PLASMAS: HOW MUCH COLLABORATION DO YOU NEED IN MULTI-AGENT SYSTEM?

Anonymous authors
Paper under double-blind review

ABSTRACT

Semantically similar tasks can differ substantially in whether and how much multi-agent collaboration they require, making task-specific topology generation critical for efficient multi-agent systems (MAS). Existing graph-based generators typically encode tasks into intermediate representations before predicting topologies with a separate graph model. We show that these pipelines exhibit limited sensitivity to fine-grained collaboration demands, with only marginal gains from stronger encoders or graph backbones, suggesting a *semantic bottleneck* between task understanding and topology generation. To address this limitation, we propose **PLASMAS**, a preference-aligned topology generator following the language-based graph foundation model paradigm. PLASMAS jointly models task semantics, agent roles, and topology outputs within a shared language interface. Utility-aware pairwise preferences guide direct preference optimization based on task effectiveness and execution cost. Experiments show that PLASMAS better distinguishes collaboration demands among semantically similar tasks, allocating additional collaboration where beneficial and reducing unnecessary coordination on easy tasks while maintaining strong task performance. Our code is available at `https://anonymous.4open.science/r/PLASMAS-F526/`.

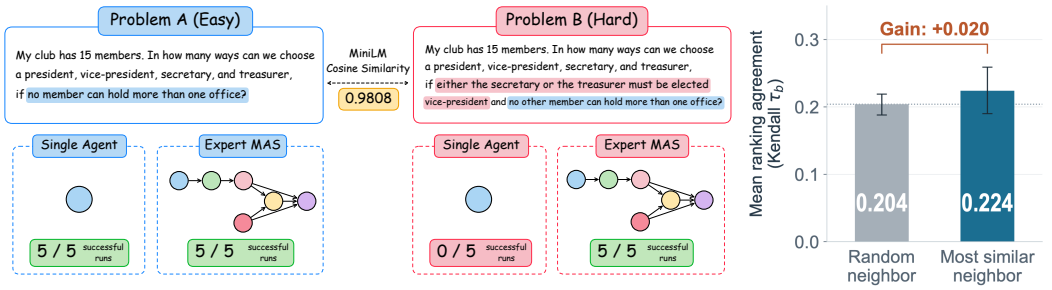
1 INTRODUCTION

Large language model (LLM)-based multi-agent systems (MAS) have emerged as a promising paradigm for solving complex reasoning, coding, and decision-making tasks through collaborative problem solving. Beyond the capabilities of individual agents, recent studies show that the *communication topology*, which determines how information is propagated and aggregated among agents, plays a critical role in both system effectiveness and efficiency (Hu et al., 2024; Zhang et al., 2025b; Wu et al., 2023). Early MAS frameworks typically rely on manually designed and task-agnostic structures, such as chains (Wei et al., 2022), trees (Yao et al., 2023), stars (Wu et al., 2023), or random graphs (Qian et al., 2025). However, the optimal collaboration pattern can vary substantially across tasks, motivating a growing shift toward *task-specific topology generation*, where communication structures are dynamically adapted to each input (Li et al., 2025; Cang et al., 2026; Jiang et al., 2026). More broadly, task-adaptive orchestration should determine not only *how* agents collaborate, but also *whether* collaboration is needed at all, allowing the system to allocate multi-agent computation only when its expected benefit justifies the additional execution cost.

Recent studies formulate task-specific MAS orchestration as a graph learning problem, representing agents as nodes and communication dependencies as edges, and condition graph pruning, generation, or optimization on each input task (Li et al., 2025; Cang et al., 2026; Jiang et al., 2026; Zhang et al., 2026; Zhou et al., 2026). More recently, OFA-MAS (Li et al., 2026) extends this paradigm across domains, showing that task semantics can systematically guide topology generation under substantially different task distributions.

Despite these advances, it remains unclear whether topology generators can distinguish semantically similar problems with different collaboration demands. Subtle changes in problem conditions can alter whether additional reasoning, verification, or complementary expertise is beneficial. The key issue is whether topology generators recognize these *topology-relevant semantics* and adjust whether and how much agents collaborate. Most existing graph-based methods first encode tasks and agent descriptions using a pretrained text encoder, then predict topologies with a downstream graph model.

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.



(a) Similar Tasks, Different Collaboration Needs (b) Limited Gain in Ranking Agreement

Figure 1: Pilot study on the semantic bottleneck. (a) Two MATH problems with high MiniLM cosine similarity (0.9808) benefit differently from collaboration: a single agent solves the first in all five trials but fails on the second in all five, whereas the same expert MAS solves both in all five trials. (b) For each question, we rank seven shared candidate topologies by success rate and measure pairwise ranking agreement using Kendall's τ_b. Nearest neighbors in MiniLM representation space show only slightly higher topology-ranking agreement than randomly sampled questions from the same subject (0.224 *vs.* 0.204; $\Delta\tau_b = 0.020$) across 651 eligible questions. Error bars denote 95% query-bootstrap confidence intervals.

This separation may introduce a *semantic bottleneck*: the pipeline may fail to capture *topology-relevant semantics* during encoding or use them effectively during structural prediction, limiting their influence on topology generation.

To probe whether generic semantic representations preserve *topology-relevant semantics*, we conduct a pilot analysis on MATH (Hendrycks et al., 2021). As shown in Fig. 1(a), two problems with a MiniLM (Wang et al., 2020) cosine similarity of 0.9808 can exhibit sharply different benefits from collaboration: both topologies (Single Agent & Expert MAS) can solve easy Problem A in all five trials, whereas only the Expert MAS succeeds on hard Problem B. Moreover, as shown in Fig. 1(b), across over 600 MATH questions, nearest neighbors under generic semantic representations show only slightly higher topology-ranking agreement than random same-subject neighbors ($\tau_b = 0.224$ *vs.* 0.204). These observations suggest that proximity in generic semantic representations does not reliably imply similar topology preferences, motivating a more direct assessment of collaboration-demand sensitivity.

Our empirical analysis (Section 2) further suggests that this limitation cannot be readily addressed by simply strengthening either component of the conventional pipeline by using stronger semantic representations or more expressive graph architectures. These observations motivate us to reconsider how *topology-relevant semantics* are incorporated into structure learning. Recent graph foundation models (GFMs) (Lin et al., 2024; Zhu et al., 2025; Kong et al., 2025), which jointly model natural-language semantics and graph structure, provide a natural alternative. We therefore formulate task-specific MAS orchestration as a unified framework, allowing communication structures to be predicted directly from natural-language tasks and agent roles within a shared language interface.

Building on this perspective, we propose **PLASMAS**, a **P**reference-a**L**igned gr**A**ph foundation model for task-**S**pecific **MAS** topology generation. PLASMAS is a unified framework that jointly models natural-language tasks, agent roles, and topology structures. By drawing on a language model's semantic reasoning capabilities, it generates structural decisions from the original task and role descriptions within a shared interface. This allows fine-grained task semantics to guide topology generation without passing through a separate text-encoder–graph-predictor interface, helping alleviate the *semantic bottleneck*. Training proceeds in two stages. First, execution-selected demonstrations teach the generator to express valid topologies and establish an initial mapping from tasks to collaboration structures.Second, utility-aware pairwise preferences compare candidate topologies by task effectiveness and execution cost, and direct preference optimization (DPO) (Rafailov et al., 2023) aligns the generator with these preferences. Together, PLASMAS is capable of distinguishing collaboration demands among semantically similar tasks, favoring minimal configurations when it is sufficient and allocating more collaboration when it improves task effectiveness.

Our main contributions are three-fold. (1) We provide empirical evidence of a *semantic bottleneck* in text-encoder–graph-learner pipelines: such generators exhibit limited sensitivity to differing collaboration demands among semantically similar tasks, with limited improvement from stronger en-

coders or graph backbones. (2) We introduce **PLASMAS**, a unified framework for task-adaptive topology generation following the language-based GFM paradigm. PLASMAS jointly models task semantics, agent roles, and topology outputs, initializes the generator with execution-selected demonstrations, and aligns it using utility-aware pairwise preferences that balance task effectiveness and execution cost. (3) Experiments show that PLASMAS achieves a favorable accuracy–cost trade-off and better distinguishes collaboration demands among semantically similar tasks.Further ablation studies support the effectiveness of its key components.

2 Empirical Analysis

To probe this potential semantic bottleneck, in this section, we examine whether existing generators distinguish collaboration demands among semantically similar tasks, and whether stronger encoders or graph backbones improve this sensitivity.

2.1 Measuring Task Sensitivity in Topology Generation

Downstream accuracy alone is insufficient to determine whether a topology generator is truly task-specific. In MATH (Hendrycks et al., 2021), candidate agents are generally designed to provide complementary reasoning capabilities, so a generator may achieve high accuracy by producing complex topologies for every problem, at substantial execution cost rather than through task-specific adaptation (Li et al., 2024; Alavi et al., 2026). Therefore we instead examine whether a topology generator allocates different structural resources to problems with different collaboration requirements.

Collaboration-demand stratification. We use two anchors: a *Single-Agent* anchor offers a structurally minimal option sufficient for simpler problems, while an *Expert MAS* anchor activates all available agents through a manually designed topology, providing stronger reasoning capabilities at higher execution cost. Comparing their success rates on different problems helps identify when a single agent suffices and when additional collaboration is beneficial. Each anchor is executed five times per problem, and their empirical success rates s_{expert} and s_{single} distinguish two groups:

- **Easy**: the Single-Agent anchor succeeds in all five trials ($s_{signle} = 1$).
- **Collaboration-Required**: the Expert MAS provides a large performance gain ($s_{\text{expert}} - s_{\text{single}} > \delta$) over the Single-Agent anchor, where δ is the success-rate improvement threshold.

We further perform stratified sampling by problem type, selecting 100 problems per group while controlling for coarse semantic categories, yielding a 200-problem stratified subset.

Evaluation Metric. We measure task sensitivity using the *Topology Sensitivity Gap (Δ_S)*, defined as the mean difference in active nodes or edges between *collaboration-required* and *easy* problems.

$$\Delta_S = \mathbb{E}_{q \in \mathcal{Q}_{\text{collab}}}[S(G_q)] - \mathbb{E}_{q \in \mathcal{Q}_{\text{easy}}}[S(G_q)]. \tag{1}$$

Here, G_q is the topology generated for problem q, and $S(G_q)$ is its number of active nodes or directed edges. Positive Δ_S indicates greater resource allocation to collaboration-required problems, while larger values indicate a stronger allocation preference for this group. For a more comprehensive assessment of allocation sensitivity, we additionally report three complementary statistics: **Cohen's** d measures standardized separation between the groups, with larger positive values indicating stronger separation; **bootstrap 95% confidence intervals** quantify uncertainty in Δ_S; and **Holm-adjusted permutation-test** p**-values** assess statistical significance ($p_{\text{Holm}} < 0.05$). These statistics characterize allocation sensitivity, without implying allocation optimality. Full definitions and statistical procedures are provided in Appendix A.

2.2 Limited Task-Specific Topology Adaptation

We conduct an empirical study of task-specific topology adaptation using graph-based topology generators and observe that, *in this setting, existing methods exhibit limited adaptability to varying collaboration demands*. We evaluate AGP (Li et al., 2025), which adapts agent participation and communication through graph pruning, and GTD (Jiang et al., 2026), which generates communication edges through task-conditioned graph diffusion, on the 200-problem stratified subset.

Method	Δ_S	Cohen's d	Bootstrap 95% CI	Perm. p	Holm p
AGP	$+0.0002$	0.067	$[-0.00069, 0.00113]$	0.629	1.0
GTD	-0.310	-0.176	$[-0.800, 0.180]$	0.203	1.0

Table 1: Task-sensitivity evaluation of existing graph-based topology generators. AGP is evaluated by node allocation. GTD is evaluated by edge allocation as its number of active agents is fixed. We use AGP's released checkpoint and reproduce GTD following its original configuration on MATH.

As shown in Table 1, AGP exhibits a negligible node-level Δ_S of $+0.0002$, while GTD yields a negative edge-level Δ_S of -0.310. Both bootstrap 95% confidence intervals cross zero, and both Holm-adjusted p-values equal 1.0. These results provide no clear evidence that either method allocates more structural resources to collaboration-required problems, consistent with the proposed *semantic bottleneck*.

2.3 Limited Gains from Stronger Graph Learners

Furthermore, from empirical study we observe that *increasing model capacity alone does not consistently improve topology sensitivity*. Following AGP (Li et al., 2025), we strengthen the text encoder and graph backbone while keeping data construction, supervision, and the topology objective fixed. Table 2 shows inconsistent gains, with no Easy–Collaboration-Required gap significant after Holm correction. This may reflect insufficient encoding of fine-grained collaboration requirements and a representation mismatch between semantic encoding and structural prediction, which larger components still may not resolve. These findings are consistent with a *semantic bottleneck*, motivating more direct integration of semantic reasoning into topology generation. Implementation details are provided in Appendix B.

Encoder	Graph Backbone	Δ_S	Cohen's d	Bootstrap 95% CI	Perm. p	Holm p
MiniLM	GCN	$+0.0833$	0.2933	$[+0.0052,\ 0.1608]$	0.0421	0.1552
ModernBERT	GCN	$+0.0191$	0.0883	$[-0.0411,\ 0.0782]$	0.5273	0.7483
Finetuned ModernBERT	GCN	$+0.0949$	0.2962	$[+0.0046,\ 0.1834]$	0.0388	0.1552
Finetuned ModernBERT	Graph Transformer	$+0.1282$	0.1264	$[-0.1595,\ 0.4003]$	0.3742	0.7483

Table 2: Controlled scaling of the conventional text-encoder–graph-learner pipeline.

3 The Proposed Method: PLASMAS

Motivated by the observations in Section 2, we introduce **PLASMAS**, a unified framework for preference-aligned, task-adaptive multi-agent topology generation. Following the language-based graph foundation model (GFM) paradigm, PLASMAS unify task semantics, agent roles, and topology generation within a shared language interface. As illustrated in Figure 2, the learning pipeline then comprises three steps: *Topology Generator Initialization* establishes the topology output format and an initial mapping from task semantics to topologies; *Utility-Aware Topology Preference Construction* builds pairwise preferences from candidate effectiveness and execution cost; *Topology preference alignment* further uses these pairs to refine the generator's output decisions. In the following part, we first introduce the shared language interface, then detail the three training steps.

Topology formulation. Let x denote a natural-language task and $\mathcal{R} = \{(r_j, d_j)\}_{j=1}^{M}$ a pool of available agents, where r_j is an agent identifier and d_j describes its role. We represent a communication topology as a directed graph $G = (V, E)$, where V specifies the participating agents and E specifies their communication dependencies. Each node inherits the textual attributes of its associated role. An edge (u, v) makes the output of agent u available to agent v. We consider directed acyclic graphs with a designated output agent f: every other active node must have a directed path to f. The singleton graph containing only f is admissible, allowing the generator to allocate minimal collaboration when appropriate. We denote the resulting space of admissible topologies by $\mathcal{G}(\mathcal{R})$.

A shared interface for semantics and structure. We formulate topology generation as conditional sequence modeling. The input $z = \mathrm{prompt}(x, \mathcal{R})$ combines the task and role descriptions with a fixed template specifying the output format and validity constraints. A serialization function

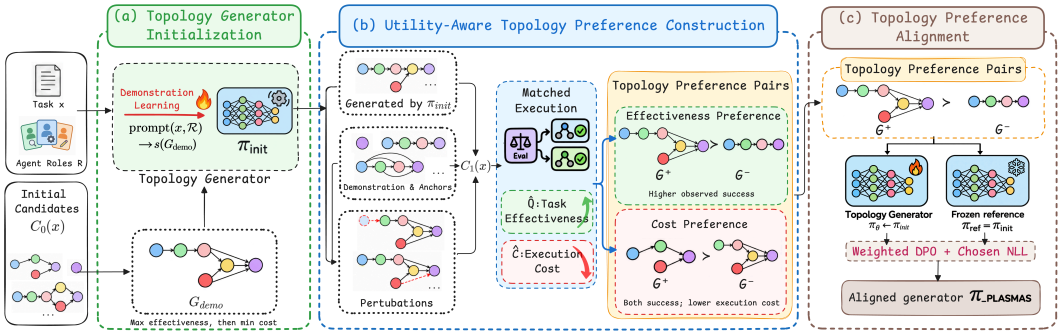


Figure 2: Overview of PLASMAS. (a) The topology generator is first initialized using demonstrations selected by task effectiveness and execution cost. (b) Demonstrations, anchor topologies, generator outputs, and local perturbations form a refreshed candidate pool, which is evaluated under matched execution conditions to construct effectiveness and cost preference pairs. (c) The generator is then aligned using weighted DPO and preferred-sequence NLL, with the initialized policy serving as the frozen reference.

$s : \mathcal{G}(\mathcal{R}) \to \mathcal{Y}$ maps each topology to an active-node list and a directed-edge list, preserving agent identities and edge directions. Here, $\mathcal{Y}$ is the space of serialized topology sequences. For $y = s(G)$, the generator is

$$\pi_\theta(y \mid z) = \prod_{t=1}^{|y|} \pi_\theta(y_t \mid z, y_{<t}). \tag{2}$$

The same serialization protocol is used during initialization, preference alignment, and inference.

3.1 Topology Generator Initialization

We initialize the generator through supervised learning on execution-selected demonstrations, establishing both the **topology output protocol** and **an initial task-to-topology mapping**.

We construct demonstrations from a candidate set $\mathcal{C}_0(x) \subset \mathcal{G}(\mathcal{R})$ spanning different agent compositions and communication patterns. Executing each candidate provides an empirical task-effectiveness estimate $\widehat{Q}(x, G)$ and an execution cost estimate $\widehat{C}(x, G)$. Demonstration selection prioritizes the highest observed effectiveness and uses lower cost to resolve ties:

$$\mathcal{C}_{\text{best}}(x) = \arg\max_{G \in \mathcal{C}_0(x)} \widehat{Q}(x, G), \qquad G^{\text{demo}}(x) \in \arg\min_{G \in \mathcal{C}_{\text{best}}(x)} \widehat{C}(x, G). \tag{3}$$

They form $\mathcal{D}_{\text{demo}} = \{(z, s(G^{\text{demo}}(x)))\}$.

We optimize the completion-only supervised objective

$$\mathcal{L}_{\text{init}}(\theta) = -\mathbb{E}_{(z,y)\sim\mathcal{D}_{\text{demo}}} \left[\frac{1}{|y|} \sum_{t=1}^{|y|} \log \pi_\theta(y_t \mid z, y_{<t}) \right]. \tag{4}$$

The resulting policy π_{init} learns to generate serialized topologies from task and role descriptions. Following the instruction-learning formulation of language-based GFMs (Lin et al., 2024; Zhu et al., 2025), supervision is applied only to the target topology tokens. Generated topologies are explicitly validated before execution.

However, a single demonstration per task does not explicitly supervise the relative effectiveness and cost of alternative topologies. We therefore refine the initialized generator through topology preference alignment.

3.2 Utility-Aware Topology Preference Construction

Execution-grounded topology comparisons. We construct a refreshed candidate set $\mathcal{C}_1(x)$ containing the demonstration topology $G^{\text{demo}}(x)$ and other candidate anchors, topologies generated

by π_{init}, and valid local perturbations of both. After deduplication, candidates are evaluated under matched execution conditions. For K repeated executions, we estimate

$$\widehat{Q}(x, G) = \frac{1}{K} \sum_{k=1}^{K} q_k(x, G), \qquad \widehat{C}(x, G) = \frac{1}{K} \sum_{k=1}^{K} c_k(x, G), \tag{5}$$

where $q_k(x, G) \in \{0, 1\}$ indicates task success and $c_k(x, G)$ is the total agent input/output token consumption. Agent models and execution conditions are fixed across candidates.

Effectiveness and cost preferences. **Utility-aware preference** construction considers both task effectiveness and execution cost. An effectiveness preference is accepted when one topology has a sufficiently large empirical effectiveness advantage, supported by repeated paired wins without opposing losses. A cost preference is accepted when both topologies succeed in every evaluated execution and one topology achieves a sufficiently large and consistent cost reduction. Comparisons without adequate evidence are omitted.

Let $G^+ \succ_x G^-$ denote an accepted comparison. The preference dataset is

$$\mathcal{D}_{\text{pref}} = \{(z_i, y_i^+, y_i^-, w_i) : y_i^{\pm} = s(G_i^{\pm}),\ G_i^+ \succ_{x_i} G_i^-\}. \tag{6}$$

We retain comparisons covering effectiveness gains, local reductions in team size with lower execution cost, and differences in role composition and connectivity at a fixed team size. The latter provide supervision on structural choices beyond the number of participating agents. Pair weights $w_i = \alpha_{t_i}/m_{x_i}$ account for preference type t_i and the number m_{x_i} of selected comparisons for the task, with a larger coefficient for effectiveness preferences. Here, $\alpha_{t_i} > 0$ is the weighting coefficient for preference type t_i.

3.3 Topology Preference Alignment

We align the generator with these topology preferences using direct preference optimization (DPO) (Rafailov et al., 2023). We initialize both the trainable policy and the frozen reference from π_{init}. Define $r_\theta(z, y) = \log \pi_\theta(y \mid z) - \log \pi_{\text{ref}}(y \mid z)$. The pairwise loss is

$$\ell_{\text{DPO},i}(\theta) = -\log \sigma\big(\beta \left[r_\theta(z_i, y_i^+) - r_\theta(z_i, y_i^-)\right]\big), \tag{7}$$

where σ is the sigmoid function and $\beta > 0$ controls the scale of the reference-relative preference margin. Sequence log probabilities are summed over topology completion tokens.

We additionally retain a likelihood term on preferred topology serializations:

$$\mathcal{L}_{\text{align}}(\theta) = \frac{1}{Z} \sum_i w_i \left[\ell_{\text{DPO},i}(\theta) - \frac{\lambda_{\text{NLL}}}{|y_i^+|} \log \pi_\theta(y_i^+ \mid z_i)\right], \qquad Z = \sum_i w_i. \tag{8}$$

where $\lambda_{\text{NLL}} \geq 0$ controls the contribution of the preferred-sequence likelihood term. This auxiliary term encourages likelihood on the preferred outputs alongside the comparative DPO objective. Execution feedback enters through the preference dataset, so optimization requires neither differentiation through agent execution nor a separately trained reward model.

Inference. Given a task and available roles, the generator produces a topology that is parsed and validated before agents execute in dependency order. The topology remains fixed during execution, and no candidate search is performed at inference time.

4 Experiments

We evaluate PLASMAS primarily in terms of task effectiveness and inference token cost, examining its performance–cost trade-off against existing methods. We further assess whether its generated topologies adapt to the collaboration demands of individual tasks, complementing downstream performance with the topology sensitivity analysis introduced in Section 2.1. Finally, we conduct ablation studies to assess the contribution of each component and separately evaluate topology generation overhead.

Method	GSM8K		MATH-500		MultiArith		SVAMP		MMLU	
	Acc.↑	Tokens↓	Acc.↑	Tokens↓	Acc.↑	Tokens↓	Acc.↑	Tokens↓	Acc.↑	Tokens↓
Vanilla	87.5	226	67.8	512	96.9	157	91.2	157	71.0	230
CoT	90.8	1,121	71.4	2,858	<u>97.4</u>	987	92.2	980	72.5	1,432
ComplexCoT	91.3	2,957	72.2	2,722	<u>97.4</u>	2,843	93.7	2,838	72.6	2,929
SC (CoT×5)	90.0	4,061	<u>74.2</u>	5,923	97.1	3,046	92.4	3,139	72.9	5,253
LLM-Debate	**93.0**	4,976	71.4	12,015	97.1	2,892	**95.7**	3,398	69.6	5,131
DyLAN	89.7	5,073	70.4	11,133	96.9	2,928	91.9	3,755	69.9	3,223
GPTSwarm‡	90.5	3,073	56.4	2,474	96.3	3,566	93.8	1,855	64.1	7,996
G-Designer‡	90.2	4,019	70.9	7,101	95.8	3,537	93.7	3,359	**75.8**	16,179
MaAS‡	90.8	2,464	70.0	9,581	96.9	2,784	93.6	1,887	71.4	2,157
AGP†	89.0	5,647	61.2	7,733	96.2	4,906	93.9	4,873	74.0	10,528
GTD‡	<u>92.4</u>	3,513	68.2	8,094	<u>97.4</u>	2,031	93.3	2,395	<u>74.4</u>	4,208
OFAMAS‡	87.0	4,161	65.4	6,216	96.7	3,852	93.1	3,831	72.6	4,139
PLASMAS (Ours)	90.9	825	**84.6**	3,945	**97.5**	588	<u>94.5</u>	663	73.9	666

† Reproduced using author-released checkpoints. ‡ Reimplemented following the original paper.

Table 3: Task effectiveness and inference token cost across five benchmarks. Acc. denotes final-answer accuracy (%), and Tokens denotes the average total number of input and output tokens consumed per example. Higher accuracy and lower token consumption are better. The best and second-best results in each column are shown in **bold** and <u>underlined</u>, respectively.

4.1 Experimental Setups

Benchmarks and Baselines. We evaluate PLASMAS on GSM8K (Cobbe et al., 2021), MATH-500 (Lightman et al., 2024), MultiArith (Roy & Roth, 2015), MMLU (Hendrycks et al., 2020), and SVAMP (Patel et al., 2021). Baselines include Single-agent LLM, CoT (Wei et al., 2022), Complex-CoT (Fu et al., 2022), SC (CoT×5) (Wang et al., 2022), LLM-Debate (Du et al., 2023), DyLAN (Liu et al., 2023), GPTSwarm (Zhuge et al., 2024), G-Designer (Zhang et al., 2024), MaAS (Zhang et al., 2025a), AGP (Li et al., 2025), and GTD (Jiang et al., 2026). All methods use the same evaluation examples within each benchmark. For topology sensitivity, we compare against task-adaptive methods AGP and GTD.

Implementation Details and Metrics. We use Qwen3-4B (Yang et al., 2025) for data generation and agent execution. Execution agents use temperature 0.7 and at most 2,048 output tokens per generation. We construct two 1,000-problem corpora from the MATH training split (Hendrycks et al., 2021) and the MMLU-Pro (Wang et al., 2024), respectively, with 900 training and 100 validation problems per corpus. We use Qwen3-1.7B as the language backbone of the PLASMAS topology generator. For initialization, we evaluate 12 candidate topologies per problem over five executions. Preference construction proposes ten valid candidates: four base topologies, three initialization-policy outputs, and three local perturbations. We train the topology generator using LoRA. More implementation details are provided in Appendix D.

We report final-answer accuracy and average inference token consumption per example. Token cost includes all input and output tokens used for topology generation, agent execution, and language-model-based selection or aggregation, including internal samples and collaboration rounds. For task sensitivity evaluation, we use the metrics defined in Section 2.1.

4.2 Performance Evaluation and Analysis

Overall Performance and Efficiency. Table 3 shows that PLASMAS achieves a favorable accuracy–cost trade-off across all five benchmarks. On MultiArith and SVAMP, where Vanilla already achieves high accuracy, PLASMAS reaches 97.5% and 94.5% accuracy using only 588 and 663 tokens per example, respectively. Compared with GTD, this improves accuracy while reducing token consumption by approximately 71% and 72%. On GSM8K and MMLU, PLASMAS also maintains competitive accuracy at substantially lower cost than the evaluated multi-agent baselines.

Notably, PLASMAS allocates more computation to the challenging MATH-500 benchmark while maintaining greater efficiency than the evaluated multi-agent baselines. It achieves 84.6% accuracy with 3945 tokens per example, outperforming the strongest baseline by 10.4 percentage points while

consuming fewer tokens. Together with its lower expenditure on easier benchmarks, these results suggest effective adaptation to varying collaboration demands. Moreover, with topology generators trained on MATH and MMLU-Pro, the results on GSM8K, MultiArith, and SVAMP demonstrate transfer beyond the training benchmarks.

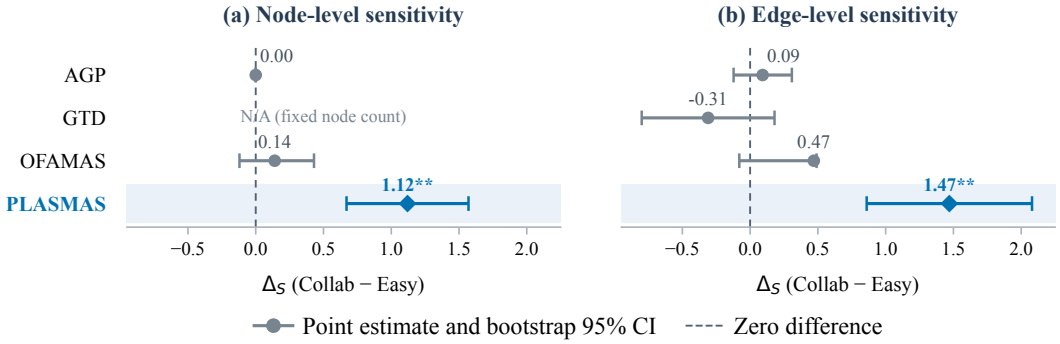


Figure 3: Topology sensitivity to collaboration demand. Points indicate node- and edge-level Δ_S (Collab minus Easy), and horizontal bars show bootstrap 95% confidence intervals. GTD has a fixed node count, so its node-level sensitivity is not applicable.** indicates Holm-adjusted $p < 0.05$ for the Easy–Collab comparison within each method.

Topology Sensitivity. We examine whether PLASMAS adjusts agent participation and communication dependencies to task-specific collaboration demands. Following Section 2.1, we compare AGP, GTD, OFAMAS, and PLASMAS using the same collaboration-demand partition and evaluation protocol. We report both node- and edge-level sensitivity for AGP, OFAMAS, and PLASMAS, and edge-level sensitivity for GTD, whose number of active agents is fixed. Figure 3 shows that PLASMAS achieves the largest positive sensitivity gaps in both agent participation and communication-edge allocation, with both 95% confidence intervals lying entirely above zero. It is also the only method with significant Easy–Collab differences after Holm correction in both metrics, supporting its stronger sensitivity to task-specific collaboration demands.

Ablation Study. We examine the contributions of generator initialization, topology preference alignment, and the preferred-sequence NLL term through four configurations: *Prompt-only* without generator training, *Initialization Only* without preference alignment, *w/o Chosen NLL* with weighted DPO alone after initialization, and full PLASMAS. The trained variants share the same initialization procedure, and both aligned variants use the same preference data.

Method	Acc.↑	Tokens↓	Metric	Δ_S	Cohen's d	95% CI	Perm. p	Holm p
Prompt-only	33.22	7,884	Nodes	0.14	0.233	$[-0.11, 0.40]$	0.31454	1.000
			Edges	0.66	0.270	$[-0.36, 1.62]$	0.21376	1.000
Initialization Only	83.06	3,102	Nodes	0.90	0.559	$[0.45, 1.34]$	0.00040	0.00120
			Edges	1.12	0.543	$[0.57, 1.70]$	0.00040	0.00120
w/o Chosen NLL	83.90	**2,991**	Nodes	1.01	0.625	$[0.58, 1.46]$	0.00020	0.00120
			Edges	1.32	0.607	$[0.74, 1.92]$	0.00020	0.00120
PLASMAS	**84.02**	3,175	Nodes	1.12	0.678	$[0.67, 1.57]$	0.00020	0.00120
			Edges	1.47	0.654	$[0.86, 2.08]$	0.00020	0.00120

Table 4: Ablation of generator initialization, preference alignment, and chosen-sequence NLL. We compare the pretrained generator without initialization or alignment (Prompt-only), initialization without DPO (Initialization Only), DPO without chosen-sequence NLL (w/o Chosen NLL), and the full method (PLASMAS).

Table 4 shows that initialization raises accuracy from 33.22% to 83.06% while substantially reducing execution token consumption. Weighted DPO further improves accuracy to 83.90% and reduces tokens by 3.6%, while increasing node- and edge-level Δ_S from 0.90 to 1.01 and from 1.12 to 1.32. Adding chosen-sequence NLL yields 84.02% accuracy and Δ_S of 1.12 and 1.47, with 6.1% more execution tokens than DPO alone. Together, these results suggest that initialization establishes effective topology generation, while preference alignment further improves task performance and differentiation of collaboration demands.

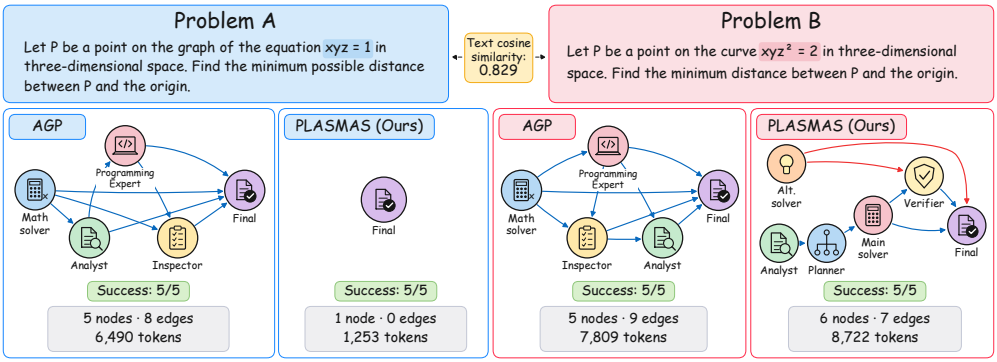


Figure 4: Case study: PLASMAS selects different collaboration scales, while AGP retains five agents for both problems. Success counts are measured over five matched trials; execution tokens are averaged per trial.

Case Study. Figure 4 illustrates how PLASMAS assigns different collaboration structures to two semantically similar MATH problems. For Problem A, PLASMAS uses a single agent and consumes 1,253 execution tokens, compared with AGP's five agents and 6,490 tokens, while both succeed in all five trials. For Problem B, PLASMAS selects six agents with seven communication edges and also succeeds in all five trials. This case illustrates task-specific variation in collaboration allocation, with substantial savings when a single agent suffices.

5 Related Work

Multi-Agent Topology Organization. LLM-based multi-agent systems have evolved from manually specified communication patterns, such as chains, hierarchical structures, and debate (Qian et al., 2024; Hong et al., 2024; Wu et al., 2023; Wang et al., 2025; Du et al., 2023), toward adaptive topology design. Recent methods formulate agent organization as computational-graph optimization, connectivity learning, structural pruning, or architecture search (Zhuge et al., 2024; Zhang et al., 2024; 2025b;a). More recent approaches further condition agent participation and communication structure on individual queries (Li et al., 2025; Cang et al., 2026; Jiang et al., 2026; Li et al., 2026). Our work focuses on this task-adaptive setting, but further considers whether multi-agent collaboration is needed at all and optimizes communication under an explicit utility–cost trade-off.

Graph Learning for MAS. Graph-based MAS methods represent agents as nodes and communication as edges, and learn task-conditioned structures through graph pruning, connectivity prediction, conditional generation, or topology-aware evaluation (Zhang et al., 2024; Li et al., 2025; Cang et al., 2026; Zhang et al., 2026; Li et al., 2026; Zhou et al., 2026; Hong et al., 2026). In parallel, graph foundation models jointly model language and graph structure, providing an interface between semantic reasoning and structural prediction (Lin et al., 2024; Kong et al., 2025; Zhu et al., 2025). We build on this direction by applying graph foundation modeling to task-specific MAS orchestration and introducing utility-aware preference learning over candidate communication structures.

6 Conclusions

In this work, we study whether task-specific topology generators can distinguish semantically similar tasks that differ in their need for multi-agent collaboration, determining both when to collaborate and how much collaboration to allocate. Our analysis provides evidence of a semantic bottleneck in existing methods, which limits their sensitivity to these fine-grained differences. To address this limitation, we propose PLASMAS, a topology generation framework following the language-based GFM paradigm that integrates language-model semantic understanding with structured topology generation and execution-grounded preference alignment. Experiments show that PLASMAS better differentiates collaboration demands among superficially similar tasks, allocating more agents and communication to tasks that benefit from collaboration while reducing unnecessary coordination on simpler tasks. These findings support task-sensitive topology generation as a means of reducing execution overhead while maintaining strong task performance.

AI USE STATEMENT

In this work, we used generative AI tools to improve the language and readability of the manuscript, assist with code implementation, and generate narration for the supplementary presentation video. Apart from code implementation, we did not use generative AI tools for any other tasks requiring mandatory disclosure under the ICLR 2027 AI policy. We reviewed the AI-assisted edits to ensure that they preserved the intended technical meaning, verified and tested the AI-assisted code for correctness, and checked the generated narration against the presentation script. We take responsibility for the final content of the paper and supplementary materials, including all AI-assisted text, code, and audio.

REFERENCES

Khashayar Alavi, Zhastay Yeltay, Lucie Flek, and Akbar Karimi. More agents improve math problem solving but adversarial robustness gap persists. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 43457–43475, 2026.

Yueyang Cang, Xiaoteng Zhang, Erlu Zhao, Zehua Ji, Yuhang Liu, Yuchen He, Zhiyuan Ning, Chen Yijun, Wenge Que, and Li Shi. Graph-grpo: Stabilizing multi-agent topology learning via group relative policy optimization. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 20222–20231, 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.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*, 2023.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. Complexity-based prompting for multi-step reasoning. *arXiv preprint arXiv:2210.00720*, 2022.

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

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, zili wang, Steven Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. Metagpt: Meta programming for a multi-agent collaborative framework. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 23247–23275, 2024.

Zhi Hong, Qian Zhang, Jiahang Sun, Zhiwei Shang, Mingze Kong, Xiangyi Wang, Yao Shu, and Zhongxiang Dai. Maspob: Bandit-based prompt optimization for multi-agent systems with graph neural networks. *arXiv preprint arXiv:2603.02630*, 2026.

Shengchao Hu, Li Shen, Ya Zhang, and Dacheng Tao. Learning multi-agent communication from graph modeling perspective. In *International Conference on Learning Representations*, volume 2024, pp. 12963–12978, 2024.

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.

Lecheng Kong, Jiarui Feng, Hao Liu, Chengsong Huang, Jiaxin Huang, Yixin Chen, and Muhan Zhang. Gofa: A generative one-for-all model for joint graph language modeling. In *International Conference on Learning Representations*, volume 2025, pp. 40792–40823, 2025.

Boyi Li, Zhonghan Zhao, Der-Horng Lee, and Gaoang Wang. Adaptive graph pruning for multi-agent communication. In *ECAI 2025: 28th European Conference on Artificial Intelligence 25-30 October 2025, Bologna, Italy-Including 14th Conference on Prestigious Applications of Intelligent Systems (PAIS 2025) Proceedings*, pp. 4305–4312. SAGE Publications 1 Oliver's Yard, 55 City Road, London, EC1Y 1SP, 2025.

Junyou Li, Qin Zhang, Yangbin Yu, Qiang Fu, and Deheng Ye. More agents is all you need. *arXiv preprint arXiv:2402.05120*, 2024.

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, 2026.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Tianqianjin Lin, Pengwei Yan, Kaisong Song, Zhuoren Jiang, Yangyang Kang, Jun Lin, Weikang Yuan, Junjie Cao, Changlong Sun, and Xiaozhong Liu. Langgfm: A large language model alone can be a powerful graph foundation model. *arXiv preprint arXiv:2410.14961*, 2024.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. A dynamic llm-powered agent network for task-oriented agent collaboration. *arXiv preprint arXiv:2310.02170*, 2023.

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, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. ChatDev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL), Long Papers*, 2024. URL https://aclanthology.org/2024.acl-long.810/. arXiv:2307.07924.

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.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

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.

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.

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

Zhao Wang, Sota Moriyama, Wei-Yao Wang, Briti Gangopadhyay, and Shingo Takamatsu. Talk structurally, act hierarchically: A collaborative framework for llm multi-agent systems. *arXiv preprint arXiv:2502.11098*, 2025.

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.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

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. *arXiv preprint arXiv:2410.11782*, 2024.

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. Multi-agent architecture search via agentic supernet. *arXiv preprint arXiv:2502.04180*, 2025a.

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, 2025b.

Zhen Zhang, Wanjing Zhou, Juncheng Li, Hao Fei, Jun Wen, and Wei Ji. Radar: Redundancy-aware diffusion for multi-agent communication structure generation. *arXiv preprint arXiv:2605.09907*, 2026.

Zhilun Zhou, Zihan Liu, Jiahe Liu, Qingyu Shao, Yihan Wang, Kun Shao, Depeng Jin, and Fengli Xu. Resmas: Resilience optimization in llm-based multi-agent systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 35176–35184, 2026.

Xi Zhu, Haochen Xue, Ziwei Zhao, Wujiang Xu, Jingyuan Huang, Minghao Guo, Qifan Wang, Kaixiong Zhou, Imran Razzak, and Yongfeng Zhang. Llm as gnn: Graph vocabulary learning for text-attributed graph foundation models. *arXiv preprint arXiv:2503.03313*, 2025.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. Language agents as optimizable graphs. *arXiv preprint arXiv:2402.16823*, 2024.

APPENDIX

A EVALUATION METRICS AND STATISTICAL PROTOCOL

Pilot construction. We stratify MATH test problems q using the empirical success rates $a_{\mathrm{F}}(q)$ and $a_{\mathrm{A}}(q)$ of the finalizer-only and expert-anchor configurations, respectively. Let $\Delta a(q) = a_{\mathrm{A}}(q) - a_{\mathrm{F}}(q)$. Using configurable thresholds, we define four descriptive strata (omitting q for brevity):

$$\text{easy}: \quad a_{\mathrm{F}} \geq \tau_{\mathrm{E}}, \tag{9}$$

$$\text{medium}: \quad \tau_{\mathrm{L}} \leq a_{\mathrm{F}} \leq \tau_{\mathrm{U}} \text{ and } |\Delta a| \leq \delta, \tag{10}$$

$$\text{collaboration-required}: \quad \Delta a > \delta, \tag{11}$$

$$\text{hard-unsolved}: \quad \max(a_{\mathrm{F}}, a_{\mathrm{A}}) \leq \tau_{\mathrm{H}}. \tag{12}$$

Here, τ_{E} specifies reliable single-agent performance, $[\tau_{\mathrm{L}}, \tau_{\mathrm{U}}]$ defines intermediate performance, δ specifies a substantial collaboration benefit, and τ_{H} specifies persistent failure.

We sample 100 questions from each stratum, yielding a 400-question pilot set. These strata reflect observed performance under the chosen anchors and evaluation budget, rather than intrinsic task difficulty. For our five-trial evaluation, we set $\tau_{\mathrm{E}} = 1.0$, $\tau_{\mathrm{L}} = 0.4$, $\tau_{\mathrm{U}} = 0.8$, $\delta = 0.4$, and $\tau_{\mathrm{H}} = 0.2$. Thus, collaboration-required problems exhibit at least three additional successes under the Expert MAS anchor. Problems outside these strata are excluded from stratified sampling. Each anchor accuracy is estimated from five runs using Qwen3-4B, temperature 0.7, a maximum of 2,048 new tokens, and seeds $\{42, 43, 44, 45, 46\}$. Answers are evaluated with `math-verify`.

Questions are sampled without replacement using seed 20260901. Each group contains 15 Algebra questions, 15 Prealgebra questions, and 14 questions from each of the remaining five categories: Counting & Probability, Geometry, Intermediate Algebra, Number Theory, and Precalculus. All evaluated models use the same fixed questions and group labels. **The primary comparison is collaboration-required versus easy, which we mainly report in the full paper.**

Topology statistics. For a generated topology G_q, we consider the number of active nodes and the number of directed edges:

$$S_{\mathrm{node}}(G_q) = \sum_{v} z_{qv}, \qquad S_{\mathrm{edge}}(G_q) = \sum_{u \neq v} A_{q,uv}, \tag{13}$$

where $z_{qv} \in \{0, 1\}$ indicates whether node v is active, and $A_{q,uv} \in \{0, 1\}$ indicates whether the generated graph contains the directed edge $u \to v$. Self-loops are excluded.

The role pool contains six nodes, including an always-active finalizer. Thus, node counts range from one to six. Including the finalizer adds the same constant to every node count and therefore does not affect the node-count gap or Cohen's d.

Topology Sensitivity Gap. Let $\mathcal{Q}_{\mathrm{collab}}$ and $\mathcal{Q}_{\mathrm{easy}}$ denote the collaboration-required and easy question sets, respectively, with $n_{collab} = n_{easy} = 100$. For a topology statistic S, define

$$\overline{S}_g = \mathbb{E}_{q \in g}[S(G_q)] = \frac{1}{n_g} \sum_{q \in g} S(G_q), \qquad \Delta_S = \overline{S}_{collab} - \overline{S}_{easy}. \tag{14}$$

Positive Δ_S indicates that collaboration-required questions receive more structural resources on average. Zero Δ_S indicates equal group means, but does not imply that the two groups have identical allocation distributions. Negative Δ_S indicates an allocation difference in the opposite direction. Δ_S retains the units of the underlying statistic: for example, $\Delta_{\mathrm{node}} = 1$ means one additional active node per collaboration-required question on average. The same definition is applied to each secondary comparison by replacing C with the corresponding target group.

It should be noted that Δ_S is an empirical metric, and larger values are not necessarily better. In principle, an ideal upper bound exists in a specific set. However, because the optimal topology is not directly measurable, this bound cannot be computed. We therefore interpret a larger Δ_S within a certain range as indicating more appropriate topology allocation between the two groups. Nevertheless, we consider $\Delta_S > 1$ necessary, since collaboration-required problems should activate at least one more node or edge than easy problems.

Standardized effect size. We report Cohen's d to express the allocation gap relative to within-group variability:

$$d_S = \frac{\overline{S}_C - \overline{S}_E}{s_{\text{pooled},S}}, \tag{15}$$

where

$$s_{\text{pooled},S} = \sqrt{\frac{(n_C - 1)s_{C,S}^2 + (n_E - 1)s_{E,S}^2}{n_C + n_E - 2}}, \tag{16}$$

and

$$s_{g,S}^2 = \frac{1}{n_g - 1} \sum_{q \in g} \left(S(G_q) - \overline{S}_g\right)^2. \tag{17}$$

A positive d_S indicates greater allocation to the collaboration-required group, and a larger positive value indicates stronger standardized separation. Cohen's d is an effect-size estimate, not a significance test or a measure of resource efficiency. We do not impose a universal effect-size threshold for successful adaptation. The statistic is defined when $s_{\text{pooled},S} > 0$.

Bootstrap confidence intervals. We estimate uncertainty in Δ_S using an independent-samples percentile bootstrap with $B = 5{,}000$ resamples. For each replicate b, we sample n_C questions with replacement from C and independently sample n_E questions with replacement from E, obtaining

$$\Delta_S^{*(b)} = \overline{S}_C^{*(b)} - \overline{S}_E^{*(b)}. \tag{18}$$

The bootstrap 95% confidence interval is

$$\text{CI}_{0.95} = \left[Q_{0.025}\left(\{\Delta_S^{*(b)}\}_{b=1}^B\right), Q_{0.975}\left(\{\Delta_S^{*(b)}\}_{b=1}^B\right)\right], \tag{19}$$

where Q_α denotes the empirical α-quantile. A narrower interval indicates greater estimation precision, and an interval entirely above zero supports a positive allocation gap. These intervals describe Δ_S, not Cohen's d, and are not adjusted for multiple comparisons.

Resampling is performed at the question level within each group, without additional stratification by mathematical category. Category balance is imposed when constructing the original pilot. Each bootstrap calculation initializes `numpy.random.default_rng(7)`.

Permutation tests. We assess statistical evidence using a two-sided permutation test with $P = 5{,}000$ permutations. Under the null hypothesis that the group labels are exchangeable with respect to S, we pool the 200 observations in the comparison and randomly reassign them to two groups of 100. Category membership is not constrained during permutation. The observed test statistic is

$$T_{\text{obs}} = |\Delta_S|. \tag{20}$$

For each permutation b, we calculate the absolute difference between the permuted group means, denoted $T^{\pi(b)}$. The Monte Carlo p-value is

$$p_{\text{perm}} = \frac{1 + \sum_{b=1}^P \mathbf{1}\left[T^{\pi(b)} \geq T_{\text{obs}}\right]}{P + 1}. \tag{21}$$

The addition of one to the numerator and denominator prevents zero Monte Carlo p-values. With $P = 5{,}000$, the smallest attainable value is $1/5{,}001 \approx 0.00020$. Each permutation calculation uses `numpy.random.default_rng(7)`. Smaller p-values indicate stronger evidence against the exchangeable-label null; they do not quantify the magnitude or practical value of the allocation difference.

Multiple-comparison correction. For each model, the correction family contains six tests: three target groups compared with easy, evaluated for both node and edge counts. Let $p_{(1)} \leq \cdots \leq p_{(m)}$ denote the ordered raw p-values, where $m = 6$. The Holm-adjusted values are

$$p_{(i)}^{\text{Holm}} = \min\left\{1, \max_{1 \leq j \leq i}\left[(m - j + 1)p_{(j)}\right]\right\}, \tag{22}$$

and are then mapped back to their original comparisons. We determine statistical significance using $p^{\text{Holm}} < 0.05$.

B Experiment Setting of Controlled Capacity-Scaling Study

For the analysis in section 2.3, we construct the training set from 1,000 MATH problems following the topology-supervision construction procedure of AGP, including candidate-topology generation, execution-based selection, and graph pruning. The agent pool, supervision format, and training objective follow the same setup. We compare MiniLM and ModernBERT semantic encoders with GCN and a three-layer Graph Transformer (hidden dimension 128, four attention heads). All encoders use masked mean pooling without additional ℓ_2 normalization.

In the frozen setting, the semantic encoder remains fixed during graph-model training. For ModernBERT adaptation, we apply LoRA ($r = 8$, $\alpha = 16$, dropout 0.05) to the `attn.Wqkv` and `attn.Wo` projections of all 22 Transformer blocks and jointly optimize the LoRA parameters with the graph predictor. Models are trained for 50 epochs with batch size 4 using AdamW, with learning rates of 2×10^{-5} for the graph predictor and 1×10^{-4} for LoRA parameters. All experiments are conducted on a single NVIDIA GeForce RTX 4090 GPU. Evaluation uses the held-out 400-problem collaboration-demand subset defined in Appendix section A.

C PLASMAS Training Procedure

To provide a clearer step-by-step implementation perspective, we summarize the entire workflow of the proposed PLASMAS framework in algorithm 1.

D Training Data and Implementation Details

Data construction. We construct two corpora of 1,000 problems each: mathematical reasoning problems from the MATH training split (Hendrycks et al., 2021) and general reasoning problems from the non-mathematical subjects of MMLU-Pro (Wang et al., 2024). Each 1,000-problem corpus is partitioned into 900 training and 100 validation problems. These problem-level assignments are preserved throughout generator initialization and preference alignment. Both corpora share the execution, scoring, and preference-construction procedure, while using task-specific agent pools and initial topology samplers.

Execution-agent pools. Each corpus uses a fixed pool of six execution agents, indexed from 0 to 5, with agent 5 designated as the finalizer (Table V). All execution agents use the same frozen Qwen3-4B backbone and are differentiated by fixed role-specific prompts. Training updates the topology generator rather than the execution-agent backbone.

The topology generator receives the problem and the agents' identifiers, role names, and role descriptions, but not the reference answer. Each active execution agent receives the original problem and the outputs of its direct predecessors. The finalizer is always active and produces the final answer; when selected alone, it solves the problem independently. Execution uses temperature 0.7, with a per-generation output limit of 2,048 tokens for both corpora.

Initial topology candidates. For each problem x, the initial candidate set $\mathcal{C}_0(x)$ contains 12 distinct topologies: seven predefined topologies and five random DAGs. The predefined candidates

Index	MATH role	MMLU-Pro role
0	Problem analyst	Terminology curator
1	Strategy planner	Disciplinary memory specialist
2	Primary solver	Case–rule specialist
3	Alternative solver	Distractor auditor
4	Symbolic/proof verifier	Evidence calibrator
5	Finalizer	Finalizer

Table V: Fixed execution-agent pools. MATH roles analyze constraints, plan and derive solutions, and verify mathematical reasoning. MMLU-Pro roles clarify terminology, recall disciplinary facts, apply contextual rules, audit distractors, and calibrate evidence before final option selection. The indices define the forward order used to construct initial candidate topologies.

Algorithm 1 Training procedure of PLASMAS

Input: Training tasks $\mathcal{D} = \{x_i\}_{i=1}^{N}$; agent-role pool $\mathcal{R}$; pretrained topology generator π_0; number of repeated executions K

Output: Aligned topology generator π_{PLASMAS}

▷ **Stage 1: Topology Generator Initialization**
Initialize demonstration set: $\mathcal{D}_{\text{demo}} \leftarrow \emptyset$
for each task $x \in \mathcal{D}$ **do**
  Construct initial candidate set $\mathcal{C}_0(x) \subset \mathcal{G}(\mathcal{R})$
  for each topology $G \in \mathcal{C}_0(x)$ **do**
    Execute G for K runs
    Estimate task effectiveness $\widehat{Q}(x, G)$ and execution cost $\widehat{C}(x, G)$
  end for
  Select the highest-effectiveness candidates: $\mathcal{C}_{\text{best}}(x) \leftarrow \arg\max_{G \in \mathcal{C}_0(x)} \widehat{Q}(x, G)$
  Select the lowest-cost demonstration: $G^{\text{demo}}(x) \in \arg\min_{G \in \mathcal{C}_{\text{best}}(x)} \widehat{C}(x, G)$
  Add $\big(\text{prompt}(x, \mathcal{R}), s(G^{\text{demo}}(x))\big)$ to $\mathcal{D}_{\text{demo}}$
end for
Optimize π_0 on $\mathcal{D}_{\text{demo}}$ using $\mathcal{L}_{\text{init}}$ in Eq. equation 4
Obtain the initialized generator π_{init}
▷ **Stage 2: Topology Preference Alignment**
Initialize preference set $\mathcal{D}_{\text{pref}} \leftarrow \emptyset$
for each task $x \in \mathcal{D}$ **do**
  Generate candidate topologies with π_{init} and construct valid local perturbations
  Form the refreshed candidate set $\mathcal{C}_1(x)$ from $G^{\text{demo}}(x)$, anchor topologies, generator outputs, and local perturbations
  Deduplicate and validate $\mathcal{C}_1(x)$
  for each topology $G \in \mathcal{C}_1(x)$ **do**
    Execute G for K runs under matched execution conditions
    Estimate $\widehat{Q}(x, G)$ and $\widehat{C}(x, G)$
  end for
  Construct accepted effectiveness and cost preferences: $G^+ \succ_x G^-$
  Retain eligible effectiveness, local-cost, and same-size structural comparisons
  for each retained pair (G_i^+, G_i^-) **do**
    Compute pair weight $w_i \leftarrow \alpha_{t_i}/m_{x_i}$
    Set $z_i \leftarrow \text{prompt}(x, \mathcal{R})$, $y_i^+ \leftarrow s(G_i^+)$, and $y_i^- \leftarrow s(G_i^-)$
    Add (z_i, y_i^+, y_i^-, w_i) to $\mathcal{D}_{\text{pref}}$
  end for
end for
Initialize trainable policy $\pi_\theta \leftarrow \pi_{\text{init}}$
Initialize and freeze reference policy $\pi_{\text{ref}} \leftarrow \pi_{\text{init}}$
Optimize π_θ on $\mathcal{D}_{\text{pref}}$ using $\mathcal{L}_{\text{align}}$ in Eq. equation 8
$\pi_{\text{PLASMAS}} \leftarrow \pi_\theta$
return π_{PLASMAS}

comprise a finalizer-only baseline, a two-agent specialist–finalizer baseline, and five task-specific collaboration templates. Their edge sets are listed in table VI.

For MATH, each random topology contains between two and six active agents. The active count is sampled with probabilities proportional to $(0.25, 0.25, 0.20, 0.10, 0.05)$ for sizes $(2, 3, 4, 5, 6)$, respectively. The active set always includes the finalizer and one uniformly chosen solver, either agent 2 or agent 3; the remaining active roles are sampled uniformly without replacement. Processing active non-finalizer agents in reverse index order, we connect each to a uniformly chosen later active agent that already reaches the finalizer. Each remaining possible forward edge is then independently added with probability 0.3. Duplicate topologies are rejected and resampled.

For MMLU-Pro, we enumerate legal forward DAGs, exclude the seven predefined templates, and uniformly sample one topology at each active size from two through six. Thus, the five random candidates span five different team sizes.

Corpus	Template	Directed edges $(\text{source}, \text{target})$
MATH	Finalizer only	$\varnothing$
	Two-agent baseline	$(2,5)$
	Expert anchor	$(0,1),(1,2),(2,4),(3,4),(2,5),(3,5),(4,5)$
	Primary pipeline	$(0,1),(1,2),(2,4),(2,5),(4,5)$
	Dual-solver review	$(2,4),(3,4),(2,5),(3,5),(4,5)$
	Dual-solver direct	$(2,5),(3,5)$
	Planned primary	$(0,1),(1,2),(2,5)$
MMLU-Pro	Finalizer only	$\varnothing$
	Two-agent baseline	$(1,5)$
	Expert anchor	$(0,3),(1,3),(2,3),(1,4),(2,4),(3,4),(4,5)$
	Terminology contrast	$(0,3),(3,5)$
	Fact triangulation	$(1,4),(2,4),(4,5)$
	Independent evidence	$(0,5),(1,5),(2,5)$
	Distractor calibration	$(1,3),(3,4),(4,5)$

Table VI: The seven predefined initial topology candidates for each corpus. Agent indices refer to table V. The active set contains all listed edge endpoints together with the mandatory finalizer, agent 5. Five additional random DAGs complete the initial set of 12 candidates per problem.

All initial candidates allow edges only from lower to higher agent indices, guaranteeing acyclicity. The finalizer must be active, and every other active agent must have a directed path to it. Candidates are checked for valid edge endpoints, absence of self-loops, acyclicity, and finalizer reachability. Deduplication uses the active-agent set and the edge set.

Demonstration selection. Each initial candidate topology is evaluated over $K = 5$ executions. Empirical effectiveness $\widehat{Q}$ is the fraction of correct final answers, and execution cost $\widehat{C}$ is the mean total number of input and output tokens consumed by the execution agents. We select the lowest-cost topology among those with the highest observed effectiveness, yielding 900 training and 100 validation demonstrations per corpus.

Refreshed preference candidates. After initialization, we construct $\mathcal{C}_1(x)$ from four base topologies: the demonstration topology, the finalizer-only topology, the expert anchor, and the complete forward DAG over all six agents. We additionally generate three topologies using the initialization policy: one greedy output and two samples at temperature 0.7, with a maximum of 256 generated tokens each.

We generate three perturbed topologies from the six preceding candidates other than the complete forward DAG. For each source topology, we visit the five non-finalizer roles in a seeded random order and independently toggle each role's inclusion relative to the original source topology. Removing a role also removes its incident edges. After each toggle, we retain all surviving edges and add a direct edge from every remaining non-finalizer agent to the finalizer. Each perturbation therefore modifies node membership and communication dependencies while preserving acyclicity and finalizer reachability. We retain three valid, distinct perturbations.

All retained candidates are reevaluated over five executions using matched execution seeds within each problem. Execution statistics from initialization are not reused. Comparisons with incomplete executions or unparsed answers are excluded.

Preference acceptance. An effectiveness preference requires an empirical success-rate gap of at least 0.4, at least three paired wins, and no paired losses across the five executions. A paired win occurs when the preferred topology succeeds and the rejected topology fails; a paired loss is the reverse.

A cost preference requires both topologies to succeed in all five executions. The preferred topology must consume at least 20% fewer tokens on average and have lower token consumption in at least four paired executions. These finite-sample criteria screen execution evidence and do not constitute statistical significance or equivalence tests.

Pair selection and weighting. We retain at most three distinct preference pairs per problem: a local-cost comparison, an effectiveness comparison, and a same-size structural comparison. The local-cost comparison favors a smaller preferred team and prioritizes nearby team sizes. The effectiveness comparison prioritizes the preferred topology's observed effectiveness, followed by a smaller preferred team and a rejected team of nearby size. The same-size comparison holds team size fixed while allowing role composition and connectivity to differ, prioritizing effectiveness preferences over cost preferences. Structural proximity is used for subsequent tie-breaking, measured by $|V^+ \triangle V^-| + |E^+ \triangle E^-|$, with smaller values preferred. Already selected pairs are excluded, and a slot remains empty when no eligible comparison is available.

For pair i, we use

$$w_i = \frac{\alpha_{t_i}}{m_{x_i}}, \qquad \alpha_{\text{effectiveness}} = 1, \qquad \alpha_{\text{cost}} = 0.25, \tag{23}$$

where t_i denotes its preference type and m_{x_i} is the number of selected pairs for its problem.

Optimization. We adapt the topology generator using LoRA with rank 16 and scaling factor 32 on attention and feed-forward projections. Generator initialization uses a learning rate of 5×10^{-5} for 10 epochs, with checkpoint selection based on validation completion loss.

For preference alignment, we merge the initialization adapter into the backbone and retain the resulting model as the frozen reference. A separate policy is initialized from the same model and optimized through a fresh LoRA adapter using AdamW. We use a learning rate of 10^{-6}, $\beta = 0.1$, $\lambda_{\text{NLL}} = 0.1$, a microbatch size of one, 16 gradient accumulation steps, and at most 10 epochs. The objective combines weighted DPO with preferred-sequence negative log-likelihood.

Checkpoint selection. We evaluate checkpoints at epochs $\{2, 3, 5, 8, 10\}$ on the 100 validation problems, with five executions per problem. Only checkpoints that produce valid topologies, allowing one corrective generation, are eligible for selection. We select the checkpoint with the lowest average inference token cost among those within one percentage point of the best validation accuracy observed across the compared policies. If no checkpoint in a configuration meets this criterion, we select its highest-accuracy checkpoint. Token cost includes both topology generation and agent execution. All selections use validation results only.