TopoDIM: One-shot Topology Generation of Diverse Interaction Modes for Multi-Agent Systems

Rui Sun[1] **Jie Ding**[1] **Chenghua Gong**[1] **Tianjun Gu**[2]
Yihang Jiang[1] **Juyuan Zhang**[1] **Liming Pan**[1*] **Linyuan Lü**[1*]
[1]University of Science and Technology of China, Hefei, China
[2]East China Normal University, Shanghai, China
{rrsun,jieding25,gongchenghua,jiang_yihang,zhangjuyuan2020}@mail.ustc.edu.cn,
51275901043@stu.ecnu.edu.cn, {pan_liming,linyuan.lv}@ustc.edu.cn

Abstract

Optimizing communication topology in LLM–based multi-agent system is critical for enabling collective intelligence. Existing methods mainly rely on spatio-temporal interaction paradigms, where the sequential execution of multi-round dialogues incurs high latency and computation. Motivated by the recent insights that evaluation and debate mechanisms can improve problem-solving in multi-agent systems, we propose **TopoDIM**, a framework for one-shot **Topo**logy generation with **D**iverse **I**nteraction **M**odes. Designed for decentralized execution to enhance adaptability and privacy, TopoDIM enables agents to autonomously construct heterogeneous communication without iterative coordination, achieving token efficiency and improved task performance. Experiments demonstrate that TopoDIM reduces total token consumption by 46.41% while improving average performance by 1.50% over state-of-the-art methods. Moreover, the framework exhibits strong adaptability in organizing communication among heterogeneous agents. Code is available at: https://github.com/Sundiasy/TopoDIM.

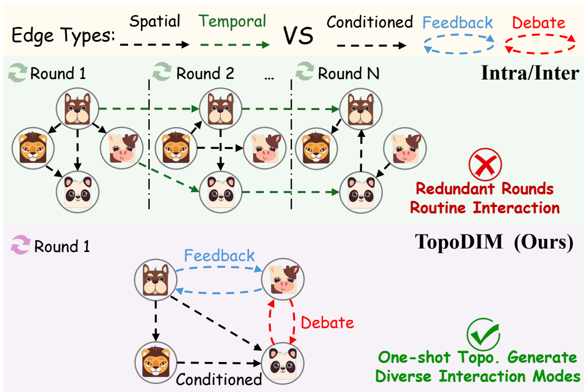


Figure 1: Illustration of hybrid intra/inter-round dialogue method versus TopoDIM. TopoDIM models complex interactions via one-shot topology generation, efficiently cutting potential token overhead.

1 Introduction

Large language model (LLM)–based multi-agent systems (MAS) exhibit collective intelligence that can surpass the capabilities of a single LLM, achieving strong performance in mathematical computation (Lei et al., 2024), code generation (Islam et al., 2024), software development (He et al., 2025), and scientific discovery (Ghareeb et al., 2025). Prior studies (Zhuge et al., 2024; Zhang et al., 2024b; Li et al., 2025) have demonstrated that carefully designed interaction topologies enhance the task-processing capability of MAS through effective collaboration.

Research on MAS with structured communication topologies has predominantly relied on dialogue-based interaction paradigms, as illustrated in Figure 1. In earlier frameworks (Wu et al., 2023; Du et al., 2023; Zheng et al., 2023; Liu et al., 2024), agent connections are homogeneous and restricted to a single interaction mode, with information flow governed by static communication graphs lacking dynamic adaptation. While recent studies (Zhuge et al., 2024; Zhang et al., 2024b) have achieved performance gains by hybridizing intra- and inter-round dialogues within spatio-temporal graphs, this multi-round design inherently imposes additional computational and token costs. Although mitigation strategies such as edge pruning (Zhang et al., 2024a), dynamic agent selection (Wang et al., 2025a,b; Li et al., 2025), and proxy-based optimization (Jiang et al., 2025) attempt to balance efficiency and performance, structural redundancy remains unavoidable. Specifically, each inter-round interaction triggers a subsequent intra-round dialogue phase, recursively compounding communication overhead (Chen et al., 2025; Zeng et al., 2025).

Single-round dialogue schemes offer the advantage of reduced communication overhead; however,

* Corresponding Author.

Findings of the Association for Computational Linguistics: ACL 2026, pages 4252–4269
July 2-7, 2026 ©2026 Association for Computational Linguistics

maintaining high task performance within such a constrained framework remains a challenge (Hu et al., 2026c). Drawing on recent insights that evaluation and debate mechanisms can improve problem-solving in multi-agent systems (Xue et al., 2025; Zhou et al., 2025), we posit that explicitly modeling diverse interactions through a heterogeneous communication graph yields a balance between efficiency and performance. Thus, we propose TopoDIM (One-shot **Topo**logy Generation with **D**iverse **I**nteraction **M**odes). Specifically, TopoDIM is characterized by the following key features:

❶ **Diverse Interaction Modes**. TopoDIM leverages three efficient collaborative argumentation primitives (Scardamalia and Bereiter, 2006), enabling agents to engage in complementary forms of information exchange beyond homogeneous message passing. ❷ **One-shot Heterogeneous Topology Generation.** TopoDIM employs a heterogeneous graph encoder to capture agent- and task-specific contexts, coupled with an autoregressive decoder that generates a communication topology with multiple interaction modes in a single inference step, eliminating iterative dialogue rounds. ❸ **Decentralized Architecture.** To facilitate adaptive decision-making and mitigate privacy risks (Yang et al., 2025), TopoDIM distills the global topology optimizer into lightweight local networks deployed on individual agents, enabling autonomous and fully decentralized topology construction.

Compared to existing approaches, TopoDIM substantially reducing computational overhead while promoting more diverse collaboration patterns, ultimately leading to improved task-solving performance in multi-agent systems. Our main contributions are summarized as follows:

❶ ***Observation.*** We suggest that replacing iterative topology generation in both intra- and inter-round dialogue processes with a one-shot heterogeneous topology formulation significantly improves communication efficiency without performance trade-offs.

❷ ***Framework.*** We propose TopoDIM, a decentralized framework that integrates a heterogeneous graph encoder with an autoregressive decoder to generate multi-relational communication topologies in one shot, enabling autonomous agent-level decision-making.

❸ ***Evaluation.*** Extensive experiments demonstrate that TopoDIM consistently enhances communication efficiency, task performance, and structural robustness across both homogeneous and heterogeneous multi-agent settings, surpassing strong task-adaptive cooperation baselines.

2 Methodology Overview

TopoDIM dynamically orchestrates conditioned, feedback, and dialectical behaviors within a single execution to leverage the complementary strengths of LLM agents and heterogeneous interactions. The schematic overview of TopoDIM is presented in Figure 2.

2.1 Communication Topology and Pipeline

Communication topology. We model LLM-based MAS as a directed heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, r)$. The node set $\mathcal{V} = \{v_1, \ldots, v_N\}$ represents the collection of agents. As illustrated in Figure 2, each agent v_i is instantiated by a base language model LM_i, a role Role_i (e.g., math-solver), a context $\mathrm{Context}_i$ which includes the knowledge and dialogues history, and a set of external tools Tool_i (e.g., file-reader). The edge set $\mathcal{E}$ denotes agent interactions, where $r : \mathcal{E} \to \mathcal{R}$ serves as an edge-type mapping function. Specifically, a directed edge $e_{ij} \in \mathcal{E}$ signifies an interaction from v_i to v_j governed by a protocol $r_{ij} := r(e_{ij}) \in \mathcal{R}$. The definition of $\mathcal{R}$ will be discussed in Section 3.1.

Communication pipeline. Given a query q, a scheduling function ψ maps $\mathcal{G}$ to an execution sequence $\psi \to \sigma = \langle v_{\sigma(1)}, \ldots, v_{\sigma(N)} \rangle$. When activated, each agent v_i processes a prompt $\mathcal{P}$ to produce a solution $\mathcal{S}_i = v_i(\mathcal{P})$. The final output a is obtained by aggregating all agents' solutions: $a \leftarrow \mathrm{Aggregate}(\{\mathcal{S}_i\}_{i=1}^N)$. As a comparision, hybrid intra/inter-round dialogue paradigms framework rely on a T-round iterative simulation yielding sequential outputs $\{a^{(t)}\}_{t=1}^T$, which imposes a potential overhead on execution efficiency.

2.2 Optimization and Decentralized Design

Centralized optimization. We employ reinforcement learning to optimize the interaction graph. Specifically, we formulate this task as learning a stochastic policy π_θ that, given a task query q, seeks to construct an optimal interaction graph $\mathcal{G}^* = (\mathcal{V}^*, \mathcal{E}^*, r^*)$. The objective of policy π_θ is to maximize the expected reward over the distribution of possible graphs, guided by a task-related reward function $R(\cdot)$ derived from the quality of the final

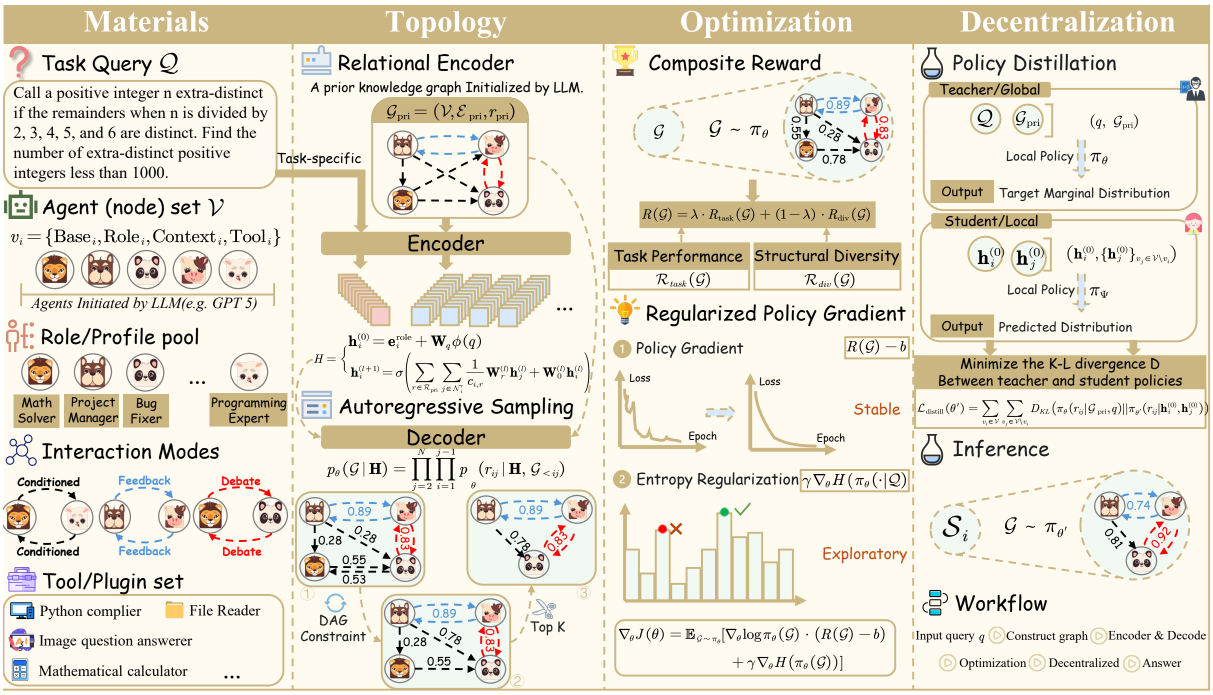


Figure 2: The framework of TopoDIM, comprising 4 components: ❶ Materials: defining agents, roles, interaction modes, and plugins; ❷ Topology: illustrating the heterogeneous topology design; ❸ Optimization: detailing topology optimization strategies, and ❹ Decentralization: describing the decentralized agent decision-making.

answer:

$$\mathcal{G}^* = \underset{\mathcal{G}}{\text{argmax}} \left[\mathbb{E}_{\mathcal{G} \sim \pi_\theta(\cdot|q)} [R(\mathcal{G})] \right]. \tag{1}$$

Decentralized design. To support adaptive decision-making while addressing potential privacy risks, we adopt a decentralized architecture where each agent $v_i \in \mathcal{V}$ maintains an independent local policy network $\pi_{\theta'}^{(i)}$ to infer its connectivity with other nodes. Conditioned on the task embedding $\mathbf{z}$ and the local states $\mathbf{h}_i$ and $\mathbf{h}_j$, the communication link between agent v_i and v_j is formulated as:

$$p_{ij}^{(r)} = \pi_{\theta'}^{(i)}(r_{ij} | \mathbf{h}_i, \mathbf{h}_j, \mathbf{z}). \tag{2}$$

3 Architecture Details

3.1 Diverse Interaction Modes

Recent studies indicate that diverse collaborative mechanisms, such as evaluation and debate, can significantly bolster the complex problem-solving capabilities of MAS (Xue et al., 2025; Zhou et al., 2025). Motivated by these findings, we formalize the interaction space $\mathcal{R}$ using three distinct edge types, as shown in Figure 3. Each edge type is instantiated via specific prompts to govern its interaction logic.

❶ **Conditioned edges.** The classical edge type where agent v_j handles the query q conditioned with the outputs of agent v_i.

❷ **Feedback edges.** Encapsulate a supervisory mechanism where agent v_j critiques or validates the intermediate outputs generated by agent v_i and v_i re-handle the query q referring to the feedback of v_j, simulating an evaluation and reflection process.

❸ **Debate edges.** Model a debate process wherein agent v_j challenges v_i's proposition for two rounds, ultimately v_j proceed the query q with the context of the debate.

Given a query q, TopoDIM aims to learn a policy π_θ that determines the optimal relation $r_{ij}^* \in \mathcal{R}' := \mathcal{R} \cup \{\emptyset\}$ between the agent pair (v_i, v_j), where $\emptyset$ signifies the non-existence of an edge.

$$r_{ij}^* = \underset{r \in \mathcal{R}'}{\arg\max}\, \pi_\theta (r | v_i, v_j, q). \tag{3}$$

3.2 Heterogeneous Interaction Topology

We formulate the generation of heterogeneous topologies as a conditional autoregressive decision process. The learnable policy π_θ defines a distribution $p_\theta(\mathcal{G}|q)$ over feasible communication graphs.

Semantics-aware relational encoder. To establish a robust latent representation for each agent, we employ a relational graph convolutional network (Schlichtkrull et al., 2018; Wang et al., 2024a) over a prior knowledge graph $\mathcal{G}_{\text{pri}} = (\mathcal{V}, \mathcal{E}_{\text{pri}}, r_{\text{pri}})$, which is initialized by an advanced LLM (e.g., GPT-5). The node update rule at layer $l + 1$ is defined as $\mathbf{h}_i^{\ell+1} = \sigma(\hat{\mathbf{h}}_i^{\ell})$, where

$$\hat{\mathbf{h}}_i^{\ell} = \sum_{r\in\mathcal{R}_{\text{pri}}} \sum_{j\in\mathcal{N}_i^r} \frac{1}{c_{i,r}} \mathbf{W}_r^{(l)}\mathbf{h}_j^{(l)} + \mathbf{W}_0^{(l)}\mathbf{h}_i^{(l)}, \quad (4)$$

$\mathcal{N}_i^r$ denotes the set of neighbors of node v_i under relation r, and $c_{i,r} = |\mathcal{N}_i^r|$. The initial node embeddings combine role and task information:

$$\mathbf{h}_i^{(0)} = \mathbf{e}_i^{\text{role}} + \mathbf{W}_q\phi(q), \quad (5)$$

where $\mathbf{e}_i^{\text{role}}$ is a learnable role embedding and $\phi(q)$ is the feature representation of the task query $q \in Q$ obtained from a pre-trained sentence encoder. The final set of node embeddings from the encoder, $\{\mathbf{h}_1, \ldots, \mathbf{h}_N\}$, constitutes the conditional representation $\mathbf{H}$.

Autoregressive edge sampling decoder. Given the conditional representation $\mathbf{H}$, a decoder generates topology $\mathcal{G}$ sequentially. With a prespecified ordering of nodes, the joint probability of observing a graph $\mathcal{G}$ reads:

$$\begin{aligned} p_\theta(\mathcal{G} \mid \mathbf{H}) &= \prod_{j=2}^{N}\prod_{i=1}^{j-1} p_\theta(r_{ij} \mid \mathbf{H}, \mathcal{G}_{<ij}) \\ &= \prod_{j=2}^{N}\prod_{i=1}^{j-1} \frac{\exp\left(\mathbf{w}_{r_{ij}}^\top[\mathbf{h}_i \| \mathbf{h}_j]/\tau\right)}{\sum_{r\in\mathcal{R}'}\exp\left(\mathbf{w}_r^\top[\mathbf{h}_i\|\mathbf{h}_j]/\tau\right)}, \end{aligned} \quad (6)$$

where $\|$ denotes concatenation, and τ is the temperature. We sample the relation type r_{ij} using inverse transform sampling by generating $u \sim \text{Uniform}(0,1)$ and setting $F(r) = \sum_{r\in\mathcal{R}'} p_\theta(r)$, where $r_{ij} = \min\{r : F(r) \geq u\}$ represents the cumulative probability.

Structural constraints. To ensure logical consistency and prevent circular dependencies, we enforce an acyclic constraint via a dynamic mask $\mathbf{M}$. Specifically, if the presence of edge (v_i, v_j) violates the acyclic constraint in any existing path from v_i to v_j, the probability of this node pair is masked as:

$$p(r_{ij} \mid \cdot) \leftarrow p(r_{ij} \mid \cdot) \odot \mathbf{M}_{ij}, \quad (7)$$

where $\mathbf{M}_{ij} = \mathbb{I}(v_i \not\leadsto v_j)$ indicates if v_i and v_j are not path-connected. Detailed discussions on structural constraint is shown in Appendix B.

Adaptive sparsification. Inspired by connectivity sparsification methods (Zhang et al., 2024a; Wang et al., 2025b) that enhance task performance through structural reduction, we employ an adaptive pruning mechanism to retain only salient interactions. Specifically, we identify the subset of edges $\mathcal{E}_{\text{final}}$ based on confidence scores, subject to a sparsity budget ratio α:

$$\mathcal{E}_{\text{final}} = \text{TopK}\left(\{p(r_{ij})\}_{\forall i,j}, 1-\alpha\right). \quad (8)$$

To identify the most relevant contributors for each task, TopoDIM filters out inactive nodes that lack connectivity within the selected edge subset. The final node set $\mathcal{V}_{\text{final}}$ is derived by retaining only those vertices involved in at least one valid interaction:

$$\mathcal{V}_{\text{final}} = \{v \mid \exists(h,t,r) \in \mathcal{E}_{\text{final}}, v \in \{h,t\}\}. \quad (9)$$

Execution order. Upon establishing the final set of agents $\mathcal{V}_{\text{final}}$ and edges $\mathcal{E}_{\text{final}}$, the interaction process follows a breadth-first manner. Starting from the root node (characterized by zero in-degree), each agent interacts with its neighbors in a sequence determined by ascending node indices.

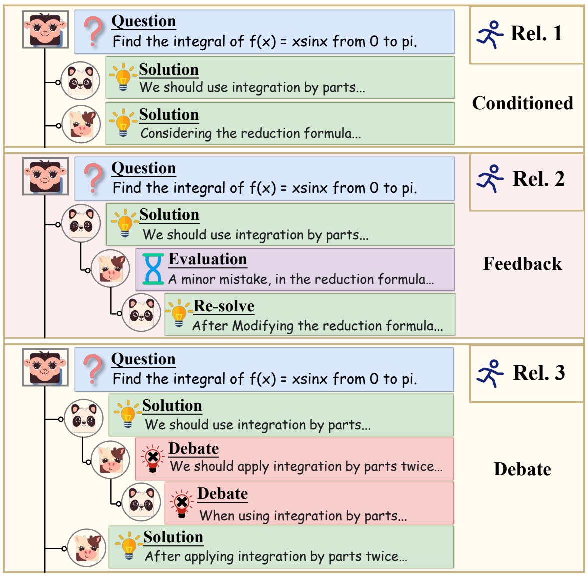


Figure 3: Interaction modes of TopoDIM. TopoDIM selects three effective interaction modes aiming to optimize the execution sequence among the agents.

3.3 Diversity-Aware Optimization Strategy

We optimize a graph generation policy π_θ via reinforcement learning, aiming to maximize a composite objective that balances task success with structural diversity. For each generated graph $\mathcal{G} \sim \pi_\theta$, the agent ensembles execute the collaborative task to yield a binary performance metric $R_{\text{task}}(\mathcal{G}) \in \{0,1\}$, where 1 indicates a successful solution and 0 indicates failure. To mitigate mode collapse and prevent the policy from converging to a narrow set of interaction patterns, we introduce a structural balance reward (Ma et al., 2026b; Liu et al., 2026). This term is formulated as the Shannon entropy of the empirical edge-type distribution $p(r) = |\mathcal{E}_r|/|\mathcal{E}|$, where $\mathcal{E}_r$ denotes the set of edges with relation type r:

$$R_{\text{div}}(\mathcal{G}) = -\sum_{r\in\mathcal{R}} p(r)\log p(r). \tag{10}$$

The holistic reward signal $R(\mathcal{G})$ is a weighted interpolation of task performance and structural entropy:

$$R(\mathcal{G}) = \lambda \cdot R_{\text{task}}(\mathcal{G}) + (1-\lambda)\cdot R_{\text{div}}(\mathcal{G}), \tag{11}$$

where $\lambda \in [0,1]$ is a hyperparameter governing the trade-off between performance exploitation and diversity exploration.

Following the policy gradient theorem (Sutton et al., 1999), we maximize the expected objective $J(\theta) = \mathbb{E}_{\mathcal{G}\sim\pi_\theta}[R(\mathcal{G})]$. To stabilize training and reduce variance, we employ a moving average baseline b (Williams, 1992; Chen et al., 2024). The overall loss function also includes an entropy regularization term for the policy's output distribution to encourage exploration:

$$\begin{aligned}\nabla_\theta J(\theta) =& \mathbb{E}_{\mathcal{G}\sim\pi_\theta}\Big[\nabla_\theta \log \pi_\theta(\mathcal{G})\cdot(R(\mathcal{G})-b) \\ &+ \gamma\nabla_\theta H(\pi_\theta(\mathcal{G}))\Big],\end{aligned} \tag{12}$$

where $H(\pi_\theta)$ denotes the entropy of the policy distribution and γ is the regularization coefficient.

3.4 Decentralized Architecture Design

To facilitate scalable and autonomous deployment, each agent v_i employs a light local policy (e.g., MLP) $\pi^{(i)}_{\theta'}$ to predict a relation type with all other node $\sum_{j\neq i} v_j$, taking each ordered pair of representations$(\mathbf{h}^{(0)}_i, \mathbf{h}^{(0)}_j)$. We align local policy with the global policy π_θ via policy distillation, minimizing the Kullback-Leibler (KL) divergence D_{KL} to capture global structural dependencies:

$$\begin{aligned}\mathcal{L}_{\text{distill}}(\theta') = &\sum_{v_i\in\mathcal{V}}\sum_{v_j\in\mathcal{V}\setminus v_i} D_{KL}\big(\\ &\pi_\theta(r_{ij}|\mathcal{G}_{\text{pri}}, q)||\pi_{\theta'}(r_{ij}|\mathbf{h}^{(0)}_i, \mathbf{h}^{(0)}_j)\big),\end{aligned} \tag{13}$$

where $\pi_\theta(\cdot|\cdot)$ is the centralized policy induced marginal distribution. During inference, agents autonomously sample connections from $\pi^{(i)}_{\theta'}$ following the structural constraints, thereby combining centralized training optimality with decentralized execution efficiency.

4 Experiment

4.1 Experimental Setups

Models and benchmarks. To evaluate the general adaptability of TopoDIM to different LLMs, we employ a diverse set of models including `Gemma-3-it:12B` (Team et al., 2025), `GPT-OSS-20B` (Agarwal et al., 2025), `GPT-OSS-120B` (Agarwal et al., 2025), and `DeepSeek-V3.2-251201:671B` (Liu et al., 2025). We test the performance of TopoDIM on three types of reasoning datasets: ❶ General Reasoning: MMLU-Pro (Wang et al., 2024b); ❷ Mathematics: MultiArith (Roy and Roth, 2016), GSM8K (Cobbe et al., 2021) , AIME$_{(2023-2025)}$; ❸ Coding: HumanEval (Hendrycks et al., 2020); LiveCodeBench$_{(202305-202403)}$(Jain et al., 2024).

Baselines. We evaluate TopoDIM by comparing its task-solving performance against 6 typical methods, including ❶ Single-agent prompting methods: Vanilla, Chain-of-Thought (Wei et al., 2022) which augment few-shot exemplars with intermediate reasoning steps. ❷ Multi-agent topology methods: LLM-Debate (Du et al., 2023), GPTSwarm (Zhuge et al., 2024), G-Designer (Zhang et al., 2024b), AgentDropout (Wang et al., 2025b).

Implementations. In our experiments, `Gemma-3` and `GPT-OSS` are deployed locally on an NVIDIA RTX 4090 via Ollama (Ollama, 2023), while `DeepSeek-V3.2` is accessed through its official APIs. We implement a temperature annealing schedule decaying from 2.0 to 0.5 and maintain a fixed learning rate of 0.01 for both centralized and decentralized training. Specific parameters include Top-K sparsity budget ratio $\alpha \in \{0.3, 0.5, 0.7\}$, $\lambda \in \{0.3, 0.5, 0.8\}$, and $\eta \in \{0.01, 0.03, 0.05\}$. The feature extraction function $\phi(\cdot)$ is instantiated with `all-MiniLM-L6-v2` (Wang et al., 2020), yielding 384-dimensional embeddings. Each experimental run is initialized with 5 agents. For all benchmarks, the query budgets for optimization (M) and and decentralized knowledge distillation (M') are selected from $\{40, 80\}$. Furthermore, we employ `GPT-5`(OpenAI, 2025) to generate prompt descriptions for three distinct interaction types, and the local policy is parameterized by an MLP. All remaining configurations and agent settings follow the protocols established in (Zhang et al., 2024b).

4.2 Main Results

Task performance. Table 1 demonstrates that TopoDIM outperforms various base-

Table 1: Performance comparison with three different base LLMs including `Gemma-3-it:12B`, `GPT-OSS-120B`, and `DeepSeek-V3.2`. **Ada.**, **DIM.**, and **Dec.** indicates whether it is task-adaptive, whether it supports diverse interaction modes, and whether it is decentralized architecture respectively. The **bold** and $\underline{\text{underlined}}$ fonts show the best and the second best result. ✗, (✓), and ✓ signifies no/partial/full support in these aspects.

Method	Ada.	DIM.	Dec.	LiveCodeBench	MMLU-Pro	GSM8K	MultiArith	AIME	HumanEval	Avg.
Base model: Gemma-3-it:12B										
Vanilla	✗	✗	✗	24.46	51.24	87.37	96.68	11.28	82.23	58.88
CoT	✗	✗	✗	24.90$_{\uparrow 0.44}$	51.95$_{\uparrow 0.71}$	87.83$_{\uparrow 0.46}$	97.01$_{\uparrow 0.33}$	12.14$_{\uparrow 0.86}$	82.85$_{\uparrow 0.62}$	59.45$_{\uparrow 0.57}$
LLM-Debate	✗	✗	✗	25.47$_{\uparrow 1.01}$	52.66$_{\uparrow 1.42}$	88.46$_{\uparrow 1.09}$	97.62$_{\uparrow 0.94}$	11.88$_{\uparrow 0.60}$	83.85$_{\uparrow 1.62}$	59.99$_{\uparrow 1.11}$
GPTSwarm	(✓)	✗	✗	26.22$_{\uparrow 1.76}$	53.14$_{\uparrow 1.90}$	89.38$_{\uparrow 2.01}$	97.85$_{\uparrow 1.17}$	12.50$_{\uparrow 1.22}$	84.14$_{\uparrow 1.91}$	60.54$_{\uparrow 1.66}$
AgentDropout	✓	(✓)	✗	26.75$_{\uparrow 2.29}$	53.79$_{\uparrow 2.55}$	90.14$_{\uparrow 2.77}$	98.20$_{\uparrow 1.52}$	$\underline{13.24}_{\uparrow 1.96}$	84.73$_{\uparrow 2.50}$	61.14$_{\uparrow 2.27}$
G-Designer	✓	(✓)	✗	$\underline{27.08}_{\uparrow 2.62}$	$\underline{54.25}_{\uparrow 3.01}$	$\underline{90.27}_{\uparrow 2.90}$	$\underline{98.45}_{\uparrow 1.77}$	12.86$_{\uparrow 1.58}$	$\underline{85.47}_{\uparrow 3.24}$	$\underline{61.40}_{\uparrow 2.52}$
TopoDIM	✓	✓	✓	**27.38**$_{\uparrow 2.92}$	**55.08**$_{\uparrow 3.84}$	**91.86**$_{\uparrow 4.49}$	**98.85**$_{\uparrow 2.17}$	**13.58**$_{\uparrow 2.30}$	**86.62**$_{\uparrow 4.39}$	**62.23**$_{\uparrow 3.35}$
Base model: GPT-OSS:120B										
Vanilla	✗	✗	✗	81.58	74.49	95.90	100	74.77	90.04	86.13
CoT	✗	✗	✗							
LLM-Debate	✗	✗	✗	82.92$_{\uparrow 1.34}$	75.53$_{\uparrow 1.04}$	95.72$_{\downarrow 0.18}$	100$_{\uparrow 0.00}$	75.38$_{\uparrow 0.61}$	91.43$_{\uparrow 1.39}$	86.83$_{\uparrow 0.70}$
GPTSwarm	(✓)	✗	✗	83.59$_{\uparrow 2.01}$	76.88$_{\uparrow 2.39}$	96.50$_{\uparrow 0.60}$	100$_{\uparrow 0.00}$	77.41$_{\uparrow 2.64}$	92.30$_{\uparrow 2.26}$	87.78$_{\uparrow 1.65}$
AgentDropout	✓	(✓)	✗	85.06$_{\uparrow 3.48}$	78.62$_{\uparrow 4.13}$	96.86$_{\uparrow 0.96}$	100$_{\uparrow 0.00}$	78.13$_{\uparrow 3.36}$	92.88$_{\uparrow 2.84}$	88.59$_{\uparrow 2.46}$
G-Designer	✓	(✓)	✗	$\underline{85.63}_{\uparrow 4.05}$	$\underline{79.08}_{\uparrow 4.59}$	$\underline{97.08}_{\uparrow 1.18}$	$\underline{100}_{\uparrow 0.00}$	$\underline{78.62}_{\uparrow 3.85}$	$\underline{93.47}_{\uparrow 3.43}$	$\underline{88.98}_{\uparrow 2.85}$
TopoDIM	✓	✓	✓	**87.28**$_{\uparrow 5.70}$	**80.11**$_{\uparrow 5.62}$	**98.34**$_{\uparrow 2.44}$	**100**$_{\uparrow 0.00}$	**80.34**$_{\uparrow 5.57}$	**95.83**$_{\uparrow 5.79}$	**90.32**$_{\uparrow 4.19}$
Base model: DeepSeek-V3.2-251201:671B										
Vanilla	✗	✗	✗	78.35	78.68	96.31	100	64.37	89.46	83.53
CoT	✗	✗	✗	78.84$_{\uparrow 0.49}$	78.92$_{\uparrow 0.24}$	96.58$_{\uparrow 0.27}$	100$_{\uparrow 0.00}$	64.08$_{\downarrow 0.29}$	89.84$_{\uparrow 0.38}$	84.71$_{\uparrow 0.18}$
LLM-Debate	✗	✗	✗	79.66$_{\uparrow 1.31}$	80.14$_{\uparrow 1.46}$	96.82$_{\uparrow 0.51}$	100$_{\uparrow 0.00}$	64.84$_{\uparrow 0.47}$	90.67$_{\uparrow 1.21}$	85.36$_{\uparrow 0.83}$
GPTSwarm	(✓)	✗	✗	80.40$_{\uparrow 2.05}$	81.75$_{\uparrow 3.07}$	97.14$_{\uparrow 0.83}$	100$_{\uparrow 0.00}$	67.20$_{\uparrow 2.83}$	92.32$_{\uparrow 2.86}$	86.47$_{\uparrow 1.94}$
AgentDropout	✓	(✓)	✗	81.19$_{\uparrow 2.84}$	82.37$_{\uparrow 3.69}$	$\underline{97.76}_{\uparrow 1.45}$	100$_{\uparrow 0.00}$	68.17$_{\uparrow 3.80}$	93.15$_{\uparrow 3.69}$	87.11$_{\uparrow 2.58}$
G-Designer	✓	(✓)	✗	$\underline{81.63}_{\uparrow 3.28}$	$\underline{82.83}_{\uparrow 4.15}$	97.48$_{\uparrow 1.17}$	$\underline{100}_{\uparrow 0.00}$	$\underline{68.62}_{\uparrow 4.25}$	$\underline{93.54}_{\uparrow 4.08}$	$\underline{87.35}_{\uparrow 2.82}$
TopoDIM	✓	✓	✓	**83.26**$_{\uparrow 4.91}$	**84.80**$_{\uparrow 6.12}$	**98.52**$_{\uparrow 2.21}$	**100**$_{\uparrow 0.00}$	**69.93**$_{\uparrow 5.56}$	**94.86**$_{\uparrow 5.20}$	**88.53**$_{\uparrow 4.03}$

lines across multiple datasets, including single models and state-of-the-art multi-agent systems. Specifically, with `Gemma-3-it:12B` and `DeepSeek-V3.2-251201`, TopoDIM yields average performance gains of 1.35% ↑ and 1.38% ↑ over the strongest baselines, respectively. Notably, TopoDIM also enhances the capabilities of the reasoning model (`GPT-OSS:120B`) on complex tasks, resulting in a 1.50% ↑ increase in average predictive accuracy.

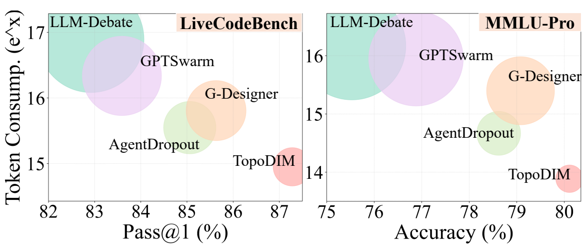


Figure 4: Performance-cost trade-off between TopoDIM and state-of-the-art methods. The bubble size is proportional to token consumption.

Computation efficiency. We compare the token consumption of TopoDIM against other baselines on MMLU-Pro and LiveCodeBench using `GPT-OSS:120B`, with the results illustrated in Figure 4. TopoDIM significantly reduces token expenditure, saving a total of 1.42M and 2.44M tokens on MMLU-Pro and LiveCodeBench, respectively, while maintaining competitive performance. A comprehensive efficiency analysis is provided in Appendix D.

Heterogeneous agents adaptability. To validate the adaptability of TopoDIM within a heterogeneous MAS, we construct a collaborative framework comprising agents with varying capabilities: three `GPT-OSS-20B` agents and two `GPT-OSS-120B` agents, with the latter serves as the final decision-maker. As shown in Table 2, the results demonstrate that TopoDIM achieves SOTA performance across all datasets, yielding an average improvement of 1.86% ↑ over existing homogeneous topology frameworks. TopoDIM leverages its adaptive sparsification design to effectively prune agents that make negligible contributions, thereby optimizing communication efficiency. A more comprehensive evaluation is provided in Appendix E.

4.3 Framework Analysis

We conduct comprehensive experiments using `GPT-OSS:120B` to validate the framework design

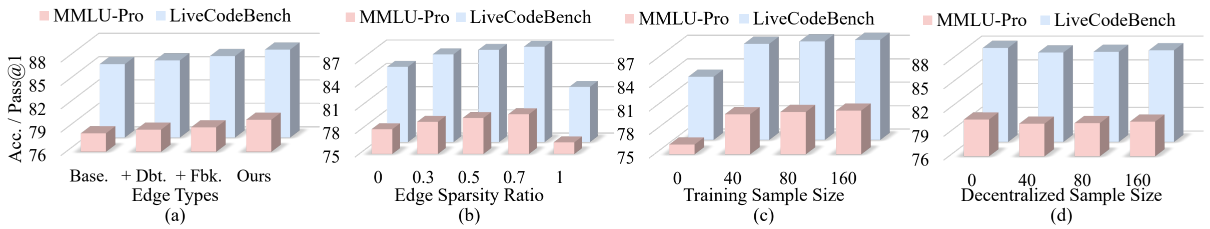


Figure 5: Dependence on hyperparameter and architecture designs. (a) Edge diversity vs. accuracy/pass@1, (b) edge sparsity vs. accuracy/pass@1, (c) training sample size vs. accuracy/pass@1, and (d) decentralized sample size vs. accuracy/pass@1.

of our proposed TopoDIM.

Edge diversity. Figure 5(a) demonstrates the impact of edge diversity on the collaborative problem-solving capabilities of TopoDIM. Specifically, `Base.` denotes the configuration with only conditioned edges, while `+ Fbk./Dbt.` corresponds to the incremental addition of feedback and debate edge types. Our systematic investigation reveals a positive correlation between edge diversity and performance. Notably, integrating all available edge types improves the prediction accuracy/pass@1 by 2.21%↑ on MMLU-Pro and 2.18%↑ on LiveCodeBench, respectively.

Edge sparsity. The amount of edges, controlled by the TopK sparsity budget ratio $\alpha \in \{0, 0.3, 0.5, 0.7, 1.0\}$, significantly impacts performance, as depicted in Figure 5(b). The optimal accuracy/pass@1 (80.11% on MMLU-Pro and 87.28% on LiveCodeBench) is achieved at $\alpha = 0.7$, which reveals a crucial trade-off: a dense graph risks in creating redundant communication channels, leading to model hallucination and ultimately impairing decision-making performance.

Training sample size. We evaluate the data efficiency of TopoDIM by varying the training sample size M (Figure 5(c)). The results demonstrate a monotonic improvement in performance as the number of samples increases, with the most significant gains occurring within the first 40 samples.

Decentralized sample size. To evaluate the effectiveness of the distillation process, we vary the number of decentralized samples M'(from 0 to 160, where 0 indicates the direct use of the centralized policy to generate topology). As illustrated in Figure 5(d), TopoDIM achieves robust performance with as few as 40 samples, demonstrating the efficiency of the knowledge transfer.

Ablation studies. To analyze TopoDIM's architecture and validate the necessity of its core components, we perform a series of ablation studies. First, to evaluate our **`heterogeneous interactions topology design`**, we consider two configurations: ❶ ***w/ Rand***, which features randomly generated heterogeneous edges, and ❷ ***w/o Graph***, which replaces $\mathcal{G}_{\text{pri}}$ with a fully connected graph (using random edge types). These modifications result in average performance drops of 5.60%↓ and 1.99%↓, respectively, underscoring the significance of the graph prior and our topology design. Second, to evaluate the **`topology optimization strategy`**, we examine the contributions of the baseline and entropy regularization in optimization phases. Removing the baseline b: ❸ ***w/o Baseline***, which enhances training stability, leads to an accuracy decrease of 0.89%↓. Similarly, removing the entropy regularization term: ❹ ***w/o ER***, which facilitates policy exploration, causes accuracy to fall by 1.38% ↓. These studies demonstrate the critical role of each component in TopoDIM.

Table 2: Performance comparison of heterogeneous agents distinguished by skill in MAS.

Method	LiveCodeBench	MMLU-Pro	AIME
Mode: Heterogeneous Skilled LLMs			
Vanilla	81.58	74.49	74.77
AgentDropout	$82.92_{\uparrow 1.34}$	$77.35_{\uparrow 2.86}$	$76.71_{\uparrow 1.94}$
G-Designer	$83.86_{\uparrow 2.28}$	$78.18_{\uparrow 3.69}$	$77.10_{\uparrow 2.32}$
TopoDIM	**84.52**$_{\uparrow 2.94}$	**79.52**$_{\uparrow 5.03}$	**79.39**$_{\uparrow 4.62}$

Table 3: Ablation study on the impact of topology and optimization strategy.

Method	LiveCodeBench	MMLU-Pro	AIME
TopoDIM	87.28	80.11	80.34
w/ Rand	$82.52_{\downarrow 4.76}$	$76.36_{\downarrow 3.75}$	$75.71_{\downarrow 4.63}$
w/o Graph	$85.49_{\downarrow 1.79}$	$78.62_{\downarrow 1.49}$	$78.77_{\downarrow 1.57}$
w/o Baseline	$86.57_{\downarrow 0.71}$	$79.45_{\downarrow 0.66}$	$79.53_{\downarrow 0.81}$
w/o ER	$86.42_{\downarrow 0.86}$	$78.97_{\downarrow 1.14}$	$78.95_{\downarrow 1.39}$

4.4 Case study

We conduct a case study on LiveCodeBench using `GPT-OSS-120B` to evaluate the performance and op-

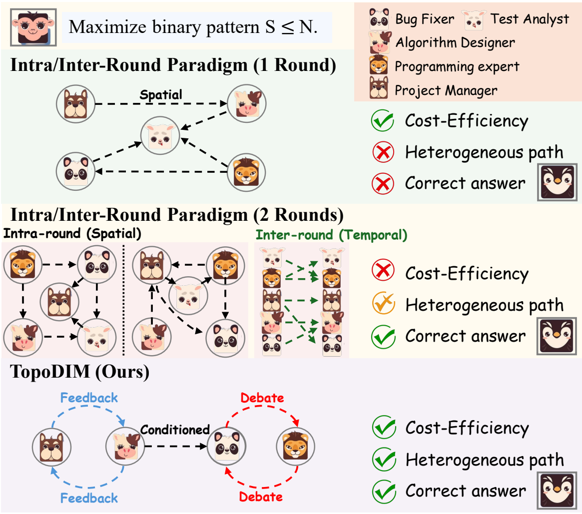


Figure 6: TOPODIM vs. Intra/inter-round dialogues methods on LiveCodeBench. Collaboration with diversity modes synthesizes various perspectives to ensure robust solutions with cost-efficiency.

timized communication structures of TOPODIM against existing methods. Figure 6 visualizes the topologies generated by different approaches for a representative query. The top panel depicts a typical intra-round dialogue graph. When handling complex tasks, such methods are often prone to information loss, leading to suboptimal performance in ambiguous scenarios. The middle panel illustrates an intra/inter-round paradigm that facilitates agent collaboration through multi-turn dialogues. However, this approach incurs additional token overhead and increases susceptibility to hallucination. In contrast, TOPODIM (bottom panel) dynamically constructs a sparse yet informative topology. By selectively activating conditional, feedback, and debate interaction patterns, it effectively preserves critical context without unnecessary token consumption, thereby successfully resolving complex reasoning challenges. Detailed dialogues are provided in Appendix G.

5 Related Work

5.1 LLM-based MAS

Multi-agent systems(MAS) catalyze a paradigm shift from single-agent strategies (Yao et al., 2022; Zhang et al., 2026b) to collective intelligence for complex problems (Gong and Li, 2025; Ma et al., 2026a; Long et al., 2026). Pioneering frameworks demonstrated the efficacy of communicative role-playing and social simulation (Li et al., 2023; Hu et al., 2026b), while task-oriented architectures have formalized collaborative workflows in graph generation (Zhang et al., 2026a), software engineering (Hong et al., 2023), and logical reasoning (Chen et al., 2023) by integrating dynamic role assignment (Ye et al., 2025), iterative debate (Liang et al., 2023; Zeng et al., 2025; Choi et al., 2025), and shared memory (Wang and Chen, 2025; Zhang et al., 2025). Additionally, recent work has introduced benchmarks for evaluating MAS (Zhang et al., 2026c). Recent advancements have corroborated decentralized architectures (Yang et al., 2025) and diversified interaction patterns (Xue et al., 2025) to improve task proficiency and adaptability. TOPODIM orchestrates decentralized structure and diverse interactions, demonstrating superior performance.

5.2 Topology Design of MAS

Communication topologies in LLM-based MAS are generally categorized into intra-round, inter-round, and hybrid structures (Zhang et al., 2024a). Early research primarily organizes either intra- or inter-round dialogues, including chain (Qian et al., 2024a; Holt et al., 2025), tree (Wu et al., 2023), layered (Du et al., 2023; Qian et al., 2024b), and filtered graphs (Zheng et al., 2023; Zhuge et al., 2024). Recent advancements focus on generating task-specific adaptive topologies by integrating both intra- and inter-round interactions(Wang et al., 2025a; Zhou et al., 2025; Li et al., 2025; Jiang et al., 2025; Wu et al., 2026). For instance, G-Designer (Zhang et al., 2024b) employs variational autoencoders to generate adaptive graph structures. While recent methods attempt to mitigate these costs via edge pruning (Zhang et al., 2024a) and dropout (Wang et al., 2025b), TOPODIM eliminates redundant dialogue rounds while maintaining diverse interactions, making a promising performance.

6 Conclusion

In this paper, we aim at solving the structural redundancy of the hybrid intra/inter-round dialogue paradigm in MAS. We introduce TOPODIM, a novel decentralized framework for one-shot heterogeneous topology generation tailored adaptively to task queries. Experiments show that TOPODIM achieves SOTA performance in overall performance and cost-efficiency. Meanwhile, its adaptability to heterogeneous agent types allows to avoid the bucket effect. We hope our work will inspire future research on exploring synergy between cognitive mechanisms and interaction topology for scalable collective intelligence.

Limitations

Interaction constraints. While TopoDIM demonstrates competitive performance through conditioning, feedback, and debate mechanisms, our current focus prioritizes these direct interaction forms to maintain an optimal balance between predictive accuracy and token efficiency. Consequently, the scope of this work does not extend to complex organizational methods, such as dynamic coalition formation, where subsets of agents spontaneously align to address specific sub-problems.

Heterogeneous agents adaptability. We demonstrate TopoDIM outperforms baselines in heterogeneous agent settings, nevertheless finding the combination of high-performance and lightweight LLMs is non-trivial. In practice, we observed that simply introducing random agents yields small performance gains when tasks observably exceed agents capabilities. Furthermore, the deployment of heterogeneous LLMs imposes infrastructure challenges, as serving models with varying architectures and weights requires engineering support.

Computational Overhead. Compared to single-LLM approaches, MAS inevitably incurs more token consumption and latency due to extensive inter-agent communication (Hu et al., 2026a). While recent works attempt to mitigate this by organizing cooperation within the latent space (Zou et al., 2025; Fu et al., 2025), we adhere to a topology-based paradigm to ensure compatibility with proprietary models. Future work will explore effective mechanisms to further optimize cost-efficiency while improving performance.

Ethical Considerations

This work relies exclusively on publicly available datasets and benchmarks, ensuring that no private or personally identifiable information is involved. While we did not observe emergent harmful behaviors or unintended autonomous actions during evaluation, we acknowledge that LLM-based multi-agent systems may inherit risks such as bias propagation or hallucinations. Crucially, our proposed framework serves as a coordination layer designed to enhance cooperative efficiency; it operates within the safety boundaries of the base LLMs without bypassing their existing guardrails. Finally, AI assistants were employed solely for grammatical error correction and rephrasing.

Acknowledgement

This work was supported by the National Natural Science Foundation of China (T2293771), the STI 2030–Major Projects (2022ZD0211400), and National Natural Science Foundation of China (No. 42230406).

References

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, and 1 others. 2025. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*.

Jinchi Chen, Jie Feng, Weiguo Gao, and Ke Wei. 2024. Decentralized natural policy gradient with variance reduction for collaborative multi-agent reinforcement learning. *Journal of Machine Learning Research*, 25(172):1–49.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, and 1 others. 2023. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *The Twelfth International Conference on Learning Representations*.

Weize Chen, Jiarui Yuan, Chen Qian, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2025. Optima: Optimizing effectiveness and efficiency for llm-based multi-agent system. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 11534–11557.

Hyeong Kyu Choi, Xiaojin Zhu, and Yixuan Li. 2025. Debate or vote: Which yields better decisions in multi-agent large language models? *arXiv preprint arXiv:2508.17536*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2023. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first International Conference on Machine Learning*.

Tianyu Fu, Zihan Min, Hanling Zhang, Jichao Yan, Guohao Dai, Wanli Ouyang, and Yu Wang. 2025. Cache-to-cache: Direct semantic communication between large language models. *arXiv preprint arXiv:2510.03215*.

Ali Essam Ghareeb, Benjamin Chang, Ludovico Mitchener, Angela Yiu, Caralyn J Szostkiewicz, Jon M Laurent, Muhammed T Razzak, Andrew D White, Michaela M Hinks, and Samuel G Rodriques. 2025. Robin: A multi-agent system for automating scientific discovery. *arXiv preprint arXiv:2505.13400*.

Chenghua Gong and Xiang Li. 2025. Agents with foundation models: advance and vision. *Frontiers of Computer Science*, 19(4):194330.

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

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Samuel Holt, Max Ruiz Luyten, and Mihaela van der Schaar. 2025. L2mac: Large language model automatic computer for extensive code generation. *arXiv preprint arXiv:2310.02003*.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, and 1 others. 2023. Metagpt: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*.

Jinwei Hu, Yi Dong, Youcheng Sun, and Xiaowei Huang. 2026a. Tapas are free! training-free adaptation of programmatic agents via llm-guided program synthesis in dynamic environments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 29477–29485.

Jinwei Hu, Xinmiao Huang, Youcheng Sun, Yi Dong, and Xiaowei Huang. 2026b. Lying with truths: Open-channel multi-agent collusion for belief manipulation via generative montage. *arXiv preprint arXiv:2601.01685*.

Junan Hu, Shudan Guo, Wenqi Liu, Jianhua Yin, and Yinwei Wei. 2026c. Context-agent: Dynamic discourse trees for non-linear dialogue. *arXiv preprint arXiv:2604.05552*.

Md Ashraful Islam, Mohammed Eunus Ali, and Md Rizwan Parvez. 2024. Mapcoder: Multi-agent code generation for competitive problem solving. *arXiv preprint arXiv:2405.11403*.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*.

Eric Hanchen Jiang, Guancheng Wan, Sophia Yin, Mengting Li, Yuchen Wu, Xiao Liang, Xinfeng Li, Yizhou Sun, Wei Wang, Kai-Wei Chang, and 1 others. 2025. Dynamic generation of multi-llm agents communication topologies with graph diffusion models. *arXiv preprint arXiv:2510.07799*.

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

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

Shiyuan Li, Yixin Liu, Qingsong Wen, Chengqi Zhang, and Shirui Pan. 2025. Assemble your crew: Automatic multi-agent communication topology design via autoregressive graph generation. *arXiv preprint arXiv:2507.18224*.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. 2023. Encouraging divergent thinking in large language models through multi-agent debate. *arXiv preprint arXiv:2305.19118*.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, and 1 others. 2025. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*.

Shiyu Liu, Yongjing Yin, Jianhao Yan, Yunbo Tang, Qinggang Zhang, Bei Li, Xin Chen, Jingang Wang, Xunliang Cai, and Jinsong Su. 2026. Bapo: Boundary-aware policy optimization for reliable agentic search. *arXiv preprint arXiv:2601.11037*.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. 2024. A dynamic llm-powered agent network for task-oriented agent collaboration. In *First Conference on Language Modeling*.

Yunbo Long, Yuhan Liu, and Liming Xu. 2026. Emomas: Emotion-aware multi-agent system for high-stakes edge-deployable negotiation with bayesian orchestration. *arXiv preprint arXiv:2604.07003*.

Shichao Ma, Yunhe Guo, Jiahao Su, Qihe Huang, Zhengyang Zhou, and Yang Wang. 2026a. Talk2image: A multi-agent system for multi-turn image generation and editing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 32437–32445.

Shichao Ma, Zhiyuan Ma, Ming Yang, Xiaofan Li, Xing Wu, Jintao Du, Yu Cheng, Weiqiang Wang, Qiliang Liu, Zhengyang Zhou, and 1 others. 2026b. Tspo: Breaking the double homogenization dilemma in multi-turn search policy optimization. *arXiv preprint arXiv:2601.22776*.

Mathematical Association of America. 1983–2025. American invitational mathematics examination (aime). Accessed via Art of Problem Solving or [Insert Dataset Name].

Ollama. 2023. Ollama: Large language model runner. Software available at https://github.com/ollama/ollama.

OpenAI. 2025. Gpt-5 system card. https://openai.com/index/gpt-5-system-card/. Accessed: 2025-08-13.

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

Chen Qian, Zihao Xie, Yifei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, and 1 others. 2024b. Scaling large language model-based multi-agent collaboration. *arXiv preprint arXiv:2406.07155*.

Subhro Roy and Dan Roth. 2016. Solving general arithmetic word problems. *arXiv preprint arXiv:1608.01413*.

Marlene Scardamalia and Carl Bereiter. 2006. Knowledge building. *The Cambridge*.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, and 1 others. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Fengyi Wang, Guanghui Zhu, Chunfeng Yuan, and Yihua Huang. 2024a. Llm-enhanced cascaded multi-level learning on temporal heterogeneous graphs. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 512–521.

Song Wang, Zhen Tan, Zihan Chen, Shuang Zhou, Tianlong Chen, and Jundong Li. 2025a. Anymac: Cascading flexible multi-agent collaboration via next-agent prediction. *arXiv preprint arXiv:2506.17784*.

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

Yu Wang and Xi Chen. 2025. Mirix: Multi-agent memory system for llm-based agents. *arXiv preprint arXiv:2507.07957*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, and 1 others. 2024b. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290.

Zhexuan Wang, Yutong Wang, Xuebo Liu, Liang Ding, Miao Zhang, Jie Liu, and Min Zhang. 2025b. Agentdropout: Dynamic agent elimination for token-efficient and high-performance llm-based multi-agent collaboration. *arXiv preprint arXiv:2503.18891*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*, 3(4).

Xingjian Wu, Xvyuan Liu, Junkai Lu, Siyuan Wang, Xiangfei Qiu, Yang Shu, Jilin Hu, Chenjuan Guo, and Bin Yang. 2026. St-evo: Towards generative spatio-temporal evolution of multi-agent communication topologies. *arXiv preprint arXiv:2602.14681*.

Xiangyuan Xue, Yifan Zhou, Guibin Zhang, Zaibin Zhang, Yijiang Li, Chen Zhang, Zhenfei Yin, Philip Torr, Wanli Ouyang, and Lei Bai. 2025. Comas: Co-evolving multi-agent systems via interaction rewards. *arXiv preprint arXiv:2510.08529*.

Yingxuan Yang, Huacan Chai, Shuai Shao, Yuanyi Song, Siyuan Qi, Renting Rui, and Weinan Zhang. 2025. Agentnet: Decentralized evolutionary coordination for llm-based multi-agent systems. *arXiv preprint arXiv:2504.00587*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

Rui Ye, Xiangrui Liu, Qimin Wu, Xianghe Pang, Zhenfei Yin, Lei Bai, and Siheng Chen. 2025. X-mas: Towards building multi-agent systems with heterogeneous llms. *arXiv preprint arXiv:2505.16997*.

Yuting Zeng, Weizhe Huang, Lei Jiang, Tongxuan Liu, Xitai Jin, Chen Tianying Tiana, Jing Li, and Xiaohua Xu. 2025. S^2-mad: Breaking the token barrier to enhance multi-agent debate efficiency. *arXiv preprint arXiv:2502.04790*.

Guibin Zhang, Muxin Fu, Guancheng Wan, Miao Yu, Kun Wang, and Shuicheng Yan. 2025. G-memory: Tracing hierarchical memory for multi-agent systems. *arXiv preprint arXiv:2506.07398*.

Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Xu Yu, and Tianlong Chen. 2024a. Cut the crap: An economical communication pipeline for llm-based multi-agent systems. *arXiv preprint arXiv:2410.02506*.

Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. 2024b. G-designer: Architecting multi-agent communication topologies via graph neural networks. *arXiv preprint arXiv:2410.11782*.

Peixuan Zhang, Zijian Jia, Ziqi Cai, Shuchen Weng, Si Li, and Boxin Shi. 2026a. Recontraster: Making your posters stand out with regional contrast. *arXiv preprint arXiv:2604.10442*.

Wenyuan Zhang, Xinghua Zhang, Haiyang Yu, Shuaiyi Nie, Bingli Wu, Juwei Yue, Tingwen Liu, and Yongbin Li. 2026b. Expseek: Self-triggered experience seeking for web agents. *arXiv preprint arXiv:2601.08605*.

Yuzhe Zhang, Feiran Liu, Yi Shan, Xinyi Huang, Xin Yang, Yueqi Zhu, Xuxin Cheng, Cao Liu, Ke Zeng, Terry Jingchen Zhang, and 1 others. 2026c. Silobench: A scalable environment for evaluating distributed coordination in multi-agent llm systems. *arXiv preprint arXiv:2603.01045*.

Chuanyang Zheng, Zhengying Liu, Enze Xie, Zhenguo Li, and Yu Li. 2023. Progressive-hint prompting improves reasoning in large language models. *arXiv preprint arXiv:2304.09797*.

Han Zhou, Xingchen Wan, Ruoxi Sun, Hamid Palangi, Shariq Iqbal, Ivan Vulić, Anna Korhonen, and Sercan Ö Arık. 2025. Multi-agent design: Optimizing agents with better prompts and topologies. *arXiv preprint arXiv:2502.02533*.

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

Jiaru Zou, Xiyuan Yang, Ruizhong Qiu, Gaotang Li, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, James Zou, Mengdi Wang, and Ling Yang. 2025. Latent collaboration in multi-agent systems. *arXiv preprint arXiv:2511.20639*.

A Algorithm workflow

The algorithm workflow of TopoDIM is illustrated in Algorithm 1. Specifically, TopoDIM comprises two training stages. In stage 1, TopoDIM iterates through all training queries and employs policy optimization to update the encoder and decoder parameters, incorporating structural constraints and adaptive sparsification. In stage 2, TopoDIM performs decentralized policy distillation using the same training set as in Stage 1, updating the parameters of lightweight networks which are deployed on each agent itself for a purpose of flexible and private decision for . During the inference phase, TopoDIM generates a communication topology and traverses it in a breadth-first manner. By facilitating agent cooperation through three semantically rich interaction modes, the method achieves competitive performance on complex tasks.

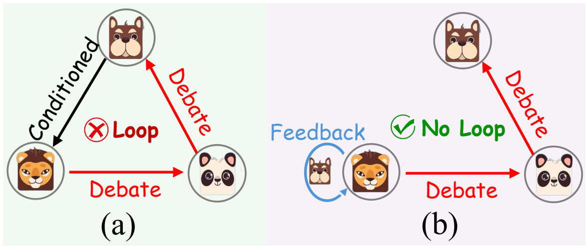


Figure 7: Illustration of acyclic property for three edge types, including (a) loop collaboration with conditioned and debate edges and (b) no loop collaboration with three edge types. The solid arrows only indicate communication directions.

B Acyclic Property

We impose acyclic structural constraints exclusively on conditioned and debate edges, leaving feedback edges unconstrained. Specifically, structural constraints are applied exclusively to conditioned and debate edges, which must remain acyclic to preclude circular dependencies (Figure 7(a)). In contrast, feedback edges are allowed for arbitrary connectivity (Figure 7(b)). During traversal, feedback edges are prioritized for immediate execution: they are excluded from degree calculations and removed upon visit. Meanwhile other edge types adhere to standard breadth-first scheduling.

C Data Statistics

We summarized the data statistics in Table 4. In this paper we conduct experiments across three representative categories of datasets, incorporating challenging benchmarks such as MMLU-Pro, AIME, and LiveCodeBench to make a comprehensive evaluation of our proposed TOPODIM. For the AIME benchmark, given the limited volume of available problems, we utilize the entire collection of samples from 2023 to 2025 for training (40 samples) and evaluation (50 samples) (Mathematical Association of America, 1983–2025). For LiveCodeBench, we employ Release V1 version of for a moderately challenging evaluation on code generation tasks, which comprises data collected between May 2023 and March 2024. For MMLU-Pro, we adopt the same data processing pipeline as applied to MMLU in previous work (Zhang et al., 2024b), selecting 153 samples from the test set as our evaluation subset. The experimental settings for the remaining datasets, including MultiArith, HumanEval, and GSM8K, remain consistent with prior research (Wang et al., 2025b) to ensure fair comparison.

Table 4: Dataset descriptions and statistics.

Category	Dataset	Answer Type	Metric	#Test
General reasoning	MMLU-Pro	Multi-choice	Acc.	153
Math reasoning	AIME_{23-25}	Number	Acc.	50
	MultiArith	Number	Acc.	600
	GSM8K	Number	Acc.	1,319
Code generation	$\text{LiveCodeBench}_{v1}$	Code	Pass@1	360
	HumanEval	Code	Pass@1	164

D Cost Efficiency

We provide a more detailed display for cost-efficiency of TOPODIM, shown in Table 6. TOPODIM cuts down the token expenditure, saving maximum prompt and completion token by 57.82% and 22.05% compared to the most efficient framework. Moreover, we conducted experiments to evaluate GPU resource overhead during the inference phase. By randomly instantiating 1,000 nodes, we demonstrated that our decentralized design, where each agent operates a lightweight network, requires merely 6.36 GB of memory, achieving significant cost efficiency.

E Heterogeneous Agents Adaptability

To provide a more comprehensive evaluation of TOPODIM's adaptability to heterogeneous agents, we conducted experiments on two additional configurations involving agents with distinct sources and capacities. As detailed in Table 5, we combined `GPT-OSS:20B` with `DeepSeek-V3.2` and `Gemma3-it:12b`, respectively. In these settings, TOPODIM consistently outperformed other baselines across three representative datasets, achieving average improvements of 1.32% and 1.15%, respectively. It is worth noting that identifying effective combinations of high-performance and lightweight LLMs is non-trivial; naively introducing random agents with heterogeneous skills can sometimes be counterproductive (e.g., the performance degradation observed with G-Designer). In contrast, TOPODIM leverages its adaptive sparsification design to effectively prune agents that make negligible contributions, thereby optimizing communication efficiency.

Table 5: Performance comparison of heterogeneous agents distinguished by sources and capacities in MAS.

Method	LiveCodeBench	MMLU-Pro	AIME
Decision-maker: `GPT-OSS:20B`; **Collabration:** 3 `Gemma3-it:12B`, 2 `GPT-OSS:20B`			
Vanilla	70.40	68.84	29.57
AgentDropout	71.26$_{\uparrow 0.86}$	69.57$_{\uparrow 0.73}$	29.80$_{\uparrow 0.23}$
G-Designer	71.18$_{\uparrow 0.78}$	70.46$_{\uparrow 1.62}$	29.24$_{\downarrow 0.33}$
TOPODIM	**72.23**$_{\uparrow 1.83}$	**71.20**$_{\uparrow 2.36}$	**30.18**$_{\uparrow 0.61}$
Decision-maker: `DeepSeek-V3.2`; **Collabration:** 3 `GPT-OSS:20B`, 2 `DeepSeek-V3.2`			
Vanilla	78.38	78.68	64.37
AgentDropout	79.52$_{\uparrow 1.14}$	80.10$_{\uparrow 1.42}$	64.55$_{\uparrow 0.18}$
G-Designer	80.17$_{\uparrow 1.79}$	80.68$_{\uparrow 2.00}$	65.23$_{\uparrow 0.86}$
TOPODIM	**82.46**$_{\uparrow 4.08}$	**82.37**$_{\uparrow 3.69}$	**66.59**$_{\uparrow 2.22}$

F Robustness Analysis

We evaluated the robustness of TOPODIM by conducting experiments on LiveCodeBench with `GPT-OSS:120B`. An agent is randomly selected and injected with prompts that drive it to generate failure outputs. As illustrated in Figure 8, thanks to its heterogeneous adaptability, TOPODIM exhibits remarkable robustness to these attacks, limiting the performance degradation to 0.3 points. Conversely, LLM-Debate incurs a drop of 1.2 due to its fixed communication topology. Meanwhile, dynamic topology frameworks like GPTSwarm, G-Designer, and AgentDropout also display competitive structural robustness, with an average drop of 0.43. This adaptive capability is vital for identifying malicious agents, underscoring the superiority of dynamic topologies in maintaining structural integrity.

Table 6: Comparison of token consumption. Ctok. and Ptok. indicate prompt and complete token consumption.

Consumption	LLM-Debate		GPTSwarm		G-Designer		AgentDropout		TopoDIM	
	MMLU-Pro	LiveCodeBench	MMLU-Pro	LiveCodeBench	MMLU-Pro	LiveCodeBench	MMLU-Pro	LiveCodeBench	MMLU-Pro	LiveCodeBench
Ptok.	7.31 M	13.72 M	6.14 M	9.46 M	3.93 M	6.37 M	2.09 M	5.30 M	0.88 M	3.02 M
Ctok.	2.87 M	4.63 M	2.40 M	3.15 M	1.56 M	2.21 M	0.98 M	1.85 M	0.76 M	1.69 M
Total	10.18 M	18.36 M	8.54 M	12.61 M	5.49 M	8.58 M	3.06 M	7.16 M	1.64 M	4.71 M

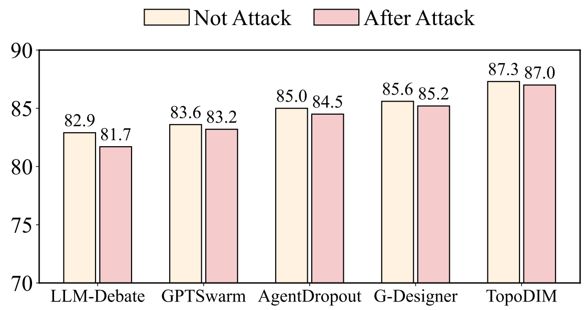


Figure 8: Pass@1 performance comparison of TopoDIM and other multi-agent baselines on LiveCodeBench pre- and post-prompt attacks. Dynamic topology frameworks exhibit competitive structural robustness, demonstrating small performance degradation.

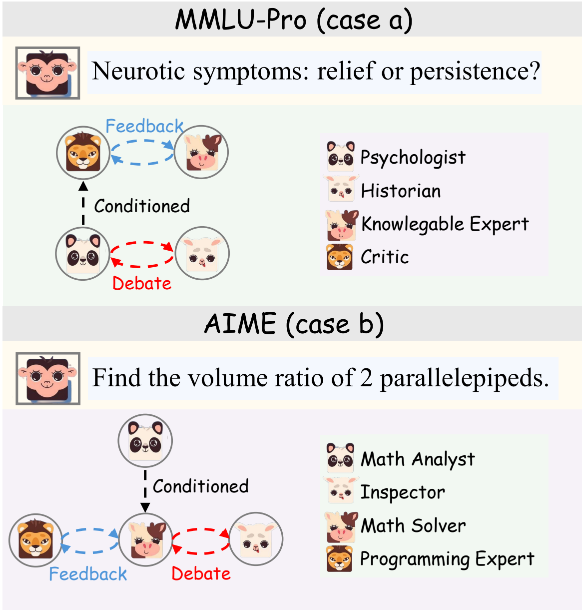


Figure 9: Case studies of the communication topologies designed by TopoDIM on MMLU-Pro (case a) and AIME (case b) benchmarks.

G Case Study

Task-specific Patterns. We conduct case studies with `GPT-OSS-120B` on AIME (math) and MMLU-Pro (general) to illustrate task-specific collaboration patterns, as shown in Figure 9. For an MMLU-Pro psychology question, TopoDIM generates an efficient topology that coordinates multiple agents to solve the problem. The psychologist and historian first engage in a debate. The critic then reviews the psychologist's response and consults a knowledgeable expert for professional feedback. For an AIME problem, the math analyst first examines the question and passes the result to the math solver. Conditioned on this output, the math solver works out the solution and requests feedback from a programming expert for verification. Finally, the math solver and inspector enter a debate to further refine the solution.

Specific Cases. While the main text briefly compares the communication sequence of TopoDIM with two typical intra- and inter-round dialogue paradigms using `GPT-OSS:120B` on a LiveCodeBench question, this section presents the detailed cooperative dialogues among different agents, as illustrated in Figures 10 through 14. Upon receiving a question q, TopoDIM effectively generates a communication graph to orchestrate agent collaboration. As shown in Figure 11, the algorithm designer outlines the solution strategy and provides the initial code. The bug fixer then generates a revised solution conditioned on the designer's output. Subsequently, a programming expert engages in a debate with the bug fixer to produce robust code that efficiently solves q, as depicted in Figure 12. Next, the project manager proposes a separate solution, which undergoes a detailed evaluation by the algorithm designer. Based on this critical feedback, the project manager refines the solution, as illustrated in Figure 13. Finally, a decision-maker reviews the collective contributions from all participants and delivers the final answer, as depicted in Figure 14. These interactions exemplify the iterative self-correction capability facilitated by the multi-type interactions.

Algorithm 1 Workflow of TopoDIM

Input: Initial graph $\mathcal{G}_{\text{pri}}$, query $\mathcal{Q}$
Output: Communication topology $\hat{\mathcal{G}}$

// Stage 1: Centralized
for node i **in** $\{1, 2, ..., N\}$ **do**
  $\mathbf{h}_i^{(0)} \leftarrow \mathbf{e}_i^{\text{role}} + \mathbf{W}_q \phi(q)$
end for
for query q **in** $\mathcal{Q}$ **do**
  // Encoder
  $\mathbf{h}_i^{l+1} \leftarrow \sigma\big(\sum_{r \in \mathcal{R}_{\text{pri}}} \sum_{j \in \mathcal{N}_i^r} \frac{1}{c_{i,r}} \mathbf{W}_r^{(l)} \mathbf{h}_j^{(l)} + \mathbf{W}_0^{(l)} \mathbf{h}_i^{(l)}\big)$
  $\mathbf{H} \leftarrow [\mathbf{h}_1^{l+1}, \mathbf{h}_2^{l+1}, ..., \mathbf{h}_N^{l+1}]$
  // Decoder
  $p_\theta(\mathcal{G} \mid \mathbf{H}) = \prod_{j=2}^{N} \prod_{i=1}^{j-1} p_\theta(r_{ij} \mid \mathbf{H}, \mathcal{G}_{<ij})$
  $r_{ij} \leftarrow \min\{r \; : \; F(r) \; \geq \; u\}, F(r) \leftarrow \sum_{r'=0}^{r} p_\theta(r')$
  // Structural constraints
  $p(r_{ij} \mid \cdot) \leftarrow p(r_{ij} \mid \cdot) \odot \mathbf{M}_{ij}$
  // Sparsification
  $\mathcal{E}_{\text{final}} = \text{TopK}\left(\{p(r_{ij})\}_{\forall i,j}, 1 - \alpha\right)$
  $\mathcal{V}_{\text{final}} = \{v \mid \exists (h, t, r) \in \mathcal{E}_{\text{final}}, v \in \{h, t\}\}$
  $\mathcal{G}_{\text{final}} \leftarrow (\mathcal{V}_{\text{final}}, \mathcal{E}_{\text{final}}, r_{\text{final}})$
  for node i **in** $\psi(\mathcal{G}_{\text{final}})$ **do** // Breadth-first traversal
    $\mathcal{S}_i \leftarrow v_i(\mathcal{P})$
  end for
  $a \leftarrow \text{Aggregate}(\{\mathcal{S}\}_{i=1}^{N})$ // No multi-rounds dialogues
  $\theta^{q+1} \leftarrow \theta^q - \delta \nabla_{\theta^q} J(\theta)$ // δ represents the learning rate during centralized training
end for
// Stage 2: Decentralized
for query q **in** $\mathcal{Q}$ **do**
  $\mathcal{L}_{\text{distill}}(\theta') \leftarrow \sum_{v_i \in \mathcal{V}} \sum_{v_j \in \mathcal{V} \setminus v_i} D_{KL}\big(\pi_\theta(r_{ij} | \mathcal{G}_{\text{pri}}, q) \, || \, \pi_{\theta'}(r_{ij} | \mathbf{h}_i^{(0)}, \mathbf{h}_j^{(0)})\big)$ // No token overhead
  $\theta'^{q+1} \leftarrow \theta'^q - \beta \nabla_{\theta'^q} \mathcal{L}_{\text{distill}}$ // β represents the learning rate during decentralized training
end for

Question

You are given an integer N and a string S consisting of '0', '1', and '?'.

Let T be the set of values that can be obtained by replacing each '?' in S with '0' or '1' and interpreting the result as a binary integer.

For instance, if $S = ?0?$, we have $T = \{000_{(2)}, 001_{(2)}, 100_{(2)}, 101_{(2)}\} = \{0,1,4,5\}$.

Print (as a decimal integer) the greatest value in T less than or equal to N.If T does not contain a value less than or equal to N, print '-1' instead.

Figure 10: Question overview of case study.

Solution (Algorithm Designer) (1/9)

Algorithm Overview (≈45 words)
Use depth-first search with memoisation over the string positions. State `(i, tight)` means we processed the first `i` bits; `tight` is `True` if the prefix equals the prefix of `N`. For each allowed bit (respecting '?') we skip choices that would exceed `N` when `tight` is `True`. The recursion returns the maximal suffix value; combine with the current bit weight. If no value ≤ `N` exists, return `-1`.

Complexity – `O(L·2)` time and `O(L·2)` memory, where `L ≤ 60`.

```python
<CODE HERE>
```

Solution (Bug Fixer) (Conditioned Algorithm Designer)(2/9)

```python
<CODE HERE>
```

Figure 11: Detailed dialogues generated in the Conditioned mode.

Solution (Bug Fixer) (Conditioned Algorithm Designer)(2/9)

```python
<CODE HERE>
```


Debate (Programming Expert) (Debate with Bug Fixer)(3/9)

```python
<CODE HERE>
```

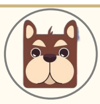
Debate (Bug Fixer) (Debate with Programming Expert)(4/9)

```python
<CODE HERE>
```

Solution (Programming Expert) (Conditioned Bug Fixer)(8/9)

```python
<CODE HERE>
```

Figure 12: Detailed dialogues generated in the Debate mode.



Solution (Project Manager) (5/9)

Design Overview

* Use a greedy DP with two states:
  `equal` – built prefix equals N's prefix so far.
  `less` – already smaller than N, so remaining bits can be maximised.

* Iterate bits of S from most-significant to least-significant, updating the best value for each state.
* If the maximal possible value (replace all '?' by 1) ≤ N, output it directly.
* Otherwise, after processing all bits, the answer is the larger of the two state values; if none exist, output -1.

Complexity – O(|S|) time, O(1) extra space.
Complexity – `O(L·2)` time and `O(L·2)` memory, where `L ≤ 60`.

```python
&lt;CODE HERE&gt;
```

Evaluation (Algorithm Designer)(Evaluate Project Manager) (6/9)

Correctness: ✅ The DP correctly tracks "equal" vs "already-less" prefixes, handling '?' choices and respecting the N-limit. Edge cases (all-zero, no feasible value, N larger than any possible) are covered.

Complexity: O(|S|) time, O(1) extra space.

Overall: The implementation is sound and meets the problem requirements.

Re-solute (Project Manager)(Feedback Algorithm Designer) (7/9)

Use a DP with two states while scanning the pattern from most-significant to least-significant bit:

* `eq` – maximum value achievable when the built prefix is exactly equal to the prefix of `N`.
* `lt` – maximum value achievable when the built prefix is already strictly less than the prefix of `N`.

For each character we try the allowed bits (0/1). Transitions respect the current state and the corresponding bit of `N`.
At the end the answer is `max(eq, lt)`. If both are impossible, output `-1`.
Complexity: **O(|S|)** time, **O(1)** extra space.
```python
&lt;CODE HERE&gt;
```

Figure 13: Detailed dialogues generated in the Feedback mode.

Solution (Decision-maker) (Conditioned all agents)(9/9)

```python
def solve() -> None:
    import sys
    data = sys.stdin.read().strip().split()
    if not data:
        return
    S = data[0]
    N = int(data[1])
    L = len(S)
    # Maximum value achievable by setting all '?' to 1
    max_possible = 0
    for i, ch in enumerate(S):
        shift = L - 1 - i
        if ch == '1' or ch == '?':
            max_possible |= (1 << shift)

    # If N is large enough, answer is simply max_possible
    if N >= max_possible:
        print(max_possible)
        return
    # Bits of N aligned to length L (MSB first)
    Nbits = [(N >> (L - 1 - i)) & 1 for i in range(L)]
    NEG = -1  # sentinel for impossible state
    eq = 0    # value when built prefix equals N's prefix so far
    lt = NEG  # value when built prefix is already less than N's prefix
    for i, ch in enumerate(S):
        allowed = [0, 1] if ch == '?' else [int(ch)]
        nb = Nbits[i]
        new_eq = NEG
        new_lt = NEG
        # Transition from eq state
        if eq != NEG:
            for b in allowed:
                if b == nb:
                    cand = (eq << 1) | b
                    if cand > new_eq:
                        new_eq = cand
                elif b < nb:
                    cand = (eq << 1) | b
                    if cand > new_lt:
                        new_lt = cand
                # b > nb would exceed N, ignore
        # Transition from lt state (already less, can choose any allowed bit)
        if lt != NEG:
            for b in allowed:
                cand = (lt << 1) | b
                if cand > new_lt:
                    new_lt = cand

        eq, lt = new_eq, new_lt
    ans = max(eq, lt)
    print(ans if ans != NEG else -1)
if __name__ == "__main__":
    solve()
```

Figure 14: Detailed decision-making overview of case study.