OpenDGO: Benchmarking Multi-Agent Organizations Across Diverse Application Domains

Xunkai Li*, Miao Zhang*, Zekai Chen*, Xuaner Chen, Yu liu, Rui Qian, Xun Wu, Hongchao Qin, Rong-Hua Li

arXiv:2610.17143v1 [cs.LG] 12 Oct 2026

Abstract

Large language model (LLM)-based multi-agent systems are increasingly applied across diverse domains, yet different applications require different forms of agent organization. Existing studies typically adopt predefined organizations, such as Star, Chain, and Tree, or learn communication graphs to improve performance on specific tasks. However, these methods are largely designed and evaluated within a limited set of application domains, providing little understanding of whether an organization effective in one domain remains suitable for others with different collaboration requirements. We present **OpenDGO**, a controlled benchmark and analysis framework covering six real-world domains and three collaboration settings: Sequential Dependency, Parallel Decomposition, and Information Integration. Under matched tasks, agents, tools, and resource limits, OpenDGO compares fixed and learned organizations while separately examining communication graphs, role assignments, input assignments, resource budgets, and component failures. Our results show that there is no "one-organization-for-all" solution: preferred organizations vary across applications, and even the same communication graph can yield different outcomes under different role and input assignments. These findings highlight the need for application-dependent organization design, aiming to understand *which organization is suitable for a given application, under what conditions, and why.*

CCS Concepts

• **Computing methodologies** → **Multi-agent systems**.

Keywords

large language models, multi-agent systems, organization design, communication topology, cross-domain benchmarking, robustness

ACM Reference Format:

Xunkai Li*, Miao Zhang*, Zekai Chen*, Xuaner Chen, Yu liu, Rui Qian, Xun Wu, Hongchao Qin, Rong-Hua Li. 2027. OpenDGO: Benchmarking Multi-Agent Organizations Across Diverse Application Domains. In *Proceedings of The ACM Web Conference 2027 (WWW '27).* ACM, New York, NY, USA, 16 pages.

1 Introduction

Large language model (LLM)-based multi-agent systems (MASs) are increasingly used in web-centric applications, such as web search, online recommendation, and document question answering, by coordinating multiple specialized agents to solve complex problems [13, 14, 18]. A basic design choice is how these agents should be organized, including which agents can communicate with each other and which role each agent plays in the communication graph (e.g., which agent is responsible for a specific function, such as implementation, testing, or verification). Common choices include centralized Star structures, ordered Chain structures, hierarchical Trees, and irregular graphs learned automatically for a given task. Different application domains naturally favor different forms of organization. In Coding, issue analysis, implementation, and testing form an ordered workflow, making Chain- or Tree-like organizations intuitive choices for passing intermediate results between stages. Finance similarly requires extracted numbers to move reliably through calculation and verification. Legal and Scientific Research instead require several agents to examine different documents, rules, or claims before their conclusions are combined, which can benefit from more distributed communication. Recommendation often allows user history, reviews, and item attributes to be analyzed in parallel before aggregation, making highly centralized sequential communication less necessary. These application domains therefore place different demands on the organization of an agent team, making a single organization unlikely to work equally well across all of them. The central question is thus *which type of multi-agent organization is appropriate for a given application domain, and why.*

Existing studies have mainly focused on evaluating which organizations perform well within a specific application domain, while paying little attention to whether the same organizations remain effective in other domains with different collaboration requirements [1, 10, 15]. For general-purpose MASs, this distinction matters: an organization suited to the sequential workflow of Coding may not work equally well for Legal analysis or Recommendation, where agents need to collaborate differently. Furthermore, the communication graph is not the only factor affecting performance. Even within a fixed Tree, placing a verifier at the center rather than at a peripheral node changes how it communicates with other agents. Similarly, assigning different inputs, such as code files or financial tables, to different agents can affect their collaboration. To examine these factors, we evaluate the same organizations across diverse domains, keeping the source task, agent model, tools, and resource limits fixed within each comparison while separately varying the communication graph, role assignment, and input assignment. This allows us to assess how organization preferences change across domains and identify the factors behind these differences.

Our motivating observations in Fig.1 support this view. Organization rankings vary considerably across domains and collaboration settings, and neither fixed nor learned organizations consistently dominate all conditions. The same communication graph can also produce different results when role assignment or input assignment changes. For example, under a fixed Tree graph, assigning a key role to a central node can be preferable in some settings, while assigning it to a bridge node becomes more effective in others. These results suggest that organization quality is not determined by the graph

WWW '27, Dublin, Ireland
2027.

shape alone, but by how the communication graph, role assignment, and input assignment jointly match the collaboration required by the application.

Motivated by these findings, we present **OpenDGO**, a controlled benchmark and analysis framework for multi-agent organization. OpenDGO covers six application domains—Coding, Finance, Biomedical, Legal, Scientific Research, and Recommendation—and summarizes their collaboration requirements through three representative settings: *Sequential Dependency*, *Parallel Decomposition*, and *Information Integration*. Under matched source tasks, agents, tools, and resource limits, OpenDGO compares six fixed organizations with recent learned organization methods. It further varies the communication graph, role assignment, input assignment, resource budget, and component failures separately, allowing the effect of each factor to be examined under the same evaluation setting. We also relate organization performance to graph properties and actual communication traces to explain why different organizations behave differently across applications. OpenDGO therefore aims to answer a concrete question: *for a given application and collaboration requirement, which organization should be used, under what resource conditions, and why?*

Our Contributions. *(1) Comprehensive Benchmark.* We introduce OpenDGO, a controlled benchmark spanning six real-world domains, three collaboration settings, six fixed organizations, and recent learned organization methods, enabling organization evaluation across heterogeneous applications rather than within a single domain. *(2) Controlled Analysis.* We separately examine communication graph, role assignment, input assignment, resource budget, and failures, and connect these factors with graph properties and communication traces to explain their effects. *(3) Valuable Insights.* We show that no organization serves as a one-for-all solution: preferred organizations vary across applications, the same communication graph can behave differently under different role and input assignments, and practical choices involve clear quality–cost–robustness trade-offs.

2 Preliminaries

2.1 Notations and Organization Paradigms

Multi-Agent Tasks. Let x denote a source problem from domain $d \in \mathcal{D}$, where $\mathcal{D}$ includes Coding, Finance, Biomedical, Legal, Scientific Research, and Recommendation. We denote its multi-agent form as $\mathcal{T}_{x,s}$, which preserves the original question q and answer target while specifying the roles and inputs required for collaboration. Here, $s \in \mathcal{S}$ represents one of three collaboration settings:

$$\mathcal{S} = \{\text{Sequential}, \text{Parallel}, \text{Integration}\}. \tag{1}$$

Sequential Dependency requires agents to complete dependent steps in order; *Parallel Decomposition* allows independent subtasks to be processed concurrently; and *Information Integration* requires agents to combine and reconcile findings from different sources. These settings describe how agents need to collaborate without prescribing their organization.

Multi-Agent Organizations. Given $\mathcal{T}_{x,s}$, an organization is defined as

$$O = (G, R, P, I, B), \tag{2}$$

where $G = (V, E)$ is the communication graph, R denotes agent roles, P assigns roles to agents, I assigns initial task inputs, and B is the execution budget. Each vertex represents an agent, and edges specify permitted communication. At round t, agent i produces

$$h_i^{(t)} = \mathcal{M}\left(q, r_i, \iota_i, \{h_j^{(t-1)} : j \in \mathcal{N}_G(i)\}; B\right), \tag{3}$$

where $\mathcal{M}$ is the underlying LLM, r_i and ι_i are the assigned role and task inputs, and $\mathcal{N}_G(i)$ denotes the communication neighbors. A designated agent combines the accessible outputs to produce the final answer.

Organization Paradigms. We consider three paradigms. *Single-Agent Reasoning* uses one agent with access to all task inputs. *Fixed Organization* adopts a predefined graph, including Star, Chain, Tree, Ring, Mesh, and Community. *Learned Organization* constructs task-dependent communication graphs, represented by G-Designer, ARG-Designer, and GoAgent [1, 10, 15]. These paradigms support comparisons under a unified evaluation setting.

2.2 Related Work

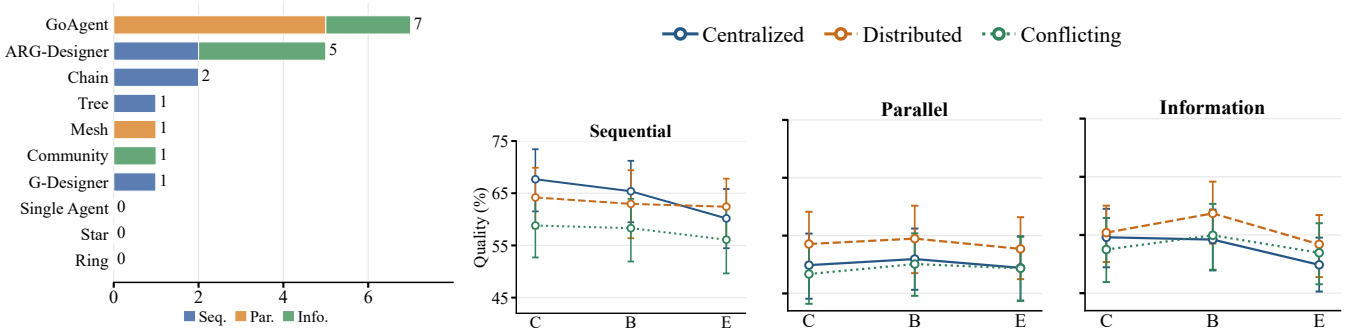
Recent benchmarks have begun to evaluate how LLM agents collaborate rather than only whether they complete tasks. MultiAgentBench compares representative communication structures, including Star, Chain, and Tree [18], while AgentsNet examines coordination across different network structures and agent scales [6]. More recent benchmarks investigate distributed coordination (SILO-BENCH), Web-agent collaboration (AgentWebBench), and collaboration under partial observability (MASBench) [4, 16, 17]. However, these benchmarks mainly focus on specific environments or collaboration mechanisms, providing limited understanding of how organization preferences change across heterogeneous application domains and different role and input assignments. OpenDGO addresses this gap through controlled cross-domain comparisons to understand which organizations are effective under different application conditions and why.

3 Motivating Observations

Existing studies provide limited understanding of whether the same organization works across different application domains and how role and input assignments affect its performance. We examine these questions through two controlled analyses, as shown in Fig. 1.

Observation 1: Organization preference varies across applications. As shown in Fig. 1a, GoAgent and ARG-Designer win 12 of 18 domain–task conditions, while fixed organizations retain five wins. Among fixed organizations, Tree achieves the highest average quality on Sequential Dependency (62.6%), whereas Community leads on Parallel Decomposition (67.0%) and Information Integration (66.9%). These results show that no organization consistently dominates across applications with different collaboration requirements.

Observation 2: Communication graph alone is insufficient. We fix the six-agent Tree and vary role and input assignments. As shown in Fig. 1b–1d, Sequential tasks favor Center + Centralized (67.7%), dropping to 60.2% when the key role moves to Edge. Parallel Decomposition and Information Integration instead favor Bridge + Distributed (54.5% and 58.7%). Thus, organization effectiveness



(a) Organization wins. (b) Sequential tasks. (c) Parallel tasks. (d) Information Integration.

Figure 1: Organization preferences and role–input assignments. (a) Organization wins across 18 domain–task conditions. (b–d) Effects of role and input assignments in a fixed Tree. Whiskers show intervals; C, B, and E denote Center, Bridge, and Edge.

depends not only on the communication graph but also on how roles and inputs are assigned to agents.

4 Organization Design and Analysis

OpenDGO aims to understand when a multi-agent organization is effective and why. We separate the collaboration required by a task from the organization used to support it, and relate controlled organization changes to communication behavior and task performance. Fig. 2 summarizes the overall framework.

4.1 Task Construction

Source Tasks. OpenDGO covers six domains: Coding, Finance, Biomedical, Legal, Scientific Research, and Recommendation. For each source instance, we preserve its original question, answer target, task inputs, and evaluator, while dividing the required work among six domain-specific roles with corresponding input assignments.

Collaboration Structures. Each task is instantiated under three representative settings. *Sequential Dependency* requires agents to complete dependent steps in order. *Parallel Decomposition* distributes independent subtasks across agents before combining their outputs. *Information Integration* requires agents to combine and reconcile information from different sources. These settings specify how agents collaborate without prescribing their communication graph.

Task Dependencies. We represent the required collaboration as a task dependency graph $H_{x,s}$, whose vertices denote agent outputs and edges represent their dependencies. It specifies which outputs are needed by subsequent agents without determining how those agents communicate, allowing the same task requirements to be evaluated under different organizations.

4.2 Controlled Organization Design

Organization Space. Following Sec. 2.1, an organization $O = (G, R, P, I, B)$ specifies the communication graph, agent roles, role assignment, input assignment, and execution budget. We consider six fixed organizations—Star, Chain, Tree, Ring, Mesh, and Community—together with three learned methods: G-Designer, ARG-Designer, and GoAgent. A Single-Agent setting with access to all task inputs serves as the reference.

Controlled Interventions. OpenDGO examines organizational factors under matched source tasks, agent models, tools, and evaluation protocols. Varying G compares communication graphs, while fixing G and changing P or I isolates the effects of role and input assignments. We further examine graph properties through matched graph variants and evaluate different resource budgets and component failures. Other conditions are held fixed within each comparison.

Execution Traces. Agents exchange messages only through permitted edges, and a designated agent produces the final answer from the inputs and messages available to it. Each execution records communication paths, intermediate outputs, message rounds, and token usage, allowing us to examine how different organizations operate.

4.3 Graph-Based Collaboration Analysis

To explain organization effectiveness, we examine three aspects of communication: information delivery, delivery resilience, and communication burden.

Information Delivery. Let $\mathcal{U}^*$ denote the set of task inputs required for the final decision, and $A(u)$ the agents initially holding input u. The minimum number of communication rounds needed for all required inputs to reach the final decision agent is bounded below by

$$L_{\text{route}} = \max_{u \in \mathcal{U}^*} \min_{v \in A(u)} \text{dist}_G(v, v_{\text{syn}}), \tag{4}$$

where v_{syn} denotes the final decision agent. This shows that delivery efficiency depends on both communication paths and input assignment.

Delivery Resilience. Communication paths may become unavailable when agents or connections fail. For the agents S^* that uniquely hold required inputs, we characterize their connection to the final decision agent by

$$\lambda^* = \min_{u \in S^*} \lambda_G(u, v_{\text{syn}}), \tag{5}$$

where λ_G denotes edge connectivity. More alternative paths can improve resilience against connection failures, although they cannot

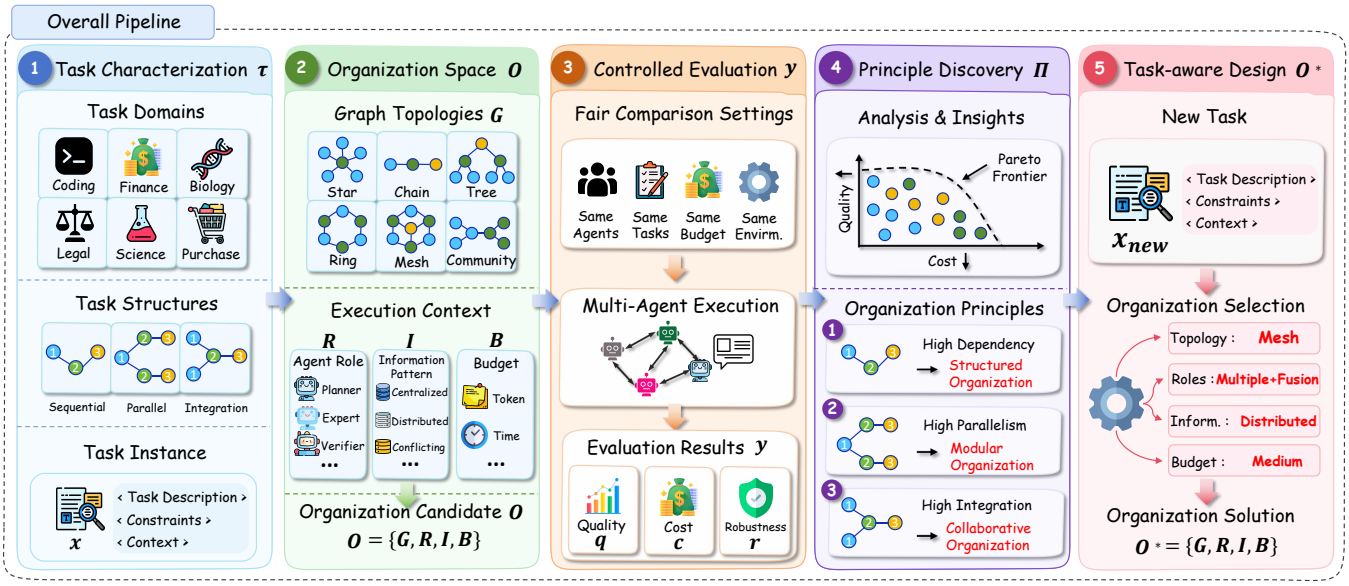


Figure 2: Overview of the OpenDGO framework. Task characterization, organization design, and controlled evaluation inform principle discovery and task-aware organization selection.

compensate for the failure of an agent holding a unique required input.

Communication Burden. Additional connections may shorten communication paths but increase message processing. Assuming messages are sent to all permitted neighbors, the total message volume is

$$C_{\text{msg}} = \sum_{t=0}^{T-1} \sum_{i \in V} \deg_G(i)\, \ell(h_i^{(t)}), \tag{6}$$

where $\ell(h_i^{(t)})$ denotes the length of the message generated by agent i at round t. Dense graphs may therefore reduce communication distance while increasing redundant messages and processing costs.

Interaction Behavior. Graph properties describe possible communication paths, whereas execution traces reveal how agents actually communicate. We measure input coverage, propagation latency, duplicate-message rate, conflict resolution, cross-group communication, and final aggregation load to examine how organizational differences translate into task performance.

4.4 Evaluation Protocol

Task Quality. Each domain retains its original evaluation metrics, including resolve rate, exact match or accuracy, claim-verification accuracy, and Hit@1 where appropriate.

Coordination Behavior. Input coverage and propagation latency measure information delivery; duplicate-message rate and aggregation load capture communication overhead; conflict resolution and cross-group communication characterize collaboration between agents.

Cost and Reliability. Token usage, message volume, and execution time measure organization cost. Controlled agent and communication-link failures evaluate the ability of different organizations to maintain performance under disruptions.

5 Experiments and Analyses

We examine how organization choice depends on task demands, input placement, and deployment constraints through four questions. **Q1:** How do single-agent, fixed, and learned organizations compare across domains and task structures? **Q2:** How do role placement and information allocation affect a fixed topology? **Q3:** Which structural properties distinguish useful organizations for different tasks? **Q4:** How does organization choice change with token budgets, component failures, and unreliable inputs? These analyses progress from comparative effectiveness to organizational factors, structural explanation, and deployment trade-offs.

5.1 Experimental Setup

Datasets and evaluation. We cover Coding with SWE-bench Verified [3, 9], Finance with FinQA [2], Biomedical with BioASQ Task B [11], Legal with LegalBench [7], Scientific Research with SciFact [12], and Recommendation with Amazon Reviews 2023 [8]. The domain-specific scores include resolve rate, exact match or accuracy, claim-verification accuracy, and Hit@1. Each domain covers Sequential Dependency, Parallel Decomposition, and Information Integration.

Methods and comparison protocol. We compare Single Agent; six fixed organizations (Star, Chain, Tree, Ring, Mesh, and Community); and G-Designer [15], ARG-Designer [10], and GoAgent [1]. The single agent receives the complete set of task inputs available collectively to the six-agent systems. The target protocol pairs 500 tasks per domain–structure cell across methods, with the same model, tools, budget, stopping rule, scorer, and five seeds. Construction and execution details are in Appendix A.

Supplementary analyses. We supplement Q1 with a backbone-transfer analysis and Q4 with theoretical cost accounting and an input perturbation stress test. Table 2 compares organization choices

Table 1: Organization benchmark (mean ± SD (%)). Bold and underlining mark the best and second-best result within each task-structure panel and domain.

Category	Method	Task Quality (%)						
		Code	Fin.	Bio.	Legal	Sci.	Rec.	Avg.
(a) Sequential Dependency								
Single	Single Agent	28.2±1.9	52.6±2.6	59.6±2.5	65.2±3.7	71.2±1.1	37.2±2.8	52.3±1.4
Fixed	Star	24.8±1.3	48.8±3.1	55.8±1.8	61.8±3.0	67.6±3.5	34.0±2.8	48.8±0.6
	Chain	**37.6**±2.7	**62.4**±5.8	67.6±3.0	73.6±3.2	78.4±3.7	44.4±4.9	60.7±0.9
	Tree	34.2±1.7	60.6±3.0	**71.2**±2.1	77.8±2.6	81.2±2.7	50.8±3.8	62.6±1.4
	Ring	29.4±2.8	53.2±4.5	61.0±2.4	66.0±3.1	71.8±3.7	38.2±4.3	53.3±1.9
	Mesh	31.0±2.5	56.8±5.0	64.8±2.9	70.2±3.6	75.6±4.1	42.6±6.1	56.8±1.8
	Community	35.8±4.3	59.4±4.8	69.4±5.4	76.4±4.1	79.8±4.3	48.8±5.1	61.6±2.2
Learned	G-Designer	37.2±4.2	62.0±6.0	71.0±2.8	78.2±5.0	**82.6**±6.9	51.2±5.1	63.7±2.5
	ARG-Designer	37.4±5.1	62.2±6.9	70.8±3.5	**79.0**±4.9	82.2±6.1	**52.0**±5.8	**63.9**±1.8
	GoAgent	36.6±3.7	61.4±6.7	70.4±5.5	77.6±4.0	81.4±4.1	50.4±6.1	63.0±1.6
(b) Parallel Decomposition								
Single	Single Agent	27.4±1.3	49.0±2.6	55.2±2.2	61.2±2.8	69.8±2.8	38.0±4.6	50.1±1.3
Fixed	Star	28.6±3.7	50.4±4.8	57.4±2.2	64.0±2.5	69.8±2.5	41.6±3.4	52.0±1.5
	Chain	21.8±3.6	43.0±6.1	49.8±2.1	55.8±5.3	61.6±4.1	29.0±3.6	43.5±1.1
	Tree	27.0±2.8	49.2±3.1	56.2±3.1	62.2±4.5	71.8±2.8	36.6±3.7	50.5±1.3
	Ring	33.0±3.5	57.0±4.3	64.4±3.3	68.8±4.1	77.4±4.8	46.2±5.5	57.8±1.0
	Mesh	41.6±6.5	62.8±5.3	71.8±3.4	76.2±4.3	**84.0**±4.0	56.8±5.7	65.5±2.1
	Community	39.4±3.3	64.6±5.6	74.6±3.7	79.4±4.9	82.6±4.8	61.6±5.6	67.0±1.4
Learned	G-Designer	42.2±5.4	65.0±6.8	75.2±4.7	80.2±5.3	83.8±5.6	62.0±6.6	68.1±1.6
	ARG-Designer	42.8±3.9	65.8±5.1	76.0±5.7	80.8±4.6	83.6±5.6	62.8±5.0	68.6±2.0
	GoAgent	**43.6**±4.7	**66.6**±6.3	**76.6**±4.9	**81.6**±6.2	83.8±2.2	**63.6**±5.8	**69.3**±2.3
(c) Information Integration								
Single	Single Agent	27.6±2.8	49.8±4.1	57.2±2.0	62.2±2.7	67.0±2.9	35.2±5.7	49.8±0.6
Fixed	Star	25.2±4.5	45.4±4.4	52.4±3.5	57.6±4.0	61.2±3.3	31.2±5.7	45.5±0.7
	Chain	23.2±4.2	43.2±5.4	50.2±4.3	55.4±3.0	59.2±4.5	29.0±5.3	43.4±1.7
	Tree	30.4±1.2	52.4±4.9	60.4±4.2	65.2±1.8	69.6±4.6	39.0±6.2	52.8±0.8
	Ring	33.2±3.8	56.6±5.7	65.2±3.3	69.8±4.6	74.8±5.4	45.6±4.7	57.5±1.5
	Mesh	40.2±4.2	63.4±5.8	72.8±4.8	77.8±4.7	83.6±3.8	55.8±5.2	65.6±1.6
	Community	41.0±3.7	64.2±5.4	74.2±4.1	79.0±4.3	83.0±6.1	**60.2**±7.1	66.9±1.1
Learned	G-Designer	42.0±5.5	65.4±6.5	75.6±5.1	80.6±5.2	84.6±5.9	59.8±5.0	68.0±2.4
	ARG-Designer	42.6±5.5	**66.0**±6.1	**76.2**±6.1	**81.2**±7.4	85.0±5.3	60.0±5.2	**68.5**±2.8
	GoAgent	**43.2**±6.0	65.6±7.2	75.8±5.8	80.8±5.9	**85.8**±6.2	59.6±5.6	**68.5**±0.8

and transfer regret across backbones; Fig. 8 compares quality retention under four types of input perturbations. Table 6 in Appendix B.8 reports analytical costs rather than measured latency or memory.

5.2 Organization Effectiveness (Q1)

To address **Q1**, we evaluate ten methods across six domains and three collaboration settings, and further examine whether organization preferences transfer across LLM backbones.

Overall Performance. Table 1 shows that ARG-Designer leads on Sequential Dependency (63.9%), GoAgent on Parallel Decomposition (69.3%), and both on Information Integration (68.5%), outperforming the best fixed organizations by 1.3, 2.3, and 1.6 percentage points, respectively. Among fixed organizations, Tree leads on Sequential tasks, while Community performs best on Parallel and Integration tasks. However, no organization dominates across domains: learned methods win 13 of 18 conditions (Fig. 1a), while domain-ranking similarities range from 0.07 to 0.72 (Fig. 4a). These results highlight substantial variation in organization preferences across applications.

Backbone Sensitivity and Transfer. We assess cross-backbone generalization using six fixed organizations under matched roles, inputs, tools, and execution policies. Let $\bar{Q}_{M,s}(o)$ denote the average quality of organization o under backbone M and setting s, and $o^*_{M,s} = \arg\max_o \bar{Q}_{M,s}(o)$. Transfer regret is defined as

$$\mathrm{Regret}(M_0 \to M) = \frac{1}{3}\sum_{s\in\mathcal{S}}\left[\bar{Q}_{M,s}(o^*_{M,s}) - \bar{Q}_{M,s}(o^*_{M_0,s})\right]. \quad (7)$$

Table 2 shows that preferred organizations vary across backbones, with transfer regrets of 0.43 and 0.20 percentage points for Llama and Qwen-2.5-32B, respectively. These modest losses suggest that organization preferences can change with model capabilities without substantial performance degradation. Details are provided in Appendix B.2.

5.3 Role and Input Assignment (Q2)

To address **Q2**, we fix the six-agent Tree and vary expert placement (Center, Bridge, Edge) and input assignment (Centralized, Distributed, Conflicting), while keeping other settings unchanged.

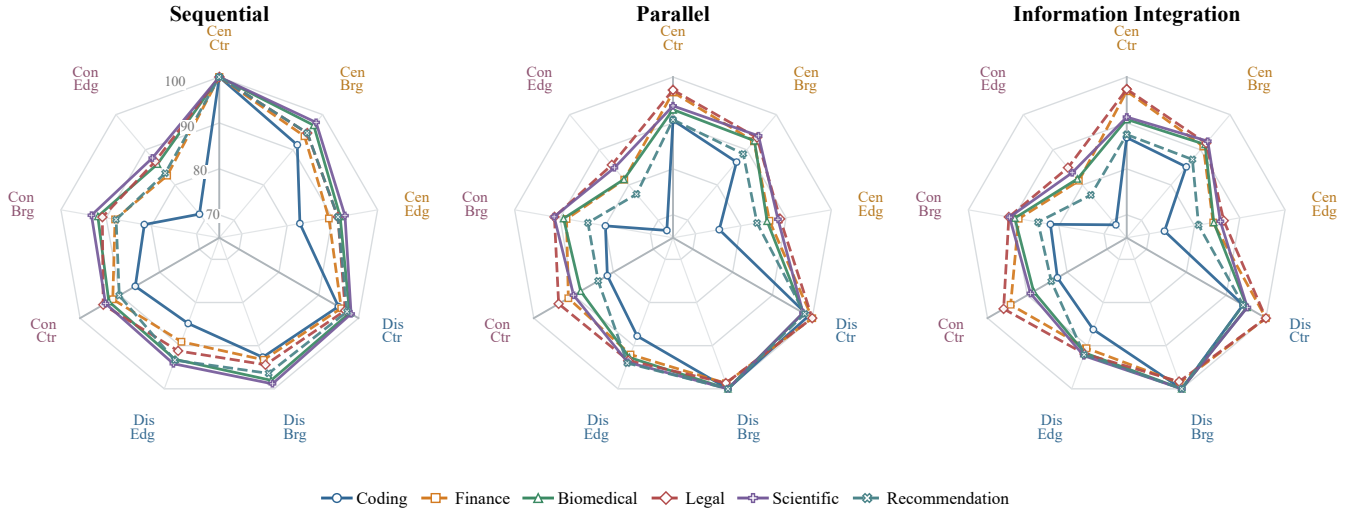


Figure 3: Domain-specific role–input configurations in a fixed Tree. Each spoke combines input assignment (Cen: Centralized, Dis: Distributed, Con: Conflicting) and expert placement (Ctr: Center, Brg: Bridge, Edg: Edge).

Task-dependent configurations. Fig. 1(b–d) shows that Sequential Dependency favors Center + Centralized (67.7%), dropping to 60.2% with Edge placement. Parallel Decomposition and Information Integration instead favor Bridge + Distributed (54.5% and 58.7%). These results indicate task-dependent preferences, although some intervals overlap.

Domain variation. Fig. 3 compares nine configurations across domains. Center + Centralized leads on Sequential tasks, while Distributed inputs perform better on Parallel and Integration tasks. For the latter, Finance and Legal favor Center, whereas the other four domains favor Bridge. These findings highlight the joint importance of role and input assignments. Additional results and normalization details are provided in Appendices B.4 and B.3.

5.4 Structure and Task Demands (Q3)

To address **Q3**, we examine how graph properties relate to organization effectiveness. Fig. 4 reports domain-ranking similarity and standardized structural associations across collaboration settings. Colored intervals indicate positive or negative associations, while gray intervals cross zero. Details are provided in Appendices B.5 and B.6.

Structural Differences. Fig. 5 shows that Star, Chain, and Tree share the same density (0.33) but differ in diameter (2, 5, and 4). Mesh minimizes communication distance, whereas Community exhibits stronger subgroup separation (modularity 0.36). Thus, graph density alone cannot explain organization effectiveness.

Task-Specific Associations. As shown in Fig. 4b, Sequential Dependency is associated with shorter diameter (−0.38) and higher connectivity (+0.22), favoring reliable information transfer. Parallel Decomposition shows positive associations with modularity (+0.34) and clustering (+0.27), supporting subgroup collaboration. Information Integration instead benefits from connectivity (+0.42) and cycle redundancy (+0.36), with a negative association for diameter (−0.19). These results reveal task-dependent structural preferences, although the associations do not establish causality.

5.5 Quality, Cost, and Robustness (Q4)

To address **Q4**, we examine organization efficiency and robustness under different budgets, component failures, and input perturbations, with protocols detailed in Appendices B.7 and B.9.

Theoretical Costs. Table 6 in Appendix B.8 compares computation, communication, and memory costs. For u agents, v edges, T rounds, average input length b, and message length ℓ, the execution cost is approximated by

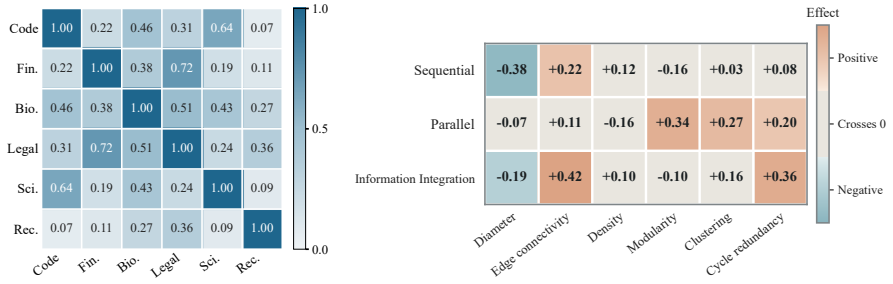
$$\mathcal{X}(u,v) = \Theta\left(uT\, C_M\left(b + \frac{2v}{u}\ell\right)\right), \tag{8}$$

where $C_M(s)$ denotes the cost of an LLM call with s input tokens. Mesh has three times the edge-based message volume of Star, Chain, or Tree under matched rounds and message lengths, while learned organizations introduce additional graph-generation costs. Detailed complexity and memory analyses are provided in Appendix B.8.

Quality–Cost Trade-offs. Fig. 6 shows different Pareto-optimal organizations: Star–Chain–Tree for Sequential, Star–Ring–Community for Parallel, and Star–Tree–Ring–Community for Information Integration. Community achieves 67.0% and 66.9% on the latter two settings with 15–16k tokens, outperforming Mesh despite its higher cost of 21.7–22.4k tokens.

Failure Robustness. Fig. 7 shows that quality declines as agent or communication-link failures increase. At 50% failure pressure, Mesh and Ring retain the highest relative quality, while Star, Chain, and Tree degrade more sharply. Thus, greater connectivity can improve robustness at the cost of additional communication.

Input Robustness. We further test Information Integration under missing, distracting, conflicting, and targeted input perturbations, with corruption ratios $p \in \{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$. Quality



(a) Domain-ranking similarity. **(b) Graph-property effects.**

Figure 4: Domain rankings and structural effects. (a) Mean Kendall's τ_b; (b) standardized partial effects on quality.

Backbone	Best Seq.	Best Par.	Best Info.	Regret (pp) ↓
Qwen3-8B -Instruct	**Tree** (62.6)	**Community** (67.0)	**Community** (66.9)	0.00
Llama-3.1-8B -Instruct	**Chain** (43.8)	**Community** (48.6)	**Mesh** (47.2)	0.43
Qwen2.5-1.5B -Instruct	**Tree** (48.9)	**Mesh** (54.1)	**Community** (52.8)	0.20

Table 2: Backbone choice and transfer regret.

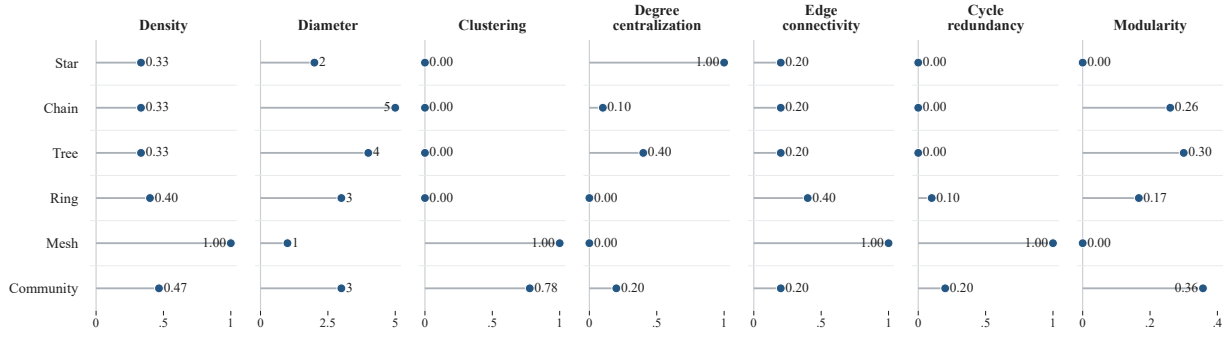


Figure 5: Structural profiles of six organizations. Markers show raw graph statistics on property-specific scales; larger values do not necessarily indicate better performance.

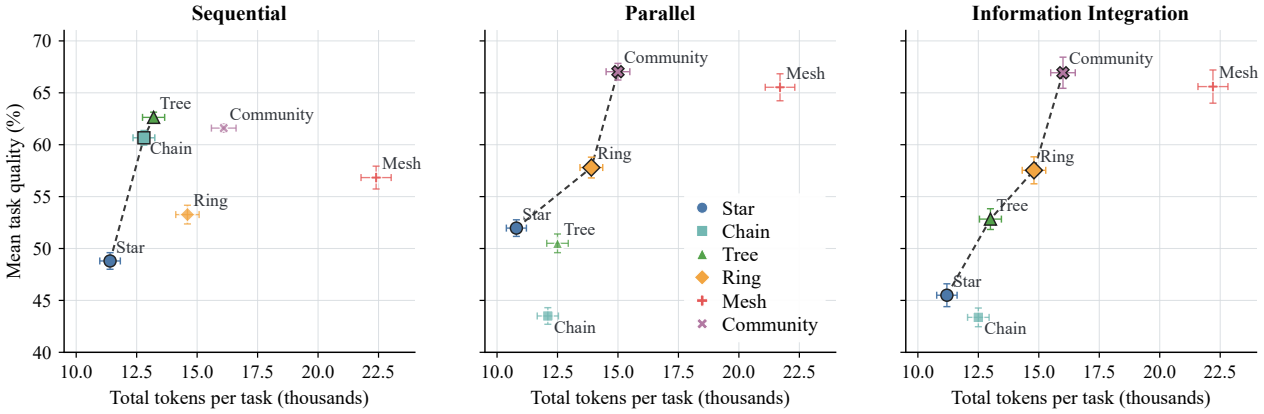


Figure 6: Quality–token Pareto trade-offs. Points show means, bars indicate one sample SD over five runs, and dashed lines connect non-dominated means.

retention is defined as

$$\mathrm{Ret}_{o,d,c}(p) = 100\,\frac{Q_{o,d,c}(p)}{Q_{o,d}(0)}, \tag{9}$$

where o, d, and c denote organization, domain, and perturbation type. As shown in Fig. 8, Mesh retains 87.3% under missing inputs and 81.0% under targeted corruption at $p = 0.5$, while Community leads against distracting and conflicting information (84.5% and 75.8%). These results suggest that communication redundancy improves tolerance to missing inputs but does not necessarily prevent the spread of incorrect information. Details are provided in Appendix B.9.

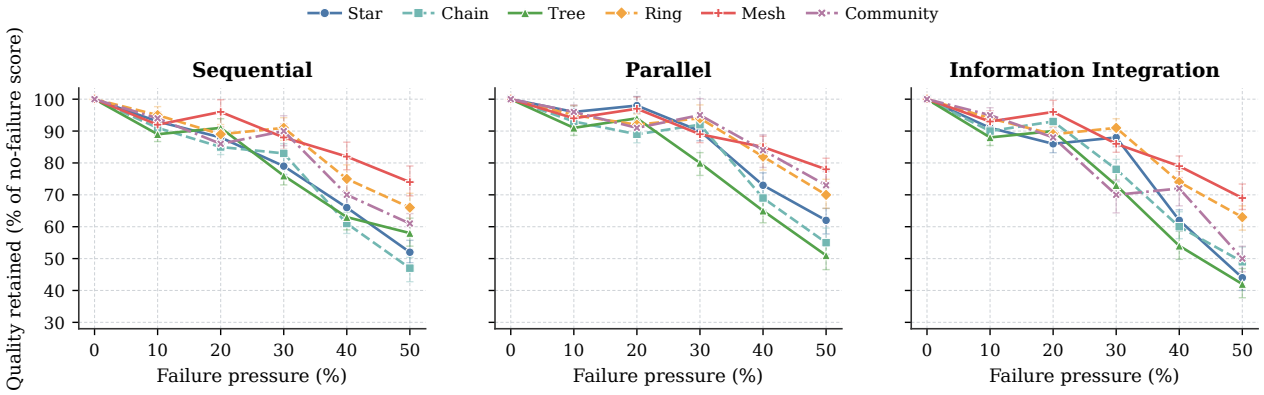


Figure 7: Quality retained under component failures. Results are normalized by no-failure quality; curves show means and bands indicate one sample SD.

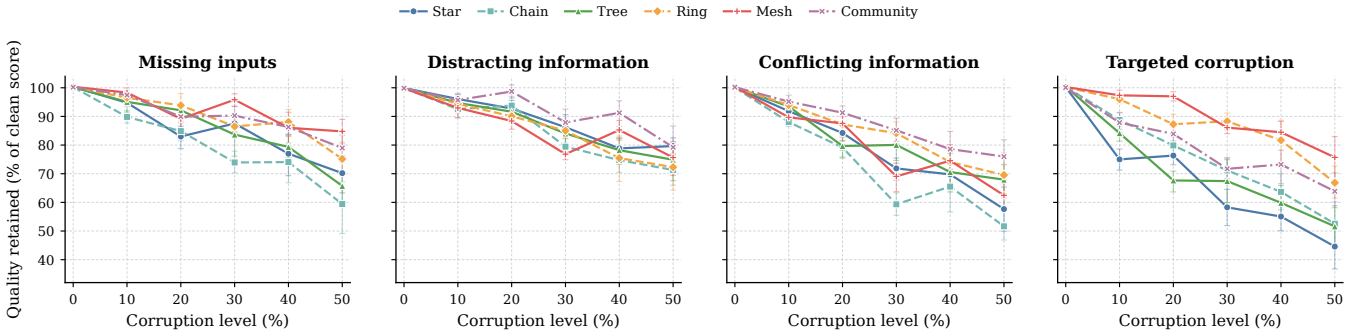


Figure 8: Input robustness under Information Integration. Four settings examine missing inputs, distracting information, conflicting information, and targeted input corruption. Curves show six-domain mean quality retention with 95% bootstrap intervals.

6 Conclusion and Future Directions

In this work, we present **OpenDGO** to systematically study how LLM-based multi-agent systems should be organized. Across six real-world domains and three representative collaboration settings, we compare fixed and learned organizations and further examine role placement, information allocation, graph structure, resource budgets, and component failures. Our results show that organization effectiveness is strongly conditioned on the collaboration required by the task: no single topology consistently dominates, and the same graph can behave differently when roles or information are rearranged. Based on these findings, we highlight several directions for future multi-agent organization research.

Task-Conditioned Organization. Our results suggest that searching for a universally strong communication topology is unlikely to be sufficient. Sequential Dependency, Parallel Decomposition, and Information Integration tasks impose different information-flow requirements and therefore favor different structural properties. Future systems should infer collaboration demand from the task and adapt their organization accordingly, rather than applying the same topology to every instance.

Organization Beyond Topology. Communication structure alone does not determine team effectiveness. Role placement and information allocation can substantially change the behavior of the same graph, indicating that topology, capability placement, and information distribution should be considered jointly. Future organization designers should therefore move beyond optimizing only G toward jointly reasoning over (G, P, I) and, when necessary, resource constraints B.

Behavior-Grounded Organization Design. Static graph statistics provide only an incomplete explanation of multi-agent collaboration. Our analysis shows that structural properties matter through their effects on information delivery, propagation delay, redundancy, and information integration during execution. Future methods could directly use message-level traces and information-flow signals to design organizations, so that communication structures are optimized for the collaboration they actually induce.

Resource-Aware Adaptive Organization. Higher connectivity can improve information access and failure tolerance, but it also increases communication and synthesis costs. The preferred organization may therefore change with token budgets, latency requirements, or component availability. A promising direction is dynamic organization, where communication remains sparse

when local reasoning is sufficient, expands when distributed information must be integrated, and reconfigures when agents or communication links become unavailable.

References

[1] Hongjiang Chen, Xin Zheng, Yixin Liu, Pengfei Jiao, Shiyuan Li, Huan Liu, Zhidong Zhao, Ziqi Xu, Ibrahim Khalil, and Shirui Pan. 2026. GoAgent: Group-of-Agents Communication Topology Generation for LLM-based Multi-Agent Systems. arXiv:2603.19677 [cs.LG] https://arxiv.org/abs/2603.19677

[2] Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 2021. FinQA: A Dataset of Numerical Reasoning over Financial Data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 3697–3711. doi:10.18653/v1/2021.emnlp-main.300

[3] Neil Chowdhury, James Aung, Chan Jun Shern, Oliver Jaffe, Dane Sherburn, Giulio Starace, Evan Mays, Rachel Dias, Marwan Aljubeh, Mia Glaese, Carlos E. Jimenez, John Yang, Leyton Ho, Tejal Patwardhan, Kevin Liu, and Aleksander Madry. 2024. Introducing SWE-bench Verified. https://openai.com/index/introducing-swe-bench-verified/.

[4] Qizhi Chu, Zekai Yu, Sijie Wen, Yang Liu, Chen Qian, Cheng Yang, Chuan Shi, and Zhiyuan Liu. 2026. MASBench: Benchmarking LLM-based Multi-Agent Collaboration under Partial Observability. arXiv:2610.04672 [cs.AI] https://arxiv.org/abs/2610.04672

[5] Reinhard Diestel. 2025. *Graph Theory* (6 ed.). Graduate Texts in Mathematics, Vol. 173. Springer, Heidelberg, Germany. https://diestel-graph-theory.com/

[6] Florian Grötschla, Luis Müller, Jan Tönshoff, Mikhail Galkin, and Bryan Perozzi. 2025. AgentsNet: Coordination and Collaborative Reasoning in Multi-Agent LLMs. arXiv:2507.08616 [cs.MA] https://arxiv.org/abs/2507.08616

[7] Neel Guha, Julian Nyarko, Daniel E. Ho, Christopher Ré, Adam Chilton, Aditya K, Alex Chohlas-Wood, Austin Peters, Brandon Waldon, Daniel Rockmore, Diego Zambrano, Dmitry Talisman, Enam Hoque, Faiz Surani, Frank Fagan, Galit Sarfaty, Gregory Dickinson, Haggai Porat, Jason Hegland, Jessica Wu, Joe Nudell, Joel Niklaus, John Nay, Jonathan Choi, Kevin Tobia, Margaret Hagan, Megan Ma, Michael Livermore, Nikon Rasumov-Rahe, Nils Holzenberger, Noam Kolt, Peter Henderson, Sean Rehaag, Sharad Goel, Shang Gao, Spencer Williams, Sunny Gandhi, Tom Zur, Varun Iyer, and Zehua Li. 2023. LegalBench: A Collaboratively Built Benchmark for Measuring Legal Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*, Vol. 36. Neural Information Processing Systems Foundation, Inc., New Orleans, Louisiana, USA, 44123–44279. doi:10.52202/075280-1915

[8] Yupeng Hou, Jiacheng Li, Xiangjun Fu, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. 2026. Bridging Language and Items for Retrieval and Recommendation: Benchmarking LLMs as Semantic Encoders. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, San Diego, California, United States, 3251–3265. doi:10.18653/v1/2026.acl-long.147

[9] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. SWE-bench: Can Language Models Resolve Real-World GitHub Issues?. In *International Conference on Learning Representations*. International Conference on Learning Representations, Vienna, Austria, 51 pages. https://proceedings.iclr.cc/paper_files/paper/2024/hash/edac78c3e300629acfe6cbe9ca88fb84-Abstract-Conference.html

[10] Shiyuan Li, Yixin Liu, Qingsong Wen, Chengqi Zhang, and Shirui Pan. 2026. Assemble Your Crew: Automatic Multi-agent Communication Topology Design via Autoregressive Graph Generation. *Proceedings of the AAAI Conference on Artificial Intelligence* 40, 28 (2026), 23142–23150. doi:10.1609/aaai.v40i28.39481

[11] George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R. Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, Yannis Almirantis, John Pavlopoulos, Nicolas Baskiotis, Patrick Gallinari, Thierry Artiéres, Axel-Cyrille Ngonga Ngomo, Norman Heino, Eric Gaussier, Liliana Barrio-Alvers, Michael Schroeder, Ion Androutsopoulos, and Georgios Paliouras. 2015. An Overview of the BIOASQ Large-Scale Biomedical Semantic Indexing and Question Answering Competition. *BMC Bioinformatics* 16 (2015), 138. doi:10.1186/s12859-015-0564-6

[12] David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. 2020. Fact or Fiction: Verifying Scientific Claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Online, 7534–7550. doi:10.18653/v1/2020.emnlp-main.609

[13] Kairui Yang, Minghao An, Xunkai Li, Ziheng Yi, Zekai Chen, Guangyuan He, and Rong-Hua Li. 2026. MACE: Memory-Agent Co-Evolution with Adaptive Memory Graphs for Multi-Agent Systems. arXiv:2609.21533 [cs.LG] https://arxiv.org/abs/2609.21533

[14] Kairui Yang, Ziheng Yi, Xunkai Li, Minghao An, Zhanke Liu, Zekai Chen, and Rong-Hua Li. 2026. MAGIC: Mixed-Granularity Agent Graphs via Incremental Construction with Dense-Reward Reinforcement Learning. arXiv:2609.26667 [cs.LG] https://arxiv.org/abs/2609.26667

[15] Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. 2025. G-Designer: Architecting Multi-agent Communication Topologies via Graph Neural Networks. In *Proceedings of the 42nd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 267)*. PMLR, Vancouver, Canada, 76678–76692. https://proceedings.mlr.press/v267/zhang25cu.html

[16] Yuzhe Zhang, Feiran Liu, Yi Shan, Xinyi Huang, Xin Yang, Yueqi Zhu, Xuxin Cheng, Cao Liu, Ke Zeng, Terry Jingchen Zhang, and Wenyuan Jiang. 2026. SILO-BENCH: A Scalable Environment for Evaluating Distributed Coordination in Multi-Agent LLM Systems. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, San Diego, California, United States, 29379–29398. doi:10.18653/v1/2026.acl-long.1354

[17] Shanshan Zhong, Kate Shen, and Chenyan Xiong. 2026. AgentWebBench: Benchmarking Multi-Agent Coordination in Agentic Web. In *Proceedings of the 43rd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 306)*. PMLR, Seoul, South Korea, 164502–164544. https://proceedings.mlr.press/v306/zhong26n.html

[18] Kunlun Zhu, Hongyi Du, Zhaochen Hong, Xiaocheng Yang, Shuyi Guo, Zhe Wang, Zhenhailong Wang, Cheng Qian, Xiangru Tang, Heng Ji, and Jiaxuan You. 2025. MultiAgentBench: Evaluating the Collaboration and Competition of LLM Agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vienna, Austria, 8580–8622. doi:10.18653/v1/2025.acl-long.421

A Methodology in Detail

A.1 Benchmark Unit and Estimand

For organization a, let $Y(T_{x,s}, a, M, z) \in [0, 1]$ be the domain scorer's output for source task x, structural view s, shared model M, and seed z. The paired contrast against organization b is

$$\widehat{\Delta}_{d,s}(a, b) = \frac{1}{|\mathcal{X}_{d,s}|K} \sum_{x \in \mathcal{X}_{d,s}} \sum_{z=1}^{K} \big[Y(T_{x,s}, a, M, z) - Y(T_{x,s}, b, M, z) \big]. \tag{10}$$

Pairs share the task, seed, combined inputs, model, tools, scorer, and nominal limits. Source-ID cluster resampling preserves all paired configurations and structural views. Comparisons of native frameworks additionally vary their execution policies; these are system comparisons rather than topology-only contrasts. A framework enters the controlled graph comparison only if its role set, input boundary, scheduling, and synthesis policy can be matched.

A.2 Task Construction and Structural Matching

Each package records the dataset version, split, instance and source IDs, selection rule, and a hash of frozen model-visible input. Gold answers, hidden tests, and support labels remain evaluator-only. Where redistribution is restricted, IDs, conversion rules, and hashes provide the reconstruction record. Selection and conversion precede held-out scoring.

Sequential packages specify ordered intermediate products; Parallel packages specify independently analyzable views and their integration; Information Integration packages partition complementary units and, where applicable, verified conflicting or distracting units. The imposed dependency graph describes the conversion, not the source dataset's original purpose. A conflicting view retains a valid answer path. Packages are screened for an automatic target and sufficient information in the union of their views. Difficulty stratum, input length, input-unit count, intermediate decisions, and source category are recorded. Cross-structure comparisons use matched sources where possible; different source compositions are recorded otherwise.

Role responsibilities are domain-specific: Coding separates issue analysis, navigation, implementation, testing, regression review, and integration; Finance separates question, table, and text analysis, calculation, verification, and integration; Biomedical separates question typing, retrieval, extraction, reasoning, fact checking, and integration. Legal uses issue, fact, and rule analysis, exception review, counterargument review, and integration; Scientific Research uses claim analysis, retrieval, support and refutation analysis, rationale checking, and integration; Recommendation uses profiling, review and metadata analysis, scoring, constraint review, and integration. Private views determine access independently of role titles.

A.3 Six-Agent Communication Graphs

All fixed organizations use $V = \{0, 1, 2, 3, 4, 5\}$. An undirected edge permits messages in both directions. Table 4 fixes the edge sets; graph labels do not imply different prompts or models.

The Community graph has two triangles joined by bridge 23. Synthesizer location and scheduling are held fixed within paired comparisons and logged with the graph ID. Learned or native systems record their realized routes and policy differences rather than inheriting the fixed-graph assumptions.

A.4 Execution and Information Boundaries

Equation 3 applies after initialization $h_i^{(0)} = F_M(q, r_i, e_i, \varnothing; B)$. Messages expose only neighbor outputs; there is no unrestricted transcript at synthesis. Logs identify sender, recipient, round, delivered content or its hash, and token usage. Tool events and shared state are recorded separately. Coding uses a frozen repository snapshot and submits a patch to the official evaluator; any shared workspace access is outside the pure message-routing bound.

Final answers use a frozen schema and evaluator. Token accounting sums actual model inputs and outputs, including tool responses and final synthesis; received-message tokens already counted as inputs are not added again. Routed-token exposure in Eq. 6 is a separate diagnostic. Invalid attempted outputs remain in the scoring denominator under the domain scorer. Prompt, parser, truncation, and stopping choices are fixed before held-out comparisons.

A.5 Controlled Intervention Details

For Q2, the Tree places the designated expert at Center (vertex 0), Bridge (1 or 2), or Edge (3, 4, or 5), with a compensating role permutation. The selected vertex and synthesizer are recorded because members of a named class can have different distances. Centralized allocation bundles decisive units at a designated holder; Distributed allocation separates complementary units; Conflicting allocation separates gold-supporting and verified contradictory or distracting units. Inventory and target remain fixed, with unit source, destination, and truth status recorded.

For mean score $\bar{Y}_{d,s}(p, i)$, a position contrast holds allocation fixed; an allocation contrast holds position fixed. Their interaction is

$$\begin{aligned} \Gamma_{d,s} = &[\bar{Y}_{d,s}(C, Cen) - \bar{Y}_{d,s}(E, Cen)] \\ &- [\bar{Y}_{d,s}(C, Dis) - \bar{Y}_{d,s}(E, Dis)], \end{aligned} \tag{11}$$

where C/E mean Center/Edge and Cen/Dis mean Centralized/Distributed. Non-Tree position classes and symmetry controls are specified in Appendix B.4. Q3 graph models and trace definitions appear in Appendices B.5 and B.6; Q4 caps and fault sampling appear in Appendix B.7. Each fault record identifies the disabled component and its relationship to required sources and routes.

A.6 Derivations and Scope of the Graph Perspectives

Routing bound. Consider a support unit absent from the public question, shared context, and independently available tools. At step zero, its source-dependent information is available only to $A(e)$. Inductively, after t exchanges it can influence only vertices within distance t of $A(e)$, since each update reads neighbors' preceding states. Its provenance cannot reach v_{syn} before $\min_{u \in A(e)} \text{dist}_G(u, v_{\text{syn}})$. Taking the maximum across designated units gives Eq. 4; an unreachable unit has infinite distance. With replicated inputs the minimum is over all holders. The argument gives an earliest possible arrival, not a guarantee of faithful forwarding, conflict resolution,

Table 3: Domain-specific realizations of the three task structures. Each row starts from a source problem with a fixed answer target.

Domain	Sequential Dependency	Parallel Decomposition	Information Integration
Coding	Locate, patch, test	Inspect files or failures separately	Reconcile issue, code, and test results
Finance	Extract figures, compute, verify	Inspect table and narrative separately	Combine complementary numerical facts
Biomedical	Retrieve, extract, verify	Review distinct articles	Resolve complementary or conflicting snippets
Legal	Identify issue, apply rule	Review facts, rules, exceptions	Reconcile rule, facts, and counterarguments
Scientific	Retrieve, classify, check rationale	Review candidate abstracts	Weigh support against refutation
Rec.	Profile, filter, rank	Score candidates by distinct signals	Combine history, reviews, and metadata

Table 4: Exact fixed six-vertex graphs. Edge ij joins vertices i and j.

Graph	Edges	$\|E\|$
Star	01, 02, 03, 04, 05	5
Chain	01, 12, 23, 34, 45	5
Tree	01, 02, 13, 14, 25	5
Ring	01, 12, 23, 34, 45, 50	6
Mesh	all $\binom{6}{2}$ pairs	15
Community	01, 02, 12, 34, 35, 45, 23	7

or correct inference. A final synthesis that reads neighbors counts as an additional exchange step when reporting this bound.

Critical-cut condition. For surviving source and synthesis vertices, a cut of size $\lambda_G(u, v_{\text{syn}})$ can disconnect their information path. Fewer deletions cannot disconnect it. Taking the minimum across uniquely held required sources proves the stated condition $k < \lambda^*$. Menger's theorem provides the equivalent edge-disjoint-path characterization [5]. This protects connectivity under edge deletions; internal-vertex failures require internally vertex-disjoint paths and endpoint failures require input or decision redundancy. For example, Community has cycles within both subgroups but a single bridge between them. Information that must cross that bridge is unprotected against its deletion regardless of those local cycles.

Communication exposure. At each exchange, vertex i sends its $\ell(h_i^{(t)})$ tokens to $\deg_G(i)$ recipients. Summing gives Eq. 6, with $2T|E|$ directed deliveries if every permitted message is sent. Dropped or truncated messages reduce exposure; asynchronous or selective routing requires summing realized deliveries instead. Equal output lengths would make exposure proportional to edge count, but output lengths and reasoning quality can change with organization. Consequently, density supplies a communication opportunity measure rather than a sufficient predictor of cost or performance.

B Additional Experimental Details

B.1 Organization Generality and Domain Sensitivity

The Q1 diagnostics are displayed in the main paper: Fig. 1a complements the domain–structure results in Table 1. GoAgent wins seven conditions, ARG-Designer five, and fixed graphs five; G-Designer accounts for the remaining win. Fig. 4a orders the six fixed organizations by domain and task/seed and averages Kendall's τ_b coefficients across the 15 domain pairs. Finance–Legal has the highest displayed similarity (0.72), whereas Coding–Recommendation (0.07), Scientific–Recommendation (0.09), and Finance–Recommendation (0.11) have the lowest. These summaries do not substitute for an evaluation of transfer to unseen tasks.

B.2 Backbone-Transfer Protocol

The comparison in Table 2 changes only the execution backbone: task IDs, input inventory, six-node graph, role assignment, information allocation, synthesizer, tools, communication rounds, and scoring remain fixed. Shared word or character limits for messages and final answers avoid equating different tokenizers; actual token use is recorded separately. Exact checkpoint revisions, decoding and thinking modes, and failure/retry policies must be logged.

The best fixed organization is selected separately for each task structure using the unweighted six-domain average. Single Agent and learned designers are excluded from that argmax. Source-backbone choices should be frozen on a development set before target-backbone testing. Equation 7 then compares the target model's own best graph with the transferred graph on the same held-out tasks. Target best-graph selection should use a separate selection split where possible; otherwise its test-set argmax is an optimistic oracle reference and must be reported as such. Raw quality, per-structure regret, and paired intervals are needed alongside the mean. Source-ID cluster bootstrap retains all backbones, graphs, and seeds of a source together. Close mean scores without a paired interval do not justify a claim of a reliably different winner.

For the Llama row, the per-structure losses are 0.4, 0.0, and 0.9 pp, yielding $(0.4 + 0 + 0.9)/3 = 0.43$ pp. The Qwen-2.5-32B row has losses 0.0, 0.6, and 0.0 pp, yielding 0.20 pp. The three task structures receive equal weight in the mean.

B.3 Normalization of the Tree Configuration Profiles

To compare configuration sensitivity across domains with different score levels, Fig. 3 normalizes each domain–task profile separately. For configuration c among the nine role–information combinations, the displayed value is $100\,\bar{Y}_{d,s}(c)/\max_{c'} \bar{Y}_{d,s}(c')$, where $\bar{Y}$ is the five-seed mean. The highest value in each profile is therefore 100, and all within-profile orderings are preserved. The radial scale starts

at 65. Curve area and values across domains are not measures of absolute task quality. These profiles display means without uncertainty intervals and do not establish significant differences between nearby configurations.

B.4 Role–Information Interventions Beyond Tree

The Q2 intervention is repeated on the five remaining fixed topologies with the task set, seeds, roles, input inventory, prompts, and budget held fixed. Positions are defined by structural equivalence: Star compares Hub and Leaf; Chain compares Center, Relay, and End; Community compares Core, Bridge, and Peripheral. All vertices of Ring and Mesh are automorphism-equivalent, so their displayed rotations or node sets are repeated placement controls rather than distinct centrality classes. Information remains Centralized, Distributed, or Conflicting under the same construction used for Tree.

B.5 Controlled Graph-property Analysis

For each six-node organization, density measures communication volume; diameter measures worst-case propagation distance; mean clustering measures local cohesion; Freeman degree centralization measures hub dependence; normalized edge connectivity measures disconnection resistance; normalized independent-cycle redundancy measures alternative routing; and greedy modularity measures subgroup separation.

Graph statistics cannot generally be varied independently. We therefore build matched connected six-node variants that alter one focal property while minimizing changes in the others. The same six focal properties—diameter, edge connectivity, density, modularity, clustering, and cycle redundancy—are to be evaluated under every task structure. Within each task, a separate standardized partial-effect model will be fitted for each property, adjusting for residual changes in other graph properties and domain fixed effects while agents, roles, input allocation, inputs, and budget remain fixed.

For a connected six-node graph with m edges, density is $m/15$ and normalized cycle redundancy is $(m-5)/10$, so redundancy equals $1.5\,\text{density} - 0.5$. Separate focal-property models must omit this collinear counterpart from their controls; the two coefficients cannot be interpreted as independently identified effects of adding edges. Constant or singular graph controls must likewise be excluded. The message-trace measures below provide diagnostics for the proposed mechanisms.

B.6 Interaction-Behavior Measures

To organize the explanation of quality differences, the analysis follows

$$\phi(G) \longrightarrow \mathbf{m}(T_{x,s}, O) \longrightarrow Y(T_{x,s}, O, M, z), \tag{12}$$

where $\phi(G)$ contains graph properties, $\mathbf{m}$ contains observed interaction measures, and Y is the domain score. This is an analytical sequence linking structure, behavior, and outcome, not an identified causal chain. Analyses within domains retain task structure; pooled analyses include domain differences and assess domain interactions where identifiable.

Graph statistics describe communication opportunities, whereas message traces show which opportunities are used. We therefore compute the six mediators in Table 5 from delivered messages, input-unit IDs, and final-synthesis events. All text normalization, similarity thresholds, and denominators are fixed before comparing organizations. Runs with no applicable conflict or community partition are marked not applicable for the relevant metric rather than assigned a favorable zero.

For each task structure, we first regress a standardized behavior measure on a standardized graph property with domain fixed effects and the matched-variant controls used in Appendix B.5. We then add that behavior measure to the corresponding quality model. We report the property–behavior coefficient, the behavior–quality coefficient conditional on the property, and the change in the property coefficient. Bootstrap resampling keeps every task's organizations and seeds together. This analysis diagnoses a proposed mechanism; correlated graph properties and trace measures preclude a claim of fully identified causal mediation.

B.7 Budget and Fault Protocols

Token accounting. We count agent inputs, outputs, inter-agent messages, and final aggregation. An organization is Pareto-efficient if no alternative has no greater mean token cost and no lower mean quality, with at least one strict improvement. Common per-task caps range from 6k to 24k tokens and reserve the same final-answer allowance. When collaboration budget is exhausted, later messages are suppressed and the synthesizer uses the state already received. Caps are enforced at message boundaries, so crossing a threshold may enable a complete additional round and need not yield a smooth curve.

Fault injection. At pressure $p \in \{0, 10, 20, 30, 40, 50\}\%$, each trial selects agent or active message-channel failure with equal probability and disables proportion p before execution. Failed agents emit no output and failed channels drop messages without retry. Locations vary by seed, so damage depends on whether an integrator, articulation point, or redundant component is sampled. Quality is normalized by each topology's no-failure score to separate resilience from baseline accuracy.

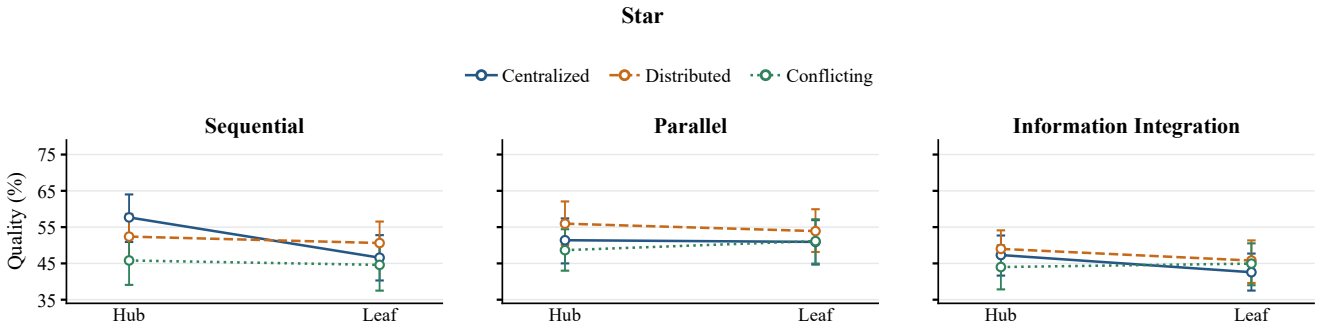
B.8 Organization-Cost Definitions and Assumptions

Table 6 adapts the supplied theoretical complexity table. T denotes exchanges rather than the role set R in Sec. 2.1; b_{all} denotes the complete single-agent input rather than the execution budget B. The call cost $C_M(s)$ includes a fixed generation allowance. Initialization, a separate final synthesis, tool execution, and shared-state access must be accounted for separately when material to measured costs.

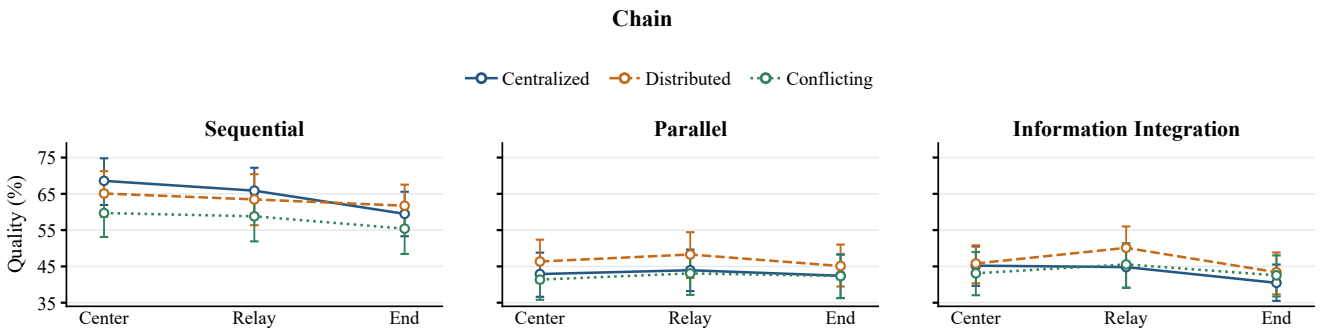
Equation 8 uses mean message lengths and balanced contexts. Without this approximation, the exchange cost is expressed as

$$\Theta\left(\sum_{t=1}^{T}\sum_{i=1}^{u} C_M\left(b_i + \sum_{j\in\mathcal{N}(i)} \ell(h_j^{(t-1)})\right)\right). \tag{13}$$

Thus a nonlinear call-cost function and unequal degrees can distinguish Star, Chain, and Tree even when their edge counts agree. The table's balanced-context expression is not a universal hardware



(a) Star: hub versus leaf placement.



(b) Chain: center, relay, and end placement.

Figure 9: Q2 extension to sparse non-Tree topologies. Curves show the three information allocations; whiskers show intervals.

Table 5: Q3 interaction measures. $\mathcal{E}^*$: designated supporting information; R: round cap.

Measure	Operational definition	Interpretation
Information coverage	$\lvert\mathcal{E}^* \cap \mathcal{E}_{\text{syn}}\rvert/\lvert\mathcal{E}^*\rvert$ at final synthesis	Required information that reaches the decision point
Propagation latency	First round t when all designated support units reach the synthesizer; incomplete cases are censored at $R+1$	Delay introduced by routing paths
Duplicate-message rate	Fraction of delivered messages with token-set Jaccard similarity at least 0.90 to an earlier message in the run	Repetition and communication waste
Contradiction resolution	Source-grounded resolved conflicts divided by injected, answer-relevant conflicts	Ability to reconcile conflicting information
Cross-group flow	Information-bearing messages crossing the fixed community cut divided by all information-bearing messages	Exchange between specialized groups
Synthesizer load	Input and received-message tokens processed in the final synthesis event	Aggregation burden at the decision point

complexity bound. If each edge carries messages in both directions at every exchange, routed volume is $\Theta(Tv\ell)$. Sparse/selective routing instead sums realized deliveries. Trace state is $\mathcal{S}(u,v) = \Theta(ub + T(u+v)\ell)$ under comparable local-output and message lengths; for a connected graph with $u \geq 2$, $v \geq u-1$ reduces this to $\Theta(ub + Tv\ell)$. This includes local outputs even when a learned graph contains isolated agents. The bound excludes backbone parameters and KV caches and does not model measured device memory.

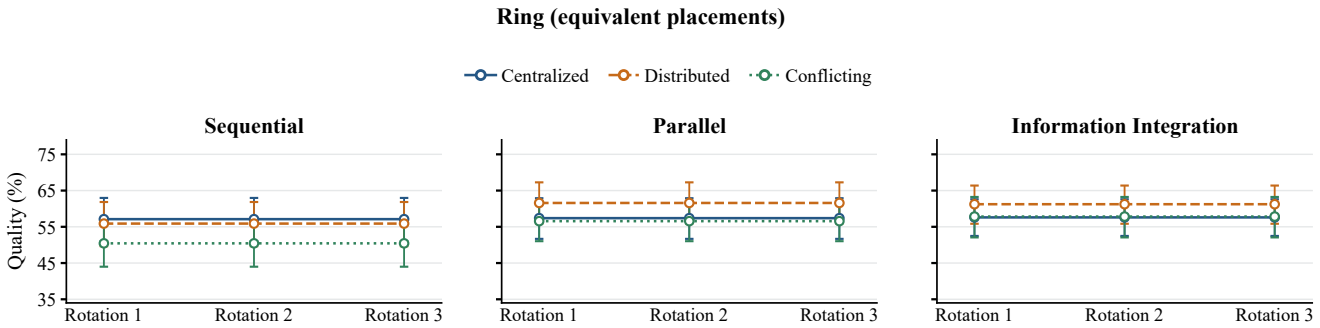
Designer-generation terms. Let d_h be hidden width, L_G the GNN encoder depth, m_0 the template graph's edge count, q_r the role-pool size, K_g the number of candidate groups, and g the number selected. Under the reference implementations assumed by the supplied analysis, the generation costs are

$$\mathcal{G}_G = \Theta\big(L_G(m_0 d_h + n d_h^2) + n^2 d_h\big), \tag{14}$$

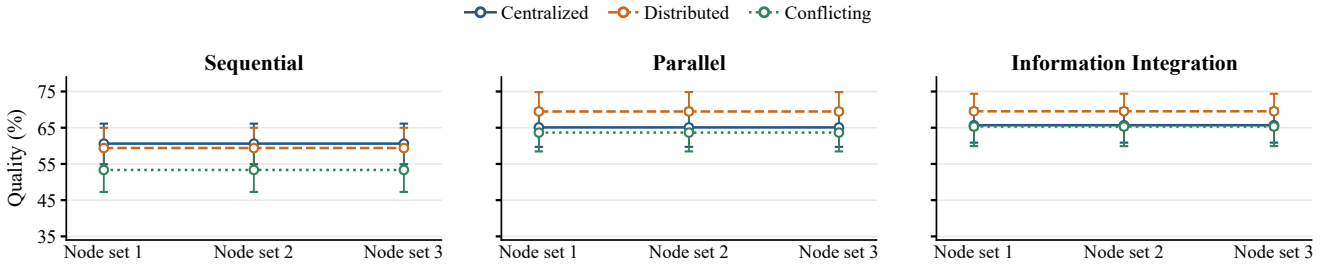
$$\mathcal{G}_A = \Theta\big(n q_r d_h + n^2 d_h^2\big), \tag{15}$$

$$\mathcal{G}_O = \Theta\big(g K_g d_h + g^2 d_h^2\big). \tag{16}$$

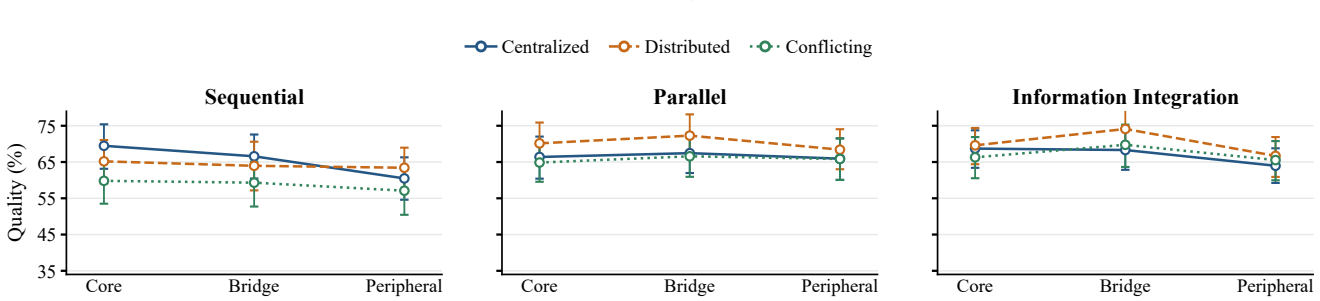
The G-Designer expression assumes a message-passing VGAE encoder with a dense pairwise decoder. ARG-Designer counts n role choices and $n(n-1)/2$ autoregressive candidate-edge decisions. GoAgent counts g selections over K_g groups and $g(g-1)/2$ group-edge decisions; its LLM-based group-discovery



(a) Ring: the three rotations are structurally equivalent controls.



(b) Mesh: the three node sets are structurally equivalent controls.



(c) Community: core, bridge, and peripheral placement.

Figure 10: Q2 extension to cyclic, dense, and modular topologies. Curve and interval semantics match Fig. 9

prompt of length s_g is shown separately in the online column. These are implementation-dependent reference bounds; sparse decoders replace dense pairwise terms by evaluated candidate edges.

The corresponding auxiliary states are

$$\mathcal{M}_G = \Theta(nd_h + m_0 + n^2), \tag{17}$$

$$\mathcal{M}_A = \Theta(q_r d_h + nd_h + n^2), \tag{18}$$

$$\mathcal{M}_O = \Theta(K_g d_h + gd_h + g^2). \tag{19}$$

For N_{tr} examples and E_{tr} epochs, designer fitting is $\mathcal{F}_i = \Theta(N_{\text{tr}} E_{\text{tr}} \mathcal{G}_i)$ when each example incurs a comparable forward/backward generation cost. Backbone training, reward evaluation, topology search, and MAS supervision rollouts are outside this expression. Methods using different objectives or training pipelines must report those additional costs empirically. The comparison separates routed tokens, backbone computation, and designer work; none is a direct substitute for measured latency or total API billing.

B.9 Input-Perturbation Stress-Test Protocol

Experiment scope. The stress test in Fig. 8 uses Information Integration because complementary and conflicting inputs have explicit semantics. The six domains and six fixed organizations are retained, with the primary backbone, roles, allocation rules, rounds, and budgets frozen.

Perturbations. For a pool of k atomic input units and $p \in \{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$, missing inputs remove a proportion p while retaining at least one unit. Distractors add approximately pk units that resemble the task context but neither support the gold answer

Table 6: Theoretical costs of representative organizations. Per-task analysis assumes undirected routing and retained interaction traces. $\mathcal{X}$ and $\mathcal{S}$ summarize execution and trace state; $\mathcal{F}_i$ counts offline designer fitting only. Shared backbone parameters, KV cache, backbone training, and supervision rollouts are excluded.

Organization	Active edges	Offline designer fitting	Online cost per task	Routed message tokens	Runtime state
Single Agent	0	–	$\Theta(C_M(b_{\text{all}}))$	0	$\Theta(b_{\text{all}})$
Star	$n-1$	–	$\mathcal{X}(n, n-1)$	$\Theta(T(n-1)\ell)$	$\mathcal{S}(n, n-1)$
Chain	$n-1$	–	$\mathcal{X}(n, n-1)$	$\Theta(T(n-1)\ell)$	$\mathcal{S}(n, n-1)$
Tree	$n-1$	–	$\mathcal{X}(n, n-1)$	$\Theta(T(n-1)\ell)$	$\mathcal{S}(n, n-1)$
Ring	n	–	$\mathcal{X}(n, n)$	$\Theta(Tn\ell)$	$\mathcal{S}(n, n)$
Mesh	$n(n-1)/2$	–	$\mathcal{X}(n, n(n-1)/2)$	$\Theta(Tn^2\ell)$	$\mathcal{S}(n, n(n-1)/2)$
Community	m_c	–	$\mathcal{X}(n, m_c)$	$\Theta(Tm_c\ell)$	$\mathcal{S}(n, m_c)$
G-Designer	$\widehat{m}_G$	$\mathcal{F}_G$	$\Theta(\mathcal{G}_G) + \mathcal{X}(\widehat{n}_G, \widehat{m}_G)$	$\Theta(T\widehat{m}_G\ell)$	$\Theta(\mathcal{M}_G) + \mathcal{S}(\widehat{n}_G, \widehat{m}_G)$
ARG-Designer	$\widehat{m}_A$	$\mathcal{F}_A$	$\Theta(\mathcal{G}_A) + \mathcal{X}(\widehat{n}_A, \widehat{m}_A)$	$\Theta(T\widehat{m}_A\ell)$	$\Theta(\mathcal{M}_A) + \mathcal{S}(\widehat{n}_A, \widehat{m}_A)$
GoAgent	$\widehat{m}_O$	$\mathcal{F}_O$	$\Theta(C_M(s_g) + \mathcal{G}_O) + \mathcal{X}(\widehat{n}_O, \widehat{m}_O)$	$\Theta(T\widehat{m}_O\ell)$	$\Theta(\mathcal{M}_O) + \mathcal{S}(\widehat{n}_O, \widehat{m}_O)$

nor explicitly contradict the supplied information. Conflicts introduce false versions of selected input units through minimal numerical, entity, negation, temporal, or causal changes, validated against the unchanged question and target. Gold answers and support labels remain evaluator-only. Unit IDs, truth status, and before/after text are logged.

Targeted corruption applies the same count of conflicting units after allocation, prioritizing Star's hub, Chain's high-betweenness relay, Tree's root/internal relays, or Community's inter-group bridge. Ring targets rotate and Mesh targets follow a frozen seed because their vertices are structurally equivalent. Target nodes keep executing; only their inputs change. This is a placement stress test, so cross-topology differences combine structure with the topology-specific targeting policy. Non-targeted perturbations are generated before allocation and shared by all graphs for a given task, type, ratio, and corruption seed. A single clean condition is shared across the four perturbation types.

Aggregation and uncertainty. Equation 9 is calculated within each domain before the six-domain macro average. The denominator must be positive; undefined ratios are reported explicitly rather than assigned a favorable value. Let $\overline{\text{Ret}}_{o,c}(p)$ denote this macro average. Normalized robustness AUC is

$$\text{RAUC}_{o,c} = \frac{1}{p_{\max}} \int_0^{p_{\max}} \overline{\text{Ret}}_{o,c}(p)\, dp, \qquad p_{\max} = 0.5, \tag{20}$$

computed by trapezoidal integration on the six specified ratios and reported on the retention-percentage scale. The displayed summary gives Mesh RAUC = 94.7 under missing inputs and 92.3 under targeting; Community reaches 93.1 with distractors and 89.5 with conflicts.

Pointwise bands show bootstrap intervals over domain/seed records. For paired task-level inference, resample source-task clusters with all organizations, corruption conditions, and seeds kept together. Recompute clean denominators and equal-domain means per replicate, then report paired differences with 95% intervals. Preserve incomplete and failed runs under the shared scoring rule. Raw quality, retained quality at $p = 0.5$, failure rate, actual tokens, and message traces distinguish input loss, error propagation, and unsuccessful conflict resolution.