can change task quality and several deployment resources simultaneously. Figure 7 locates runtime savings at distinct boundaries, from model context to serving state. Figure 10 extends the comparison to the development work required before deployment. Read together, the figures connect organizational choice, realized execution, and reusable design.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

Cross-layer methods also make causal attribution more difficult. A graph generator may change both roles and communication links, while workflow search may alter the complete execution policy. The stages reported by MASS [248], for example, are cumulative and executed in a fixed order. Their increments describe the returns along the optimized pipeline rather than independent effects of its components. Without matched component ablations, the complete workflow is the appropriate unit of interpretation.

The empirical results also show how expenditure moves across resource coordinates and lifecycle stages. MAPGD [51] reduces model calls by 33.2% but increases wall time by approximately 26.4%. CASTER [110] reduces model expenditure with workload-dependent changes in quality. At a lower system layer, TokenDance [10] reduces cache memory without changing the logical agent workflow, whereas MetaAgent [242] incurs design tokens before deployment. Cross-layer analysis therefore asks not only how much work is removed, but also where the remaining work is performed.

7.5. Task Dependence

Matched studies show that the preferred system design changes across workloads. In Figure 9, GoA-Mean produces larger gains on GPQA, MATH, and HumanEval, whereas GoA-Max performs better on MMLU, MMLU-Pro, and MedMCQA [221]. The reversal indicates that the aggregation rule interacts with the benchmark rather than providing a fixed advantage. A similar pattern appears in model routing. CASTER [110] obtains its largest cost reduction on software tasks, where it also incurs its largest quality loss. Its science and security results retain small quality gains at lower savings. A broader analysis of 260 MAS configurations further finds that scaling behavior changes with the model and coordination setting [82]. The preferred configuration is therefore determined by the specific workload and quality constraint.

Task dependence also changes which resource becomes the main bottleneck. In tool planning, Smurfs [17] improves pass rate while reducing the context processed relative to DFSDT. Coding workflows expose a different trade-off: the MapCoder comparison [68] shows that additional planning and repair can substantially increase both calls and tokens. In concurrency-bound deployments, TokenDance [10] instead targets duplicated cache state at the serving layer. Method selection should therefore follow the measured bottleneck of the target workload rather than its broad task category.

7.6. Evidence Limits

The strongest evidence in the current literature comes from matched comparisons within individual studies. These comparisons show that topology design can remove redundant interaction, runtime control can reduce selected execution work, and reusable optimization can improve later operating points. They establish meaningful quality–resource improvements under particular experimental conditions, but do not provide a common scale for ranking the three method families.

Causal attribution becomes more difficult when an intervention changes several components simultaneously. Learned topology may modify both team composition and communication, while staged workflow search updates multiple parts of the execution policy. The improvements reported by MASS [248], for example, follow a cumulative optimization pipeline rather than independent component ablations. MetaAgent [242] presents a different accounting issue: automatic design tokens are measured, whereas the human effort used to construct comparison systems is not represented in the same unit. These studies provide evidence for complete systems, with a narrower basis for attributing gains to individual components.

Reporting conventions create an additional barrier to aggregation. Most studies measure only a subset of $\boldsymbol{c}$, and C_{design} is included less consistently than deployment cost. The figures in this survey therefore preserve each study's baseline, workload, and accounting unit. They summarize within-study movements in quality and resources instead of combining paper-specific percentages into a single efficiency score. The unresolved questions of standardized accounting, causal attribution, and cross-setting transfer motivate the open problems discussed in Section 8.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

8. Open Problems

The preceding sections identify five priorities for future research, which will be discussed in the following subsections. Comparable evaluation and end-to-end accounting concern whether an efficiency gain can be measured consistently. Adaptive control and transfer under drift ask whether that gain can be preserved as requests, models, and environments change. Causal attribution concerns which component of an MAS actually produces the observed improvement. The following subsections formulate these priorities as concrete research questions and outline evaluation protocols that can be implemented with current models and benchmarks.

8.1. Comparable Evaluation

Comparable evaluation requires consistent budgets, baselines, and accounting boundaries. A fixed thinking-token limit does not ensure equal total tokens or model calls [169]. Results from large configuration studies also vary with the underlying model and task [82]. Comparisons that leave these factors uncontrolled cannot determine whether an observed gain comes from the efficiency mechanism or the experimental setup.

The baseline should correspond to the claim being tested. A method designed to improve an existing workflow can be evaluated against the unmodified MAS. A broader claim about collaboration requires a single-controller baseline with comparable access to models, information, and tools. Such a baseline may still use repeated sampling [183], tool interaction [210], or search [83] when these capabilities are available to the MAS.

Existing benchmarks provide the components needed for a shared protocol. AgentBench [111] supplies interactive environments, while AgentBoard [116] supports trajectory-level analysis. Web and desktop interaction is covered by Mind2Web [30], WebArena [249], WorkArena [32], and OSWorld [197]. Controlled MAS comparisons are supported by MultiAgentBench [253], BattleAgentBench [182], and MALLM [5].

A practical evaluation protocol could select representative tasks from these suites and use a fixed model pool to compare topology, runtime, and optimization methods. Each run should release the observed task utility Q, the relevant coordinates of $\boldsymbol{c}$, and C_{design} when offline construction is involved. Repeated runs across resource budgets would then produce quality–resource curves with uncertainty. These curves, indexed by task and model setting, would provide the common empirical basis needed for comparing efficiency mechanisms.

8.2. Adaptive Control

The marginal value of collaboration varies across requests and may change during execution. A hybrid SAS–MAS cascade demonstrates request-level escalation [42]. Related mechanisms appear in FrugalGPT [19], preference-trained model routing [130], and the next-agent policy of AnyMAC [181]. The open problem is to predict whether the expected utility of another collaborative action exceeds its incremental resource cost. This decision should be repeated during execution so that an active team can continue, escalate, or stop as new information becomes available.

Verification provides a concrete setting for this controller. Structured critics [131] and the evidence-grounded checker in MARCH [101] offer additional reliability at an added execution cost. SupervisorAgent [103] instead invokes supervision selectively. A practical experiment should execute the same requests under the learned controller, fixed always-on and never-on policies, and an oracle derived from counterfactual outcomes. All variants should receive the same verifier and evidence access. Comparing predicted and realized gains would measure controller calibration, while regret and gate overhead would quantify the cost of incorrect decisions. This protocol directly tests whether adaptive control concentrates collaboration on requests for which its realized benefit exceeds its added cost.

8.3. Transfer Under Drift

Reusable optimization is efficient only when its output remains useful beyond the setting in which it was constructed. MASS [248] and MAPRO [243] jointly optimize prompts and workflow

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

structure. MASPOB [62] and MASPO [189] instead use system-level feedback to improve prompts. TopoPrior [237] and OFA-MAS [98] move toward organizations that can be reused across tasks or domains. The open problem is whether these configurations remain effective after changes to the model pool, tool interface interface, or workload distribution. Changes in deployment prices may further alter which configuration is economically preferable.

Transfer under drift should be studied over evolving deployment conditions rather than a single source–target split. The central question is how long an optimized configuration preserves its quality–resource advantage and how much re-optimization is required when that advantage degrades. Evaluation should therefore distinguish direct transfer from continual reconstruction and include every update in the lifecycle cost. Break-even analysis must also extend across the deployment horizon, since repeated adaptation can gradually offset the savings obtained from reuse.

8.4. Causal Attribution

Current endpoint results rarely identify why an organization works. Improvements may come from the graph, the prompts attached to its roles, unequal model access, or additional samples. Formal analyses of decomposition motivate modular execution under restricted assumptions [191], while code systems such as AgentCoder and SWE-agent show that interfaces and verifiers strongly affect practical decomposition [66,204]. Information-propagation analysis and GEMMAS provide structural diagnostics [87,154], and MALLM exposes configurable personas and decision protocols [5]. These tools do not yet connect structure to a causal effect on quality and cost.

Paired trace interventions offer a feasible path forward. Researchers can freeze agent outputs and replay them through another graph, remove or paraphrase one message, randomize an edge at matched degree, or let one controller execute the same decomposition and tools. Calls and token budgets should remain fixed during the intervention. Such experiments would reveal whether a method benefits from information routing, compute allocation, role instructions, or decomposition itself, and whether the learned organization is stable across seeds and model replacement.

8.5. End-to-End Cost

Application-level tokens describe only part of deployment. A sparse message graph can still duplicate prefixes and KV state, while a token-heavy parallel graph may finish quickly at high peak demand. TokenDance studies collective KV-cache sharing and Token Coherence models artifact synchronization [10,134]. Long-lived systems add provenance and maintenance: Generative Agents and Voyager illustrate persistent memory and reusable skills [135,175], while RCR-Router selects role- and stage-specific memory views [107]. The unresolved systems problem is to optimize logical work, state reuse, and scheduling jointly without losing traceability.

This boundary becomes more important when agents act through tools or physical environments. Cooperative embodied systems expose observation and action dependencies [119,230]; ReAd measures interaction steps as well as LLM-query rounds [241]; and SOTOPIA requires social outcomes beyond task accuracy [251]. Adversarial workflows add another real cost: prompt propagation, indirect injection, and tool exploitation require detection and recovery [28,36,37,86,223,254].

A concrete evaluation can publish the logical trace separately from the serving schedule, replay the same workflow under different concurrency and cache conditions, and retain tool, recovery, and defense events in the trace. This would show which savings transfer across hardware and which depend on one deployment. For persistent or adversarial workloads, provenance retention, stale-state failures, attack reachability, false alarms, and recovery cost should appear on the same quality–resource frontier as ordinary inference.

9. Conclusion

This survey organized prior work on efficient LLM-based multi-agent systems by the primary control point of each method. Across these families, efficiency depends on selective collaboration:

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

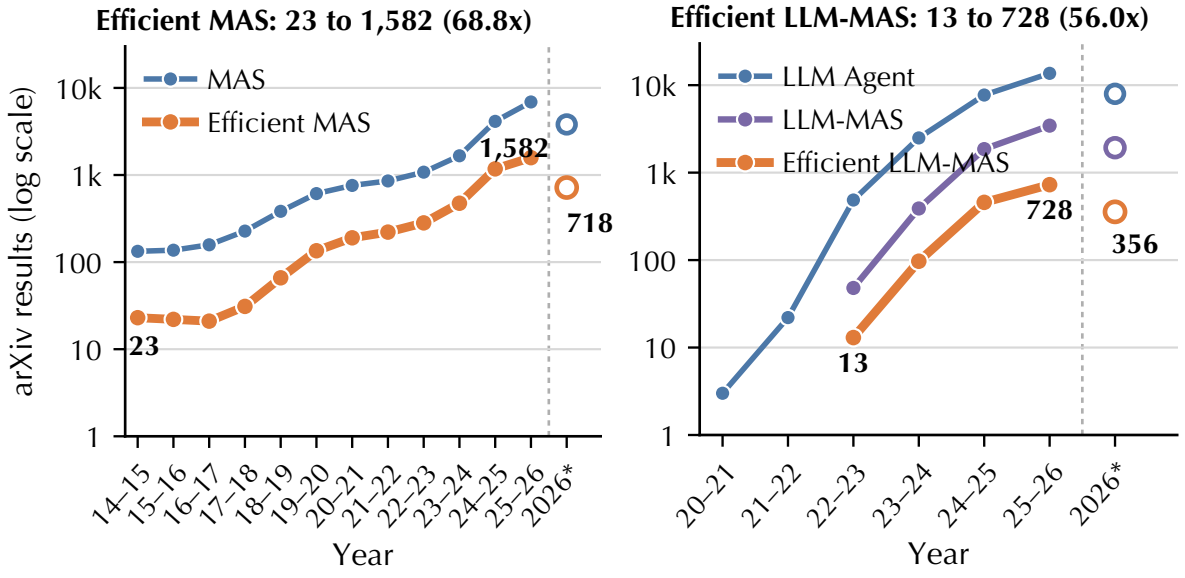
Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

redundant computation should be removed without disrupting the information paths that create value, while control overhead and offline investment must remain part of the reported cost.

The scale of the literature further shows why efficiency has become a distinct research problem. Figure 12 reports that the number of results for "Efficient Multi-Agent System" increased from 23 in 2014–15 to 1,582 in 2025–26. The corresponding "Efficient LLM Multi-Agent System" query rose from 13 results in 2022–23 to 728 in 2025–26. These keyword searches are non-exclusive and should not be interpreted as unique-paper counts or precise literature shares. Nevertheless, their absolute growth reflects the rapid expansion of research on the resource demands of multi-agent organization.

Despite this expansion, current evidence remains too heterogeneous to support a universal ranking of efficiency methods. Most studies establish gains within particular workflows and workloads, while differences in baselines and accounting boundaries limit cross-paper comparisons. The appropriate reference may be a capable single-agent system, a fixed multi-agent workflow, or an always-on version of the proposed mechanism, depending on the claimed contribution. Reliable comparison therefore requires matched access to underlying capabilities and explicit reporting of both deployment and development costs.

Returning to the question posed in Section 1, prior work identifies concrete conditions under which collaboration can justify its cost and several mechanisms for improving that trade-off. It does not yet provide a general rule that predicts the best organization across tasks and model pools. An efficient multi-agent system is therefore not simply one that uses fewer agents, but one whose collaborative contribution and associated cost remain measurable under its intended operating conditions.



(a) MAS literature **(b) LLM-agent literature**

Figure 12. Growth of efficiency-oriented MAS research. ArXiv keyword-search results are shown on logarithmic scales. The 2026 observations are incomplete and displayed as detached points; search sets may overlap.

References

1. J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai, M. Terry, Q. Le, and C. Sutton. Program Synthesis with Large Language Models, 2021. URL https://arxiv.org/abs/2108.07732. arXiv preprint.
2. J. Bai and L. Shi. MAS-PromptBench: When Does Prompt Optimization Improve Multi-Agent LLM Systems?, 2026. URL https://arxiv.org/abs/2606.23664. arXiv preprint.
3. L. Bai, Z. Huang, X. Wang, J. Sun, R. Mihalcea, E. Brynjolfsson, A. Pentland, and J. Pei. How Do AI Agents Spend Your Money? Analyzing and Predicting Token Consumption in Agentic Coding Tasks, 2026. URL https://arxiv.org/abs/2604.22750. arXiv preprint.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

4. Y. Bai, X. Lv, J. Zhang, H. Lyu, J. Tang, Z. Huang, Z. Du, X. Liu, A. Zeng, L. Hou, Y. Dong, J. Tang, and J. Li. LongBench: A Bilingual, Multitask Benchmark for Long Context Understanding, 2024. URL https://arxiv.org/abs/2308.14508. Annual Meeting of the Association for Computational Linguistics (ACL-24).
5. J. Becker, L. B. Kaesberg, N. Bauer, J. P. Wahle, T. Ruas, and B. Gipp. MALLM: Multi-Agent Large Language Models Framework, 2025. URL https://arxiv.org/abs/2509.11656. Conference on Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP Demos-25).
6. J. Becker, J. P. Wahle, T. Ruas, and B. Gipp. Misinformation Propagation in Benign Multi-Agent Systems, 2026. URL https://arxiv.org/abs/2606.16710. arXiv preprint.
7. D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The Complexity of Decentralized Control of Markov Decision Processes, 2002. URL https://doi.org/10.1287/moor.27.4.819.297. Mathematics of Operations Research.
8. B. Bertalanič and C. Fortuna. The Ringelmann Effect in Multi-Agent LLM Systems: A Scaling Law for Effective Team Size, 2026. URL https://arxiv.org/abs/2606.02646. arXiv preprint.
9. I. Besrour, J. He, T. Schreieder, and M. Färber. RAGentA: Multi-Agent Retrieval-Augmented Generation for Attributed Question Answering, 2025. URL https://arxiv.org/abs/2506.16988. ACM SIGIR Conference on Research and Development in Information Retrieval, LiveRAG Challenge (SIGIR LiveRAG-25).
10. Z. Bian, F. Wu, C. Zhang, H. Dong, Y. Liang, and Y. Zhuo. TokenDance: Scaling Multi-Agent LLM Serving via Collective KV Cache Sharing, 2026. URL https://arxiv.org/abs/2604.03143. arXiv preprint.
11. M. Carroll, R. Shah, M. K. Ho, T. L. Griffiths, S. A. Seshia, P. Abbeel, and A. Dragan. On the Utility of Learning about Humans for Human-AI Coordination, 2019. URL https://arxiv.org/abs/1910.05789. Conference on Neural Information Processing Systems (NeurIPS-19).
12. M. Cemri, M. Z. Pan, S. Yang, L. A. Agrawal, B. Chopra, R. Tiwari, K. Keutzer, A. Parameswaran, D. Klein, K. Ramchandran, M. Zaharia, J. E. Gonzalez, and I. Stoica. Why Do Multi-Agent LLM Systems Fail?, 2025. URL https://arxiv.org/abs/2503.13657. Conference on Neural Information Processing Systems (NeurIPS-25).
13. C.-M. Chan, W. Chen, Y. Su, J. Yu, W. Xue, S. Zhang, J. Fu, and Z. Liu. ChatEval: Towards Better LLM-based Evaluators through Multi-Agent Debate, 2023. URL https://arxiv.org/abs/2308.07201. arXiv preprint.
14. C.-Y. Chang, Z. Jiang, V. Rakesh, M. Pan, C.-C. M. Yeh, G. Wang, M. Hu, Z. Xu, Y. Zheng, M. Das, and N. Zou. MAIN-RAG: Multi-Agent Filtering Retrieval-Augmented Generation, 2024. URL https://arxiv.org/abs/2501.00332. Annual Meeting of the Association for Computational Linguistics (ACL-25).
15. G. Chang, E. Lin, C. Yuan, R. Cai, B. Chen, X. Xie, and Y. Zhang. Agent Network Protocol Technical White Paper, 2025. URL https://arxiv.org/abs/2508.00007. arXiv preprint.
16. H. Chen, X. Zheng, Y. Liu, P. Jiao, S. Li, H. Liu, Z. Zhao, Z. Xu, I. Khalil, and S. Pan. GoAgent: Group-of-Agents Communication Topology Generation for LLM-based Multi-Agent Systems, 2026. URL https://arxiv.org/abs/2603.19677. arXiv preprint.
17. J. Chen, J. Liang, and B. Wang. Smurfs: Multi-Agent System using Context-Efficient DFSDT for Tool Planning, 2024. URL https://arxiv.org/abs/2405.05955. Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics (NAACL-25).
18. J. C.-Y. Chen, S. Saha, and M. Bansal. ReConcile: Round-Table Conference Improves Reasoning via Consensus among Diverse LLMs, 2023. URL https://arxiv.org/abs/2309.13007. Annual Meeting of the Association for Computational Linguistics (ACL-24).
19. L. Chen, M. Zaharia, and J. Zou. FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance, 2023. URL https://arxiv.org/abs/2305.05176. Transactions on Machine Learning Research (TMLR).
20. M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba. Evaluating Large Language Models Trained on Code, 2021. URL https://arxiv.org/abs/2107.03374. arXiv preprint.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

21. W. Chen, J. Yuan, C. Qian, C. Yang, Z. Liu, and M. Sun. Optima: Optimizing Effectiveness and Efficiency for LLM-Based Multi-Agent System, 2024. URL https://arxiv.org/abs/2410.08115. Annual Meeting of the Association for Computational Linguistics (ACL-25).
22. X. Chen, Y. Liu, H. Wei, and K. Ding. LEMON: Learning Executable Multi-Agent Orchestration via Counterfactual Reinforcement Learning, 2026. URL https://arxiv.org/abs/2605.14483. arXiv preprint.
23. Y. Chen, J. Chen, C. He, Y. Li, Y. Ji, Y. Wu, D. Yang, L. Diao, L. Shou, H. Zhang, H. Li, and G. Chen. Token Economics for LLM Agents: A Dual-View Study from Computing and Economics, 2026. URL https://arxiv.org/abs/2605.09104. arXiv preprint.
24. H. Chi, Z. Feng, X. Yu, L. Luo, Y.-S. Ong, I. Tsang, H. Chen, Y. Chang, and H. Yin. EvoCF: Multi-Agent Collaboration via Agentic Memory-Driven Evolutionary Counterfactual Planning, 2026. URL https://openreview.net/forum?id=zGKkewtb2w. International Conference on Learning Representations Workshop on Multi-Agent Learning and Its Opportunities in the Era of Generative AI (ICLR MALGAI-26).
25. Y. J. Chun and I. Ahmed. What Do Agents Communicate? Characterizing Information Exchange in Multi-Agent Systems, 2026. URL https://arxiv.org/abs/2605.20548. arXiv preprint.
26. P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. Think You Have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge, 2018. URL https://arxiv.org/abs/1803.05457. arXiv preprint.
27. K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman. Training Verifiers to Solve Math Word Problems, 2021. URL https://arxiv.org/abs/2110.14168. arXiv preprint.
28. E. Debenedetti, J. Zhang, M. Balunović, L. Beurer-Kellner, M. Fischer, and F. Tramèr. AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents, 2024. URL https://arxiv.org/abs/2406.13352. Conference on Neural Information Processing Systems (NeurIPS-24).
29. S. Deng, H. Zhao, Z. Wang, W. Qian, X. Ao, G. Cheng, J. Yin, A. Y. Zomaya, and S. Dustdar. Agentic Services Computing, 2025. URL https://arxiv.org/abs/2509.24380. arXiv preprint.
30. X. Deng, Y. Gu, B. Zheng, S. Chen, S. Stevens, B. Wang, H. Sun, and Y. Su. Mind2Web: Towards a Generalist Agent for the Web, 2023. URL https://arxiv.org/abs/2306.06070. Conference on Neural Information Processing Systems (NeurIPS-23).
31. J. Dong and W. Lin. BANDMAS: Causality-Inspired Semantic Packet Scheduling for Bandwidth-Efficient Multi-Agent Collaboration, 2026. URL https://arxiv.org/abs/2608.00458. arXiv preprint.
32. A. Drouin, M. Gasse, M. Caccia, I. H. Laradji, M. D. Verme, T. Marty, L. Boisvert, M. Thakkar, Q. Cappart, D. Vazquez, N. Chapados, and A. Lacoste. WorkArena: How Capable Are Web Agents at Solving Common Knowledge Work Tasks?, 2024. URL https://arxiv.org/abs/2403.07718. International Conference on Machine Learning (ICML-24).
33. Y. Du, S. Li, A. Torralba, J. B. Tenenbaum, and I. Mordatch. Improving Factuality and Reasoning in Language Models through Multiagent Debate, 2023. URL https://arxiv.org/abs/2305.14325. International Conference on Machine Learning (ICML-24).
34. A. Ehtesham, A. Singh, G. K. Gupta, and S. Kumar. A survey of agent interoperability protocols: Model Context Protocol (MCP), Agent Communication Protocol (ACP), Agent-to-Agent Protocol (A2A), and Agent Network Protocol (ANP), 2025. URL https://arxiv.org/abs/2505.02279. arXiv preprint.
35. W. Fan, T. Tognoli, H. P. Zou, C. Miao, Y. Wang, and X. Zhang. TodyComm: Task-Oriented Dynamic Communication for Multi-Round LLM-based Multi-Agent System, 2026. URL https://arxiv.org/abs/2602.03688. arXiv preprint.
36. R. Fang, R. Bindu, A. Gupta, and D. Kang. LLM Agents can Autonomously Exploit One-day Vulnerabilities, 2024. URL https://arxiv.org/abs/2404.08144. arXiv preprint.
37. R. Fang, R. Bindu, A. Gupta, Q. Zhan, and D. Kang. LLM Agents can Autonomously Hack Websites, 2024. URL https://arxiv.org/abs/2402.06664. arXiv preprint.
38. L. Feng, L. Zheng, S. He, F. Zhang, and B. An. Dr. MAS: Stable Reinforcement Learning for Multi-Agent LLM Systems, 2026. URL https://arxiv.org/abs/2602.08847. arXiv preprint.
39. M. Fu, X. Xue, Y. Li, Z. He, S. Huang, X. Qu, Y. Cheng, and Y. Yang. LatentMem: Customizing Latent Memory for Multi-Agent Systems, 2026. URL https://arxiv.org/abs/2602.03036. arXiv preprint.
40. T. Fu, Z. Min, H. Zhang, J. Yan, G. Dai, W. Ouyang, and Y. Wang. Cache-to-Cache: Direct Semantic Communication Between Large Language Models, 2025. URL https://arxiv.org/abs/2510.03215. International Conference on Learning Representations (ICLR-26).

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

41. S. Gandhi, M. Patwardhan, L. Vig, and G. Shroff. BudgetMLAgent: A Cost-Effective LLM Multi-Agent system for Automating Machine Learning Tasks, 2024. URL https://arxiv.org/abs/2411.07464. International Conference on AI-ML Systems (AIMLSystems-24).
42. M. Gao, Y. Li, B. Liu, Y. Yu, P. Wang, C.-Y. Lin, and F. Lai. Single-agent or Multi-agent Systems? Why Not Both?, 2025. URL https://arxiv.org/abs/2505.18286. arXiv preprint.
43. W. Gao, X. Ding, H. Liu, and T. Xing. Where Reasoning Diverges: Localized Multi-Agent Debate for Multi-Hop Question Answering, 2026. URL https://arxiv.org/abs/2608.01463. arXiv preprint.
44. M. Geva, D. Khashabi, E. Segal, T. Khot, D. Roth, and J. Berant. Did Aristotle Use a Laptop? A Question Answering Benchmark with Implicit Reasoning Strategies, 2021. URL https://arxiv.org/abs/2101.02235. Transactions of the Association for Computational Linguistics (TACL), 2021.
45. W. Gou and Z. Liu. Dynamic Trust-Aware Sparse Communication Topology for LLM-Based Multi-Agent Consensus, 2026. URL https://arxiv.org/abs/2606.01828. arXiv preprint.
46. T. Guo, X. Chen, Y. Wang, R. Chang, S. Pei, N. Chawla, O. Wiest, and X. Zhang. Large Language Model based Multi-Agents: A Survey of Progress and Challenges, 2024. URL https://arxiv.org/abs/2402.01680. International Joint Conference on Artificial Intelligence (IJCAI-24).
47. X. Guo, S. Wang, C. Ji, X. Zhao, W. Xi, Y. Liu, Q. Li, C. Deng, and J. Feng. Towards Generalized Routing: Model and Agent Orchestration for Adaptive and Efficient Inference, 2025. URL https://arxiv.org/abs/2509.07571. arXiv preprint.
48. Z. Guo, S. Cheng, H. Wang, S. Liang, Y. Qin, P. Li, Z. Liu, M. Sun, and Y. Liu. StableToolBench: Towards Stable Large-Scale Benchmarking on Tool Learning of Large Language Models, 2024. URL https://aclanthology.org/2024.findings-acl.664/. Findings of the Association for Computational Linguistics: ACL 2024 (ACL Findings-24).
49. D. Han, C. Couturier, D. M. Diaz, X. Zhang, V. Rühle, and S. Rajmohan. LEGOMem: Modular Procedural Memory for Multi-agent LLM Systems for Workflow Automation, 2025. URL https://arxiv.org/abs/2510.04851. International Conference on Autonomous Agents and Multiagent Systems (AAMAS-26).
50. S. Han, Q. Zhang, Y. Yao, W. Jin, Z. Xu, and C. He. LLM Multi-Agent Systems: Challenges and Open Problems, 2024. URL https://arxiv.org/abs/2402.03578. arXiv preprint.
51. Y. Han, Y. Han, S. Huang, G. Liu, Z. Zhou, B. Liu, Y. Zhang, I. N. Shi, L. He, and T. Shi. MAPGD: Multi-Agent Prompt Gradient Descent for Collaborative Prompt Optimization, 2025. URL https://arxiv.org/abs/2509.11361. arXiv preprint.
52. X. Hao, Z. Wu, Y.-X. Qiu, C. Xiao, R. Xu, S. Zheng, and J. Qin. Not All Flips Are Conformity: Decomposing Stance Convergence in Multi-Agent LLM Debate, 2026. URL https://arxiv.org/abs/2606.00820. arXiv preprint.
53. Z. Hao, T. Wang, H. Dong, Z. Liu, H. Wang, X. Lin, Q. Lin, C. Wang, H. Dong, and J. Chen. Evolve as a Team: Collaborative Self-Evolution for LLM-based Multi-Agent Systems, 2026. URL https://arxiv.org/abs/2605.29790. arXiv preprint.
54. C. He, R. Luo, Y. Bai, S. Hu, Z. L. Thai, J. Shen, J. Hu, X. Han, Y. Huang, Y. Zhang, J. Liu, L. Qi, Z. Liu, and M. Sun. OlympiadBench: A Challenging Benchmark for Promoting AGI with Olympiad-Level Bilingual Multimodal Scientific Problems, 2024. URL https://arxiv.org/abs/2402.14008. Annual Meeting of the Association for Computational Linguistics (ACL-24).
55. L. He, L. He, D. He, and C. Li. Conflict-Resilient Multi-Agent Reasoning via Signed Graph Modeling, 2026. URL https://arxiv.org/abs/2605.19418. arXiv preprint.
56. D. Hendrycks, S. Basart, S. Kadavath, M. Mazeika, A. Arora, E. Guo, C. Burns, S. Puranik, H. He, D. Song, and J. Steinhardt. Measuring Coding Challenge Competence With APPS, 2021. URL https://arxiv.org/abs/2105.09938. Conference on Neural Information Processing Systems (NeurIPS-21), Datasets and Benchmarks Track.
57. D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. Measuring Massive Multitask Language Understanding, 2021. URL https://arxiv.org/abs/2009.03300. International Conference on Learning Representations (ICLR-21).
58. D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt. Measuring Mathematical Problem Solving With the MATH Dataset, 2021. URL https://arxiv.org/abs/2103.03874. Conference on Neural Information Processing Systems (NeurIPS-21), Datasets and Benchmarks Track.
59. X. Ho, A.-K. D. Nguyen, S. Sugawara, and A. Aizawa. Constructing A Multi-hop QA Dataset for Comprehensive Evaluation of Reasoning Steps, 2020. URL https://arxiv.org/abs/2011.01060. International Conference on Computational Linguistics (COLING-20).

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

60. S. Holt, M. R. Luyten, and M. van der Schaar. L2MAC: Large Language Model Automatic Computer for Extensive Code Generation, 2023. URL https://arxiv.org/abs/2310.02003. International Conference on Learning Representations (ICLR-24).
61. S. Hong, M. Zhuge, J. Chen, X. Zheng, Y. Cheng, C. Zhang, J. Wang, Z. Wang, S. K. S. Yau, Z. Lin, L. Zhou, C. Ran, L. Xiao, C. Wu, and J. Schmidhuber. MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework, 2023. URL https://arxiv.org/abs/2308.00352. arXiv preprint.
62. Z. Hong, Q. Zhang, J. Sun, Z. Shang, M. Kong, X. Wang, Y. Shu, and Z. Dai. MASPOB: Bandit-Based Prompt Optimization for Multi-Agent Systems with Graph Neural Networks, 2026. URL https://arxiv.org/abs/2603.02630. International Conference on Machine Learning (ICML-26).
63. S. Hu, L. Shen, Y. Zhang, and D. Tao. Learning Multi-Agent Communication from Graph Modeling Perspective, 2024. URL https://arxiv.org/abs/2405.08550. International Conference on Learning Representations (ICLR-24).
64. Y. Hu, Y. Cai, Y. Du, X. Zhu, X. Liu, Z. Yu, Y. Hou, S. Tang, and S. Chen. Self-Evolving Multi-Agent Collaboration Networks for Software Development, 2024. URL https://arxiv.org/abs/2410.16946. International Conference on Learning Representations (ICLR-25).
65. W. Hua, L. Fan, L. Li, K. Mei, J. Ji, Y. Ge, L. Hemphill, and Y. Zhang. War and Peace (WarAgent): Large Language Model-based Multi-Agent Simulation of World Wars, 2023. URL https://arxiv.org/abs/2311.17227. arXiv preprint.
66. D. Huang, J. M. Zhang, M. Luck, Q. Bu, Y. Qing, and H. Cui. AgentCoder: Multi-Agent-based Code Generation with Iterative Testing and Optimisation, 2023. URL https://arxiv.org/abs/2312.13010. arXiv preprint.
67. J. Huang, Z. Zhang, K. Shi, Y. Ye, and C. Zhang. EvolveRouter: Co-Evolving Routing and Prompt for Multi-Agent Question Answering, 2026. URL https://arxiv.org/abs/2604.05149. arXiv preprint.
68. M. A. Islam, M. E. Ali, and M. R. Parvez. MapCoder: Multi-Agent Code Generation for Competitive Problem Solving, 2024. URL https://arxiv.org/abs/2405.11403. Annual Meeting of the Association for Computational Linguistics (ACL-24).
69. N. Jain, K. Han, A. Gu, W.-D. Li, F. Yan, T. Zhang, S. Wang, A. Solar-Lezama, K. Sen, and I. Stoica. LiveCodeBench: Holistic and Contamination Free Evaluation of Large Language Models for Code, 2025. URL https://arxiv.org/abs/2403.07974. International Conference on Learning Representations (ICLR-25).
70. H. Jeon, H. Ha, and J.-J. Kim. LRAgent: Efficient KV Cache Sharing for Multi-LoRA LLM Agents, 2026. URL https://arxiv.org/abs/2602.01053. International Conference on Machine Learning (ICML-26).
71. D. Jiang, X. Ren, and B. Y. Lin. LLM-Blender: Ensembling Large Language Models with Pairwise Ranking and Generative Fusion, 2023. URL https://arxiv.org/abs/2306.02561. Annual Meeting of the Association for Computational Linguistics (ACL-23).
72. E. H. Jiang, M. Li, G. Wan, S. Yin, Y. Wu, X. Liang, X. Li, Y. Sun, W. Wang, K.-W. Chang, and Y. N. Wu. Dynamic Generation of Multi-LLM Agents Communication Topologies with Graph Diffusion Models, 2025. URL https://arxiv.org/abs/2510.07799. Annual Meeting of the Association for Computational Linguistics (ACL-26).
73. E. H. Jiang, L. Li, R. Sun, X. Liang, Y. Li, Y. Wu, H. Luo, H. Li, Z. Zhang, Z. Kang, K.-W. Chang, and Y. N. Wu. Agent Q-Mix: Selecting the Right Action for LLM Multi-Agent Systems through Reinforcement Learning, 2026. URL https://arxiv.org/abs/2604.00344. arXiv preprint.
74. Y. Jiang, J. Yuan, C. Tu, P. Ye, and T. Chen. LSTM-MAS: A Long Short-Term Memory Inspired Multi-Agent System for Long-Context Understanding, 2026. URL https://arxiv.org/abs/2601.11913. arXiv preprint.
75. C. E. Jimenez, J. Yang, A. Wettig, S. Yao, K. Pei, O. Press, and K. Narasimhan. SWE-bench: Can Language Models Resolve Real-World GitHub Issues?, 2024. URL https://arxiv.org/abs/2310.06770. International Conference on Learning Representations (ICLR-24).
76. B. Jin, T. Collins, D. Yu, M. Cemri, S. Zhang, M. Li, J. Tang, T. Qin, Z. Xu, J. Lu, G. Yin, J. Han, and Z. Wang. Controlling Performance and Budget of a Centralized Multi-agent LLM System with Reinforcement Learning, 2025. URL https://arxiv.org/abs/2511.02755. arXiv preprint.
77. T. Joo, S. Ishida, I. Sosnovik, B. Lim, S. Rezaei-Shoshtari, A. Gaier, and R. Giaquinto. Graph of Agents: Principled Long Context Modeling by Emergent Multi-Agent Collaboration, 2025. URL https://arxiv.org/abs/2509.21848. arXiv preprint.
78. M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension, 2017. URL https://arxiv.org/abs/1705.03551. Annual Meeting of the Association for Computational Linguistics (ACL-17).

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

79. L. B. Kaesberg, J. Becker, J. P. Wahle, T. Ruas, and B. Gipp. Voting or Consensus? Decision-Making in Multi-Agent Debate, 2025. URL https://arxiv.org/abs/2502.19130. Findings of the Association for Computational Linguistics: ACL 2025 (ACL Findings-25).
80. Z. Kang, X. Zhao, and D. Song. Scalable Best-of-N Selection for Large Language Models via Self-Certainty, 2025. URL https://arxiv.org/abs/2502.18581. Conference on Neural Information Processing Systems (NeurIPS-25).
81. T. Khot, H. Trivedi, M. Finlayson, Y. Fu, K. Richardson, P. Clark, and A. Sabharwal. Decomposed Prompting: A Modular Approach for Solving Complex Tasks, 2022. URL https://arxiv.org/abs/2210.02406. International Conference on Learning Representations (ICLR-23).
82. Y. Kim, K. Gu, C. Park, C. Park, S. Schmidgall, A. A. Heydari, Y. Yan, Z. Zhang, Y. Zhuang, Y. Liu, M. Malhotra, P. P. Liang, H. W. Park, Y. Yang, X. Xu, Y. Du, S. Patel, T. Althoff, D. McDuff, and X. Liu. Towards a Science of Scaling Agent Systems, 2025. URL https://arxiv.org/abs/2512.08296. arXiv preprint.
83. J. Y. Koh, S. McAleer, D. Fried, and R. Salakhutdinov. Tree Search for Language Model Agents, 2024. URL https://arxiv.org/abs/2407.01476. Transactions on Machine Learning Research (TMLR).
84. B. Kriuk and L. Ng. Q-KVComm: Efficient Multi-Agent Communication Via Adaptive KV Cache Compression, 2025. URL https://arxiv.org/abs/2512.17914. International Conference on Multi-Agent Systems for Collaborative Intelligence (ICMSCI-26).
85. T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, K. Toutanova, L. Jones, M. Kelcey, M.-W. Chang, A. M. Dai, J. Uszkoreit, Q. Le, and S. Petrov. Natural Questions: A Benchmark for Question Answering Research, 2019. URL https://aclanthology.org/Q19-1026/. Transactions of the Association for Computational Linguistics (TACL), 2019.
86. D. Lee and M. Tiwari. Prompt Infection: LLM-to-LLM Prompt Injection within Multi-Agent Systems, 2024. URL https://arxiv.org/abs/2410.07283. arXiv preprint.
87. J. Lee, R. Chang, D. Kwon, H. Singh, and N. Verma. GEMMAS: Graph-based Evaluation Metrics for Multi Agent Systems, 2025. URL https://arxiv.org/abs/2507.13190. arXiv preprint.
88. H. Y. Leong, Y. Li, Y. Wu, W. Ouyang, W. Zhu, J. Gao, and W. Han. AMAS: Adaptively Determining Communication Topology for LLM-based Multi-Agent System, 2025. URL https://arxiv.org/abs/2510.01617. Conference on Empirical Methods in Natural Language Processing (EMNLP-25).
89. B. Li, Z. Zhao, D.-H. Lee, and G. Wang. Adaptive Graph Pruning for Multi-Agent Communication, 2025. URL https://arxiv.org/abs/2506.02951. European Conference on Artificial Intelligence (ECAI-25).
90. G. Li, H. A. A. K. Hammoud, H. Itani, D. Khizbullin, and B. Ghanem. CAMEL: Communicative Agents for "Mind" Exploration of Large Language Model Society, 2023. URL https://arxiv.org/abs/2303.17760. Conference on Neural Information Processing Systems (NeurIPS-23).
91. J. Li, Q. Zhang, Y. Yu, Q. Fu, and D. Ye. More Agents Is All You Need, 2024. URL https://arxiv.org/abs/2402.05120. Transactions on Machine Learning Research (TMLR).
92. J. Li, X. Xiao, Y. Zhang, C. Liu, L. Zhao, X. Liao, Y. Ji, J. Wang, J. Gu, Y. Ge, W. Xu, X. Fang, X. Xu, T. Zhao, Y. Kim, T. Wang, J. Hamm, S. Krishnaswamy, J. Huan, and C. K. Reddy. Agent Harness Engineering: A Survey, 2026. URL https://openreview.net/forum?id=eONq7FdiHa. OpenReview preprint.
93. M. Li, Y. Zhao, B. Yu, F. Song, H. Li, H. Yu, Z. Li, F. Huang, and Y. Li. API-Bank: A Comprehensive Benchmark for Tool-Augmented LLMs, 2023. URL https://arxiv.org/abs/2304.08244. Conference on Empirical Methods in Natural Language Processing (EMNLP-23).
94. Q. Li and X. Wang. Sponge tool attack: Stealthy denial-of-efficiency against tool-augmented agentic reasoning, 2026. URL https://arxiv.org/abs/2601.17566. International Conference on Machine Learning (ICML-26).
95. S. Li, Y. He, H. Guo, X. Bu, G. Bai, J. Liu, J. Liu, X. Qu, Y. Li, W. Ouyang, W. Su, and B. Zheng. GraphReader: Building Graph-based Agent to Enhance Long-Context Abilities of Large Language Models, 2024. URL https://arxiv.org/abs/2406.14550. Conference on Empirical Methods in Natural Language Processing (EMNLP-24).
96. S. Li, H. Hasson, and J. Ghosh. OMAC: A Holistic Optimization Framework for LLM-Based Multi-Agent Collaboration, 2025. URL https://arxiv.org/abs/2505.11765. International Conference on Machine Learning (ICML-26).
97. S. Li, Y. Liu, Q. Wen, C. Zhang, and S. Pan. Assemble Your Crew: Automatic Multi-agent Communication Topology Design via Autoregressive Graph Generation, 2025. URL https://arxiv.org/abs/2507.18224. 40th Annual AAAI Conference on Artificial Intelligence (AAAI-26).

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

98. S. Li, Y. Liu, Y. Zheng, M. Li, Q. V. H. Nguyen, and S. Pan. OFA-MAS: One-for-All Multi-Agent System Topology Design based on Mixture-of-Experts Graph Generative Models, 2026. URL https://arxiv.org/abs/2601.12996. ACM Web Conference (WWW-26).
99. Y. Li, D. Choi, J. Chung, N. Kushman, J. Schrittwieser, R. Leblond, T. Eccles, J. Keeling, F. Gimeno, A. D. Lago, T. Hubert, P. Choy, C. de Masson d'Autume, I. Babuschkin, X. Chen, P.-S. Huang, J. Welbl, S. Gowal, A. Cherepanov, J. Molloy, D. J. Mankowitz, E. S. Robson, P. Kohli, N. de Freitas, K. Kavukcuoglu, and O. Vinyals. Competition-Level Code Generation with AlphaCode, 2022. URL https://arxiv.org/abs/2203.07814. Science, 2022.
100. Y. Li, Y. Du, J. Zhang, L. Hou, P. Grabowski, Y. Li, and E. Ie. Improving Multi-Agent Debate with Sparse Communication Topology, 2024. URL https://arxiv.org/abs/2406.11776. Conference on Empirical Methods in Natural Language Processing (EMNLP-24).
101. Z. Li, Y. Zhang, P. Cheng, J. Song, M. Zhou, H. Li, S. Hu, Y. Qin, E. Zhao, X. Jiang, and G. Jiang. MARCH: Multi-Agent Reinforced Self-Check for LLM Hallucination, 2026. URL https://arxiv.org/abs/2603.24579. Annual Meeting of the Association for Computational Linguistics (ACL-26).
102. T. Liang, Z. He, W. Jiao, X. Wang, Y. Wang, R. Wang, Y. Yang, S. Shi, and Z. Tu. Encouraging Divergent Thinking in Large Language Models through Multi-Agent Debate, 2023. URL https://arxiv.org/abs/2305.19118. Conference on Empirical Methods in Natural Language Processing (EMNLP-24).
103. F. Lin, S. Chen, R. Fang, H. Wang, and T. Lin. Stop Wasting Your Tokens: Towards Efficient Runtime Multi-Agent Systems, 2025. URL https://arxiv.org/abs/2510.26585. International Conference on Learning Representations (ICLR-26).
104. W. Ling, D. Yogatama, C. Dyer, and P. Blunsom. Program Induction by Rationale Generation: Learning to Solve and Explain Algebraic Word Problems, 2017. URL https://arxiv.org/abs/1705.04146. Annual Meeting of the Association for Computational Linguistics (ACL-17).
105. M. L. Littman. Markov Games as a Framework for Multi-Agent Reinforcement Learning, 1994. URL https://doi.org/10.1016/B978-1-55860-335-6.50027-1. 11th International Conference on Machine Learning (ICML-94).
106. J. Liu, L. Cui, H. Liu, D. Huang, Y. Wang, and Y. Zhang. LogiQA: A Challenge Dataset for Machine Reading Comprehension with Logical Reasoning, 2020. URL https://arxiv.org/abs/2007.08124. International Joint Conference on Artificial Intelligence (IJCAI-20).
107. J. Liu, Z. Kong, C. Yang, F. Yang, T. Li, P. Dong, J. Nanjekye, H. Tang, G. Yuan, W. Niu, W. Zhang, P. Zhao, X. Lin, D. Huang, and Y. Wang. RCR-Router: Efficient Role-Aware Context Routing for Multi-Agent LLM Systems with Structured Memory, 2025. URL https://arxiv.org/abs/2508.04903. arXiv preprint.
108. S. Liu, T. Chen, Z. Liang, X. Lyu, and C. Amato. LLM Collaboration With Multi-Agent Reinforcement Learning, 2025. URL https://arxiv.org/abs/2508.04652. arXiv preprint.
109. S. Liu, T. Chen, R. Amiri, and C. Amato. Learning Decentralized LLM Collaboration with Multi-Agent Actor Critic, 2026. URL https://arxiv.org/abs/2601.21972. arXiv preprint.
110. S. Liu, X. Yuan, T. Chen, Z. Zhan, Z. Han, D. Zheng, W. Zhang, and S. Cao. CASTER: Breaking the Cost-Performance Barrier in Multi-Agent Orchestration via Context-Aware Strategy for Task Efficient Routing, 2026. URL https://arxiv.org/abs/2601.19793. arXiv preprint.
111. X. Liu, H. Yu, H. Zhang, Y. Xu, X. Lei, H. Lai, Y. Gu, H. Ding, K. Men, K. Yang, S. Zhang, X. Deng, A. Zeng, Z. Du, C. Zhang, S. Shen, T. Zhang, Y. Su, H. Sun, M. Huang, Y. Dong, and J. Tang. AgentBench: Evaluating LLMs as Agents, 2023. URL https://arxiv.org/abs/2308.03688. International Conference on Learning Representations (ICLR-24).
112. Y. Liu, Y. Huang, J. Yao, S. Feng, Z. Gu, K. Du, H. Li, Y. Cheng, J. Jiang, S. Lu, M. Musuvathi, and E. Choukse. DroidSpeak: KV Cache Sharing for Cross-LLM Communication and Multi-LLM Serving, 2024. URL https://arxiv.org/abs/2411.02820. USENIX Symposium on Networked Systems Design and Implementation (NSDI-26).
113. Y. Liu, J. Zhang, Y. Wu, M. Takac, S. Lahlou, X. Chen, and N. Lukas. Breaking the Martingale Curse: Multi-Agent Debate via Asymmetric Cognitive Potential Energy, 2026. URL https://arxiv.org/abs/2603.06801. arXiv preprint.
114. Z. Liu, Y. Zhang, P. Li, Y. Liu, and D. Yang. A Dynamic LLM-Powered Agent Network for Task-Oriented Agent Collaboration, 2023. URL https://arxiv.org/abs/2310.02170. Conference on Language Modeling (COLM-24).
115. R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments, 2017. URL https://proceedings.neurips.cc/paper_files/paper/20

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

17/hash/68a9750337a418a86fe06c1991a1d64c-Abstract.html. Conference on Neural Information Processing Systems (NeurIPS-17).

116. C. Ma, J. Zhang, Z. Zhu, C. Yang, Y. Yang, Y. Jin, Z. Lan, L. Kong, and J. He. AgentBoard: An Analytical Evaluation Board of Multi-turn LLM Agents, 2024. URL https://arxiv.org/abs/2401.13178. Conference on Neural Information Processing Systems (NeurIPS-24).
117. A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhumoye, Y. Yang, S. Gupta, B. P. Majumder, K. Hermann, S. Welleck, A. Yazdanbakhsh, and P. Clark. Self-Refine: Iterative Refinement with Self-Feedback, 2023. URL https://arxiv.org/abs/2303.17651. Conference on Neural Information Processing Systems (NeurIPS-23).
118. A. Maharana, D.-H. Lee, S. Tulyakov, M. Bansal, F. Barbieri, and Y. Fang. Evaluating Very Long-Term Conversational Memory of LLM Agents, 2024. URL https://arxiv.org/abs/2402.17753. Annual Meeting of the Association for Computational Linguistics (ACL-24).
119. Z. Mandi, S. Jain, and S. Song. RoCo: Dialectic Multi-Robot Collaboration with Large Language Models, 2023. URL https://arxiv.org/abs/2307.04738. IEEE International Conference on Robotics and Automation (ICRA-24).
120. A. Mazumder, S. R. Dipta, N. J. Lia, T. Khan, K. R. Hossain, N. Shri, S. Debsarkar, H. Tasnim, G. G. T. Shawon, D. Mitra, S. A. Rani, A. J. I. Anik, and A. N. Khan. AgentCollabBench: Diagnosing When Good Agents Make Bad Collaborators, 2026. URL https://arxiv.org/abs/2605.08647. arXiv preprint.
121. Y. Meng, S. S. Narvekar, L. Tang, R. R. Vaje, Y. Zhang, M. Ye, and Z. Xi. EquiMem: Calibrating Shared Memory in Multi-Agent Debate via Game-Theoretic Equilibrium, 2026. URL https://arxiv.org/abs/2605.09278. arXiv preprint.
122. G. Mialon, C. Fourrier, C. Swift, T. Wolf, Y. LeCun, and T. Scialom. GAIA: a benchmark for General AI Assistants, 2023. URL https://arxiv.org/abs/2311.12983. arXiv preprint.
123. X. Mou, C. Qian, W. Liu, L. Yan, Y. Hu, X. Huang, and Z. Wei. EcoLANG: Efficient and Effective Agent Communication Language Induction for Social Simulation, 2025. URL https://aclanthology.org/2025.findings-emnlp.284/. Findings of the Association for Computational Linguistics: EMNLP 2025 (EMNLP Findings-25).
124. D. V. Mulders, M. Bogaert, and D. V. den Poel. Digital Pantheon: Simulating and Auditing Coalition Formation with LLM Agents, 2026. URL https://arxiv.org/abs/2607.15095. AIDEM Workshop at the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD AIDEM-26).
125. T. Nguyen, P. Chin, and Y.-W. Tai. MA-RAG: Multi-Agent Retrieval-Augmented Generation via Collaborative Chain-of-Thought Reasoning, 2025. URL https://arxiv.org/abs/2505.20096. arXiv preprint.
126. Z. Nie, R. Shen, X. Yu, B. Yin, J. Zhang, and X. Hu. SkillGraph: Self-Evolving Multi-Agent Collaboration with Multimodal Graph Topology, 2026. URL https://arxiv.org/abs/2604.17503. arXiv preprint.
127. B. Niu, Y. Song, K. Lian, Y. Shen, Y. Yao, K. Zhang, and T. Liu. Flow: Modularized Agentic Workflow Automation, 2025. URL https://arxiv.org/abs/2501.07834. International Conference on Learning Representations (ICLR-25).
128. R. Niu, X. Shu, and Y. Zhao. Reliability-Contagion Feasibility in LLM Multi-Agent Networks, 2026. URL https://arxiv.org/abs/2607.21912. arXiv preprint.
129. M. Okawa. Emergence of Biased Consensus in Multi-Agent LLM Debates, 2026. URL https://arxiv.org/abs/2608.02827. International Conference on Machine Learning (ICML-26).
130. I. Ong, A. Almahairi, V. Wu, W.-L. Chiang, T. Wu, J. E. Gonzalez, M. W. Kadous, and I. Stoica. RouteLLM: Learning to Route LLMs with Preference Data, 2024. URL https://arxiv.org/abs/2406.18665. arXiv preprint.
131. O. Ozer, Y. Wang, G. Wu, D. Dosti, H. Zhang, and V. D. L. Rue. MAR:Multi-Agent Reflexion Improves Reasoning Abilities in LLMs, 2025. URL https://arxiv.org/abs/2512.20845. arXiv preprint.
132. C. Packer, S. Wooders, K. Lin, V. Fang, S. G. Patil, I. Stoica, and J. E. Gonzalez. MemGPT: Towards LLMs as Operating Systems, 2023. URL https://arxiv.org/abs/2310.08560. arXiv preprint.
133. S. Pan and M. Luo. How Task Structure Limits Multi-Agent Success: An Information-Theoretic Analysis, 2026. URL https://arxiv.org/abs/2606.13733. arXiv preprint.
134. V. Parakhin. Token Coherence: Adapting MESI Cache Protocols to Minimize Synchronization Overhead in Multi-Agent LLM Systems, 2026. URL https://arxiv.org/abs/2603.15183. arXiv preprint.
135. J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein. Generative Agents: Interactive Simulacra of Human Behavior, 2023. URL https://arxiv.org/abs/2304.03442. ACM Symposium on User Interface Software and Technology (UIST-23).

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

136. C. Parmar, A. Mehta, H. Wu, J. Ramamurthy, and S. Medhekar. When Helping Hurts and How to Fix It: Multi-Agent Debate for Data Cleaning, 2026. URL https://arxiv.org/abs/2606.02866. arXiv preprint.
137. A. Patel, S. Bhattamishra, and N. Goyal. Are NLP Models Really Able to Solve Simple Math Word Problems?, 2021. URL https://arxiv.org/abs/2103.07191. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-21).
138. S. G. Patil, T. Zhang, X. Wang, and J. E. Gonzalez. Gorilla: Large Language Model Connected with Massive APIs, 2023. URL https://arxiv.org/abs/2305.15334. arXiv preprint.
139. R. Pryzant, D. Iter, J. Li, Y. Lee, C. Zhu, and M. Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 7957–7968, 2023.
140. C. Qian, W. Liu, H. Liu, N. Chen, Y. Dang, J. Li, C. Yang, W. Chen, Y. Su, X. Cong, J. Xu, D. Li, Z. Liu, and M. Sun. ChatDev: Communicative Agents for Software Development, 2023. URL https://arxiv.org/abs/2307.07924. Annual Meeting of the Association for Computational Linguistics (ACL-24).
141. C. Qian, Z. Xie, Y. Wang, W. Liu, K. Zhu, H. Xia, Y. Dang, Z. Du, W. Chen, C. Yang, Z. Liu, and M. Sun. Scaling Large Language Model-based Multi-Agent Collaboration, 2024. URL https://arxiv.org/abs/2406.07155. International Conference on Learning Representations (ICLR-25).
142. Y. Qin, S. Liang, Y. Ye, K. Zhu, L. Yan, Y. Lu, Y. Lin, X. Cong, X. Tang, B. Qian, S. Zhao, L. Hong, R. Tian, R. Xie, J. Zhou, M. Gerstein, D. Li, Z. Liu, and M. Sun. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs, 2023. URL https://arxiv.org/abs/2307.16789. International Conference on Learning Representations (ICLR-24).
143. A. S. Rao and M. P. Georgeff. BDI Agents: From Theory to Practice, 1995. URL https://cdn.aaai.org/ICMAS/1995/ICMAS95-042.pdf. First International Conference on Multi-Agent Systems (ICMAS-95).
144. T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. Foerster, and S. Whiteson. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning, 2018. URL https://proceedings.mlr.press/v80/rashid18a.html. 35th International Conference on Machine Learning (ICML-18).
145. D. Rein, B. L. Hou, A. C. Stickland, J. Petty, R. Y. Pang, J. Dirani, J. Michael, and S. R. Bowman. GPQA: A Graduate-Level Google-Proof Q&A Benchmark, 2024. URL https://arxiv.org/abs/2311.12022. Conference on Language Modeling (COLM-24).
146. X. Ren, Q. Zang, Z. Wang, C. Forder, S. Deb, P. Carroll, and Z. Guo. AgentRadio: Passive Awareness for Long-Horizon Multi-Agent Collaboration, 2026. URL https://arxiv.org/abs/2607.28430. arXiv preprint.
147. A. Rezazadeh, Z. Li, A. Lou, Y. Zhao, W. Wei, and Y. Bao. Collaborative Memory: Multi-User Memory Sharing in LLM Agents with Dynamic Access Control, 2025. URL https://arxiv.org/abs/2505.18279. arXiv preprint.
148. S. Roy and D. Roth. Solving General Arithmetic Word Problems, 2015. URL https://aclanthology.org/D15-1202/. Conference on Empirical Methods in Natural Language Processing (EMNLP-15).
149. M. Salim, J. Latendresse, S. Khatoonabadi, and E. Shihab. Tokenomics: Quantifying Where Tokens Are Used in Agentic Software Engineering, 2026. URL https://arxiv.org/abs/2601.14470. arXiv preprint.
150. A. Salve, S. Attar, M. Deshmukh, S. Shivpuje, and A. M. Utsab. A Collaborative Multi-Agent Approach to Retrieval-Augmented Generation Across Diverse Data, 2024. URL https://arxiv.org/abs/2412.05838. arXiv preprint.
151. L. Sander, F. Pan, V. Zolfaghari, A. Schamschurko, N. Petrovic, and A. Knoll. Scaling LLM-Driven Multi-Agent Systems: Design Principles and Architectural Scalability Analysis, 2026. URL https://arxiv.org/abs/2607.27942. arXiv preprint.
152. W. Seo, J. Lee, J. Koh, W. Choi, H. An, J. Park, S. lee, H. Chen, and Y. Bu. MA-SAPO: Multi-Agent Reasoning for Score-Aware Prompt Optimization, 2025. URL https://arxiv.org/abs/2510.16635. arXiv preprint.
153. M. Shen, C. Shang, S. Shahriar, D. Kulshreshtha, Y. Zhang, S. Swamy, and Y. Qi. Relational Priors as Convergence Pressure in LLM-Based Multi-Agent Systems, 2026. URL https://arxiv.org/abs/2608.03239. arXiv preprint.
154. X. Shen, Y. Liu, Y. Dai, Y. Wang, R. Miao, Y. Tan, S. Pan, and X. Wang. Understanding the Information Propagation Effects of Communication Topologies in LLM-based Multi-Agent Systems, 2025. URL https://arxiv.org/abs/2505.23352. Conference on Empirical Methods in Natural Language Processing (EMNLP-25).
155. X. Shen, Q. Zhang, S. Wang, Z. Tan, X. Zhao, L. Yao, V. Tadiparthi, H. N. Mahjoub, E. M. Pari, K. Lee, and T. Chen. Metacognitive Self-Correction for Multi-Agent System via Prototype-Guided Next-Execution Reconstruction, 2025. URL https://arxiv.org/abs/2510.14319. arXiv preprint.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

156. Y. Shi, M. Wang, Y. Cao, H. Lai, J. Lan, X. Han, Y. Wang, J. Geng, Z. Li, Z. Xia, X. Chen, C. Li, J. Xu, W. Duan, and Y. Zhu. Aime: Towards Fully-Autonomous Multi-Agent Framework, 2025. URL https://arxiv.org/abs/2507.11988. arXiv preprint.
157. N. Shinn, F. Cassano, E. Berman, A. Gopinath, K. Narasimhan, and S. Yao. Reflexion: Language Agents with Verbal Reinforcement Learning, 2023. URL https://arxiv.org/abs/2303.11366. Conference on Neural Information Processing Systems (NeurIPS-23).
158. M. Shridhar, X. Yuan, M.-A. Côté, Y. Bisk, A. Trischler, and M. Hausknecht. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning, 2021. URL https://arxiv.org/abs/2010.03768. International Conference on Learning Representations (ICLR-21).
159. A. Singh, A. Ehtesham, S. Kumar, T. T. Khoei, and A. V. Vasilakos. Agentic Retrieval-Augmented Generation: A Survey on Agentic RAG, 2025. URL https://arxiv.org/abs/2501.09136. arXiv preprint.
160. R. G. Smith. The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver, 1980. URL https://doi.org/10.1109/TC.1980.1675516. IEEE Transactions on Computers.
161. Y. Song, W. Xiong, D. Zhu, W. Wu, H. Qian, M. Song, H. Huang, C. Li, K. Wang, R. Yao, Y. Tian, and S. Li. RestGPT: Connecting Large Language Models with Real-World RESTful APIs, 2023. URL https://arxiv.org/abs/2306.06624. arXiv preprint.
162. Z. Sprague, X. Ye, K. Bostrom, S. Chaudhuri, and G. Durrett. MuSR: Testing the Limits of Chain-of-Thought with Multistep Soft Reasoning, 2024. URL https://arxiv.org/abs/2310.16049. International Conference on Learning Representations (ICLR-24), Spotlight.
163. M. Suzgun, N. Scales, N. Sch"arli, S. Gehrmann, Y. Tay, H. W. Chung, A. Chowdhery, Q. V. Le, E. H. Chi, D. Zhou, and J. Wei. Challenging BIG-Bench Tasks and Whether Chain-of-Thought Can Solve Them, 2023. URL https://arxiv.org/abs/2210.09261. Annual Meeting of the Association for Computational Linguistics (Findings of ACL-23).
164. Y. Talebirad and A. Nadiri. Multi-Agent Collaboration: Harnessing the Power of Intelligent LLM Agents, 2023. URL https://arxiv.org/abs/2306.03314. arXiv preprint.
165. A. Talmor, J. Herzig, N. Lourie, and J. Berant. CommonsenseQA: A Question Answering Challenge Targeting Commonsense Knowledge, 2019. URL https://arxiv.org/abs/1811.00937. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-19).
166. J. Tang, J. Zhang, Q. Lv, S. Liu, J. Yang, C. Tang, and K. Wang. HiVA: Self-Organized Hierarchical Variable Agent via Goal-Driven Semantic-Topological Evolution, 2026. URL https://ojs.aaai.org/index.php/AAAI/article/view/37190. 40th Annual AAAI Conference on Artificial Intelligence (AAAI-26).
167. X. Tang, T. Qin, T. Peng, Z. Zhou, D. Shao, T. Du, X. Wei, P. Xia, F. Wu, H. Zhu, G. Zhang, J. Liu, X. Wang, S. Hong, C. Wu, H. Cheng, C. Wang, and W. Zhou. Agent KB: Leveraging Cross-Domain Experience for Agentic Problem Solving, 2025. URL https://arxiv.org/abs/2507.06229. arXiv preprint.
168. J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal. FEVER: A Large-Scale Dataset for Fact Extraction and VERification, 2018. URL https://arxiv.org/abs/1803.05355. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-18).
169. D. Tran and D. Kiela. Single-Agent LLMs Outperform Multi-Agent Systems on Multi-Hop Reasoning Under Equal Thinking Token Budgets, 2026. URL https://arxiv.org/abs/2604.02460. arXiv preprint.
170. K.-T. Tran, D. Dao, M.-D. Nguyen, Q.-V. Pham, B. O'Sullivan, and H. D. Nguyen. Multi-Agent Collaboration Mechanisms: A Survey of LLMs, 2025. URL https://arxiv.org/abs/2501.06322. arXiv preprint.
171. H. Trivedi, N. Balasubramanian, T. Khot, and A. Sabharwal. MuSiQue: Multihop Questions via Single-hop Question Composition, 2022. URL https://arxiv.org/abs/2108.00573. Transactions of the Association for Computational Linguistics (TACL), 2022.
172. H. Trivedi, T. Khot, M. Hartmann, R. Manku, V. Dong, E. Li, S. Gupta, A. Sabharwal, and N. Balasubramanian. AppWorld: A Controllable World of Apps and People for Benchmarking Interactive Coding Agents, 2024. URL https://arxiv.org/abs/2407.18901. Annual Meeting of the Association for Computational Linguistics (ACL-24).
173. H. Wan, J. Wu, M. Luo, F. Li, N. Wang, N. F. Chen, and M.-Y. Kan. The Deliberative Illusion: Diagnosing Factual Attrition and Stance Homogenization in Multi-Agent LLM Deliberation, 2026. URL https://arxiv.org/abs/2606.03032. arXiv preprint.
174. D. Wang, D. Yin, R. Desai, L. Li, A. Celikyilmaz, and A. Ni. Learning to Interrupt in Language-based Multi-agent Communication, 2026. URL https://arxiv.org/abs/2604.06452. arXiv preprint.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

175. G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar. Voyager: An Open-Ended Embodied Agent with Large Language Models, 2023. URL https://arxiv.org/abs/2305.16291. Transactions on Machine Learning Research (TMLR).
176. J. Wang, J. Wang, B. Athiwaratkun, C. Zhang, and J. Zou. Mixture-of-Agents Enhances Large Language Model Capabilities, 2024. URL https://arxiv.org/abs/2406.04692. International Conference on Learning Representations (ICLR-25).
177. J. Wang, S. Zhao, H. Wang, Y. Fan, and T. Liu. Optimal-Agent-Selection: State-Aware Routing Framework for Efficient Multi-Agent Collaboration, 2025. URL https://arxiv.org/abs/2511.02200. arXiv preprint.
178. M. F. Wang, H. Xie, G. Wang, D. Zhang, and J. O. Woo. Budgeted Act-or-Defer Multi-Agent LLM Deliberation with Local Reliability Bounds, 2026. URL https://arxiv.org/abs/2606.29654. arXiv preprint.
179. Q. Wang, Z. Tang, Z. Jiang, N. Chen, T. Wang, and B. He. AgentTaxo: Dissecting and Benchmarking Token Distribution of LLM Multi-Agent Systems, 2025. URL https://openreview.net/forum?id=0iLbiYYIpC. International Conference on Learning Representations Workshop on Foundation Models in the Wild (ICLR FM-Wild-25).
180. R. Wang, P. Jansen, M.-A. Côté, and P. Ammanabrolu. ScienceWorld: Is Your Agent Smarter than a 5th Grader?, 2022. URL https://arxiv.org/abs/2203.07540. Conference on Empirical Methods in Natural Language Processing (EMNLP-22).
181. S. Wang, Z. Tan, Z. Chen, S. Zhou, T. Chen, and J. Li. AnyMAC: Cascading Flexible Multi-Agent Collaboration via Next-Agent Prediction, 2025. URL https://arxiv.org/abs/2506.17784. Conference on Empirical Methods in Natural Language Processing (EMNLP-25).
182. W. Wang, D. Zhang, T. Feng, B. Wang, and J. Tang. BattleAgentBench: A Benchmark for Evaluating Cooperation and Competition Capabilities of Language Models in Multi-Agent Systems, 2024. URL https://arxiv.org/abs/2408.15971. arXiv preprint.
183. X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou. Self-Consistency Improves Chain of Thought Reasoning in Language Models, 2022. URL https://arxiv.org/abs/2203.11171. International Conference on Learning Representations (ICLR-23).
184. X. Wang, J. Wang, F. Zhang, Y. Hu, D. Zhang, Y. Ye, Y. Ban, J. Han, and R. Wang. MasFACT: Continual Multi-Agent Topology Learning via Geometry-Aware Posterior Transfer, 2026. URL https://arxiv.org/abs/2605.17361. arXiv preprint.
185. Y. Wang and X. Chen. MIRIX: Multi-Agent Memory System for LLM-Based Agents, 2025. URL https://arxiv.org/abs/2507.07957. arXiv preprint.
186. Y. Wang, X. Ma, G. Zhang, Y. Ni, A. Chandra, S. Guo, W. Ren, A. Arulraj, X. He, Z. Jiang, T. Li, M. Ku, K. Wang, A. Zhuang, R. Fan, X. Yue, and W. Chen. MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark, 2024. URL https://arxiv.org/abs/2406.01574. Conference on Neural Information Processing Systems (NeurIPS-24), Datasets and Benchmarks Track.
187. Y. Wang, J. Zhao, H. Xie, H. Ma, Y. Lei, S. Liu, X. Song, Z. Zhang, and H. Zhang. MetaGen: Self-Evolving Roles and Topologies for Multi-Agent LLM Reasoning, 2026. URL https://arxiv.org/abs/2601.19290. arXiv preprint.
188. Z. Wang, Y. Wang, X. Liu, L. Ding, M. Zhang, J. Liu, and M. Zhang. AgentDropout: Dynamic Agent Elimination for Token-Efficient and High-Performance LLM-Based Multi-Agent Collaboration, 2025. URL https://arxiv.org/abs/2503.18891. arXiv preprint.
189. Z. Wang, X. Liu, L. Wang, Z. Shan, Y. Wang, Z. Song, and M. Zhang. MASPO: Joint Prompt Optimization for LLM-based Multi-Agent Systems, 2026. URL https://arxiv.org/abs/2605.06623. International Conference on Machine Learning (ICML-26).
190. Z. Wang, C. Pei, Y. Liu, J. Li, Y. Huo, Q. Zhou, H. Si, H. Cui, Z. Liu, G. Xie, F. Sun, D. Pei, and D. Lo. Agent System Operations: Categorization, Challenges, and Future Directions, 2026. URL https://arxiv.org/abs/2606.01581. arXiv preprint.
191. Y. Wolf, B. Rothberg, D. Shteyman, and A. Shashua. Compositional Hardness of Code in Large Language Models – A Probabilistic Perspective, 2024. URL https://arxiv.org/abs/2409.18028. arXiv preprint.
192. J. Wu, Y. Sun, T. Xie, S. Chen, J. Bao, Y. Xu, G. Du, I. Heo, A. Gutfraind, and X. Wang. Cross-Modal Memory Compression for Efficient Multi-Agent Debate, 2026. URL https://arxiv.org/abs/2602.00454. arXiv preprint.
193. Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, A. H. Awadallah, R. W. White, D. Burger, and C. Wang. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation, 2023. URL https://arxiv.org/abs/2308.08155. arXiv preprint.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

194. S. Wu and K. Shu. Memory in LLM-based Multi-agent Systems: Mechanisms, Challenges, and Collective Intelligence, 2025. URL https://doi.org/10.36227/techrxiv.176539617.79044553/v1. TechRxiv preprint.
195. Z. Xi, Y. Ding, W. Chen, B. Hong, H. Guo, J. Wang, D. Yang, C. Liao, X. Guo, W. He, S. Gao, L. Chen, R. Zheng, Y. Zou, T. Gui, Q. Zhang, X. Qiu, X. Huang, Z. Wu, and Y.-G. Jiang. AgentGym: Evolving Large Language Model-based Agents across Diverse Environments, 2024. URL https://arxiv.org/abs/2406.04151. arXiv preprint.
196. J. Xie, K. Zhang, J. Chen, T. Zhu, R. Lou, Y. Tian, Y. Xiao, and Y. Su. TravelPlanner: A Benchmark for Real-World Planning with Language Agents, 2024. URL https://arxiv.org/abs/2402.01622. International Conference on Machine Learning (ICML-24).
197. T. Xie, D. Zhang, J. Chen, X. Li, S. Zhao, R. Cao, T. J. Hua, Z. Cheng, D. Shin, F. Lei, Y. Liu, Y. Xu, S. Zhou, S. Savarese, C. Xiong, V. Zhong, and T. Yu. OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments, 2024. URL https://arxiv.org/abs/2404.07972. Conference on Neural Information Processing Systems (NeurIPS-24).
198. C. Xu, Y. Hu, R. Wang, X. Lin, W. Wang, D. Liu, and F. Feng. TacoMAS: Test-Time Co-Evolution of Topology and Capability in LLM-based Multi-Agent Systems, 2026. URL https://arxiv.org/abs/2605.09539. arXiv preprint.
199. J. Xu, A. Koesdwiady, S. Bei, Y. Han, B. Huang, D. Wang, Y. Chen, Z. Wang, P. Wang, P. Li, and Y. Ding. Rethinking the Value of Multi-Agent Workflow: A Strong Single Agent Baseline, 2026. URL https://arxiv.org/abs/2601.12307. arXiv preprint.
200. W. Xu, Z. Liang, K. Mei, H. Gao, J. Tan, and Y. Zhang. A-MEM: Agentic Memory for LLM Agents, 2025. URL https://arxiv.org/abs/2502.12110. Conference on Neural Information Processing Systems (NeurIPS-25).
201. Z. Xu, S. Zhu, J. Wang, J. Wang, B. Athiwaratkun, C. Wang, J. Zou, and C. Zhang. When Does Divide and Conquer Work for Long Context LLM? A Noise Decomposition Framework, 2025. URL https://arxiv.org/abs/2506.16411. International Conference on Learning Representations (ICLR-26).
202. C. Yan, Z. Yue, F. Zhao, E. Lin, L. Jia, H. Tong, M. Lyu, C. Sun, and Y. Zeng. When Truth Is Distributed: Misinformation Derails Collective Fact Recovery in LLM-Based Multi-Agent Systems, 2026. URL https://arxiv.org/abs/2608.03421. arXiv preprint.
203. B. Yang, X. He, H. Gao, Y. Cao, X. Li, and D. Hsu. CodeAgents: A Token-Efficient Framework for Codified Multi-Agent Reasoning in LLMs, 2025. URL https://arxiv.org/abs/2507.03254. arXiv preprint.
204. J. Yang, C. E. Jimenez, A. Wettig, K. Lieret, S. Yao, K. Narasimhan, and O. Press. SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering, 2024. URL https://arxiv.org/abs/2405.15793. Conference on Neural Information Processing Systems (NeurIPS-24).
205. W. Yang, J. Pang, S. Li, P. Bogdan, S. Tu, and J. Thomason. Maestro: Learning to Collaborate via Conditional Listwise Policy Optimization for Multi-Agent LLMs, 2025. URL https://arxiv.org/abs/2511.06134. arXiv preprint.
206. W. Yang, D. Cao, J. Pang, M. Weng, and Y. Liu. Adaptive Collaboration with Humans: Metacognitive Policy Optimization for Multi-Agent LLMs with Continual Learning, 2026. URL https://arxiv.org/abs/2603.07972. arXiv preprint.
207. Y. Yang, H. Chai, S. Shao, Y. Song, S. Qi, R. Rui, and W. Zhang. AgentNet: Decentralized Evolutionary Coordination for LLM-based Multi-Agent Systems, 2025. URL https://arxiv.org/abs/2504.00587. Conference on Neural Information Processing Systems (NeurIPS-25).
208. Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D. Manning. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering, 2018. URL https://arxiv.org/abs/1809.09600. Conference on Empirical Methods in Natural Language Processing (EMNLP-18).
209. S. Yao, H. Chen, J. Yang, and K. Narasimhan. WebShop: Towards Scalable Real-World Web Interaction with Grounded Language Agents, 2022. URL https://arxiv.org/abs/2207.01206. Conference on Neural Information Processing Systems (NeurIPS-22).
210. S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. ReAct: Synergizing Reasoning and Acting in Language Models, 2022. URL https://arxiv.org/abs/2210.03629. International Conference on Learning Representations (ICLR-23).
211. S. Yao, N. Shinn, P. Razavi, and K. Narasimhan. τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains, 2024. URL https://arxiv.org/abs/2406.12045. arXiv preprint.
212. T. Yao, Z. Li, and Z. Shen. HieraMAS: Optimizing Intra-Node LLM Mixtures and Inter-Node Topology for Multi-Agent Systems, 2026. URL https://arxiv.org/abs/2602.20229. arXiv preprint.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

213. H. Ye, Z. Gao, M. Ma, Q. Wang, Y. Fu, M.-Y. Chung, Y. Lin, Z. Liu, J. Zhang, D. Zhuo, and Y. Chen. KVCOMM: Online Cross-context KV-cache Communication for Efficient LLM-based Multi-agent Systems, 2025. URL https://arxiv.org/abs/2510.12872. Conference on Neural Information Processing Systems (NeurIPS-25).
214. Z. Yin, Q. Sun, C. Chang, Q. Guo, J. Dai, X. Huang, and X. Qiu. Exchange-of-Thought: Enhancing Large Language Model Capabilities through Cross-Model Communication, 2023. URL https://arxiv.org/abs/2312.01823. Conference on Empirical Methods in Natural Language Processing (EMNLP-23).
215. Z. Yin, C. Gao, C. Fan, W. Yang, Y. Xue, and L. Zhang. A Comprehensive Empirical Evaluation of Agent Frameworks on Code-centric Software Engineering Tasks, 2025. URL https://arxiv.org/abs/2511.00872. arXiv preprint.
216. S. Yu, X. Xu, K. Deng, L. Li, and L. Tian. Tree of Agents: Improving Long-Context Capabilities of Large Language Models through Multi-Perspective Reasoning, 2025. URL https://arxiv.org/abs/2509.06436. Conference on Empirical Methods in Natural Language Processing (EMNLP-25).
217. W. Yu, L. Zhou, X. Dong, S. S. Barath, D. Staunton, B.-J. Yoon, X. Qian, J. Caverlee, and S. Ji. When Do Multi-Agent Systems Help? An Information Bottleneck Perspective, 2026. URL https://arxiv.org/abs/2607.16133. arXiv preprint.
218. Z. Yu, N. Yu, H. Zhang, W. Ni, M. Yin, J. Yang, Y. Zhao, and J. Zhao. Multi-Agent Memory from a Computer Architecture Perspective: Visions and Challenges Ahead, 2026. URL https://arxiv.org/abs/2603.10062. arXiv preprint.
219. Y. Yue, G. Zhang, B. Liu, G. Wan, K. Wang, D. Cheng, and Y. Qi. MasRouter: Learning to Route LLMs for Multi-Agent Systems, 2025. URL https://arxiv.org/abs/2502.11133. Annual Meeting of the Association for Computational Linguistics (ACL-25).
220. S. Yuen, F. G. Medina, T. Su, Y. Du, and A. J. Sobey. Intrinsic Memory Agents: Heterogeneous Multi-Agent LLM Systems through Structured Contextual Memory, 2025. URL https://arxiv.org/abs/2508.08997. arXiv preprint.
221. S. Yun, J. Peng, P. Li, W. Fan, J. Chen, J. Zou, G. Li, and T. Chen. Graph-of-Agents: A Graph-based Framework for Multi-Agent LLM Collaboration, 2026. URL https://arxiv.org/abs/2604.17148. International Conference on Learning Representations (ICLR-26).
222. Y. Zeng, W. Huang, L. Jiang, T. Liu, X. Jin, C. T. Tiana, J. Li, and X. Xu. S^2-MAD: Breaking the Token Barrier to Enhance Multi-Agent Debate Efficiency, 2025. URL https://arxiv.org/abs/2502.04790. Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics (NAACL-25).
223. Q. Zhan, Z. Liang, Z. Ying, and D. Kang. InjecAgent: Benchmarking Indirect Prompt Injections in Tool-Integrated Large Language Model Agents, 2024. URL https://arxiv.org/abs/2403.02691. Findings of the Association for Computational Linguistics: ACL 2024 (ACL Findings-24).
224. C. Zhang, K. Yang, S. Hu, Z. Wang, G. Li, Y. Sun, C. Zhang, Z. Zhang, A. Liu, S.-C. Zhu, X. Chang, J. Zhang, F. Yin, Y. Liang, and Y. Yang. ProAgent: Building Proactive Cooperative Agents with Large Language Models, 2023. URL https://arxiv.org/abs/2308.11339. 38th Annual AAAI Conference on Artificial Intelligence (AAAI-24).
225. G. Zhang, Y. Yue, Z. Li, S. Yun, G. Wan, K. Wang, D. Cheng, J. X. Yu, and T. Chen. Cut the Crap: An Economical Communication Pipeline for LLM-based Multi-Agent Systems, 2024. URL https://arxiv.org/abs/2410.02506. International Conference on Learning Representations (ICLR-25).
226. G. Zhang, Y. Yue, X. Sun, G. Wan, M. Yu, J. Fang, K. Wang, T. Chen, and D. Cheng. G-Designer: Architecting Multi-agent Communication Topologies via Graph Neural Networks, 2024. URL https://arxiv.org/abs/2410.11782. International Conference on Machine Learning (ICML-25).
227. G. Zhang, K. Chen, G. Wan, H. Chang, H. Cheng, K. Wang, S. Hu, and L. Bai. EvoFlow: Evolving Diverse Agentic Workflows On The Fly, 2025. URL https://arxiv.org/abs/2502.07373. arXiv preprint.
228. G. Zhang, M. Fu, G. Wan, M. Yu, K. Wang, and S. Yan. G-Memory: Tracing Hierarchical Memory for Multi-Agent Systems, 2025. URL https://arxiv.org/abs/2506.07398. Conference on Neural Information Processing Systems (NeurIPS-25).
229. G. Zhang, L. Niu, J. Fang, K. Wang, L. Bai, and X. Wang. Multi-agent Architecture Search via Agentic Supernet, 2025. URL https://arxiv.org/abs/2502.04180. International Conference on Machine Learning (ICML-25).
230. H. Zhang, W. Du, J. Shan, Q. Zhou, Y. Du, J. B. Tenenbaum, T. Shu, and C. Gan. Building Cooperative Embodied Agents Modularly with Large Language Models, 2023. URL https://arxiv.org/abs/2307.02485. International Conference on Learning Representations (ICLR-24).

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

231. H. Zhang, Y. Tian, and T. Zhang. Enhancing Multi-Agent Communication through Attention Steering with Context Relevance, 2026. URL https://arxiv.org/abs/2605.30136. arXiv preprint.
232. J. Zhang, X. Xu, N. Zhang, R. Liu, B. Hooi, and S. Deng. Exploring Collaboration Mechanisms for LLM Agents: A Social Psychology View, 2023. URL https://arxiv.org/abs/2310.02124. Annual Meeting of the Association for Computational Linguistics (ACL-24).
233. J. Zhang, J. Xiang, Z. Yu, F. Teng, X. Chen, J. Chen, M. Zhuge, X. Cheng, S. Hong, J. Wang, B. Zheng, B. Liu, Y. Luo, and C. Wu. AFlow: Automating Agentic Workflow Generation, 2024. URL https://arxiv.org/abs/2410.10762. International Conference on Learning Representations (ICLR-25).
234. R. Zhang, X. Zhao, R. Wang, S. Chen, G. Zhang, A. Zhang, K. Wang, and Q. Wen. SafeSieve: From Heuristics to Experience in Progressive Pruning for LLM-based Multi-Agent Communication, 2025. URL https://arxiv.org/abs/2508.11733. 40th Annual AAAI Conference on Artificial Intelligence (AAAI-26).
235. S. Zhang, M. Yin, J. Zhang, J. Liu, Z. Han, J. Zhang, B. Li, C. Wang, H. Wang, Y. Chen, and Q. Wu. Which Agent Causes Task Failures and When? On Automated Failure Attribution of LLM Multi-Agent Systems, 2025. URL https://arxiv.org/abs/2505.00212. International Conference on Machine Learning (ICML-25).
236. T. Zhang, P. Zhao, Q. Chen, J. Wan, C. Chen, X. He, C. Wang, and R. Hong. Taming "Zombie" Agents: A Markov State-Aware Framework for Resilient Multi-Agent Evolution, 2026. URL https://arxiv.org/abs/2605.17348. Annual Meeting of the Association for Computational Linguistics (ACL-26).
237. T. Zhang, Z. Zhou, J. Wan, T. Hu, C. Wang, X. He, and R. Hong. Learning Transferable Topology Priors for Multi-Agent LLM Collaboration Across Domains, 2026. URL https://arxiv.org/abs/2605.17359. arXiv preprint.
238. W. Zhang, L. Zeng, Y. Xiao, Y. Li, C. Cui, Y. Zhao, R. Hu, Y. Liu, Y. Zhou, and B. An. AgentOrchestra: Orchestrating Multi-Agent Intelligence with the Tool-Environment-Agent(TEA) Protocol, 2025. URL https://arxiv.org/abs/2506.12508. arXiv preprint.
239. X. Zhang, Y. Chen, S. Hu, Z. Xu, J. Chen, M. K. Hao, X. Han, Z. L. Thai, S. Wang, Z. Liu, and M. Sun. ∞Bench: Extending Long Context Evaluation Beyond 100K Tokens, 2024. URL https://arxiv.org/abs/2402.13718. Annual Meeting of the Association for Computational Linguistics (ACL-24).
240. Y. Zhang, R. Sun, Y. Chen, T. Pfister, R. Zhang, and S. Ö. Arik. Chain of Agents: Large Language Models Collaborating on Long-Context Tasks, 2024. URL https://arxiv.org/abs/2406.02818. Conference on Neural Information Processing Systems (NeurIPS-24).
241. Y. Zhang, S. Yang, C. Bai, F. Wu, X. Li, Z. Wang, and X. Li. Towards Efficient LLM Grounding for Embodied Multi-Agent Collaboration, 2024. URL https://arxiv.org/abs/2405.14314. Annual Meeting of the Association for Computational Linguistics (ACL-25).
242. Y. Zhang, X. Liu, and C. Xiao. MetaAgent: Automatically Constructing Multi-Agent Systems Based on Finite State Machines, 2025. URL https://arxiv.org/abs/2507.22606. International Conference on Machine Learning (ICML-25).
243. Z. Zhang, L. Ge, H. Li, W. Zhu, C. Zhang, and Y. Ye. MAPRO: Recasting Multi-Agent Prompt Optimization as Maximum a Posteriori Inference, 2025. URL https://arxiv.org/abs/2510.07475. Conference of the European Chapter of the Association for Computational Linguistics (EACL-26).
244. Z. Zhang, W. Zhou, J. Li, H. Fei, J. Wen, and W. Ji. RADAR: Redundancy-Aware Diffusion for Multi-Agent Communication Structure Generation, 2026. URL https://arxiv.org/abs/2605.09907. International Conference on Machine Learning (ICML-26).
245. A. Zhao, D. Huang, Q. Xu, M. Lin, Y.-J. Liu, and G. Huang. ExpeL: LLM Agents Are Experiential Learners, 2023. URL https://arxiv.org/abs/2308.10144. 38th Annual AAAI Conference on Artificial Intelligence (AAAI-24).
246. J. Zhao, C. Zu, H. Xu, Y. Lu, W. He, Y. Ding, T. Gui, Q. Zhang, and X. Huang. LongAgent: Scaling Language Models to 128k Context through Multi-Agent Collaboration, 2024. URL https://arxiv.org/abs/2402.11550. arXiv preprint.
247. W. Zhong, L. Guo, Q. Gao, H. Ye, and Y. Wang. MemoryBank: Enhancing Large Language Models with Long-Term Memory, 2023. URL https://arxiv.org/abs/2305.10250. 38th Annual AAAI Conference on Artificial Intelligence (AAAI-24).
248. H. Zhou, X. Wan, R. Sun, H. Palangi, S. Iqbal, I. Vulić, A. Korhonen, and S. Ö. Arık. Multi-Agent Design: Optimizing Agents with Better Prompts and Topologies, 2025. URL https://arxiv.org/abs/2502.02533. International Conference on Learning Representations (ICLR-26).

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2026 doi:10.20944/preprints202609.1639.v1

249. S. Zhou, F. F. Xu, H. Zhu, X. Zhou, R. Lo, A. Sridhar, X. Cheng, T. Ou, Y. Bisk, D. Fried, U. Alon, and G. Neubig. WebArena: A Realistic Web Environment for Building Autonomous Agents, 2023. URL https://arxiv.org/abs/2307.13854. International Conference on Learning Representations (ICLR-24).
250. W. Zhou, Y. E. Jiang, L. Li, J. Wu, T. Wang, S. Qiu, J. Zhang, J. Chen, R. Wu, S. Wang, S. Zhu, J. Chen, W. Zhang, X. Tang, N. Zhang, H. Chen, P. Cui, and M. Sachan. Agents: An Open-source Framework for Autonomous Language Agents, 2023. URL https://arxiv.org/abs/2309.07870. arXiv preprint.
251. X. Zhou, H. Zhu, L. Mathur, R. Zhang, H. Yu, Z. Qi, L.-P. Morency, Y. Bisk, D. Fried, G. Neubig, and M. Sap. SOTOPIA: Interactive Evaluation for Social Intelligence in Language Agents, 2023. URL https://arxiv.org/abs/2310.11667. International Conference on Learning Representations (ICLR-24).
252. Z. Zhou, C. Li, X. Chen, S. Wang, Y. Chao, Z. Li, H. Wang, R. An, Q. Shi, Z. Tan, X. Han, X. Shi, Z. Liu, and M. Sun. LLM×MapReduce: Simplified Long-Sequence Processing using Large Language Models, 2024. URL https://arxiv.org/abs/2410.09342. Annual Meeting of the Association for Computational Linguistics (ACL-25).
253. K. Zhu, H. Du, Z. Hong, X. Yang, S. Guo, Z. Wang, Z. Wang, C. Qian, X. Tang, H. Ji, and J. You. MultiAgentBench: Evaluating the Collaboration and Competition of LLM agents, 2025. URL https://arxiv.org/abs/2503.01935. Annual Meeting of the Association for Computational Linguistics (ACL-25).
254. Y. Zhu, A. Kellermann, A. Gupta, P. Li, R. Fang, R. Bindu, and D. Kang. Teams of LLM Agents can Exploit Zero-Day Vulnerabilities, 2024. URL https://arxiv.org/abs/2406.01637. Conference of the European Chapter of the Association for Computational Linguistics (EACL-26).
255. M. Zhuge, W. Wang, L. Kirsch, F. Faccio, D. Khizbullin, and J. Schmidhuber. Language Agents as Optimizable Graphs, 2024. URL https://arxiv.org/abs/2402.16823. International Conference on Machine Learning (ICML-24).
256. J. Zou, R. Qiu, G. Li, X. Yang, K. Tieu, P. Lu, K. Shen, H. Tong, Y. Choi, J. He, J. Zou, M. Wang, and L. Yang. Latent Collaboration in Multi-Agent Systems, 2025. URL https://arxiv.org/abs/2511.20639. International Conference on Machine Learning (ICML-26).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.