LEARNING TOPOLOGY PREFERENCES FOR ADAPTIVE COMMUNICATION TOPOLOGY GENERATION IN MULTI-AGENT SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM)-based multi-agent systems (MAS) have demonstrated strong capabilities in solving complex tasks, where communication topology plays a critical role in determining performance. However, most methods leverage execution feedback at the whole-graph level, leaving the fine-grained structural decisions underlying topology generation insufficiently explored. In particular, the relative preferences revealed by comparative topology executions are rarely captured as explicit guidance for generation. To this end, we propose Topology Preference-guided Adaptive Generation (TopoPAG), a framework that transforms execution-derived topology preferences into task-specific structural guidance for autoregressive MAS topology generation. Specifically, TopoPAG learns preferences from the comparative utilities of candidate topologies and embeds them into a compact continuous structural space using graph descriptors. The predicted preference provides an initial topology representation beyond discrete template selection. During generation, this representation is adaptively refined based on the partially constructed graph to guide subsequent role and communication-link predictions. Furthermore, the learned preference weights graph generation, allowing all evaluated topologies to contribute according to their relative utility. Experiments on six reasoning and coding benchmarks demonstrate that TopoPAG consistently produces more effective and communication-efficient topologies. Our code is available at `https://anonymous.4open.science/r/TopoPAG-7853`.

1 INTRODUCTION

Large language models (LLMs) have substantially advanced the capabilities of autonomous agents in reasoning, planning, tool use, and task execution wang2024survey,ferrag2026llm. However, a single agent remains limited by its individual perspective, susceptibility to reasoning errors, and difficulty in handling complex tasks that require diverse expertise or coordinated subtasks. LLM-based multi-agent systems (MAS) mitigate these limitations by enabling multiple agents with complementary roles to collaborate toward a common objective (Li et al., 2024). The effectiveness of such systems depends not only on the capabilities of individual agents, but also on how agents are organized and how information is exchanged among them (Tran et al., 2025). Consequently, communication topology, which determines the interaction structure and information flow among agents, has become a central design factor in LLM-based MAS(Leong et al., 2025; Li et al., 2026a;b).

Early studies on LLM-based MAS typically relied on manually specified communication patterns, such as chains and trees, to organize agent interactions (Wu et al., 2023; Hong et al., 2024). Although these predefined topologies provide simple and effective coordination mechanisms, a fixed communication structure cannot accommodate the diverse collaboration requirements of different tasks. This limitation has motivated a shift toward automated topology design, where the communication graph is adapted according to task characteristics rather than predetermined in advance (Zhang et al., 2025c;b). Recent approaches have further extended this direction from modifying or selecting existing graph structures to directly generating task-specific collaboration topologies, including the participating agents, their roles, and communication links (Shang et al., 2025; Li et al., 2026b; Jiang et al., 2026). These advances substantially improve the flexibility of MAS construction

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.

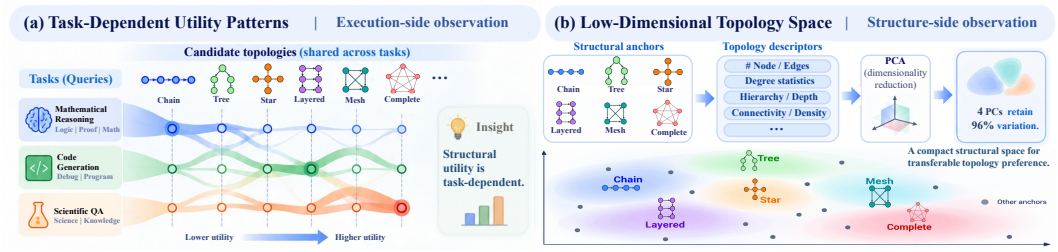


Figure 1: Key observations motivating TopoPAG: task-dependent topology utility and low-dimensional structural organization of candidate topologies.

and establish topology generation as a promising alternative to manually designed collaboration protocols. However, as topology design becomes increasingly data-driven and generative, an important question remains regarding how the execution experience collected from different topologies should be exploited to guide the generation process.

Despite their different modeling and optimization strategies, existing topology design methods largely rely on execution outcomes as global signals for assessing and improving multi-agent collaboration (Zhang et al., 2025b; Li et al., 2026a). Such feedback can support topology selection, optimization, or generation, but provides limited supervision about how different topological characteristics contribute to task performance. When multiple candidate topologies are evaluated on the same task, their relative utilities reveal richer comparative information about task-specific structural preferences. As illustrated in Figure 1(a), different task categories exhibit distinct utility patterns over shared candidate topologies, suggesting that topology effectiveness is inherently task-dependent. However, existing objectives typically exploit these outcomes to optimize individual topologies or model parameters rather than explicitly learning transferable topology preferences (Jiang et al., 2026; Li et al., 2026b). Without an explicit topology-level representation, such preference information cannot directly guide fine-grained decisions over agents, roles, and communication links, leaving the connection between preferred and generated topologies vague. This raises a natural question: *Can we explicitly learn task-specific topology preferences from execution feedback and use them as direct guidance for adaptive communication topology generation?*

To address these, we propose **Topo**logy **P**reference-guided **A**daptive **G**eneration (**TopoPAG**), which explicitly transforms execution feedback into topology preferences for adaptive communication topology generation. TopoPAG models the relative preference among candidate topologies instead of treating their execution outcomes as isolated graph-level signals. To make such preferences transferable across tasks, we characterize each candidate using task- and role-independent topology descriptors and organize them in a compact continuous topology space (Chen et al., 2025b). As shown in Figure 1(b), topology descriptors reveal a compact low-dimensional organization among the topology anchors, where four principal components preserve 96% of the variance. This continuous representation allows comparative execution utilities to be converted into task-specific topology preferences that generalize to unseen tasks. In addition, TopoPAG uses the predicted preference to construct an initial topology representation instead of selecting a predefined graph. During autoregressive generation, this representation is continuously updated according to the evolving partial topology. The updated representation directly conditions subsequent role and communication-link predictions, allowing execution-derived preferences to guide topology construction throughout generation. Together, these designs establish a direct pathway from comparative execution experience to adaptive topology generation.

TopoPAG further incorporates the learned topology preference into training by weighting graph imitation according to the relative utility of evaluated topologies, allowing diverse structural experiences to contribute during optimization. Extensive experiments across diverse reasoning benchmarks demonstrate that TopoPAG consistently improves task performance and communication efficiency over representative topology design methods. Further analyses show that the learned topology preferences reflect task-dependent structural tendencies and that the adaptive representations evolve consistently with topology generation. Our main contributions are summarized as follows:

- We formulate communication topology design from the perspective of topology preference learning, explicitly extracting task-specific preferences from comparative MAS executions and enabling their transfer to unseen tasks.

- We propose the TopoPAG framework, which maps topology preferences into a continuous topology representation and adaptively updates this representation during topology generation to guide role and communication-link prediction.
- Extensive experiments on diverse reasoning and coding benchmarks demonstrate the effectiveness and efficiency of TopoPAG, while further analyses reveal the structural meaning of the learned preferences and adaptive representations.

2 PRELIMINARIES

2.1 LLM-BASED MAS AS COLLABORATION GRAPHS

Given a task query $q_i \in \mathcal{Q}$, we formulate an LLM-based MAS collaboration as a directed acyclic graph (DAG) $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The node set $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ represents the participating agents. Each agent v_l is instantiated with a role $r_l \in \mathcal{R}$, where $\mathcal{R}$ denotes a predefined role pool and r_l specifies the expertise and responsibility of the agent. The edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ defines the directed communication dependencies among agents. In particular, an edge $e_{ml} = (v_m, v_l) \in \mathcal{E}$ indicates that the output of agent v_m is provided to agent v_l as contextual information for subsequent reasoning. During MAS execution, agents are activated according to the dependency structure specified by $\mathcal{G}$. Each agent performs its role-specific reasoning based on the task query and the information received from its predecessors, and its output can in turn be propagated to downstream agents. Therefore, $\mathcal{G}$ jointly specifies the agent composition and the information flow through which collaborative reasoning is carried out.

2.2 AUTOREGRESSIVE TOPOLOGY GENERATION

The goal of topology generation is to learn a conditional distribution $p_\theta(\mathcal{G} \mid q_i)$ that constructs an appropriate collaboration graph for the given task query. Following the autoregressive graph generation paradigm, the graph is constructed incrementally by adding one agent and its associated communication links at each generation step. Let $\mathcal{G}_{<l}$ denote the partial graph containing the agents and communication links generated before step l. Node generation instantiates a new agent v_l by selecting its role $r_l \in \mathcal{R}$, while edge generation determines its incoming communication links from previously generated agents. Accordingly, the conditional graph distribution is factorized as:

$$p_\theta(\mathcal{G} \mid q_i) = \prod_{l=1}^{|\mathcal{V}|} \underbrace{p_\theta(r_l \mid \mathcal{G}_{<l}, q_i)}_{\text{Node Generation}} \prod_{m=1}^{l-1} \underbrace{p_\theta(e_{ml} \mid r_l, \mathcal{G}_{<l}, q_i)}_{\text{Edge Generation}}. \tag{1}$$

Here, e_{ml} denotes the directed link from a previously generated agent v_m to the newly generated agent v_l. This factorization transforms topology generation into a sequence of conditional node and edge decisions, while allowing each decision to depend on the task and the graph constructed so far.

3 METHODOLOGY

TopoPAG learns task-specific topology preferences from comparative anchor executions and uses them to guide autoregressive topology generation. The learned preferences are embedded into a continuous topology space and adaptively refined during graph construction to condition role and communication-link predictions. The overall framework is illustrated in Figure 2.

3.1 TOPOLOGY PREFERENCE LEARNING

Autoregressive generation defines how topologies are constructed, but not which structures are preferred for a task. We learn such preferences from comparative executions of candidate topology anchors on training queries and predict them from task queries.

Let $\{\mathcal{G}_t\}_{t=1}^{T}$ denote a set of representative candidate topologies, including common collaboration patterns such as chain, tree, star, and so on. We treat them as *structural anchors* and characterize each $\mathcal{G}_t$ by a task- and role-independent descriptor $\mathbf{x}_t \in \mathbb{R}^D$. These descriptors capture complementary aspects of topology scale and connectivity, hierarchy, information flow, degree heterogeneity,

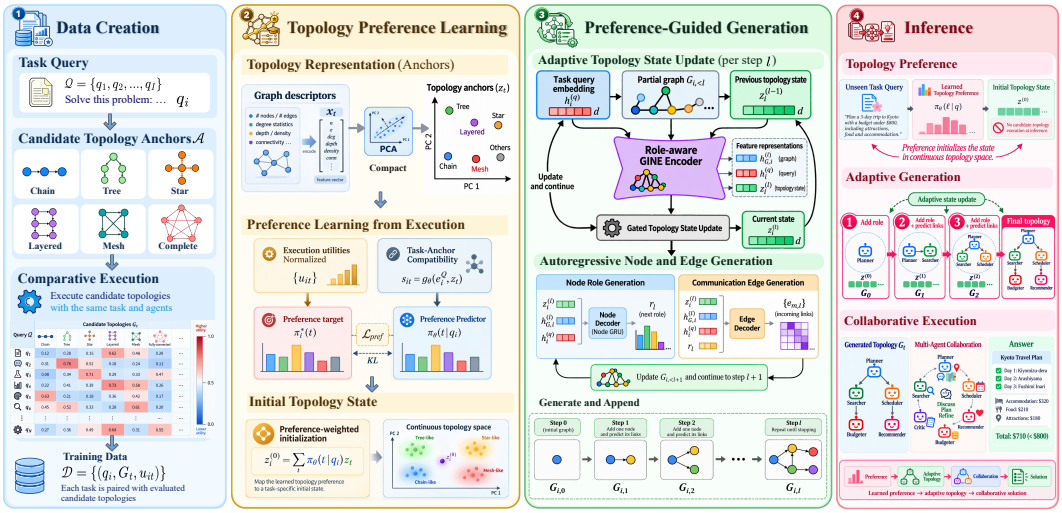


Figure 2: Overview of our proposed TopoPAG.

centralization, and path-based reachability; their complete definitions are provided in Appendix A. To capture the structural relationships among these anchors, we standardize their descriptors and construct a compact continuous topology space using principal component analysis (PCA). Let $\widetilde{\mathbf{X}} = [\widetilde{\mathbf{x}}_1^\top; \widetilde{\mathbf{x}}_2^\top; \ldots; \widetilde{\mathbf{x}}_T^\top] \in \mathbb{R}^{T \times D}$. Its singular value decomposition yields:

$$\widetilde{\mathbf{X}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top, \quad \mathbf{z}_t = \mathbf{V}_K^\top \widetilde{\mathbf{x}}_t \in \mathbb{R}^K, \tag{2}$$

where $\mathbf{V}_K \in \mathbb{R}^{D \times K}$ contains the first K principal directions. As observed in Figure 1, the anchor topologies exhibit a compact low-dimensional organization under these descriptors. This compact coordinate system therefore preserves the dominant structural variation among the anchors while providing a continuous basis for representing topology preference.

With this structural space established, we next evaluate the suitability of each anchor for individual tasks. For each training task q_i, we execute all T anchors and obtain binary correctness outcomes $y_{it} \in 0, 1$. The resulting execution utility is defined as:

$$u_{it} = y_{it} - \lambda_c \bar{c}_t, \quad \bar{c}_t = \mathrm{Norm}\left(|\mathcal{V}_t|/V_{\max} + |\mathcal{E}_t|/E_{\max}\right), \tag{3}$$

where λ_c controls the trade-off between task correctness and topology complexity, $\mathrm{Norm}(\cdot)$ denotes min–max normalization over the anchor set, and $E_{\max} = V_{\max}(V_{\max} - 1)/2$.

The utilities $u_{it}{}_{t=1}^{T}$ induce a relative preference distribution over the structural anchors. Rather than performing hard selection based on the maximum utility, we preserve the underlying preference structure through an entropy-regularized formulation. Let Δ_T denote the probability simplex over the T anchors. We define the entropy-regularized preference distribution as:

$$p_i^* = \arg\max_{\boldsymbol{\pi} \in \Delta_T} \left\{ \sum_{t=1}^{T} \pi_t u_{it} + \tau_u \mathcal{H}(\boldsymbol{\pi}) \right\}, \quad p_i^*(t) = \frac{\exp(u_{it}/\tau_u)}{\sum_{t'=1}^{T} \exp(u_{it'}/\tau_u)}, \tag{4}$$

where $\mathcal{H}(\boldsymbol{\pi}) = -\sum_t \pi_t \log \pi_t$ denotes the entropy of the distribution, and $\tau_u > 0$ controls its concentration. This formulation assigns larger probabilities to high-utility anchors while preserving the relative preference information encoded by all evaluated topologies.

The execution-derived preference in Equation 4 is available only for tasks evaluated with the structural anchors. To generalize this preference to unseen tasks, we obtain a query representation $\mathbf{e}_i^Q$ by projecting the pretrained sentence-transformer embedding of q_i, and learn a task-conditioned compatibility score $s_{it} = g_\theta(\mathbf{e}_i^Q, \mathbf{z}_t)$ for each structural anchor t. The predicted preference is:

$$p_\theta(t \mid q_i) = \frac{\exp(s_{it}/\tau_u)}{\sum_{t'=1}^{T} \exp(s_{it'}/\tau_u)}, \quad \mathcal{L}_{\text{pref}} = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} D_{\mathrm{KL}}\left(p_i^* \,\|\, p_\theta(\cdot \mid q_i)\right), \tag{5}$$

where $\mathcal{I}$ denotes the set of training tasks with non-trivial utility variation across the candidate topologies. The learned predictor thus transfers the topology preferences revealed by comparative executions to unseen tasks without requiring the candidate topologies to be executed again.

3.2 PREFERENCE-GUIDED TOPOLOGY GENERATION

The task-specific preference learned in Section 3.1 characterizes the structural patterns favored by the input task. TopoPAG converts this preference into a continuous topology representation and uses it as shared structural guidance throughout graph generation. For task q_i, the initial topology representation is obtained by aggregating the anchor coordinates according to the predicted preference:

$$\mathbf{z}_i^{(0)} = \sum_{t=1}^{T} p_\theta(t \mid q_i)\mathbf{z}_t = \mathbf{V}_K^\top \left(\sum_{t=1}^{T} p_\theta(t \mid q_i)\widetilde{\mathbf{x}}_t \right). \tag{6}$$

The second equality follows from the linear PCA projection in Equation 2. Hence, $\mathbf{z}_i^{(0)}$ summarizes the preference-weighted structural characteristics of the task, combining information from multiple structural anchors into a single topology-level representation.

As generation proceeds, the partial graph $\mathcal{G}_{<l}$ provides additional evidence about how the preferred structure has been instantiated through the roles and communication links generated so far. We encode the partial graph with GINE (Hu et al., 2020) to obtain node representations $\{\mathbf{h}_m^{(l)}\}_{m<l}$ and a pooled graph representation $\mathbf{h}_{\mathcal{G}}^{(l)}$. Combining the task representation $\mathbf{e}_i^Q$, the current graph prefix, and its preceding value, TopoPAG updates the topology representation as:

$$\widetilde{\mathbf{z}}_i^{(l)} = \mathrm{MLP}_{\mathrm{cand}}\left([\mathbf{e}_i^Q; \mathbf{h}_{\mathcal{G}}^{(l)}; \mathbf{z}_i^{(l-1)}] \right), \quad \mathbf{g}_i^{(l)} = \sigma\Big(\mathrm{MLP}_{\mathrm{gate}}\left([\mathbf{e}_i^Q; \mathbf{h}_{\mathcal{G}}^{(l)}; \mathbf{z}_i^{(l-1)}] \right) \Big), \tag{7}$$

$$\mathbf{z}_i^{(l)} = (\mathbf{1} - \mathbf{g}_i^{(l)}) \odot \mathbf{z}_i^{(l-1)} + \mathbf{g}_i^{(l)} \odot \widetilde{\mathbf{z}}_i^{(l)}. \tag{8}$$

Here, $\widetilde{\mathbf{z}}_i^{(l)}$ captures the topology adjustment suggested by the current context, while $\mathbf{g}_i^{(l)}$ controls how this adjustment is integrated with the preceding representation. The resulting $\mathbf{z}_i^{(l)}$ therefore coordinates the task-level topology preference with the structure already realized in the current prefix.

The adaptive topology representation subsequently conditions agent generation. Following the autoregressive formulation in Section 2.2, we maintain a recurrent state that integrates the task representation, the current partial graph, and the topology representation:

$$\mathbf{h}_{i,l}^N = \mathrm{GRU}\left([\mathbf{e}_i^Q; \mathbf{h}_{\mathcal{G}}^{(l)}; \mathbf{z}_i^{(l)}], \mathbf{h}_{i,l-1}^N \right). \tag{9}$$

Each candidate role $r \in \mathcal{R}$ is encoded from its textual description using the same pretrained sentence transformer, yielding a semantic role representation $\mathbf{e}_r^R \in \mathbb{R}^{d_R}$. Together with a learnable STOP representation, these form the augmented role set $\mathcal{R}^+ = \mathcal{R} \cup \{\textsc{Stop}\}$. We compute the compatibility between the current node state and each candidate role as:

$$p_\theta\left(r_l = r \mid q_i, \mathcal{G}_{<l}, \mathbf{z}_i^{(l)} \right) = \underset{r \in \mathcal{R}^+}{\mathrm{Softmax}}\left(\left\langle \mathbf{W}_N \mathbf{h}_{i,l}^N, \mathbf{e}_r^R \right\rangle / \sqrt{d_R} \right). \tag{10}$$

The role prediction therefore jointly reflects task semantics, the current collaboration context, and the adaptive topology representation, while STOP determines when node generation terminates.

After generating r_l, TopoPAG uses the same topology representation to determine how the new agent connects to the existing graph. We construct an edge query from the generated role and the current topology representation, and compare it with the representation of each previous agent:

$$\mathbf{q}_{i,l}^E = \mathrm{MLP}_E\left([\mathbf{z}_i^{(l)}; \mathbf{e}_{r_l}^R] \right), \quad \mathbf{k}_{i,m}^E = \mathbf{W}_E \mathbf{h}_m^{(l)}, \tag{11}$$

$$s_{iml}^E = \mathrm{MLP}_{\mathrm{edge}}\left([\mathbf{q}_{i,l}^E; \mathbf{k}_{i,m}^E; \mathbf{q}_{i,l}^E \odot \mathbf{k}_{i,m}^E] \right), \tag{12}$$

$$p_\theta\left(e_{ml} = 1 \mid r_l, q_i, \mathcal{G}_{<l}, \mathbf{z}_i^{(l)} \right) = \sigma\left(s_{iml}^E \right), \quad m < l. \tag{13}$$

Each candidate edge is modeled independently, allowing the new agent to receive information from multiple predecessors. In this way, the adaptive topology representation consistently guides both role selection and communication-link generation, translating task-level topology preference into fine-grained graph construction.

3.3 Training Objective and Inference

The adaptive topology representation introduced in Section 3.2 evolves with the partially generated graph and directly guides subsequent role and communication-link generation. To anchor this evolving representation to the topology space defined in Section 3.1, we use the structural coordinate of each training anchor as supervision. Specifically, when $\mathcal{G}_t$ is generated under teacher forcing for task q_i, we encourage its terminal topology representation to match the corresponding anchor coordinate:

$$\mathcal{L}_{\text{topo}} = \frac{1}{|\mathcal{D}|} \sum_{(i,t)\in\mathcal{D}} \left\| \mathbf{z}_{it}^{(L_t)} - \mathbf{z}_t \right\|_2^2, \tag{14}$$

where $\mathcal{D}$ denotes the set of training task–anchor pairs and L_t is the generation length of $\mathcal{G}_t$. Moreover, the anchor executions provide a natural supervision signal for weighting candidate topologies during generator training. Rather than treating all anchor graphs equally, we use the execution-derived preference $p_i^*(t)$ from Equation 4 to weight their generation likelihood:

$$\mathcal{L}_{\text{gen}} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} p_i^*(t) \log p_\theta \left(\mathcal{G}_t \mid q_i \right), \tag{15}$$

where N denotes the number of training tasks. Following the autoregressive factorization in Section 2.2, the graph likelihood is decomposed into role and edge prediction terms. Therefore, all evaluated anchors contribute to training, while anchors with higher execution utility receive larger weights. This formulation transfers the relative topology preference revealed by comparative executions into the graph-generation objective. Combining topology preference learning, representation grounding, and preference-weighted graph generation, the overall objective is formulated as:

$$\mathcal{L} = \lambda_{\text{pref}} \mathcal{L}_{\text{pref}} + \lambda_{\text{topo}} \mathcal{L}_{\text{topo}} + \lambda_{\text{gen}} \mathcal{L}_{\text{gen}}, \tag{16}$$

where λ_{pref}, λ_{topo}, and λ_{gen} are hyperparameters that controls the relative contributions.

As shown in Appendix B, given an unseen query q, TopoPAG predicts its preference distribution over the structural anchors and initializes the topology representation according to Equation 6. Starting from an empty graph, it iteratively updates this representation and generates agent roles with their incoming links. The anchors are not executed during inference; instead, their topology coordinates serve as structural references for initializing the task-specific representation.

4 Experiments

4.1 Experimental Setup

Datasets and Metrics. We evaluate TopoPAG on six benchmarks. MMLU (Hendrycks et al., 2021) evaluates general reasoning; GSM8K (Cobbe et al., 2021), MultiArith (Roy & Roth, 2015), SVAMP (Patel et al., 2021), and AQuA (Ling et al., 2017) assess mathematical reasoning; and HumanEval (Chen et al., 2021) measures code generation. We report accuracy for all reasoning benchmarks and Pass@1 for HumanEval. Dataset statistics are provided in Appendix C.

Baselines. To evaluate the effectiveness of TopoPAG, we compare it against representative baselines from three categories: **1) single-agent reasoning methods**, including Vanilla, Chain-of-Thought (CoT) (Wei et al., 2022), and Self-Consistency (SC) (Wang et al., 2022); **2) fixed-topology MASs**, including Chain, Tree, Complete, Random(Qian et al., 2025), and LLM-Debate (Du et al., 2024); and **3) learnable topology methods**, including AgentPrune (Zhang et al., 2025a), AgentDropout (Wang et al., 2025b), G-Designer (Zhang et al., 2025b), ARG-Designer (Li et al., 2026a), and OFA-MAS (Li et al., 2026b). Detailed baseline configurations are provided in Appendix D.

Implementation Details. Our main experiments primarily use GPT-4o (gpt-4o-2024-08-06) through the OpenAI API, while additional backbone evaluations are conducted with GPT-4o-mini and GPT-5.4-nano. For all multi-agent baseline methods, a summarizer agent produces the final response after $K = 3$ interaction rounds. We adopt Sentence-BERT (all-MiniLM-L6-v2) (Wang et al., 2020) as the node encoder, with an embedding dimension of $D = 384$. Following common settings in LLM-based MAS (Zhuge et al., 2024; Zhang et al., 2025b; Li et al., 2026a), explicit agent profiles generated are used for all multi-agent methods. We sample $B \in \{40, 60\}$ training queries from each dataset. Further implementation details are reported in Appendix E.

Table 1: Performance comparison (%) on six benchmarks. The best results are highlighted in bold.

Method	MMLU	GSM8K	AQuA	MultiArith	SVAMP	HumanEval	Average
Vanilla	80.39	82.30	71.06	93.09	86.55	71.39	80.80
CoT	81.69 ↑1.30	86.50 ↑4.20	73.58 ↑2.52	93.25 ↑0.16	87.36 ↑0.81	74.67 ↑3.28	82.84 ↑2.04
SC (CoT)	83.66 ↑3.27	81.60 ↓0.70	75.63 ↑4.57	94.12 ↑1.03	88.59 ↑2.04	79.83 ↑8.44	83.91 ↑3.11
Chain	83.01 ↑2.62	88.30 ↑6.00	74.05 ↑2.99	93.27 ↑0.18	87.17 ↑0.62	81.37 ↑9.98	84.53 ↑3.73
Tree	81.04 ↑0.65	85.20 ↑2.90	71.23 ↑0.17	93.68 ↑0.59	88.91 ↑2.36	80.53 ↑9.14	83.43 ↑2.63
Complete	82.35 ↑1.96	80.10 ↓2.20	72.95 ↑1.89	94.53 ↑1.44	84.01 ↓2.54	79.03 ↑7.64	82.16 ↑1.36
Random	84.31 ↑3.92	86.90 ↑4.60	76.48 ↑5.42	94.08 ↑0.99	87.54 ↑0.99	82.66 ↑11.27	85.33 ↑4.53
LLM-Debate	84.96 ↑4.57	91.40 ↑9.10	77.65 ↑6.59	96.36 ↑3.27	90.11 ↑3.56	84.70 ↑13.31	87.53 ↑6.73
AgentPrune	85.07 ↑4.68	91.10 ↑8.80	80.51 ↑9.45	94.65 ↑1.56	90.58 ↑4.03	86.75 ↑15.36	88.11 ↑7.31
AgentDropout	85.62 ↑5.23	91.70 ↑9.40	80.94 ↑9.88	95.60 ↑2.51	91.04 ↑4.49	85.98 ↑14.59	88.48 ↑7.68
G-designer	86.13 ↑5.74	92.84 ↑10.54	81.40 ↑10.34	96.38 ↑3.29	92.76 ↑6.21	88.02 ↑16.63	89.59 ↑8.79
ARG-Designer	85.40 ↑5.01	93.15 ↑10.85	83.64 ↑12.58	98.13 ↑5.04	94.33 ↑7.78	90.08 ↑18.69	90.79 ↑9.99
OFA-MAS	86.93 ↑6.54	93.33 ↑11.03	82.07 ↑11.01	98.08 ↑4.99	93.65 ↑7.10	90.11 ↑18.72	90.70 ↑9.90
TopoPAG	**88.24 ↑7.85**	**93.84 ↑11.54**	**87.38 ↑16.32**	**99.46 ↑6.37**	**96.15 ↑9.60**	**93.11 ↑21.72**	**93.03 ↑12.23**

4.2 Overall Performance and Efficiency

Performance Comparison. Table 1 reports the performance of TopoPAG against single-agent, fixed-topology, and learnable-topology baselines across six benchmarks. Fixed communication patterns exhibit substantial variation across tasks, with some configurations even falling below the single-agent baseline, suggesting that no predefined structure is consistently effective for different reasoning requirements. Learnable topology methods alleviate this rigidity by adapting communication to individual tasks, with ARG-Designer and OFA-MAS serving as the strongest existing competitors. TopoPAG, however, consistently surpasses both methods and achieves the best performance on all six benchmarks. It improves over ARG-Designer and OFA-MAS by 2.24 and 2.33 percentage points on average, respectively. The gains are observed across general reasoning, mathematical reasoning, and code generation rather than being driven by a single benchmark, with particularly pronounced improvements of 3.74 points on AQuA and 3.00 points on HumanEval over the previous best results. Overall, these results demonstrate that TopoPAG consistently produces more effective task-adaptive communication structures across diverse reasoning settings.

Token Efficiency. Figure 3 compares reasoning performance and token cost on MMLU and GSM8K. On GSM8K, TopoPAG achieves the highest accuracy with the fewest prompt tokens among all methods. On MMLU, it again attains the best accuracy while using substantially fewer tokens than communication-intensive baselines such as LLM-Debate and Complete. Although some learned-topology methods are more token-efficient on MMLU, their lower accuracy indicates that reducing communication alone does not guarantee effective collaboration. Overall, TopoPAG achieves a favorable accuracy–efficiency trade-off, showing that its gains do not rely on increased interaction. Instead, its adaptive topology generation selectively determines agent participation and communication links based on the evolving structural preference, reducing unnecessary communication while preserving effective information flow.

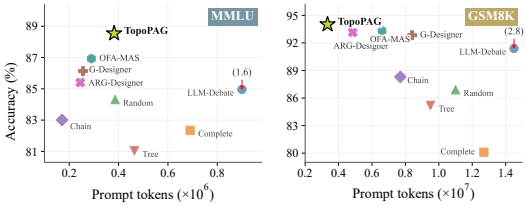


Figure 3: Trade-off between accuracy and token consumption on MMLU and GSM8K.

4.3 Ablation and Robustness

Ablation Study. We evaluate four TopoPAG variants: **w/o Preference** removes task-specific topology preference learning; **w/o Adaptation** disables dynamic topology-state updates; **w/o Grounding** removes the auxiliary alignment between the terminal topology representation and its corresponding anchor; and **w/o Weighting** replaces preference-weighted graph imitation with uniform weighting. As shown in Ta-

Table 2: Ablation study of the key components.

Model	MMLU	GSM8K	HumanEval	Avg.
Vanilla	80.39	82.30	71.39	78.03
TopoPAG	**88.24**	**93.84**	**93.11**	**91.73**
w/o Preference	85.78	92.20	92.23	90.07
w/o Adaptation	84.97	91.88	91.36	89.40
w/o Grounding	87.43	93.04	92.42	90.96
w/o Weighting	85.62	91.45	91.95	89.67

ble 2, all variants consistently underperform the full model across the three benchmarks, confirming the complementary roles of these components. Removing adaptation causes the largest average drop of 2.33 percentage points, while removing preference learning leads to a 1.66-point decline, highlighting the importance of both inferring task-specific structural preferences and adapting them to the partially generated graph. Preference-weighted imitation contributes a 2.06-point improvement by emphasizing higher-utility topologies, whereas topology grounding provides a smaller but consistent gain by preserving the structural semantics of the evolving representation. Overall, these results support the full TopoPAG design: execution-derived preferences provide a structural prior, adaptive updates translate it into sequential generation decisions, and the auxiliary objectives maintain alignment with informative execution feedback.

Robustness Analysis. Following Zhuge et al. (2024), we evaluate robustness to a system-prompt attack by replacing a key agent role with an adversarial prompt that injects misleading information during reasoning. As shown in Figure 4, fixed topologies such as Complete, Debate, Tree, and Random suffer substantial accuracy drops, while learnable topology methods are generally more resilient but remain affected by corrupted information. TopoPAG exhibits the smallest degradation among all methods, dropping only 2.5 percentage points from 88.2% to 85.7%, while retaining the highest post-attack accuracy. Its degradation is also smaller than that of strong adaptive baselines, including OFA-MAS, ARG-Designer, and G-Designer. These results indicate that TopoPAG is less sensitive to misleading information from compromised agents, consistent with its preference-guided adaptive generation, which continuously adjusts topology-level guidance as the collaboration evolves.

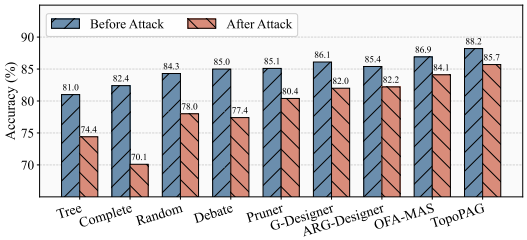


Figure 4: Robustness of different communication topology methods under prompt attacks.

4.4 Performance Across LLM Backbones

Since collaboration topology can interact with the capability of the underlying LLM, we examine whether TopoPAG generalizes beyond the GPT-4o backbone used in our main experiments. We evaluate GPT-5.4-nano and GPT-4o-mini on MMLU, AQuA, and HumanEval, representing general reasoning, mathematical reasoning, and code generation, respectively, and compare TopoPAG against Vanilla, ARG-Designer, and OFA-MAS. As shown in Table 3, TopoPAG consistently achieves the best performance across both backbones and all three benchmarks. With GPT-5.4-nano, it reaches an average score of 90.93%, outperforming the strongest competing topology method by 3.44 percentage points. With GPT-4o-mini, TopoPAG improves the best baseline from 83.83% to 87.15%, a gain of 3.32 points. These improvements span reasoning, mathematics, and coding, indicating that execution-derived topology preferences provide effective structural guidance across different LLM backbones. Overall, preference-guided topology adaptation remains beneficial as backbone capability changes, suggesting that it complements rather than depends on a particular LLM.

4.5 Case Study

Appendix F presents a HumanEval case study showing how TopoPAG turns task-specific structural preferences into adaptive generation guidance. Despite sharing the same role pool, the two tasks favor markedly different structures: the Collatz task prefers compact Chain-2/3 and Star-3 topologies, whereas the minimum subarray-sum task favors denser Mesh-4 and fully connected variants. Rather than selecting an anchor directly, TopoPAG projects the predicted preference distribution into a continuous topology representation that serves as the initial structural prior. For the Collatz task, this representation starts near Chain-2 and progressively moves toward an FC-3-like structure as a *Bug Fixer* is introduced and new communication links are added. With the same task and initial preference, the w/o Adaptation variant remains near the initial Chain-2 structure and fails, whereas the full model adapts toward FC-3 and succeeds. This contrast illustrates how TopoPAG translates a global structural preference into evolving role and communication decisions during generation.

Table 3: Performance comparison (%) across backbone LLMs. Best results are shown in bold.

GPT-5.4-nano

Method	MMLU	AQuA	HumanEval	Avg.
Vanilla	77.12	82.71	87.60	82.48
ARG-Designer	84.31	85.98	88.43	86.24
OFA-MAS	90.20	82.91	89.36	87.49
TopoPAG	**92.03**	**87.38**	**93.39**	**90.93**

GPT-4o-mini

Method	MMLU	AQuA	HumanEval	Avg.
Vanilla	73.47	64.02	71.10	69.53
ARG-Designer	79.61	81.68	84.73	82.01
OFA-MAS	81.70	83.28	86.52	83.83
TopoPAG	**84.31**	**84.58**	**92.56**	**87.15**

5 Related Work

Topology Design for Multi-Agent Systems. Communication topology governs information flow among LLM agents and critically affects reasoning quality and collaboration efficiency (Shen et al., 2025; Zheng et al., 2026). Existing topology design methods broadly fall into two categories: structure modification and structure synthesis (Tran et al., 2025; Zhou et al., 2026). Structure modification starts from predefined interaction patterns, such as chains, trees, or debate graphs (Du et al., 2024; Wang et al., 2025a), and removes redundant components. Representative methods include AgentPrune (Zhang et al., 2025a), which prunes unnecessary communication, and AgentDropout (Wang et al., 2025b), which dynamically suppresses redundant agents and edges. In contrast, structure synthesis directly constructs task-dependent communication graphs. G-Designer (Zhang et al., 2025b) predicts graph structures with GNNs, while ARG-Designer (Li et al., 2026a) autoregressively generates agent roles and communication links. Recent approaches further incorporate diverse structural signals, including information-flow patterns in EIB-Learner (Shen et al., 2025), guided graph diffusion in GTD (Jiang et al., 2026), and mixture-of-experts generation in OFA-MAS (Li et al., 2026b). Despite different optimization strategies (Sheng et al., 2022; Wang et al., 2026), these methods mainly address *how* to construct or refine communication graphs, while explicitly modeling *which* structures are preferable for a given task remains underexplored. TopoPAG addresses this gap by learning task-specific structural preferences from comparative topology executions and embedding them into a continuous topology space to guide subsequent generation.

Autoregressive Graph Generation. This paradigm factorizes graph construction into sequential node and edge predictions conditioned on the generated prefix (Guo & Zhao, 2022; Chen et al., 2026; Zhu et al., 2022). Although computationally tractable, such local generation may not adequately capture the global structure of the completed graph. Existing methods mainly improve the generation process itself: GraphRNN (You et al., 2018) sequentially generates nodes and edges, GRAN (Liao et al., 2019) models larger graph blocks with graph attention, and BiGG (Dai et al., 2020) exploits sparsity for scalability. Orderless Regularization (Cohen-Karlik et al., 2024) and PARD (Zhao et al., 2024) reduce sensitivity to generation order, while G2PT (Chen et al., 2025a) formulates graph generation as Transformer-based next-token prediction. In LLM-based multi-agent systems, ARG-Designer (Li et al., 2026a) applies this paradigm to autoregressively generate agent roles and communication links. Despite these advances, existing methods primarily focus on how local generation decisions are parameterized or ordered, while the desired global topology is rarely modeled as an explicit state that evolves during construction (Shi et al., 2020; Goyal et al., 2020). TopoPAG bridges this global-to-local gap by initializing a continuous topology state from task-specific execution preferences and progressively updating it with the generated prefix, thereby conditioning subsequent role and edge decisions on an evolving global structural representation.

6 Conclusion

We presented TopoPAG, a topology preference-guided framework for task-adaptive communication graph generation in LLM-based multi-agent systems. By learning task-specific topology preferences from comparative executions and embedding them into a continuous topology space, TopoPAG bridges graph-level execution feedback with fine-grained autoregressive topology decisions. The learned preference initializes a topology representation that is progressively refined during graph construction to guide role and communication-link generation, while preference-weighted imitation further leverages the relative utility of evaluated topologies. Extensive experiments on six benchmarks demonstrate consistent improvements over baselines in effectiveness and communication efficiency, with ablation, robustness, and case studies further validating the proposed design.

AI USE STATEMENT

In this work, we used generative AI tools solely to aid and polish the writing of the manuscript. We have not used generative AI tools for any other research-related tasks, including generating scientific claims, experimental results, or technical content. All AI-assisted text has been reviewed and verified by the authors. We take responsibility for the final content of this work.

ETHICS STATEMENT

This work does not involve human subjects, collection of private or sensitive data, or the release of new datasets. We conduct experiments on publicly available benchmarks and follow their corresponding usage policies. Although our method automatically generates communication topologies for LLM-based multi-agent systems, it is designed for improving task-solving efficiency and does not target applications involving high-risk decision making or sensitive domains. We are not aware of any additional ethical concerns arising from this work.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the proposed TopoPAG framework, training objectives, and inference procedure in Section 3. The experimental settings, benchmark datasets, baseline configurations, and implementation details are described in Section 4 and the Appendix. We will release the source code, experiment scripts, and configuration files to facilitate reproduction of our results.

REFERENCES

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Xiaohui Chen, Yinkai Wang, Jiaxing He, Yuanqi Du, Soha Hassoun, Xiaolin Xu, and Li-Ping Liu. Graph generative pre-trained transformer. In *Proceedings of the 42nd International Conference on Machine Learning*, ICML'25. JMLR.org, 2025a.

Yong Chen, Li Li, Wei Peng, and Songzhi Su. Vorec: Enhancing recommendation with voronoi diagram in hyperbolic space. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2081–2091, 2025b.

Yong Chen, Li Li, Nannan Zong, Zhihui Liu, and Song-Zhi Su. Ardiff: Anisotropic residual diffusion for heterogeneous graph learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 14592–14600, 2026.

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

Edo Cohen-Karlik, Eyal Rozenberg, and Daniel Freedman. Overcoming order in autoregressive graph generation. *arXiv preprint arXiv:2402.03387*, 2024.

Hanjun Dai, Azade Nazi, Yujia Li, Bo Dai, and Dale Schuurmans. Scalable deep generative modeling for sparse graphs. In *International conference on machine learning*, pp. 2302–2312. PMLR, 2020.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Nikhil Goyal, Harsh Vardhan Jain, and Sayan Ranu. Graphgen: A scalable approach to domain-agnostic labeled graph generation. In *Proceedings of the web conference 2020*, pp. 1253–1263, 2020.

Xiaojie Guo and Liang Zhao. A systematic survey on deep generative models for graph generation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5370–5390, 2022.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring Massive Multitask Language Understanding. In *International Conference on Learning Representations*, 2021. URL https://mlanthology.org/iclr/2021/hendrycks2021iclr-measuring/.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Steven Yau, Zijuan Lin, Liyang Zhou, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, volume 2024, pp. 23247–23275, 2024.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJlWWJSFDH.

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 *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 38042–38060, 2026.

Hui Yi Leong, Yuheng Li, Yuqing Wu, Wenwen Ouyang, Wei Zhu, Jiechao Gao, and Wei Han. Amas: Adaptively determining communication topology for llm-based multi-agent system. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 2061–2070, 2025.

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, 2026a.

Shiyuan Li, Yixin Liu, Yu Zheng, Mei Li, Quoc Viet Hung Nguyen, and Shirui Pan. Ofa-mas: One-for-all multi-agent system topology design based on mixture-of-experts graph generative models. In *Proceedings of the ACM Web Conference 2026*, pp. 1333–1344, 2026b.

Xinyi Li, Sai Wang, Siqi Zeng, Yu Wu, and Yi Yang. A survey on llm-based multi-agent systems: workflow, infrastructure, and challenges. *Vicinagearth*, 1(1):9, 2024.

Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, Will Hamilton, David K Duvenaud, Raquel Urtasun, and Richard Zemel. Efficient graph generation with graph recurrent attention networks. *Advances in neural information processing systems*, 32, 2019.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 158–167, 2017.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? In *Proceedings of the 2021 conference of the North American chapter of the association for computational linguistics: human language technologies*, pp. 2080–2094, 2021.

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 *International Conference on Learning Representations*, volume 2025, pp. 41488–41505, 2025.

Subhro Roy and Dan Roth. Solving general arithmetic word problems. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pp. 1743–1752, 2015.

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.

Xu Shen, Yixin Liu, Yiwei Dai, Yili Wang, Rui Miao, Yue Tan, Shirui Pan, and Xin Wang. Understanding the information propagation effects of communication topologies in llm-based multi-agent systems. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 12358–12372, 2025.

Junjie Sheng, Xiangfeng Wang, Bo Jin, Junchi Yan, Wenhao Li, Tsung-Hui Chang, Jun Wang, and Hongyuan Zha. Learning structured communication for multi-agent reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 36(2):50, 2022.

Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. GraphAF: A Flow-Based Autoregressive Model for Molecular Graph Generation. In *International Conference on Learning Representations*, 2020. URL https://mlanthology.org/iclr/2020/shi2020iclr-graphaf/.

Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O'Sullivan, and Hoang D Nguyen. Multi-agent collaboration mechanisms: A survey of llms. *arXiv preprint arXiv:2501.06322*, 2025.

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 *International Conference on Learning Representations*, volume 2026, pp. 113586–113613, 2026.

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.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Yuntao Wang, Shaolong Guo, Yanghe Pan, Zhou Su, Fahao Chen, Tom H Luan, Peng Li, Jiawen Kang, and Dusit Niyato. Internet of agents: Fundamentals, applications, and challenges. *IEEE Transactions on Cognitive Communications and Networking*, 2025a.

Zhexuan Wang, Yutong Wang, Xuebo Liu, Liang Ding, Miao Zhang, Jie Liu, and Min Zhang. Agentdropout: Dynamic agent elimination for token-efficient and high-performance llm-based multi-agent collaboration. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 24013–24035, 2025b.

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.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.

Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, pp. 5708–5717. PMLR, 2018.

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 *International Conference on Learning Representations*, volume 2025, pp. 75389–75428, 2025a.

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 *Proceedings of the 42nd International Conference on Machine Learning*, ICML'25. JMLR.org, 2025b.

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 *International Conference on Learning Representations*, volume 2025, pp. 34040–34077, 2025c.

Lingxiao Zhao, Xueying Ding, and Leman Akoglu. Pard: Permutation-invariant autoregressive diffusion for graph generation. *Advances in Neural Information Processing Systems*, 37:7156–7184, 2024.

Yujia Zheng, Zhuokai Zhao, Zijian Li, Yaqi Xie, Mingze Gao, Lizhu Zhang, and Kun Zhang. Thought communication in multiagent collaboration. *Advances in Neural Information Processing Systems*, 38:123389–123418, 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 *International Conference on Learning Representations*, volume 2026, pp. 15844–15872, 2026.

Yanqiao Zhu, Yuanqi Du, Yinkai Wang, Yichen Xu, Jieyu Zhang, Qiang Liu, and Shu Wu. A survey on deep graph generation: Methods and applications. In *Learning on graphs conference*, pp. 47–1. PMLR, 2022.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. Gptswarm: language agents as optimizable graphs. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

A Structural Topology Descriptors

Each structural anchor is characterized by $D = 15$ graph descriptors, whose complete definitions are provided in Table 4. These descriptors capture six complementary aspects of graph structure: scale and connectivity, hierarchy, information flow, degree heterogeneity, centralization, and path-based reachability. All descriptors depend solely on graph structure and are independent of task semantics, agent-role identities, and execution outcomes. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a DAG, with $n = |\mathcal{V}|$, $m = |\mathcal{E}|$, and node-wise in- and out-degrees d_v^- and d_v^+, respectively. We construct the topological layers $\{\mathcal{L}_j\}_{j=1}^{J}$ recursively, where $\mathcal{L}_1$ contains the source nodes and each subsequent layer contains the source nodes of the remaining graph after removing all preceding layers. We further denote by $\mathcal{G}^u$ the undirected projection of $\mathcal{G}$ and by $\mathrm{TC}(\mathcal{G}) = (\mathcal{V}, \mathcal{E}_{\mathrm{TC}})$ its transitive closure. Before constructing the continuous topology space in Equation 2, each descriptor dimension is standardized across the T structural anchors using statistics computed solely from the anchor set. Therefore, the structural anchor coordinates encode only topology-level variation, while task-specific information is introduced through the learned topology preference.

B Inference Procedure

Algorithm 1 details the inference procedure of TopoPAG. The predicted anchor preference is first converted into an initial topology state through preference-weighted aggregation. Starting from an empty graph, the model iteratively generates agent roles and communication links conditioned on the current state and generated prefix, followed by topology-state updates after each step. Generation terminates upon producing STOP or reaching the predefined agent limit.

C Data Statistics

Table 5 summarizes the datasets used in our experiments, including task categories, answer formats, evaluation metrics, test set sizes, and licensing information. We briefly describe each dataset below.

MMLU is a large-scale multiple-choice benchmark covering 57 subjects across humanities, STEM, and social sciences, and is used to evaluate broad knowledge and reasoning abilities.

GSM8K is a grade-school mathematical reasoning benchmark consisting of multi-step arithmetic word problems, where models are required to produce the final numerical answer.

Table 4: Topology descriptors used to construct the structural anchor space.

Category	Descriptor	Definition
Scale & Connectivity	Number of nodes	n
	Edge density	$\frac{m}{n(n-1)}$
Hierarchy	Depth	Longest path length
	Maximum width	$\max_j \lvert\mathcal{L}_j\rvert$
Information Flow	Source fraction	$\frac{1}{n}\sum_v \mathbb{I}[d_v^- = 0]$
	Sink fraction	$\frac{1}{n}\sum_v \mathbb{I}[d_v^+ = 0]$
	Branching fraction	$\frac{1}{n}\sum_v \mathbb{I}[d_v^+ > 1]$
	Merging fraction	$\frac{1}{n}\sum_v \mathbb{I}[d_v^- > 1]$
Degree Heterogeneity	In-degree deviation	$\mathrm{Std}_v(d_v^-)$
	Out-degree deviation	$\mathrm{Std}_v(d_v^+)$
Centralization	In-centralization	$\frac{\sum_v(\max_u d_u^- - d_v^-)}{\max\{(n-1)^2, 1\}}$
	Out-centralization	$\frac{\sum_v(\max_u d_u^+ - d_v^+)}{\max\{(n-1)^2, 1\}}$
Path & Reachability	Mean undirected path	$\mathrm{ASPL}(\mathcal{G}^u)$
	Undirected diameter	$\mathrm{diam}(\mathcal{G}^u)$
	Transitive-closure gap	$\frac{\lvert\mathcal{E}_{\mathrm{TC}}\rvert - m}{\max\{\lvert\mathcal{E}_{\mathrm{TC}}\rvert, 1\}}$

Algorithm 1: The Inference Algorithm of TopoPAG.

Input: Task query Q, trained model P_θ, anchor $\{\mathbf{z}_t\}_{t=1}^T$.
Output: Collaboration graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

Predict topology preference $p_\theta(t \mid Q)$;
Initialize $\mathbf{z}^{(0)} \leftarrow \sum_{t=1}^T p_\theta(t \mid Q)\mathbf{z}_t$;
Initialize $\mathcal{V} \leftarrow \emptyset, \mathcal{E} \leftarrow \emptyset, i \leftarrow 1$;
while $i \le N_{\max}$ **do**
  Sample agent role $r_i \sim P_\theta(v_i \mid \mathcal{G}_{<i}, Q, \mathbf{z}^{(i-1)})$;
  if $r_i = \textsc{Stop}$ **then**
    break;
  $\mathcal{V} \leftarrow \mathcal{V} \cup \{v_i\}$;
  for $j = i-1$ **to** 1 **do**
    Sample edge $b_{ji} \sim P_\theta(e_{ji} \mid v_i, \mathcal{G}_{<i}, Q, \mathbf{z}^{(i-1)})$;
    if $b_{ji} = 1$ **then**
      $\mathcal{E} \leftarrow \mathcal{E} \cup \{e_{ji}\}$;
  Update topology representation $\mathbf{z}^{(i)} \leftarrow \mathrm{TopoUpdate}_\theta(\mathbf{z}^{(i-1)}, Q, \mathcal{G}_{\le i})$;
  $i \leftarrow i + 1$;
return $\mathcal{G} = (\mathcal{V}, \mathcal{E})$;

MultiArith contains elementary arithmetic word problems with varying numbers of operations, focusing on basic compositional mathematical reasoning.

SVAMP is a challenge set for arithmetic reasoning that introduces variations in problem wording and structure to test robustness beyond standard templates.

Table 5: Dataset statistics and details.

Category	Dataset	Answer Type	Metric	#Test	License
General reasoning	MMLU	Multi-choice	Acc.	153	MIT License
Math reasoning	GSM8K	Number	Acc.	1,319	MIT License
	MultiArith	Number	Acc.	600	Unspecified
	SVAMP	Number	Acc.	1,000	MIT License
	AQuA	Multi-choice	Acc.	254	Apache-2.0
Code generation	HumanEval	Code	Pass@1	164	MIT License

AQuA is a multiple-choice algebraic reasoning benchmark that requires solving quantitative problems through multi-step reasoning.

HumanEval is a code generation benchmark containing programming problems with function-level specifications, where generated solutions are evaluated by execution-based correctness.

The detailed statistics and licensing information are provided in Table 5.

D Baseline Details

Single-Agent Reasoning Methods. We consider three representative single-agent baselines: (1) **Vanilla**: directly prompts a single LLM to solve the task without additional reasoning enhancement; (2) **Chain-of-Thought (CoT)** (Wei et al., 2022): prompts the LLM to generate intermediate reasoning steps before producing the final answer; and (3) **Self-Consistency (SC)** (Wang et al., 2022): samples multiple CoT reasoning trajectories and selects the final answer by majority voting.

Fixed-Topology Multi-Agent Systems. We evaluate five manually specified collaboration structures: (1) **Chain**: agents communicate sequentially along a directed chain; (2) **Tree**: agents are organized hierarchically, with information progressively aggregated toward downstream agents; (3) **Complete**: every preceding agent can communicate with every subsequent agent, enabling dense information exchange; (4) **Random** (Qian et al., 2025): constructs a random communication topology among the participating agents; and (5) **LLM-Debate** (Du et al., 2024): allows multiple agents to iteratively exchange and critique their reasoning over several rounds.

Learnable Topology Methods. We compare TopoPAG with five representative methods that learn or optimize multi-agent communication structures: (1) **AgentPrune** (Zhang et al., 2025a): removes redundant communication connections from multi-agent interaction graphs to improve collaboration efficiency; (2) **AgentDropout** (Wang et al., 2025b): dynamically suppresses redundant agents and communication links during multi-agent collaboration; (3) **G-Designer** (Zhang et al., 2025b): predicts task-dependent communication graphs using a graph-based topology designer; (4) **ARG-Designer** (Li et al., 2026a): autoregressively constructs MAS by sequentially generating agent roles and communication links; and (5) **OFA-MAS** (Li et al., 2026b): learns a unified topology generator across task domains using a mixture-of-experts architecture.

E Implementation Details

TopoPAG constructs its continuous structural space from 18 topology anchors described by 15 topology-level statistics. We select the number of retained principal components from $K \in \{4, 8\}$. The hidden dimensions of the graph encoder, topology representation, edge predictor, and recurrent node decoder are selected from $\{64, 128, 256\}$. We train the topology generator using Adam for $E \in \{50, 100, 200, 300\}$ epochs and search the learning rate from $1e^{-5}$ to $1e^{-3}$. The utility temperature is selected from $\{0.01, 0.1, 0.3, 1.0\}$, the structural-cost coefficient from $\{0.1, 0.3, 0.5\}$, and the auxiliary loss weights from $\{0.1, 0.5, 1.0, 1.5\}$. Hyperparameters and checkpoints are selected according to the total validation objective under a task-level $90\%/10\%$ training–validation split.

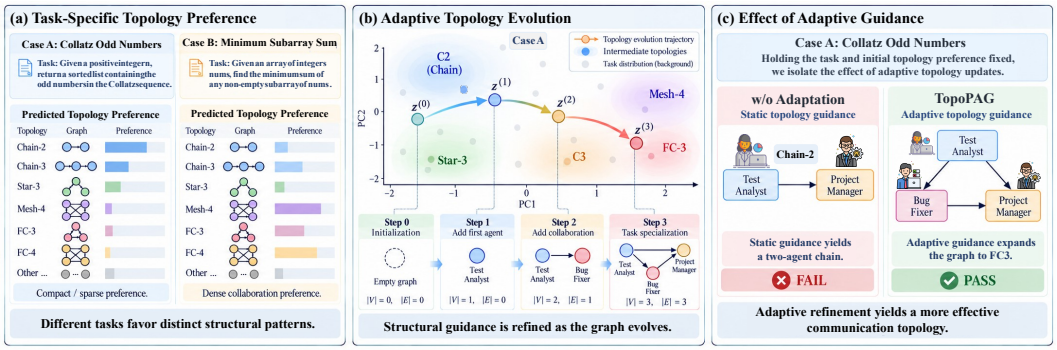


Figure 5: Case study of TopoPAG on HumanEval. (a) Topology preferences for two representative tasks. (b) Topology evolution with the generated graph prefixes. (c) TopoPAG versus w/o Adaptation under the same task and initial topology preference.

F Case Study

We conduct the case study on HumanEval to examine how TopoPAG transforms task-specific topology preferences into adaptive structural guidance for graph generation. Figure 5(a) contrasts two tasks drawn from the same role pool but exhibiting markedly different topology preferences. The Collatz task favors compact structures, including Chain-2, a two-agent chain, together with Chain-3 and Star-3. By contrast, the minimum subarray-sum task assigns greater preference to more connected structures such as Mesh-4, FC-4, and FC-3, where FC-3 denotes a three-agent fully connected (complete) DAG. The contrast indicates that the structural guidance learned by TopoPAG varies with the collaboration demands of the task, providing a task-conditioned starting point for subsequent topology generation.

Figure 5(b) traces the evolution of this guidance during generation for the Collatz case. The predicted topology preference initializes $z^{(0)}$, which is progressively updated as the collaboration graph grows, forming the trajectory $z^{(0)} \to z^{(1)} \to z^{(2)} \to z^{(3)}$. The early states remain near the Chain-2 region, matching the compact structure induced by the initial agents. As additional agents and links are introduced, the representation shifts toward the FC-3 region, reflecting a transition to a more connected coordination pattern. This evolving topology state conditions subsequent role and communication-link predictions, allowing task-level structural preferences to be reconciled with the partially constructed graph. Thus, TopoPAG transforms static topology preference into a dynamic structural signal that guides the remaining generation process.

Figure 5(c) further demonstrates the role of adaptive guidance through the **w/o Adaptation** variant. Given the same task and initial topology preference, **w/o Adaptation** preserves a static topology representation and remains at a Chain-2 structure, *Test Analyst*! $\to$!*Project Manager*, which fails to solve the task. In contrast, TopoPAG progressively shifts the topology representation toward an FC-3-like coordination pattern, guiding the generator to introduce a *Bug Fixer* and establish additional communication links among agents. The resulting FC-3 topology successfully solves the task. Together, these cases reveal the full trajectory from task-specific structural preference, through adaptive topology evolution, to concrete agent composition and communication decisions.

G Limitations

We identify the following limitations of our proposed TopoPAG: Although our experiments cover diverse reasoning and coding benchmarks, they primarily focus on static task settings with bounded interaction horizons. Extending evaluation to long-horizon scenarios, dynamically evolving environments, heterogeneous tool-augmented agents, and larger agent populations would provide a more comprehensive assessment of the generalization and scalability of topology adaptation. More broadly, how communication topologies should continuously adapt to changing task distributions, agent capabilities, and resource constraints remains an important direction for future research.