LEARNING ROLE-CONTEXT GRAPH POLICIES FOR ADAPTIVE MULTI-AGENT COLLABORATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Effective LLM-based multi-agent collaboration requires jointly deciding which agents should participate and how they should communicate. Existing methods often optimize communication over a prescribed team, potentially leading to redundant or insufficient participation for individual queries, or adapt participation through selection from available structures or sequential graph construction, limiting structural flexibility or decoding parallelism. Moreover, limited modeling of local directional role relationships can obscure the distinct effects of information reception and dissemination. To address these limitations, we propose RCGP, a query-conditioned Role-Context Graph Policy for adaptive multi-agent collaboration. To reflect their structural coupling, we formulate participation, functional roles, and directed communication within a unified graph-policy action space, allowing collaboration scale and topology to adapt jointly through parallel node–edge decisions. We further design directional role-context encodings to characterize local role diversity and neighborhood support separately in receiving and sending contexts, enriching graph-policy prediction beyond role identity and connectivity while preserving the functional asymmetry between information reception and contribution dissemination. We directly optimize the graph policy via execution-guided reinforcement learning (RL) with a structure-aware initialization. Together, the graph policy provides RL with a structured and relational collaboration space to optimize, while RL grounds the graph policy with task-aware execution utility, calibrating collaboration toward query-specific effectiveness. Extensive experiments on six benchmarks demonstrate that RCGP outperforms state-of-the-art baselines, while additional analyses show favorable cost–performance trade-offs.

1 INTRODUCTION

Large language model (LLM)-based multi-agent systems have been applied to code generation (Hong et al., 2024), mathematical reasoning (Du et al., 2024), and question answering (Zhang et al., 2024). Their collaboration can be represented as a directed graph, with nodes corresponding to agents and edges encoding communication or execution dependencies (Zhang et al., 2025d). The choice of participating agents and their communication links can affect both task performance and execution cost (Zhang et al., 2025d;b). Existing approaches to automated collaboration design include performance-guided search (Hu et al., 2025b; Zhang et al., 2025e), supervised generation (Ye et al., 2025; Li et al., 2026c), and feedback-driven policy optimization (Zhuge et al., 2024; Zhang et al., 2025d;b). Several of these methods use reinforcement learning (RL) to optimize parameterized graph policies, sampling collaboration graphs for task execution and using the resulting rewards to guide policy updates (Zhuge et al., 2024; Zhang et al., 2025d; Yu et al., 2026b; Zhang et al., 2026b). Despite this progress, existing methods still face two challenges in adaptive collaboration design.

Challenge I: Joint adaptation of participation and communication. Existing topology optimization methods typically refine communication structures within a team whose size is fixed in advance (Zhuge et al., 2024; Zhang et al., 2025d), leaving agent participation outside the topology optimization. To examine whether simply increasing this fixed scale is sufficient, we vary team size while keeping the topology optimization procedure unchanged. Figure 1 shows that more agents do not consistently improve accuracy; for example, GTD performs worse when the team size increases

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

from 7 to 9 agents. This indicates that participation itself should adapt to the query rather than be treated as a fixed resource budget. Such adaptation is inherently tied to topology, since changing the participating agents also changes the communication structures that can be realized. Recent methods support variable participation (Zhang et al., 2025b; Yao et al., 2026; Tang et al., 2026; Li et al., 2026c), but often restrict structural flexibility or introduce sequential dependencies in graph construction. This makes it important to jointly adapt participation, roles, and communication while retaining flexible and parallel graph prediction.

Challenge II: Distinguishing functional communication contexts. Beyond the action space, learning participation and communication also depends on how the policy represents local relationships between functional roles. Existing topology learners incorporate role information (Zhang et al., 2025d; Wu et al., 2026; Li et al., 2026c; Zhang et al., 2026b), but often summarize it through team-level or aggregated local statistics, obscuring its directional function. Although diverse roles provide complementary expertise (Hong et al., 2024; Chen et al., 2024), their utility depends on their position in information flow. In real collaboration, incoming roles supply information to an agent, whereas outgoing roles receive its contributions, creating a functional asymmetry between reception and dissemination. Thus, similar neighborhoods can represent different functional contexts across directions. The recent RADAR (Zhang et al., 2026b) models role-dependent local redundancy but still aggregates incoming and outgoing information rather than preserving distinct directional contexts. Therefore, separate receiving and sending role contexts are important for learning participation and communication from execution feedback.

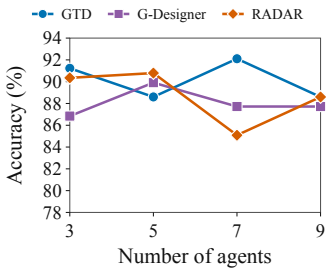


Figure 1: Accuracy of three representative methods across agent counts on HumanEval.

In this paper, we propose **R**ole-**C**ontext **G**raph **P**olicy for Adaptive Multi-Agent Collaboration (**RCGP**), a query-conditioned graph policy for adaptive multi-agent collaboration. Motivated by **Challenge I**, we formulate a unified graph-policy action space over agent participation, functional roles, and directed communication, and introduce an inactive node state that allows participation decisions to directly alter the realizable communication structure. This brings collaboration scale and topology into the same policy decision, enabling their joint adaptation through parallel node–edge prediction. Building on this action space, we develop directional role-context encodings over incoming, outgoing, shared-source, and shared-target relations. We characterize local role context from separate receiving and sending perspectives, preserving their functional asymmetry and enriching graph decisions beyond role identity and connectivity alone (addressing **Challenge II**). We then directly optimize the resulting role-context graph policy with execution-guided RL from a structure-aware initialization, allowing task-level utility to jointly refine graph decisions and the role-context representations that inform them. At inference, inactive nodes and their incident edges are pruned from the sampled graph, yielding query-specific collaboration with adaptive scale and topology. Across six benchmarks, RCGP consistently outperforms all static and autonomous baselines, with further analyses supporting its token efficiency and effectiveness across different LLM backbones. *Overall, RCGP exposes structurally coupled graph decisions and functionally asymmetric role contexts to policy learning, while execution feedback grounds both in task utility.*

2 Related Work

LLM-agent Collaboration. LLM-based multi-agent systems extend the capabilities of individual agents by integrating complementary expertise, reasoning, and feedback to solve complex tasks (Wu et al., 2024; Ma et al., 2025; Hong et al., 2024; Xu et al., 2026b). Early approaches largely organize collaboration through predefined roles and interaction protocols. For example, CAMEL (Li et al., 2023) adopts role-playing dialogues, while multi-agent debate (Du et al., 2024) and ChatEval (Chan et al., 2024) facilitate iterative reasoning and evaluation through structured discussions. MetaGPT (Hong et al., 2024) and Debate on Graph (Ma et al., 2025) coordinate specialized agents through task-specific procedures. Other approaches structure information exchange through sequential chains, layered aggregation, or prescribed directed acyclic graphs (Zhang et al., 2024; Wang

et al., 2025a; Qian et al., 2025). Although these designs demonstrate the benefits of structured collaboration, their reliance on prescribed organizational patterns can limit flexibility across tasks with different reasoning and communication demands.

Automated Multi-Agent System Design. Recent work broadly studies automated system construction and optimization (Shang et al., 2025; Li et al., 2026d; 2025b; Yu et al., 2026a; Zhao et al., 2026; Ye et al., 2025; Zhang et al., 2025g; Hu et al., 2026; Wang et al., 2026b; Li et al., 2026a; Fan et al., 2025; Rosser & Foerster, 2025; Niu et al., 2025; Wang et al., 2025b; Xiang et al., 2026; Wan et al., 2026; Jin et al., 2026; Zhou et al., 2026; Feng et al., 2025; Li et al., 2026b; Wang et al., 2026d; Hong et al., 2026; Xia et al., 2026; Chen et al., 2024; Kim et al., 2024; Yuan et al., 2025; Shao et al., 2026; Zhang et al., 2025f; Wang et al., 2026a) and adaptive agent coordination (Klein et al., 2025; Dang et al., 2025; Feng et al., 2026; Nielsen et al., 2026; Xu et al., 2026a; Li et al., 2026e; Ruan et al., 2026; Ke et al., 2026; Hu et al., 2025a; Li et al., 2025a; Vyas et al., 2025; Zhu et al., 2025; Yang et al., 2025), alongside advances in evaluation, memory, and execution management (Trirat et al., 2026; Ursekar et al., 2026; Ma et al., 2026; Huang et al., 2026; Wei et al., 2026; Chen et al., 2026; Fan et al., 2026a; Deng et al., 2025; Lin et al., 2026; Yao et al., 2025; Zhang et al., 2025a; Fioresi et al., 2026; Guan et al., 2026; Hua et al., 2026). Collaboration is commonly modeled as a directed graph, where nodes represent agents or functional components and edges encode communication or execution dependencies (Dong et al., 2026; Yun et al., 2026; Fan et al., 2026b; Wu et al., 2026; Zhang et al., 2026a; Yang et al., 2026; Jiang et al., 2026; Tastan et al., 2026; Wang et al., 2026c; Zhang et al., 2025c). Execution feedback guides structural search and evolution or optimizes graph policies through reinforcement learning (Hu et al., 2025b; Zhang et al., 2025e; Hu et al., 2025c; Tang et al., 2026; Zhuge et al., 2024; Zhang et al., 2025d; 2026b). Despite progress in automated collaboration design, existing methods often optimize communication over prescribed teams (Zhuge et al., 2024; Zhang et al., 2025d) or adapt participation through predefined composition spaces or sequential graph construction (Zhang et al., 2025b; Yao et al., 2026; Li et al., 2026c), limiting structural flexibility or decoding parallelism. They also provide limited explicit modeling of local role diversity from distinct receiving and sending perspectives, which can obscure the functional asymmetry between information reception and contribution dissemination.

3 PRELIMINARIES

3.1 TOPOLOGY STRUCTURE

We model an LLM-based multi-agent system as a directed graph $G = (V, E)$, where V is the set of agent nodes and $E \subseteq V \times V$ is the set of directed communication edges. Each node $v_i \in V$ corresponds to an LLM-based agent, formalized as

$$v_i = \{\texttt{Base}_i, \texttt{Role}_i, \texttt{State}_i, \texttt{Plugin}_i\}, \tag{1}$$

where (1) $\texttt{Base}_i$ is the language-model instance powering the agent; (2) $\texttt{Role}_i$ is its assigned agent role; (3) $\texttt{State}_i$ represents its accumulated knowledge and interaction history; and (4) $\texttt{Plugin}_i$ is the set of external tools available to it, such as web search, code execution, or file retrieval (Zhang et al., 2025d). Given a prompt $\mathcal{P}_i = (\mathcal{P}_{\text{sys},i}, \mathcal{P}_{\text{usr},i})$, agent v_i generates a response $\mathcal{R}_i = v_i(\mathcal{P}_i) = v_i(\mathcal{P}_{\text{sys},i}, \mathcal{P}_{\text{usr},i})$. Here, $\mathcal{P}_{\text{sys},i}$ is the system prompt constructed from the agent's role $\texttt{Role}_i$ and internal state $\texttt{State}_i$, while $\mathcal{P}_{\text{usr},i}$ provides the task query and may include messages from other agents and externally retrieved information. Given a query $\mathcal{Q}$, agents in an acyclic collaboration graph G execute in topological order, with each agent generating a response based on its role and state, the query, and messages from its incoming neighbors. The agent responses are aggregated to produce the system output $G(\mathcal{Q})$.

3.2 AUTOMATIC MULTI-AGENT TOPOLOGY DESIGN

For a given query $\mathcal{Q}$, we automatically construct an LLM-based multi-agent system $G = (V, E)$ that balances three objectives: i) effectiveness, by maximizing the task utility $u(G(\mathcal{Q}))$; ii) cost efficiency, by minimizing the execution cost $c(G; \mathcal{Q})$; and iii) adaptiveness, by dynamically selecting participating agents and communication structures according to the query and task requirements. We formulate these objectives as

$$\min_{G \in \mathbb{G}} \quad \mathcal{L}(G; \mathcal{Q}), \quad \mathcal{L}(G; \mathcal{Q}) = -u(G(\mathcal{Q})) + \alpha\, c(G; \mathcal{Q}). \tag{2}$$

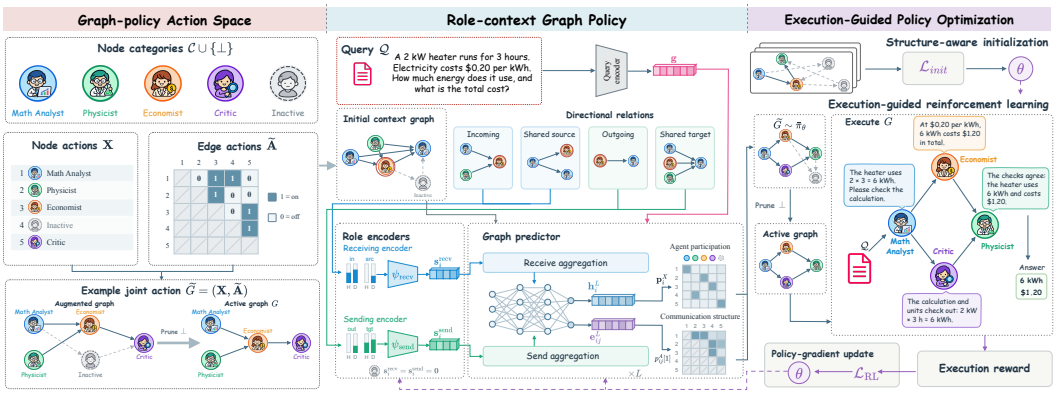


Figure 2: Overview of the RCGP framework. Given a query Q, RCGP encodes $\widetilde{G}^{(0)}$ sampled from the unified action space into receiving and sending representations for parallel node–edge prediction, prunes inactive agents and their edges, and learns through execution-guided policy optimization.

Here, $\mathbb{G}$ denotes the set of directed acyclic collaboration graphs. $G(\mathcal{Q})$ is the system output obtained by executing the constructed communication graph, $u(\cdot)$ measures task-specific utility, e.g., accuracy or Pass@1, and $c(\cdot)$ captures the computational cost of agent invocation and inter-agent communication. The coefficient $\alpha \geq 0$ controls the trade-off between utility and cost.

4 The Proposed RCGP

In this section, we propose RCGP, a *Role-Context Graph Policy* for query-adaptive multi-agent collaboration. The key idea is to capture *structural coupling* by modeling participation and communication in a unified action space, preserve *functional asymmetry* through distinct receiving and sending role contexts for graph prediction, and jointly optimize the graph policy and these contextual representations through execution-guided RL. As shown in Figure 2, RCGP consists of three components: a graph-policy action space, a role-context graph policy, and execution-guided policy optimization. We detail each module below.

4.1 Graph-Policy Action Space

Participation and communication are inherently coupled: the utility of an agent depends on the functional role it plays and how it communicates, while changing participation also changes the feasible communication structure. We therefore design a unified graph-policy action space that jointly represents participation, role assignment, and directed communication, where an inactive role category is introduced to make agent participation learnable and couple collaboration scale with the feasible communication structure. Let $\mathcal{C}$ denote the set of candidate functional roles and $\mathcal{S} = \{1, \ldots, N_{\max}\}$ index $N_{\max}$ candidate agent slots. To make participation a policy decision, we augment $\mathcal{C}$ with an inactive category $\perp$. For each slot $i \in \mathcal{S}$, the node action $X_i \in \mathcal{C} \cup \{\perp\}$ specifies either a functional role or inactivity. Let $\mathbf{X} = (X_1, \ldots, X_{N_{\max}})$ collect all node actions. Directed communication is represented by binary edge actions $\widetilde{A}_{ij} \in \{0, 1\}$ over $\Omega = \{(i, j) \mid 1 \leq i < j \leq N_{\max}\}$, where the slot order ensures acyclic execution. The graph-policy action space is

$$\mathcal{A}_G = \underbrace{(\mathcal{C} \cup \{\perp\})^{N_{\max}}}_{\text{participation and role actions}} \times \underbrace{\{0, 1\}^{|\Omega|}}_{\text{directed communication actions}}. \tag{3}$$

An augmented graph action $\widetilde{G} = (\mathbf{X}, \widetilde{\mathbf{A}}) \in \mathcal{A}_G$ therefore specifies which agents participate, which roles they take, and how they communicate. The executed collaboration graph is obtained by removing inactive slots and their incident edges. Let $z_i = \mathbb{I}[X_i \neq \perp]$ indicate whether slot i is active, and let v_i denote its corresponding agent. The active graph $G = (V, E)$ is

$$V = \{v_i \mid i \in \mathcal{S},\ z_i = 1\}, \qquad E = \{(v_i, v_j) \mid (i, j) \in \Omega,\ z_i z_j [\widetilde{\mathbf{A}}]_{ij} = 1\}. \tag{4}$$

Its collaboration scale is $|V| = \sum_{i \in \mathcal{S}} z_i$, making team size an outcome of the graph action rather than a prescribed quantity. Since an edge contributes to execution only when both endpoint agents are active, participation, role assignment, and communication are evaluated through the same graph-level outcome. This allows execution feedback to assess whether an agent should participate together with the role and communication structure in which it operates.

> **Insight.** The unified graph-policy action space places participation, roles, and communication within the same decision space. By introducing an inactive node state, collaboration scale becomes learnable, while participation decisions directly determine which communication links remain feasible.

4.2 Role-Context Graph Policy

The graph-policy action space represents participation, roles, and communication jointly, but does not describe the local functional context in which each role operates. In directed collaboration, this context is inherently asymmetric: incoming roles support information integration, whereas outgoing roles determine where an agent's contribution is used. We therefore construct separate receiving and sending role contexts and use them to inform graph prediction.

Directional role contexts. We first sample an initial augmented graph $\widetilde{G}^{(0)} = (\mathbf{X}^{(0)}, \widetilde{\mathbf{A}}^{(0)})$ from categorical priors over $\mathcal{A}_G$; details are given in Appendix B.1. Its active subgraph $G^{(0)} = (V^{(0)}, E^{(0)})$ provides the local communication structure from which role contexts are derived. Let $N = |V^{(0)}|$, let $r_i \in \mathcal{C}$ denote the role of active agent v_i, and let $\mathbf{A} \in \{0,1\}^{N \times N}$ be the adjacency matrix of $G^{(0)}$, where $[\mathbf{A}]_{ij} = 1$ denotes $v_i \to v_j$. We define four relations:

$$\mathbf{R}^{\text{in}} = \mathbf{A}^\top, \qquad \mathbf{R}^{\text{out}} = \mathbf{A}, \qquad \mathbf{R}^{\text{src}} = \mathbb{I}[\mathbf{A}^\top \mathbf{A} > 0], \qquad \mathbf{R}^{\text{tgt}} = \mathbb{I}[\mathbf{A}\mathbf{A}^\top > 0]. \tag{5}$$

Here, $\top$ denotes transpose and $\mathbb{I}[\cdot]$ is an element-wise indicator; diagonals are set to zero. $\mathbf{R}^{\text{in}}$ and $\mathbf{R}^{\text{out}}$ identify direct information sources and recipients, while $\mathbf{R}^{\text{src}}$ and $\mathbf{R}^{\text{tgt}}$ capture overlap through shared sources and recipients. These relations therefore extend the local context beyond direct connectivity while retaining its receiving–sending directionality. For relation $c \in \{\text{in}, \text{out}, \text{src}, \text{tgt}\}$, let $\mathcal{N}_c(v_i) = \{v_j \mid [\mathbf{R}^c]_{ij} = 1\}$ and let $p_i^c(r)$ be the proportion of neighbors with role $r \in \mathcal{C}$. We characterize each neighborhood by normalized role entropy H_i^c and relative size D_i^c:

$$H_i^c = -\frac{1}{\log |\mathcal{C}|} \sum_{r \in \mathcal{C}} p_i^c(r) \log(p_i^c(r) + \epsilon), \qquad D_i^c = \frac{|\mathcal{N}_c(v_i)|}{\max(N-1, 1)}. \tag{6}$$

Here, $\epsilon > 0$ stabilizes the logarithm; both statistics are zero for an empty neighborhood. H_i^c captures the diversity of functional roles in the neighborhood, whereas D_i^c records its structural support and distinguishes an empty neighborhood from a nonempty homogeneous one. This yields a compact description of how roles are locally organized under each relation. We then form separate receiving and sending role contexts:

$$\mathbf{s}_i^{\text{recv}} = \psi_{\text{recv}}\big(H_i^{\text{in}}, D_i^{\text{in}}, H_i^{\text{src}}, D_i^{\text{src}}\big), \qquad \mathbf{s}_i^{\text{send}} = \psi_{\text{send}}\big(H_i^{\text{out}}, D_i^{\text{out}}, H_i^{\text{tgt}}, D_i^{\text{tgt}}\big). \tag{7}$$

Here, ψ_{recv} and ψ_{send} are separately parameterized learnable encoders. We refer to $(\mathbf{s}_i^{\text{recv}}, \mathbf{s}_i^{\text{send}})$ as the directional role context of v_i. The former represents the role organization supporting information reception, while the latter represents the role organization surrounding contribution dissemination. Keeping the two contexts separate preserves their distinct functional meanings instead of collapsing them into a single local descriptor. Inactive slots receive zero role-context vectors.

Role-context graph prediction. Given query $\mathcal{Q}$, we obtain $\mathbf{g} = \text{Enc}(\mathcal{Q})$ and initialize node and edge representations as $\mathbf{h}_i^0 = \text{Emb}_X(X_i^{(0)})$ and $\mathbf{e}_{ij}^0 = \text{Emb}_A(\widetilde{A}_{ij}^{(0)})$. Here, Enc encodes the query text, while Emb_X and Emb_A embed node categories and edge existence, respectively. The predictor then uses the directional role contexts to evaluate candidate interactions from distinct receiving and sending views. At layer ℓ, we compute receiving and sending context vectors and use them to update

the node representations as follows:

$$\begin{aligned}
\mathbf{c}_i^{\mathrm{recv},\ell} &= \mathrm{Attn}_{\mathrm{recv}}\big(\{(\mathbf{h}_u^\ell + \mathbf{s}_u^{\mathrm{recv}}) \| \mathbf{h}_i^\ell \| \mathbf{e}_{ui}^\ell \| \mathbf{g} : (u,i) \in \Omega\}\big), \\
\mathbf{c}_i^{\mathrm{send},\ell} &= \mathrm{Attn}_{\mathrm{send}}\big(\{\mathbf{h}_i^\ell \| (\mathbf{h}_v^\ell + \mathbf{s}_v^{\mathrm{send}}) \| \mathbf{e}_{iv}^\ell \| \mathbf{g} : (i,v) \in \Omega\}\big), \\
\mathbf{h}_i^{\ell+1} &= U_\ell\Big(\mathbf{h}_i^\ell \| \mathbf{c}_i^{\mathrm{recv},\ell} \| \mathbf{c}_i^{\mathrm{send},\ell} \| \mathbf{g}\Big).
\end{aligned} \tag{8}$$

Here, $\|$ denotes concatenation, $\mathrm{Attn}_{\mathrm{recv}}$ and $\mathrm{Attn}_{\mathrm{send}}$ are learnable attention operators, and U_ℓ is the node-update map. For a candidate interaction $u \to v$, $\mathbf{s}_u^{\mathrm{recv}}$ summarizes the upstream role support behind the information available at u, while $\mathbf{s}_v^{\mathrm{send}}$ summarizes the downstream role context in which information reaching v may be further used. The predictor can therefore distinguish candidate interactions that appear similar in role identity or connectivity but occupy different functional contexts. All admissible pairs in Ω participate in prediction. After L layers, node and edge heads produce categorical distributions $\mathbf{p}_i^X$ over $\mathcal{C} \cup \{\perp\}$ and $\mathbf{p}_{ij}^A$ over edge absence and presence. The graph policy is

$$\pi_\theta\Big(\widetilde{G} \mid \widetilde{G}^{(0)}, \mathcal{Q}\Big) = \prod_{i \in \mathcal{S}} \mathbf{p}_i^X[X_i] \prod_{(i,j) \in \Omega} \mathbf{p}_{ij}^A[\widetilde{A}_{ij}], \tag{9}$$

where θ denotes the trainable parameters and $\widetilde{G} = (\mathbf{X}, \widetilde{\mathbf{A}})$ is the output graph action. Although node and edge actions are factorized for parallel sampling, their distributions are produced by the same query-conditioned, role-context-aware predictor. Participation, role assignment, and communication can therefore be predicted in parallel while remaining informed by a shared representation of the collaboration context. Detailed attention, edge-update, and prediction-head operations are provided in Appendix B.2.

We impose a nonempty-graph constraint to ensure that at least one agent participates, yielding the constrained distribution $\bar{\pi}_\theta$ (Appendix B.3). At inference, we sample $\widetilde{G} \sim \bar{\pi}_\theta$ and prune inactive nodes and their incident edges according to equation 4 to obtain a query-specific collaboration graph.

> **Insight.** Directional role context exposes the *functional asymmetry* of local information flow, allowing the graph policy to distinguish receiving and sending functions beyond role identity and connectivity alone.

4.3 Execution-Guided Policy Optimization

The graph policy specifies a structured space of collaboration decisions, but structural supervision alone does not determine whether a sampled graph is useful for a particular query. We therefore use structure-aware initialization as a warm start and then optimize the policy from graph execution.

Structure-aware initialization. We initialize the graph predictor and role-context encoders using weighted node–edge supervision on synthetic role-labeled graphs constructed from commonly used multi-agent collaboration structures (Zhang et al., 2026b). Node targets supervise participation and functional roles, while edge targets supervise directed communication. Targets with higher local role diversity receive larger weights, providing structural preferences before reinforcement learning. The complete initialization objective and target construction are provided in Appendix B.4.

Execution-guided reinforcement learning. For each query $\mathcal{Q}$, we sample and execute a collaboration graph from the initialized policy. To reward correct solutions while favoring sparse communication among successful executions, we define

$$R(G; \mathcal{Q}) = C(G; \mathcal{Q})\,[\lambda_C + \lambda_S S(G)], \tag{10}$$

where $C(G; \mathcal{Q}) \in \{0, 1\}$ indicates answer correctness. Inspired by prior work (Zhang et al., 2025d; 2026b), we use the relative edge sparsity $S(G)$ of the active graph to reward sparse communication only when the answer is correct, with $\lambda_C, \lambda_S \geq 0$ controlling the correctness reward and sparsity bonus, respectively. Policy-gradient updates use the log-probabilities of sampled actions under $\bar{\pi}_\theta$ to refine participation, role, and edge predictions. Because the receiving and sending encoders in equation 7 contribute to these action probabilities, the same execution feedback also updates how directional role contexts influence graph prediction. Thus, the graph policy provides a structured

Table 1: Performance comparison across single-agent methods, multi-agent systems, and autonomous multi-agent systems. The best results are highlighted in **bold**, and the strongest baseline results are underlined.

Methods	Auto	MMLU	GSM8K	SVAMP	AQuA	HumanEval	MBPP	Avg.
Vanilla	✗	78.20	87.47	86.74	78.92	87.72	84.62	83.94
CoT	✗	78.95$_{\uparrow 0.75}$	87.16$_{\downarrow 0.32}$	87.37$_{\uparrow 0.63}$	75.49$_{\downarrow 3.43}$	88.60$_{\uparrow 0.88}$	86.21$_{\uparrow 1.59}$	83.96
ComplexCoT	✗	79.70$_{\uparrow 1.50}$	86.92$_{\downarrow 0.55}$	87.68$_{\uparrow 0.95}$	75.98$_{\downarrow 2.94}$	87.72	81.17$_{\downarrow 3.45}$	83.19
MultiPersona	✗	79.32$_{\uparrow 1.13}$	87.55$_{\uparrow 0.08}$	87.05$_{\uparrow 0.32}$	79.41$_{\uparrow 0.49}$	88.16$_{\uparrow 0.44}$	76.39$_{\downarrow 8.22}$	82.98
LLM-Debate	✗	78.57$_{\uparrow 0.38}$	89.52$_{\uparrow 2.05}$	89.05$_{\uparrow 2.32}$	79.90$_{\uparrow 0.98}$	89.04$_{\uparrow 1.32}$	82.23$_{\downarrow 2.39}$	84.72
AgentVerse	✗	80.45$_{\uparrow 2.26}$	89.91$_{\uparrow 2.44}$	88.42$_{\uparrow 1.68}$	77.94$_{\downarrow 0.98}$	89.47$_{\uparrow 1.75}$	79.58$_{\downarrow 5.04}$	84.30
MacNet	✗	81.95$_{\uparrow 3.76}$	88.02$_{\uparrow 0.55}$	86.00$_{\downarrow 0.74}$	79.17$_{\uparrow 0.25}$	85.09$_{\downarrow 2.63}$	84.88$_{\uparrow 0.27}$	84.19
GPTSwarm	✓	76.69$_{\downarrow 1.50}$	89.20$_{\uparrow 1.73}$	88.74$_{\uparrow 2.00}$	80.88$_{\uparrow 1.96}$	89.47$_{\uparrow 1.75}$	79.58$_{\downarrow 5.04}$	84.09
G-Designer	✓	80.83$_{\uparrow 2.63}$	91.10$_{\uparrow 3.62}$	90.00$_{\uparrow 3.26}$	80.64$_{\uparrow 1.72}$	89.91$_{\uparrow 2.19}$	88.33$_{\uparrow 3.71}$	86.80
AgentPrune	✓	80.08$_{\uparrow 1.88}$	91.96$_{\uparrow 4.49}$	90.42$_{\uparrow 3.68}$	81.37$_{\uparrow 2.45}$	87.72	86.74$_{\uparrow 2.12}$	86.38
GTD	✓	80.45$_{\uparrow 2.26}$	91.41$_{\uparrow 3.94}$	90.32$_{\uparrow 3.58}$	80.15$_{\uparrow 1.23}$	88.60$_{\uparrow 0.88}$	84.08$_{\downarrow 0.53}$	85.83
MaAS	✓	81.20$_{\uparrow 3.01}$	91.17$_{\uparrow 3.70}$	89.68$_{\uparrow 2.95}$	80.39$_{\uparrow 1.47}$	90.35$_{\uparrow 2.63}$	85.41$_{\uparrow 0.80}$	86.37
ARG-Designer	✓	81.58$_{\uparrow 3.38}$	91.25$_{\uparrow 3.78}$	92.21$_{\uparrow 5.47}$	81.62$_{\uparrow 2.70}$	89.47$_{\uparrow 1.75}$	87.00$_{\uparrow 2.39}$	87.19
RADAR	✓	81.95$_{\uparrow 3.76}$	92.51$_{\uparrow 5.04}$	93.26$_{\uparrow 6.53}$	82.84$_{\uparrow 3.92}$	90.79$_{\uparrow 3.07}$	89.12$_{\uparrow 4.51}$	88.41
Ours	✓	**84.21**$_{\uparrow 6.02}$	**94.01**$_{\uparrow 6.54}$	**94.00**$_{\uparrow 7.26}$	**85.29**$_{\uparrow 6.37}$	**92.11**$_{\uparrow 4.39}$	**89.92**$_{\uparrow 5.31}$	**89.92**

role-context space for reinforcement learning, while execution feedback adjusts both the graph decisions and the contextual representations used to predict them.

> **Insight.** The graph policy provides reinforcement learning with a structured, role-context-aware collaboration space over participation, roles, and communication. Execution feedback, in turn, grounds these decisions in task utility and also adjusts how directional role contexts influence their prediction.

5 Experiments

5.1 Experimental Setup

Datasets and Metrics. To comprehensively evaluate RCGP, we conduct experiments on six benchmarks spanning general reasoning, mathematical problem solving, and code generation. Specifically, we use MMLU for general reasoning; GSM8K, SVAMP, and AQuA for mathematical problem solving; and HumanEval and MBPP for code generation. The dataset statistics and evaluation metrics are presented in Table 4 (Appendix A).

Baselines and Setup. We compare RCGP against three types of baselines: i) **single-agent methods**, including Vanilla prompting, Chain-of-Thought (CoT) (Wei et al., 2022), and complexity-based CoT (ComplexCoT) (Fu et al., 2023); ii) **static multi-agent systems**, including MultiPersona (Wang et al., 2024), LLM-Debate (Du et al., 2024), AgentVerse (Chen et al., 2024), and MacNet (Qian et al., 2025); and iii) **autonomous multi-agent systems**, including GPTSwarm (Zhuge et al., 2024), G-Designer (Zhang et al., 2025d), AgentPrune (Zhang et al., 2025c), GTD (Jiang et al., 2026), MaAS (Zhang et al., 2025b), ARG-Designer (Li et al., 2026c), and RADAR (Zhang et al., 2026b). All main experiments use `GPT-4o-mini` as the backbone. Please see Appendix C for more details.

5.2 Performance Analysis

We compare RCGP with single-agent methods, static multi-agent systems, and autonomous topology-design methods across six benchmark datasets, with results reported in Table 1.

Finding 1: RCGP is effective across different task domains. With the graph policy optimized separately for each benchmark, RCGP improves upon Vanilla in all three evaluated domains: knowledge reasoning, mathematical problem solving, and code generation. Across the six benchmarks, RCGP improves upon Vanilla by up to 7.26 percentage points, with an average improvement of 5.98

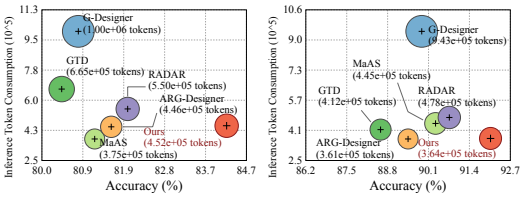
percentage points. These results support the applicability of the unified graph-policy formulation to different task domains, allowing participation, role assignment, and communication to be learned according to their respective task requirements.

Finding 2: The role-context graph policy improves upon static and autonomous collaboration. CoT and ComplexCoT yield limited average gains or degradation, while static multi-agent systems exhibit task-dependent performance. For example, MultiPersona and AgentVerse score 76.39% and 79.58% on MBPP, respectively, below Vanilla's 84.62%. Autonomous topology-design methods generally achieve stronger average performance, supporting the value of adapting collaboration structures to query requirements. For example, RADAR uses graph diffusion to generate query-adaptive topologies with reduced communication redundancy, attaining the highest baseline average of 88.41%. RCGP further outperforms RADAR on every benchmark, with the largest relative improvement of 2.96% on AQuA. These gains suggest that explicitly learning participation and preserving separate receiving and sending role contexts provide finer control over team composition and information flow than RADAR's use of node isolation and effective-size summaries. These results demonstrate the effectiveness of RCGP in improving task performance through query-adaptive multi-agent collaboration.

5.3 Token Efficiency

Figure 3 compares task performance and inference-token consumption on MMLU and HumanEval, counting both prompt and completion tokens.

Finding 3: The coupled graph policy improves the performance–cost trade-off. RCGP achieves the highest performance on MMLU and HumanEval, consuming approximately 4.52×10^5 and 3.64×10^5 tokens, respectively. Compared with RADAR, it improves performance while reducing token consumption by 17.8% and 23.8%; the corresponding savings over G-Designer reach 54.8% and 61.4%. It also outperforms ARG-Designer at nearly the same token consumption. MaAS uses fewer tokens on MMLU but achieves lower accuracy. These results suggest that jointly adapting participation and communication improves the performance–cost trade-off: pruning inactive agents and their incident edges reduces agent invocations and message-exchange tokens, while execution-guided RL uses task utility to guide coupled decisions and retain effective collaboration.



(a) MMLU (b) HumanEval

Figure 3: Performance and inference-token consumption on MMLU and HumanEval.

5.4 Effectiveness Across Different LLM Backbones

We compare RCGP with Vanilla prompting on MMLU using four LLM backbones to examine whether collaboration benefits persist across base models. The results are reported in Table 2.

Finding 4: The query-conditioned graph policy improves performance across LLM backbones. RCGP increases MMLU accuracy across all four backbones by 3.76–6.02 percentage points. GPT-4o-mini achieves the largest gain, improving from 78.20% to 84.21%. Llama-3.3-70B-Instruct improves from 83.46% to 87.22%, gaining 3.76 points, while Qwen3-32B gains 4.51 points. Even DeepSeek-V4 Flash, the strongest standalone backbone, improves from 86.47% to 90.98%. These results show that the graph policy consistently improves performance with each evaluated LLM backbone. By jointly determining agent participation, functional roles, and communication, RCGP organizes ef-

Table 2: Performance on MMLU across different LLM backbones.

LLMs	Vanilla	RCGP
GPT-4o-mini	78.20	**84.21** $\uparrow 6.02$
DeepSeek-V4 Flash	86.47	**90.98** $\uparrow 4.51$
Llama-3.3-70B-Instruct	83.46	**87.22** $\uparrow 3.76$
Qwen3-32B	81.20	**85.71** $\uparrow 4.51$

fective collaboration among agents sharing the same backbone, providing additional gains even when standalone model performance is already strong.

5.5 Ablation Study

We conduct ablation studies on MMLU, AQuA, and HumanEval to examine the key design choices of RCGP. The results are reported in Table 3.

Finding 5: Adaptive collaboration benefits from coupling participation with communication. Removing the inactive category (w/o IC), while retaining role and edge prediction, reduces the average score by 1.69% relative to the full model, with relative drops of 1.34%, 2.30%, and 1.43% on MMLU, AQuA, and HumanEval, respectively. Thus, optimizing communication over an always-active team does not recover the performance of jointly deciding participation and topology. This supports our structural motivation that changing who participates also changes the communication structures available to the policy. We further examine the benefit of adaptive participation through comparisons with different fixed numbers of active agents, as reported in Appendix D.3.

Finding 6: Functional asymmetry provides useful context for graph decisions. Removing the full role-context representation (w/o RRE) decreases performance by up to 2.87%. More specifically, collapsing receiving and sending contexts (w/o DS) yields a 2.53% relative decrease, supporting their distinct roles in graph prediction. Removing shared-source and shared-target relations (w/o SR) causes a comparable 2.52% decrease, indicating that direct connectivity alone omits useful neighborhood context. Removing role entropy while retaining neighborhood size (w/o RE) produces a role-context degradation of 2.70%. Together, these comparisons support modeling local information flow as functionally asymmetric and explicitly characterizing its local role organization.

Table 3: Ablation studies on three datasets.

Variants	MMLU	AQuA	HumanEval
Full model	**84.21**	**85.29**	**92.11**
w/o IC	$83.08_{\downarrow 1.13}$	$83.33_{\downarrow 1.96}$	$90.79_{\downarrow 1.32}$
w/o RRE	$81.95_{\downarrow 2.26}$	$83.33_{\downarrow 1.96}$	$89.47_{\downarrow 2.63}$
w/o SR	$82.71_{\downarrow 1.50}$	$82.84_{\downarrow 2.45}$	$89.47_{\downarrow 2.63}$
w/o DS	$83.46_{\downarrow 0.75}$	$83.82_{\downarrow 1.47}$	$87.72_{\downarrow 4.39}$
w/o RE	$81.20_{\downarrow 3.01}$	$82.11_{\downarrow 3.19}$	$91.23_{\downarrow 0.88}$
w/o RL	$80.45_{\downarrow 3.76}$	$82.84_{\downarrow 2.45}$	$86.84_{\downarrow 5.26}$
w/o SPI	$81.95_{\downarrow 2.26}$	$83.09_{\downarrow 2.21}$	$90.79_{\downarrow 1.32}$

Finding 7: Execution feedback grounds the learned collaboration policy. Removing execution-guided RL (w/o RL) decreases the average score by 4.39%, compared with only 2.21% without structure-aware initialization (w/o SPI). The contrast is particularly clear on HumanEval, where the relative decreases are 5.72% and 1.43%, respectively. This suggests that initialization provides a useful structural warm start, while execution feedback plays the larger role in grounding both the coupled graph decisions and the directional role contexts in task outcomes.

We provide additional experiments and analyses in Appendix D, including robustness to prompt attacks, efficiency, generated graph statistics, and qualitative case studies.

6 Conclusion

We propose RCGP, a query-conditioned graph policy based on the view that adaptive multi-agent collaboration is both structurally coupled and functionally asymmetric. Based on this perspective, we formulate participation, functional roles, and directed communication within a unified graph-policy action space, allowing collaboration scale and topology to adapt jointly through parallel node–edge decisions. We further characterize separate receiving and sending role contexts over direct and shared relations, preserving their distinct functional meanings and enriching graph decisions beyond role identity and connectivity alone. Starting from structure-aware initialization, execution-guided reinforcement learning refines both graph decisions and the role-context representations that inform them according to task-level utility. Across six benchmarks, RCGP consistently outperforms 14 baselines, while further analyses support the effectiveness of the proposed design and its competitive token efficiency. Overall, the graph policy provides RL with a structured, role-context-aware collaboration space, while RL grounds this space in task-level execution utility, moving graph decisions beyond reliance on structural priors alone.

AI USE STATEMENT

We used generative AI tools solely to improve the clarity, grammar, and presentation of this manuscript. We reviewed all AI-assisted revisions to ensure that they accurately reflect our technical content and claims. We take full responsibility for the final manuscript, including any text generated or revised with AI assistance.

REPRODUCIBILITY STATEMENT

Appendix B presents additional technical details of RCGP. Appendix C specifies the model architecture, LLM decoding settings, and optimization configuration. We plan to make the implementation code publicly available upon acceptance.

REFERENCES

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. Chateval: Towards better llm-based evaluators through multi-agent debate. In *The International conference on learning representations*, volume 2024, pp. 9079–9093, 2024.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, et al. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *The International Conference on Learning Representations*, volume 2024, pp. 20094–20136, 2024.

Yongchao Chen, Jiefeng Chen, Rui Meng, Ji Yin, Na Li, Chuchu Fan, Chi Wang, Tomas Pfister, and Jinsung Yoon. TUMIX: Multi-agent test-time scaling with tool-use mixture. In *The International Conference on Learning Representations*, 2026.

Yufan Dang, Chen Qian, Xueheng Luo, Jingru Fan, Zihao Xie, Ruijie Shi, Weize Chen, Cheng Yang, Xiaoyin Che, Ye Tian, Xuantang Xiong, Lei Han, Zhiyuan Liu, and Maosong Sun. Multi-agent collaboration via evolving orchestration. In *The Annual Conference on Neural Information Processing Systems*, 2025.

Wentao Deng, Jiahuan Pei, Zhiwei Xu, Zhaochun Ren, Zhumin Chen, and Pengjie Ren. Belief-calibrated multi-agent consensus seeking for complex NLP tasks. In *The Annual Conference on Neural Information Processing Systems*, 2025.

Jiangwen Dong, Zehui Lin, Wanyu Lin, and Mingjin Zhang. S-dag: A subject-based directed acyclic graph for multi-agent heterogeneous reasoning. In *The AAAI Conference on Artificial Intelligence*, volume 40, pp. 29394–29402, 2026.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *The International Conference on Machine Learning*, 2024.

Shengda Fan, Xin Cong, Yuepeng Fu, Zhong Zhang, Shuyan Zhang, Yuanwei Liu, Yesai Wu, Yankai Lin, Zhiyuan Liu, and Maosong Sun. WorkflowLLM: Enhancing workflow orchestration capability of large language models. In *The International Conference on Learning Representations*, 2025.

Wei Fan, JinYi Yoon, and Bo Ji. imad: Intelligent multi-agent debate for efficient and accurate llm inference. In *The AAAI Conference on Artificial Intelligence*, volume 40, pp. 29403–29411, 2026a.

Yijia Fan, Jusheng Zhang, Kaitong Cai, Jing Yang, Chengpei Tang, Jian Wang, and Keze Wang. Cost-effective communication: An auction-based method for language agent interaction. In *The AAAI Conference on Artificial Intelligence*, volume 40, pp. 29412–29420, 2026b.

Shangbin Feng, Zifeng Wang, Palash Goyal, Yike Wang, Weijia Shi, Huang Xia, Hamid Palangi, Luke Zettlemoyer, Yulia Tsvetkov, Chen-Yu Lee, et al. Heterogeneous swarms: Jointly optimizing model roles and weights for multi-llm systems. *Advances in Neural Information Processing Systems*, 38:114319–114351, 2025.

Tao Feng, Haozhen Zhang, Zijie Lei, Peixuan Han, and Jiaxuan You. Graphplanner: Graph memory-augmented agentic routing for multi-agent LLMs. In *The International Conference on Learning Representations*, 2026.

Joseph Fioresi, Parth Parag Kulkarni, Ashmal Vayani, Song Wang, and Mubarak Shah. Learning to share: Selective memory for efficient parallel agentic systems. In *The International Conference on Machine Learning*, 2026.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. Complexity-based prompting for multi-step reasoning. In *The International Conference on Learning Representations*, 2023.

Yao Guan, Lin Wang, Zhihui Lu, Ziyi Wang, Wenzhu Yan, and Qiang Duan. MOC: Multi-order communication in LLM-based multi-agent systems. In *The International Conference on Machine Learning*, 2026.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The International Conference on Learning Representations*, 2024.

Zhi Hong, Qian Zhang, Jiahang Sun, Zhiwei Shang, Mingze Kong, Xiangyi Wang, Yao Shu, and Zhongxiang Dai. MASPOB: Bandit-based prompt optimization for multi-agent systems with graph neural networks. In *The International Conference on Machine Learning*, 2026.

Mengkang Hu, Yuhang Zhou, Wendong Fan, Yuzhou Nie, Ziyu Ye, Bowei Xia, Tao Sun, Zhaoxuan Jin, Yingru Li, Zeyu Zhang, Yifeng Wang, Qianshuo Ye, Bernard Ghanem, Ping Luo, and Guohao Li. OWL: Optimized workforce learning for general multi-agent assistance in real-world task automation. In *The Annual Conference on Neural Information Processing Systems*, 2025a.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *The International Conference on Learning Representations*, 2025b.

Yue Hu, Yuzhu Cai, Yaxin Du, Xinyu Zhu, Xiangrui Liu, Zijie Yu, Yuchen Hou, Shuo Tang, and Siheng Chen. Self-evolving multi-agent collaboration networks for software development. In *The International Conference on Learning Representations*, 2025c.

Yuntong Hu, Yuting Zhang, Matthew Trager, Yi Zhang, Shuo Yang, Wei Xia, and Stefano Soatto. Evomas: Evolutionary generation of multi-agent systems. In *The International Conference on Machine Learning*, 2026.

Xingyuan Hua, Sheng Yue, Xinyi Li, Yizhe Zhao, Jinrui Zhang, and Ju Ren. Context learning for multi-agent discussion. In *The International Conference on Learning Representations*, 2026.

Jing Huang, Lidong Zhang, Mutian Bao, Yadong Li, Xingzhong Xu, Jinjian Zhang, Jie Liu, Ming Kong, and Qiang Zhu. MAS-architect: Declarative multi-agent system design via separation of concerns. In *The International Conference on Machine Learning*, 2026.

Eric Hanchen Jiang, Levina Li, Frank Wan, Xiao Liang, Sophia Yin, Yuchen Wu, Xinfeng Li, Yizhou Sun, Wei Wang, Kai-Wei Chang, et al. Dynamic generation of multi llm agents communication topologies with graph diffusion models. In *The Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 38042–38060, 2026.

Haibo Jin, Peng Kuang, Ye Yu, Xiaopeng Yuan, and Haohan Wang. Agent primitives: Reusable latent building blocks for multi-agent systems. In *The International Conference on Machine Learning*, 2026.

Zixuan Ke, Yifei Ming, Austin Xu, Ryan Chin, Xuan-Phi Nguyen, Prathyusha Jwalapuram, Jiayu Wang, Semih Yavuz, Caiming Xiong, and Shafiq Joty. MAS-orchestra: Understanding and improving multi-agent reasoning through holistic orchestration and controlled benchmarks. In *The International Conference on Machine Learning*, 2026.

Yubin Kim, Chanwoo Park, Hyewon Jeong, Yik S Chan, Xuhai Xu, Daniel McDuff, Hyeonhoon Lee, Marzyeh Ghassemi, Cynthia Breazeal, and Hae W Park. Mdagents: An adaptive collaboration of llms for medical decision-making. *Advances in Neural Information Processing Systems*, 37:79410–79452, 2024.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Lars Henning Klein, Nearchos Potamitis, Roland Aydin, Robert West, Caglar Gulcehre, and Akhil Arora. Fleet of agents: Coordinated problem solving with large language models. In *The International Conference on Machine Learning*, 2025.

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

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for "mind" exploration of large language model society. *Advances in neural information processing systems*, 36:51991–52008, 2023.

Hongwei Li, Zhun Wang, Qinrun Dai, Yuzhou Nie, Jinjun Peng, Ruitong Liu, Jingyang Zhang, Kaijie Zhu, Jingxuan He, Lun Wang, Yangruibo Ding, Yueqi Chen, Wenbo Guo, and Dawn Song. Opensage: Self-programming agent generation engine. In *The International Conference on Machine Learning*, 2026a.

Shijun Li, Hilaf Hasson, and Joydeep Ghosh. OMAC: A holistic optimization framework for LLM-based multi-agent collaboration. In *The International Conference on Machine Learning*, 2026b.

Shiyuan Li, Yixin Liu, Qingsong Wen, Chengqi Zhang, and Shirui Pan. Assemble your crew: Automatic multi-agent communication topology design via autoregressive graph generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 23142–23150, 2026c.

Yu Li, Lehui Li, Zhihao Wu, Qingmin Liao, Jianye Hao, Kun Shao, and Fengli Xu. Agentswift: Efficient llm agent design via value-guided hierarchical search. In *The AAAI Conference on Artificial Intelligence*, volume 40, pp. 31843–31851, 2026d.

Zhening Li, Armando Solar-Lezama, Yisong Yue, and Stephan Zheng. Encompass: Enhancing agent programming with search over program execution paths. In *The Annual Conference on Neural Information Processing Systems*, 2025b.

Zhuofeng Li, Haoxiang Zhang, Seungju Han, Sheng Liu, Jianwen Xie, Yu Zhang, Yejin Choi, James Zou, and Pan Lu. In-the-flow agentic system optimization for effective planning and tool use. In *The International Conference on Learning Representations*, 2026e.

Fulin Lin, Shaowen Chen, Ruishan Fang, Hongwei Wang, and Tao Lin. Stop wasting your tokens: Towards efficient runtime multi-agent systems. In *The International Conference on Learning Representations*, 2026.

Jie Ma, Zhitao Gao, Qi Chai, Wangchun Sun, Pinghui Wang, Hongbin Pei, Jing Tao, Lingyun Song, Jun Liu, Chen Zhang, et al. Debate on graph: a flexible and reliable reasoning framework for large language models. In *The AAAI Conference on Artificial Intelligence*, volume 39, pp. 24768–24776, 2025.

Ming Ma, Jue Zhang, Fangkai Yang, Yu Kang, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. Dover: Intervention-driven auto debugging for LLM multi-agent systems. In *The International Conference on Learning Representations*, 2026.

Stefan Nielsen, Edoardo Cetin, Peter Schwendeman, Qi Sun, Jinglue Xu, and Yujin Tang. Learning to orchestrate agents in natural language with the conductor. In *The International Conference on Learning Representations*, 2026.

Boye Niu, Yiliao Song, Kai Lian, Yifan Shen, Yu Yao, Kun Zhang, and Tongliang Liu. Flow: Modularized agentic workflow automation. In *The International Conference on Learning Representations*, volume 2025, pp. 74949–74977, 2025.

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

J Rosser and Jakob Nicolaus Foerster. Agentbreeder: Mitigating the AI safety risks of multi-agent scaffolds via self-improvement. In *The Annual Conference on Neural Information Processing Systems*, 2025.

Jianhao Ruan, Zhihao Xu, Yiran Peng, Fashen Ren, Zhaoyang Yu, Xinbing Liang, Jinyu Xiang, Yongru Chen, Bang Liu, Chenglin Wu, Yuyu Luo, and Jiayi Zhang. AOrchestra: Automating sub-agent creation for agentic orchestration. In *The International Conference on Machine Learning*, 2026.

Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. Agentsquare: Automatic llm agent search in modular design space. In *International Conference on Learning Representations*, volume 2025, pp. 3841–3865, 2025.

Shuai Shao, Yixiang Liu, Bingwei Lu, and Weinan Zhang. Monoscale: Scaling multi-agent system with monotonic improvement. In *The International Conference on Machine Learning*, 2026.

Jinzhou Tang, Jusheng Zhang, Qinhan Lv, Sidi Liu, Jing Yang, Chengpei Tang, and Keze Wang. Hiva: Self-organized hierarchical variable agent via goal-driven semantic-topological evolution. In *The AAAI Conference on Artificial Intelligence*, volume 40, pp. 2083–2091, 2026.

Nurbek Tastan, Samuel Horváth, and Karthik Nandakumar. Stochastic self-organization in multi-agent systems. In *The International Conference on Learning Representations*, 2026.

Patara Trirat, Wonyong Jeong, and Sung Ju Hwang. Multi-view encoders for performance prediction in llm-based agentic workflows. In *The International Conference on Learning Representations*, volume 2026, pp. 44047–44068, 2026.

Varun Ursekar, Apaar Shanker, Veronica Chatrath, Yuan Xue, and Samuel Marc Denton. VeRO: A harness for agents to optimize agents. In *The International Conference on Machine Learning*, 2026.

Kaustubh Vyas, Damien Graux, Yijun Yang, Sebastien Montella, Chenxin Diao, Wendi Zhou, Pavlos Vougiouklis, Ruofei Lai, Yang Ren, Keshuang Li, and Jeff Z. Pan. From an LLM swarm to a PDDL-empowered hive: Planning self-executed instructions in a multi-modal jungle. In *The International Conference on Learning Representations*, 2025.

Guancheng Wan, Mo Zhou, Ziyi Wang, Xiaoran Shang, Eric Hanchen Jiang, Guibin Zhang, Jinhe Bi, Yunpu Ma, Zaixi Zhang, Ke Liang, et al. Dawn: Distributed llm multi-agent workflow synthesis. In *The AAAI Conference on Artificial Intelligence*, volume 40, pp. 26099–26106, 2026.

Chenyu Wang, Zishen Wan, Hao Kang, Emma Chen, Zhiqiang Xie, Tushar Krishna, Vijay Janapa Reddi, and Yilun Du. SLM-MUX: Orchestrating small language models for reasoning. In *The International Conference on Learning Representations*, 2026a.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Y Zou. Mixture-of-agents enhances large language model capabilities. In *The International Conference on Learning Representations*, volume 2025, pp. 33944–33963, 2025a.

Kun Wang, Guibin Zhang, ManKit Ye, Xinyu Deng, Dongxia Wang, Xiaobin Hu, Jinyang Guo, Yang Liu, and Yufei Guo. MAS^2: Self-generative, self-configuring, self-rectifying multi-agent systems. In *The International Conference on Learning Representations*, volume 2026, pp. 113586–113613, 2026b.

Siyu Wang, Ruotian Lu, Zhihao Yang, Yuchao Wang, Yanzhou Zhang, Lei Xu, Qimin Xu, Guojun Yin, Cailian Chen, and Xinping Guan. Agentconductor: Topology evolution for multi-agent competition-level code generation. In *The International Conference on Machine Learning*, 2026c.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in neural information processing systems*, 33:5776–5788, 2020.

Yanbo Wang, Zixiang Xu, Yue Huang, Xiangqi Wang, Zirui Song, Lang Gao, Chenxi Wang, Robert Tang, Yue Zhao, Arman Cohan, et al. Dyflow: Dynamic workflow framework for agentic reasoning. *Advances in Neural Information Processing Systems*, 38:174148–174181, 2025b.

Zhenhailong Wang, Shaoguang Mao, Wenshan Wu, Tao Ge, Furu Wei, and Heng Ji. Unleashing the emergent cognitive synergy in large language models: A task-solving agent through multi-persona self-collaboration. In *The Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 257–279, 2024.

Zhexuan Wang, Xuebo Liu, Li Wang, Zifei Shan, Yutong Wang, Zhenxi Song, and Min Zhang. MASPO: Joint prompt optimization for LLM-based multi-agent systems. In *The International Conference on Machine Learning*, 2026d.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Yangbo Wei, Zhen Huang, Ronghao Xu, Hong Wang, and WEI W. XING. EvoMAS: Heuristics in the loop—evolving smarter agentic workflows. In *The International Conference on Machine Learning*, 2026.

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

Tongtong Wu, Yanming Li, Ziye Tang, Chen Jiang, Linhao Luo, Guilin Qi, Shirui Pan, and Gholamreza Haffari. CARD: Towards conditional design of multi-agent topological structures. In *The International Conference on Learning Representations*, 2026.

Yihan Xia, Taotao Wang, Shengli Zhang, Zhangyuhua Weng, Bin Cao, and Soung Chang Liew. Hivemind: Contribution-guided online prompt optimization of llm multi-agent systems. In *The AAAI Conference on Artificial Intelligence*, volume 40, pp. 29767–29774, 2026.

Keyi Xiang, Zeyu Feng, Zhuoyi Lin, Yueming Lyu, Shi Boyuan, Yew-Soon Ong, Ivor Tsang, and Haiyan Yin. Flowsearcher: Synthesizing memory-guided agentic workflows for web information seeking. In *The International Conference on Learning Representations*, 2026.

Jinglue Xu, Qi Sun, Peter Schwendeman, Stefan Nielsen, Edoardo Cetin, and Yujin Tang. Trinity: An evolved llm coordinator. In *The International Conference on Learning Representations*, volume 2026, pp. 70195–70224, 2026a.

Yuanyuan Xu, Wenjie Zhang, Yin Chen, Xuemin Lin, and Ying Zhang. Self-evolving agents as dynamic graph transformation: A survey and new perspective, 2026b.

Peiting Yang, Jiahao Shi, Caiyi Xu, Ming Liu, Yanxia Wu, and Rongsheng Li. Agenttailor: A semantic-aware LLM-based multi-agent system with actor-critic structure. In *The International Conference on Machine Learning*, 2026.

Yingxuan Yang, Huacan Chai, Shuai Shao, Yuanyi Song, Siyuan Qi, Renting Rui, and Weinan Zhang. Agentnet: Decentralized evolutionary coordination for llm-based multi-agent systems. *Advances in Neural Information Processing Systems*, 38:107309–107336, 2025.

Tianjun Yao, Zhaoyi Li, and Zhiqiang Shen. HieraMAS: Optimizing intra-node LLM mixtures and inter-node topology for multi-agent systems. In *The International Conference on Machine Learning*, 2026.

Yu Yao, Yiliao Song, Yian Xie, Mengdan Fan, Mingyu Guo, and Tongliang Liu. Can dependencies induced by LLM-agent workflows be trusted? In *The Annual Conference on Neural Information Processing Systems*, 2025.

Rui Ye, Shuo Tang, Rui Ge, Yaxin Du, Zhenfei Yin, Siheng Chen, and Jing Shao. MAS-GPT: Training LLMs to build LLM-based multi-agent systems. In *The International Conference on Machine Learning*, 2025.

Hongyeon Yu, Young-Bum Kim, and Yoon Kim. Flowbot: Inducing LLM workflows with bilevel optimization and textual gradients. In *The International Conference on Machine Learning*, 2026a.

Jingxuan Yu, Ju Jia, Simeng Qin, Xiaojun Jia, Siqi Ma, Yihao Huang, Yali Yuan, and Guang Cheng. Mpas: Breaking sequential constraints of multi-agent communication topologies via individual-epistemic message propagation. In *The AAAI Conference on Artificial Intelligence*, volume 40, pp. 29847–29855, 2026b.

Michelle Yuan, Khushbu Pahwa, Shuaichen Chang, Mustafa Kaba, Jiarong Jiang, Xiaofei Ma, Yi Zhang, and Monica Sunkara. Automated composition of agents: A knapsack approach for agentic component selection. *Advances in Neural Information Processing Systems*, 38:40597–40625, 2025.

Sukwon Yun, Jie Peng, Pingzhi Li, Wendong Fan, Jie Chen, James Y Zou, Guohao Li, and Tianlong Chen. Graph-of-agents: A graph-based framework for multi-agent llm collaboration. In *The International Conference on Learning Representations*, volume 2026, pp. 19745–19760, 2026.

Guibin Zhang, Muxin Fu, Kun Wang, Guancheng Wan, Miao Yu, and Shuicheng YAN. G-memory: Tracing hierarchical memory for multi-agent systems. In *The Annual Conference on Neural Information Processing Systems*, 2025a.

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, LEI BAI, and Xiang Wang. Multi-agent architecture search via agentic supernet. In *The International Conference on Machine Learning*, 2025b.

Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Yu, and Tianlong Chen. Cut the crap: An economical communication pipeline for llm-based multi-agent systems. In *The International Conference on Learning Representations*, volume 2025, pp. 75389–75428, 2025c.

Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. G-designer: Architecting multi-agent communication topologies via graph neural networks. In *The International Conference on Machine Learning*, 2025d.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, et al. Aflow: Automating agentic workflow generation. In *The International Conference on Learning Representations*, volume 2025, pp. 34040–34077, 2025e.

Jusheng Zhang, Zimeng Huang, Yijia Fan, Ningyuan Liu, Mingyan Li, Zhuojie Yang, Jiawei Yao, Jian Wang, and Keze Wang. KABB: Knowledge-aware bayesian bandits for dynamic expert coordination in multi-agent systems. In *The International Conference on Machine Learning*, 2025f.

Ruijia Zhang, Xinyan Zhao, Ruixiang Wang, Sigen Chen, Guibin Zhang, An Zhang, Kun Wang, and Qingsong Wen. Safesieve: From heuristics to experience in progressive pruning for llm-based multi-agent communication. In *The AAAI Conference on Artificial Intelligence*, volume 40, pp. 29892–29900, 2026a.

Yaolun Zhang, Xiaogeng Liu, and Chaowei Xiao. Metaagent: Automatically constructing multi-agent systems based on finite state machines. In *The International Conference on Machine Learning*, 2025g.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Ö Arık. Chain of agents: Large language models collaborating on long-context tasks. *Advances in Neural Information Processing Systems*, 37:132208–132237, 2024.

Table 4: Statistics of the six datasets used in our experiments.

Category	Dataset	Format	Metric	#Samples
General Reasoning	MMLU	Multiple choice	Accuracy	153
Mathematical Reasoning	GSM8K	Number	Exact Match	1,319
	SVAMP	Number	Exact Match	1,000
	AQuA	Multiple choice	Accuracy	254
Code Generation	HumanEval	Code	Pass@1	164
	MBPP	Code	Pass@1	427

Zhen Zhang, Wanjing Zhou, Juncheng Li, Hao Fei, Jun Wen, and Wei Ji. RADAR: Redundancy-aware diffusion for multi-agent communication structure generation. In *The International Conference on Machine Learning*, 2026b.

Mingming Zhao, Xiaokang Wei, Yuanqi Shao, Kaiwen Zhou, Lin Yang, Siwei Rao, Junhui Zhan, and Zhitang Chen. A^{2}flow: Automating agentic workflow generation via self-adaptive abstraction operators. In *The AAAI Conference on Artificial Intelligence*, volume 40, pp. 29930–29938, 2026.

Han Zhou, Xingchen Wan, Ruoxi Sun, Hamid Palangi, Shariq Iqbal, Ivan Vulić, Anna Korhonen, and Sercan Arik. Multi-agent design: Optimizing agents with better prompts and topologies. In *The International Conference on Learning Representations*, volume 2026, pp. 15844–15872, 2026.

Wei Zhu, Zhiwen Tang, and Kun Yue. SYMPHONY: Synergistic multi-agent planning with heterogeneous language model assembly. In *The Annual Conference on Neural Information Processing Systems*, 2025.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTSwarm: Language agents as optimizable graphs. In *The International Conference on Machine Learning*, 2024.

A Dataset Statistics

We present the statistics of three types of benchmark datasets in Table 4. Following (Zhang et al., 2025d), we optimize RCGP on a small subset of each benchmark and freeze the learned parameters for testing on the remaining queries. We use 20 queries for training on MMLU and 50 queries for each of the other benchmarks.

B Additional Technical Details of the Proposed Method

This section provides additional details of the initial context graph, role-context graph predictor, constrained graph sampling, structure-aware initialization, and policy-gradient optimization used in RCGP.

B.1 Initial Context Graph

The initial augmented graph $\widetilde{G}^{(0)} = (\mathbf{X}^{(0)}, \widetilde{\mathbf{A}}^{(0)})$ provides the relational context used by the graph predictor. Let q_X and q_A denote categorical priors over node and admissible edge states, respectively. We independently sample $X_i^{(0)} \sim q_X$ for each candidate slot $i \in \mathcal{S}$ and $\widetilde{A}_{ij}^{(0)} \sim q_A$ for each $(i, j) \in \Omega$; entries outside Ω are fixed to zero. The categorical priors are estimated with add-one smoothing from an augmented structural corpus constructed using commonly used multi-agent collaboration templates, as detailed in Appendix B.4.

B.2 Details of Role-Context Graph Prediction

We provide the detailed operations underlying the directional aggregation in equation 8. For candidate slot i, let $\Omega_i^{\mathrm{recv}} = \{(u, v) \in \Omega \mid v = i\}$ and $\Omega_i^{\mathrm{send}} = \{(u, v) \in \Omega \mid u = i\}$ denote its incoming

and outgoing candidate edge sets. These sets contain all admissible candidate edges, including those absent from the initial context graph.

Directional attention inputs. For a candidate edge $(u, v) \in \Omega$ at predictor layer ℓ, we construct

$$\begin{aligned} \mathbf{m}_{uv}^{\mathrm{recv},\ell} &= (\mathbf{h}_u^\ell + \mathbf{s}_u^{\mathrm{recv}}) \| \mathbf{h}_v^\ell \| \mathbf{e}_{uv}^\ell \| \mathbf{g}, \\ \mathbf{m}_{uv}^{\mathrm{send},\ell} &= \mathbf{h}_u^\ell \| (\mathbf{h}_v^\ell + \mathbf{s}_v^{\mathrm{send}}) \| \mathbf{e}_{uv}^\ell \| \mathbf{g}. \end{aligned} \tag{11}$$

Thus, receiving aggregation incorporates the source's receiving context, whereas sending aggregation incorporates the recipient's sending context. For a candidate interaction $u \to v$, these two quantities provide upstream and downstream role-context information, respectively.

Directional attention. A transformation M_ℓ, shared across the two directions, maps each attention input to a message vector. Separate attention functions a_ℓ^{recv} and a_ℓ^{send} produce direction-specific scores:

$$\widehat{\mathbf{m}}_{uv}^{d,\ell} = M_\ell(\mathbf{m}_{uv}^{d,\ell}), \qquad \eta_{uv}^{d,\ell} = a_\ell^d(\mathbf{m}_{uv}^{d,\ell}), \qquad d \in \{\mathrm{recv}, \mathrm{send}\}. \tag{12}$$

For node i, the scores are normalized over the corresponding candidate set:

$$\alpha_{uv}^{d,\ell} = \frac{\exp(\eta_{uv}^{d,\ell})}{\sum_{(p,q)\in\Omega_i^d} \exp(\eta_{pq}^{d,\ell})}, \qquad (u, v) \in \Omega_i^d. \tag{13}$$

The resulting directional aggregation is

$$\mathbf{c}_i^{d,\ell} = \sum_{(u,v)\in\Omega_i^d} \alpha_{uv}^{d,\ell} \widehat{\mathbf{m}}_{uv}^{d,\ell}, \qquad d \in \{\mathrm{recv}, \mathrm{send}\}. \tag{14}$$

If Ω_i^d is empty, we set $\mathbf{c}_i^{d,\ell} = \mathbf{0}$.

Node and edge updates. The two directional contexts are fused with the current node representation and query embedding using residual updates:

$$\begin{aligned} \widehat{\mathbf{h}}_i^{\ell+1} &= \mathrm{LN}\left(\mathbf{h}_i^\ell + F_\ell^V\left(\mathbf{h}_i^\ell \| \mathbf{c}_i^{\mathrm{recv},\ell} \| \mathbf{c}_i^{\mathrm{send},\ell} \| \mathbf{g}\right)\right), \\ \mathbf{h}_i^{\ell+1} &= \mathrm{LN}\left(\widehat{\mathbf{h}}_i^{\ell+1} + \mathrm{FFN}_\ell^V(\widehat{\mathbf{h}}_i^{\ell+1})\right). \end{aligned} \tag{15}$$

Here, F_ℓ^V is a learnable node transformation, FFN_ℓ^V is a node feed-forward network, and LN denotes layer normalization.

Each edge representation is subsequently updated from its ordered source and recipient representations:

$$\begin{aligned} \widehat{\mathbf{e}}_{uv}^{\ell+1} &= \mathrm{LN}\left(\mathbf{e}_{uv}^\ell + F_\ell^E\left(\mathbf{h}_u^{\ell+1} \| \mathbf{h}_v^{\ell+1} \| \mathbf{e}_{uv}^\ell \| \mathbf{g}\right)\right), \\ \mathbf{e}_{uv}^{\ell+1} &= \mathrm{LN}\left(\widehat{\mathbf{e}}_{uv}^{\ell+1} + \mathrm{FFN}_\ell^E(\widehat{\mathbf{e}}_{uv}^{\ell+1})\right), \end{aligned} \tag{16}$$

where F_ℓ^E and FFN_ℓ^E are the corresponding edge transformations. The ordered concatenation preserves the source–recipient distinction when updating a directed edge.

Node and edge prediction heads. After L layers, the final node and edge representations are mapped to categorical distributions through query-conditioned prediction heads:

$$\begin{aligned} \mathbf{p}_i^X &= \mathrm{softmax}\big(W_X \mathbf{h}_i^L + \Gamma_i^X(\mathbf{g})\big), \quad i \in \mathcal{S}, \\ \mathbf{p}_{ij}^A &= \mathrm{softmax}\big(W_A \mathbf{e}_{ij}^L + \Gamma_{ij}^A(\mathbf{g})\big), \quad (i, j) \in \Omega. \end{aligned} \tag{17}$$

Here, W_X and W_A are learnable projections shared across candidate slots and admissible edges, respectively. The learnable maps Γ_i^X and Γ_{ij}^A provide position-specific query contributions in the logit space, with $\Gamma_i^X(\mathbf{g}) \in \mathbb{R}^{|\mathcal{C}|+1}$ and $\Gamma_{ij}^A(\mathbf{g}) \in \mathbb{R}^2$. Softmax is applied over the corresponding categories: $\mathbf{p}_i^X$ assigns probabilities to functional roles and inactivity in $\mathcal{C} \cup \{\perp\}$, while $\mathbf{p}_{ij}^A$ assigns probabilities to edge absence and presence in $\{0, 1\}$. All prediction-head parameters are included in θ. Conditioned on the initial augmented graph $\widetilde{G}^{(0)}$ and query $\mathcal{Q}$, these factors define the graph policy in equation 9 and are conditionally independent, allowing their distributions to be evaluated in parallel after one policy evaluation. The nonempty-graph constraint is subsequently applied as described in Appendix B.3, yielding the actual sampling distribution $\bar{\pi}_\theta$.

B.3 Constrained Parallel Graph Sampling

Equation 9 factorizes the graph policy over all node and admissible edge actions, allowing their categorical distributions to be evaluated in parallel. We additionally impose a nonempty-graph constraint during sampling.

Let $\mathbf{p}_i^X[\perp]$ denote the predicted probability that slot i is inactive. We first select

$$i^\star = \arg\max_{i \in \mathcal{S}} \left(1 - \mathbf{p}_i^X[\perp]\right), \tag{18}$$

i.e., the slot with the largest predicted probability of taking an active role. For this slot, we mask the inactive category and renormalize the remaining probabilities:

$$\bar{\mathbf{p}}_{i^\star}^X[x] = \begin{cases} 0, & x = \perp, \\ \dfrac{\mathbf{p}_{i^\star}^X[x]}{1 - \mathbf{p}_{i^\star}^X[\perp]}, & x \in \mathcal{C}. \end{cases} \tag{19}$$

All other node distributions and all edge distributions remain unchanged. We denote the resulting constrained sampling distribution by $\bar{\pi}_\theta$. After sampling the augmented graph, inactive nodes and their incident edges are removed according to equation 4.

B.4 Structure-Aware Initialization

Before execution-guided optimization, we initialize the graph predictor and role-context encoders using synthetic role-labeled graph structures. Inspired by Zhang et al. (2026b), we construct candidate graphs from fully connected, mesh, star, layered, and random templates for each training query. For a template containing n active nodes, we randomly insert $N_{\max} - n$ inactive slots while preserving the relative order of active nodes. Edges incident to inactive slots are sampled with probability 0.5 subject to the topological ordering. The empirical node and edge frequencies of this augmented corpus, with add-one smoothing, define the categorical priors used to construct $\widetilde{G}^{(0)}$.

For an augmented target $\widetilde{G}_b^\star = (\mathbf{X}_b^\star, \widetilde{\mathbf{A}}_b^\star)$, let $\mathbf{p}_{b,i}^X$ and $\mathbf{p}_{b,ij}^A$ denote the node and edge distributions predicted from the corresponding initialization input. We define

$$\ell_X^{(b)} = -\frac{1}{N_{\max}} \sum_{i \in \mathcal{S}} \log \mathbf{p}_{b,i}^X[X_{b,i}^\star], \qquad \ell_A^{(b)} = -\frac{1}{|\Omega|} \sum_{(i,j) \in \Omega} \log \mathbf{p}_{b,ij}^A[\widetilde{A}_{b,ij}^\star]. \tag{20}$$

The node loss supervises both functional roles and inactivity, whereas the edge loss supervises directed connectivity. To place greater weight on targets with locally diverse direct role neighborhoods, we use

$$w(G_b^\star) = 1 + \frac{1}{2}\left(\bar{H}^{\text{in}}(G_b^\star) + \bar{H}^{\text{out}}(G_b^\star)\right). \tag{21}$$

Here, $G_b^\star$ denotes the active subgraph of $\widetilde{G}_b^\star$ obtained by removing inactive nodes and their incident edges according to equation 4. $\bar{H}^{\text{in}}$ and $\bar{H}^{\text{out}}$ are the normalized role entropies averaged over active agents with nonempty incoming and outgoing neighborhoods, respectively.

For a batch of B initialization samples, the weighted objective is

$$\mathcal{L}_{\text{init}} = \frac{\sum_{b=1}^B w(G_b^\star)\left(\ell_X^{(b)} + \lambda_A \ell_A^{(b)}\right)}{\sum_{b=1}^B w(G_b^\star)}, \tag{22}$$

where λ_A controls the relative contribution of edge supervision. This stage provides a structural initialization for the subsequent execution-guided policy optimization.

B.5 Policy-Gradient Optimization

We refine the initialized graph policy using REINFORCE with the execution reward in equation 10. For each training query q, we sample an augmented graph $\widetilde{G}_q$ from the constrained policy and execute its active subgraph to obtain the reward. Let $\widehat{A}_q$ denote the standardized and clipped reward

advantage relative to a query-specific exponential moving average baseline. For a batch $\mathcal{B}$, we minimize the following loss while holding the sampled advantages fixed:

$$\mathcal{L}_{\mathrm{RL}} = -\frac{1}{|\mathcal{B}|}\sum_{q\in\mathcal{B}}\left[\widehat{A}_q \ell_q^{\pi} + \beta_H \mathcal{H}_q\right], \tag{23}$$

where $\ell_q^{\pi} = M^{-1}\log \bar{\pi}_\theta(\widetilde{G}_q \mid \widetilde{G}_q^{(0)}, q)$, $M = N_{\max} + |\Omega|$, and $\mathcal{H}_q$ is the mean entropy of the M node and edge categorical distributions. The coefficient β_H controls entropy regularization. Log-probabilities and entropies are evaluated under the constrained policy before inactive-node pruning. Execution feedback jointly updates the graph predictor and directional role-context encoders.

C Implementation Details

All main experiments use `GPT-4o-mini` through the OpenAI API[1], with a decoding temperature of 0.2 and a maximum output budget of 1,000 tokens per call. After communication, one additional LLM call aggregates the outputs of all agents. This aggregation call is included in the reported token consumption and end-to-end inference time. All baselines are evaluated using code from their official repositories and the parameter settings recommended by their authors. Results are averaged over multiple runs.

For the proposed method, queries are encoded using frozen `all-MiniLM-L6-v2` (Wang et al., 2020). The graph predictor uses two directional aggregation layers, each with receiving and sending attention and a hidden dimension of 128. We set $N_{\max} = 5$ and use one communication round. For each training query, we instantiate fully connected, mesh, star, layered, and random templates at each base size in $\{3, \ldots, N_{\max}\}$. Both structure-aware initialization and execution-guided RL use Adam (Kingma & Ba, 2014) with $(\beta_1, \beta_2) = (0.9, 0.999)$ and learning rates of 10^{-4} and 3×10^{-5}, respectively. For structure-aware initialization, we use the reconstruction losses and graph-weighted objective in equation 22, with $\lambda_A = 1$. For the RL optimization, we set $\lambda_C = 0.9$ and $\lambda_S = 0.1$ in equation 10.

D Additional Experiments

D.1 Robustness Analysis

We evaluate robustness on MMLU by replacing the role prompts of approximately 40% of participating agents with instructions to provide deliberately incorrect information. Figure 4 reports accuracy before and after the attacks.

Finding 8: The learned collaboration graphs exhibit robustness to role-prompt attacks. Under role-prompt attacks, RCGP's accuracy decreases from 84.21% to 82.71%, a drop of 1.50 percentage points. This degradation matches RADAR's and is smaller than the decreases of 4.51 and 4.14 percentage points observed for MaAS and ARG-Designer, respectively. Meanwhile, RCGP retains the highest accuracy under attack, exceeding RADAR by 2.26 percentage points. These results support the robustness of RCGP under the evaluated attacks, suggesting that its collaboration structures tolerate partial corruption of agent role instructions while maintaining strong task performance.

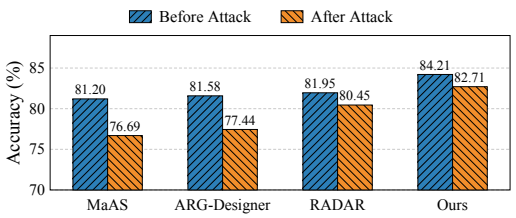


Figure 4: Accuracy of four methods before and after prompt attacks on MMLU.

D.2 Efficiency

Table 5 compares overall token consumption and wall-clock training and inference time on MMLU.

[1] `https://api.openai.com/v1/chat/completions`

Table 6: Statistics of collaboration graphs generated on MMLU. Structural statistics are reported as mean ± standard deviation across queries. $\bar{H}^c$ denotes normalized role entropy averaged over nonempty neighborhoods of relation c.

Method	Acc. (%)	N_{call}	N_{eff}	$\lvert E\rvert$	Density	Inactive Ratio	$\bar{H}^{\text{in}}$	$\bar{H}^{\text{out}}$	$\bar{H}^{\text{src}}$	$\bar{H}^{\text{tgt}}$
ARG-Designer	81.58	3.29 ± 0.80	3.29 ± 0.80	2.36 ± 0.89	0.33 ± 0.09	–	0.007 ± 0.026	0.123 ± 0.131	0.000 ± 0.000	0.000 ± 0.000
RADAR	81.95	5.00 ± 0.00	4.69 ± 0.58	5.05 ± 1.67	0.29 ± 0.07	–	0.180 ± 0.114	0.153 ± 0.100	0.122 ± 0.136	0.154 ± 0.160
RCGP	84.21	3.84 ± 0.85	3.74 ± 0.99	4.28 ± 2.34	0.38 ± 0.09	0.23 ± 0.17	0.161 ± 0.126	0.156 ± 0.115	0.127 ± 0.166	0.126 ± 0.157

Finding 9: The coupled graph policy reduces training overhead while maintaining efficient inference. RCGP consumes 1.11×10^6 tokens overall, reducing token usage by 6.7% and 34.3% relative to GTD and RADAR, respectively. Training takes 20.06 minutes, corresponding to reductions of 50.7% and 76.9% against the same baselines. Inference completes in 2.30 minutes, providing a $1.28\times$ speedup over GTD and a modest 1.7% time reduction relative to RADAR. Thus, the unified graph policy achieves the lowest measured token consumption and training and inference times among the compared methods. These results support the computational efficiency of jointly learning participation, roles, and communication, with the principal runtime advantage occurring during training.

Table 5: Comparison of wall-clock time and overall token consumption on MMLU.

Methods	Overall Token	Train	Infer
GTD	1.19×10^6	40.67 min	2.95 min
RADAR	1.69×10^6	86.82 min	2.34 min
RCGP	1.11×10^6	20.06 min	2.30 min

D.3 Effect of Fixed Collaboration Scale

We evaluate RCGP with the number of active agents fixed to $K \in \{2, 3, 4, 5\}$ per query on MMLU, AQuA, and HumanEval. Figure 5 reports the results.

Finding 10: Query-adaptive participation outperforms every evaluated fixed team size. Overall, increasing the fixed team size does not consistently improve accuracy across benchmarks. Four active agents yield the best fixed results on AQuA and HumanEval, whereas five active agents perform best on MMLU. With query-adaptive participation, the full model exceeds these best fixed results by 1.13, 0.98, and 0.88 percentage points on MMLU, AQuA, and HumanEval, respectively. These gains over the best fixed configurations support the benefit of adapting agent participation to individual queries. By including inactivity in the node action space, RCGP allows collaboration scale to vary across queries while jointly determining which agents execute and which communication links remain feasible.

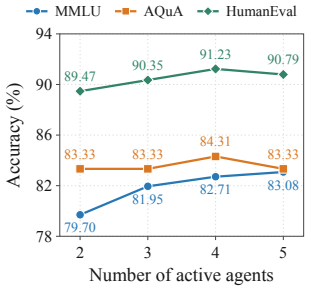
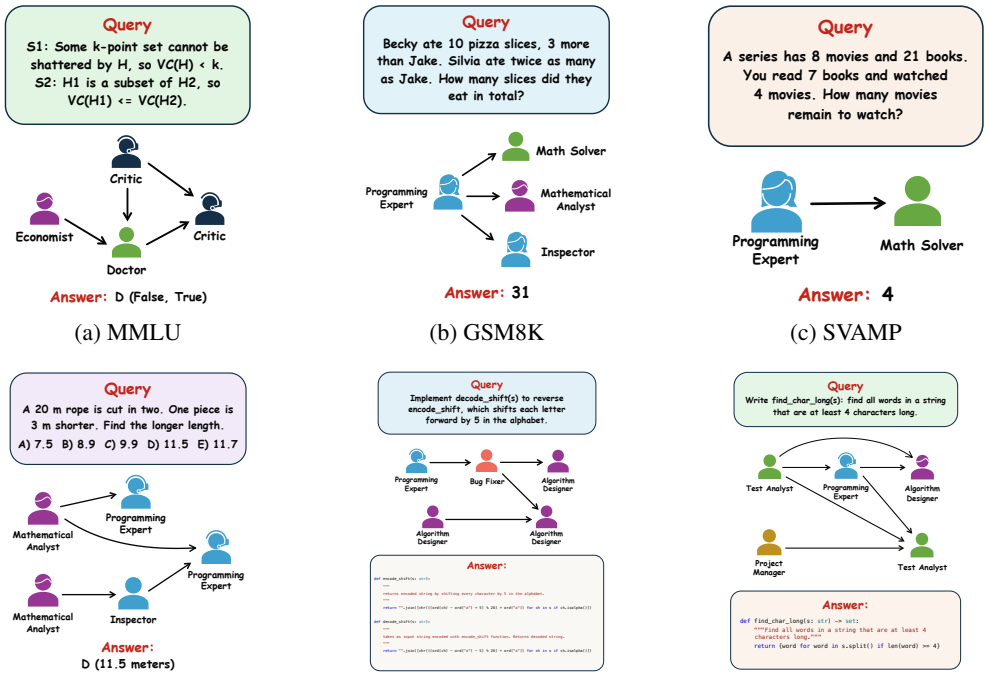


Figure 5: Accuracy with different fixed agent counts.

D.4 Distribution of Generated Collaboration Graphs

We compare the collaboration graphs generated by ARG-Designer, RADAR, and RCGP on MMLU. Table 6 reports structural statistics as mean ± standard deviation across queries. N_{call} counts executed role agents before final aggregation, while N_{eff} excludes isolated nodes. Density is computed as $\lvert E\rvert / \max\{N_{\text{eff}}(N_{\text{eff}} - 1), 1\}$. Inactive Ratio measures the proportion of augmented nodes predicted to be inactive, with "–" denoting not applicable. For each relation $c \in \{\text{in}, \text{out}, \text{src}, \text{tgt}\}$, corresponding to incoming, outgoing, shared-source, and shared-target relations, respectively, $\bar{H}^c$ denotes the average normalized role entropy over nonempty neighborhoods.

Finding 11: The coupled graph policy selects compact teams with concentrated communication. RCGP achieves the highest accuracy of 84.21% with an average of 3.84 executed agents and 4.28 communication edges. Compared with RADAR, it reduces agent calls by 23.2% and edges by approximately 15.2%, while improving accuracy by 2.26 percentage points. Its higher density shows that the smaller effective graphs use a larger fraction of the possible directed communication



(a) MMLU (b) GSM8K (c) SVAMP

(d) AQuA (e) HumanEval (f) MBPP

Figure 6: Case studies of query-specific collaboration graphs generated by RCGP.

links. Meanwhile, variation in the inactive ratio reflects different participation levels across queries. Together, these observations support learning participation and connectivity as structurally coupled decisions: the policy produces smaller participating teams while retaining communication structures that support strong task performance.

Finding 12: The role-context policy retains diversity across direct and shared neighborhoods. ARG-Designer generates the smallest graphs among the compared methods, but its mean incoming role entropy is only 0.007, with zero reported entropy for both shared-source and shared-target relations. In comparison, RCGP maintains nonzero mean role entropy across all four relations. These values remain broadly comparable to RADAR's despite using fewer effective agents, showing that compact collaboration graphs can retain heterogeneous receiving, sending, and shared-neighborhood contexts. Together with the role-context ablations in Table 3, these statistics support explicitly modeling local role organization: the learned graphs retain diverse functional relationships while achieving higher task accuracy at a smaller collaboration scale than RADAR.

D.5 Case Study

Figure 6 visualizes collaboration graphs generated by RCGP for one query from each benchmark, illustrating variation in participation and local role organization.

Finding 13: Participation decisions jointly shape collaboration scale and communication. The SVAMP example uses two agents connected by a single directed edge, while MMLU and GSM8K each use four agents, and AQuA, HumanEval, and MBPP use five. These cases illustrate how the graph policy realizes different participation levels within a unified action space. Because inactive agents and their incident edges are removed together, participation decisions determine both which agents execute and which communication links remain feasible. These examples illustrate how the policy enables variable collaboration scale, with participation decisions directly constraining the feasible communication structure.

E PROMPT

MMLU Role Profiles

```
Role: Knowlegable Expert
You are a knowlegable expert in question answering.
Please give several key entities that need to be searched in wikipedia
↪ to solve the problem, for example: catfish effect, broken window
↪ effect, Shakespeare.
If there is no entity in the question that needs to be searched in
↪ Wikipedia, you don't have to provide it

Role: Critic
You are an excellent critic.
Please point out potential issues in other agent's analysis point by
↪ point.

Role: Mathematician
You are a mathematician who is good at math games, arithmetic
↪ calculation, and long-term planning.

Role: Psychologist
You are a psychologist.
You are good at psychology, sociology, and philosophy.
You give people scientific suggestions that will make them feel
↪ better.

Role: Historian
You research and analyze cultural, economic, political, and social
↪ events in the past, collect data from primary sources and use it
↪ to develop theories about what happened during various periods of
↪ history.

Role: Doctor
You are a doctor and come up with creative treatments for illnesses or
↪ diseases.
You are able to recommend conventional medicines, herbal remedies and
↪ other natural alternatives.
You also consider the patient's age, lifestyle and medical history
↪ when providing your recommendations.

Role: Lawyer
You are good at law, politics, and history.

Role: Economist
You are good at economics, finance, and business.
You have experience on understanding charts while interpreting the
↪ macroeconomic environment prevailing across world economies.

Role: Programmer
You are good at computer science, engineering, and physics.
You have experience in designing and developing computer software and
↪ hardware.
```

GSM8K and SVAMP Role Profiles

```
Role: Math Solver
You are a math expert.
You will be given a math problem and hints from other agents.
Give your own solving process step by step based on hints.
The last line of your output contains only the final result without
↪ any units, for example: The answer is 140
You will be given some examples you may refer to.
```

````
Role: Mathematical Analyst
You are a mathematical analyst.
You will be given a math problem, analysis and code from other agents.
You need to first analyze the problem-solving process step by step,
↪  where the variables are represented by letters.
Then you substitute the values into the analysis process to perform
↪  calculations and get the results.
The last line of your output contains only the final result without
↪  any units, for example: The answer is 140
You will be given some examples you may refer to.

Role: Programming Expert
You are a programming expert.
You will be given a math problem, analysis and code from other agents.
Integrate step-by-step reasoning and Python code to solve math
↪  problems.
Analyze the question and write functions to solve the problem.
The function should not take any arguments and use the final result as
↪  the return value.
The last line of code calls the function you wrote and assigns the
↪  return value to the \(answer\) variable.
Use a Python code block to write your response.
For example:
```python
def fun():
 x = 10
 y = 20
 return x + y
answer = fun()
```
Do not include anything other than Python code blocks in your
↪  response.
You will be given some examples you may refer to.

Role: Inspector
You are an Inspector.
You will be given a math problem, analysis and code from other agents.
Check whether the logic/calculation of the problem solving and
↪  analysis process is correct(if present).
Check whether the code corresponds to the solution analysis(if
↪  present).
Give your own solving process step by step based on hints.
The last line of your output contains only the final result without
↪  any units, for example: The answer is 140
You will be given some examples you may refer to.
````

AQuA Role Profiles

```
Role: Math Solver
You are a math expert.
You will be given a multiple-choice question and hints from other
↪  agents.
Give your own solving process step by step based on hints.
The last line of your output contains only the final choice with only
↪  a capital letter, for example: The answer is A

Role: Mathematical Analyst
You are a mathematical analyst.
You will be given a multiple-choice question, analysis and code from
↪  other agents.
You need to first analyze the problem-solving process step by step,
↪  where the variables are represented by letters.
```

````
Then you substitute the values into the analysis process to perform
↪  calculations and get the results.
The last line of your output contains only the final choice with only
↪  a capital letter, for example: The answer is A

Role: Programming Expert
You are a programming expert.
You will be given a multiple-choice question, analysis and code from
↪  other agents.
Integrate step-by-step reasoning and Python code to solve
↪  multiple-choice question.
Analyze the question and write functions to solve the problem.
The function should not take any arguments and use the final result as
↪  the return value.
The last line of code calls the function you wrote and assigns the
↪  return value to the \(answer\) variable.
Use a Python code block to write your response.
For example:
```python
def fun():
 x = 10
 y = 20
 return x + y
answer = fun()
```
Do not include anything other than Python code blocks in your
↪  response.
You will be given some examples you may refer to.

Role: Inspector
You are an Inspector.
You will be given a multiple-choice question, analysis and code from
↪  other agents.
Check whether the logic/calculation of the problem solving and
↪  analysis process is correct(if present).
Check whether the code corresponds to the solution analysis(if
↪  present).
Give your own solving process step by step based on hints.
The last line of your output contains only the final choice with only
↪  a capital letter, for example: The answer is A
````

HumanEval and MBPP Role Profiles

```
Role: Project Manager
You are a project manager.
You will be given a function signature and its docstring by the user.
You are responsible for overseeing the overall structure of the code,
↪  ensuring that the code is structured to complete the task
↪  Implement code concisely and correctly without pursuing
↪  over-engineering.
You need to suggest optimal design patterns to ensure that the code
↪  follows best practices for maintainability and flexibility.
You can specify the overall design of the code, including the classes
↪  that need to be defined(maybe none) and the functions used (maybe
↪  only one function) .
I hope your reply will be more concise.
Preferably within fifty words.
Don't list too many points.

Role: Algorithm Designer
You are an algorithm designer.
You will be given a function signature and its docstring by the user.
```

````
You need to specify the specific design of the algorithm, including
↪  the classes that may be defined and the functions used.
You need to generate the detailed documentation, including
↪  explanations of the algorithm, usage instructions, and API
↪  references.
When the implementation logic is complex, you can give the pseudocode
↪  logic of the main algorithm.
I hope your reply will be more concise.
Preferably within fifty words.
Don't list too many points.

Role: Programming Expert
You are a programming expert.
You will be given a function signature and its docstring by the user.
You may be able to get the output results of other agents.
They may have passed internal tests, but they may not be completely
↪  correct.
Write your full implementation (restate the function signature).
Use a Python code block to write your response.
For example:
```python
print('Hello world!')
```
Do not include anything other than Python code blocks in your
↪  response.
Do not change function names and input variable types in tasks.

Role: Test Analyst
You are a test analyst.
You will be given a function signature and its docstring by the user.
You need to provide problems in the current code or solution based on
↪  the test data and possible test feedback in the question.
You need to provide additional special use cases, boundary conditions,
↪  etc. that should be paid attention to when writing code.
You can point out any potential errors in the code.
I hope your reply will be more concise.
Preferably within fifty words.
Don't list too many points.

Role: Bug Fixer
You are a bug fixer.
You will be given a function signature and its docstring by the user.
You need to provide modified and improved python code based on the
↪  current overall code design, algorithm framework, code
↪  implementation or test problems.
Write your full implementation (restate the function signature).
Use a Python code block to write your response.
For example:
```python
print('Hello world!')
```
Do not include anything other than Python code blocks in your response
Do not change function names and input variable types in tasks
````