TOPOLOGY AS EXPERIENCE: FEDERATED OPTIMIZATION ON COLLABORATIONS IN LLM-BASED MAS

Anonymous authors
Paper under double-blind review

ABSTRACT

Topology design for multi-agent systems built on large language models is typically optimized within individual clients, leaving useful collaboration experience from other clients underused. Sharing textual experience introduces growing records and the risk of exposing sensitive task content, while heterogeneous task distributions require client-specific collaboration strategies. To address these limitations, we propose a personalized federated framework that represents collaboration experience as a compact probabilistic topology graph. Specifically, each client combines a shared probabilistic graph that aggregates reusable communication patterns with a personalized probabilistic graph that adapts them to local tasks, while keeping raw tasks and execution traces local. To reduce the cost of topology optimization, a task-conditioned mixture-of-experts predictor learns from historical execution records and supplies inexpensive rewards for probabilistic graph training. Limited real multi-agent feedback then refines the topology policy through reinforcement learning to address the gap between predicted and executed performance. At deployment, the predictor ranks sampled topologies and selects one for execution. Across two heterogeneous three-client suites spanning coding, mathematics, and reasoning, the complete framework achieves the highest mean accuracy in both suites. Predictor pretraining improves the starting topology policy, limited real feedback further refines it, and personalized candidate search reaches 90.0% accuracy with eight candidates in a setting where the shared policy requires 512. These results show how a probabilistic graph turns cross-client collaboration experience into executable, client-adaptive policies.

1 INTRODUCTION

Large language model (LLM)-based multi-agent systems (MAS) combine the capabilities of specialized agents through information exchange and mutual verification to solve complex tasks (Li et al., 2023; Wu et al., 2024b; Hong et al., 2024). However, collaboration quality depends on how the agents are organized: indiscriminately adding agents or extending interaction rounds does not necessarily improve task performance (Wang et al., 2025). Redundant agents and communication increase inference cost (Zhang et al., 2025b), while poorly designed information flows can propagate errors and reduce answer accuracy (Shen et al., 2025). A MAS topology specifies which agents participate, how they exchange information, and how their intermediate results are combined. Thus, designing an effective collaboration topology is essential for realizing the potential of MAS.

Motivated by this, recent work has developed several approaches to automatically optimize MAS topologies. GPTSwarm (Zhuge et al., 2024) and G-Designer (Zhang et al., 2025c) optimize communication graphs; AgentPrune (Zhang et al., 2025b) and AgentDropout (Wang et al., 2025) remove redundant interactions or agents; and AFlow (Zhang et al., 2025d) and MaAS (Zhang et al., 2025a) search executable workflows or distributions over agentic architectures. These methods learn useful collaboration structures from task feedback, primarily within an individual task collection or deployment. Consequently, clients optimizing independently may waste their limited executions rediscovering patterns uncovered by other clients, with no effective communication available between clients handling similar tasks.

An example is given in Figure 1, in which client A hires LLM agents with complementary roles from the Internet of Agents and organizes them with a specific topology for its local tasks. Suppose

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.

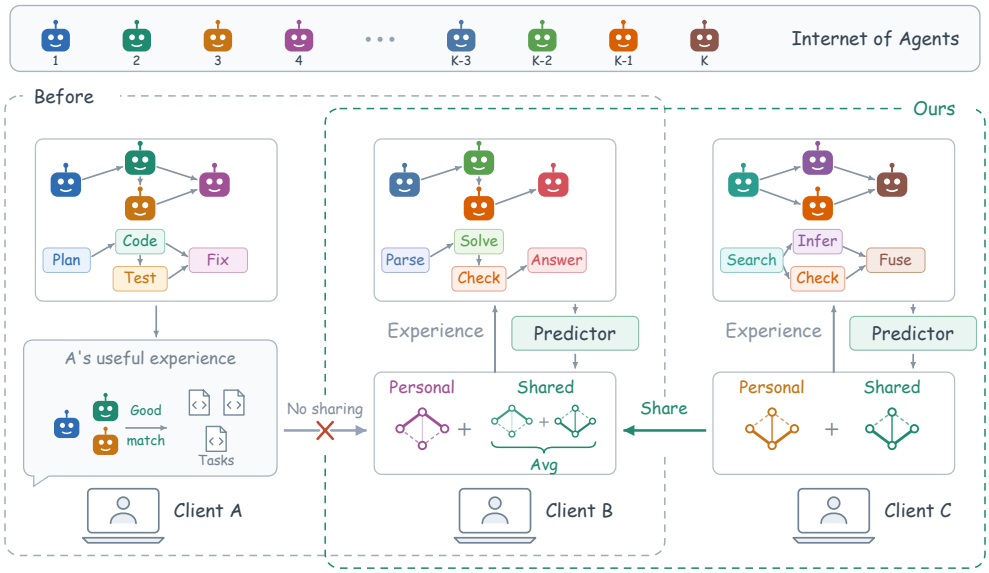


Figure 1: Topology as experience. Left: local collaboration experience remains isolated. Right: clients share probabilistic graphs while retaining personalized structure, with predictors supporting low-cost topology learning. The overlapping groups share the middle client.

that A discovers an effective collaboration pattern for a recurring type of subtask, including which agents to hire and how they should exchange intermediate results. Without an experience-sharing mechanism, this discovery remains local. When client B later encounters a similar task, B may hire comparable agents, but cannot directly reuse A's experience to construct its agent network and topology. It must therefore evaluate alternative collaborations from scratch. This motivates our study of *federated topology optimization*: enabling each client to improve its local topology policy while sharing reusable collaboration experience with other clients. Our goal is to let each client improve its local topology policy while periodically sharing useful collaboration experience, so that clients can benefit from collective discoveries and still organize agents for their own tasks.

Achieving this goal raises three questions, each paired with a corresponding design choice:

- **What should be learned as collaboration experience among clients?** In previous works, textual records are attractive because they can describe which agents participated, how information was exchanged, and why a collaboration pattern succeeded(Wu et al., 2024a). However, textual records grow as experience accumulates and may mix reusable advice with sensitive task content(Yao et al., 2026). Repeated records may also contain redundant or task-specific descriptions, making it difficult to identify which parts can be safely reused by another client. We instead represent experience as a probabilistic graph: each legal communication edge has a learnable inclusion probability, and the aligned edge parameters compactly describe a family of executable topologies. With a common agent pool and communication schema, this fixed-size representation can be aggregated and sampled directly without transferring the original textual task history.

- **How can topology policies be learned efficiently?** Evaluating candidate topologies through real MAS execution is costly, since each candidate may require multiple agent calls and intermediate information exchanges. Exploring a large topology space with such feedback can therefore consume a substantial number of executions before a useful policy is found. To reduce this cost, we design a task-conditioned predictor that is trained on historical execution records and supplies inexpensive proxy rewards for topology optimization. Because the predictor only approximates the outcome of a real MAS execution, its prediction may deviate from the observed performance of newly sampled topologies. A small amount of real execution feedback then refines the policy through reinforcement learning, allowing the learned policy to correct this prediction error while keeping the number of real executions small.

- **What should be shared across heterogeneous clients?** Blindly averaging all topology preferences can obscure structures that are useful to particular clients, whereas keeping every topology parameter local prevents clients from benefiting from collaboration patterns discovered elsewhere. Considering each client should share collaboration experience to the extent that it is useful to the other clients while retaining adaptations to its own task distribution, we design a personalized federated sharing mechanism. Specifically, each client maintains a shared probabilistic graph for reusable cross-client structure and a personalized probabilistic graph for local preferences. Federated aggregation updates the shared graph, while interpolation with the personalized graph retains client-specific structure. In this way, the shared component transfers common collaboration experience, whereas the personalized component preserves adaptations to each client's task distribution.

Our contributions can be summarized as follows. First, we study federated topology optimization as a way for clients to reuse collaboration experience discovered by others, extending topology learning beyond independent or centralized optimization. Second, we develop a personalized framework that represents experience with probabilistic graphs, uses predictor-based rewards to reduce the cost of topology search, and refines client policies with real MAS feedback. Experiments across heterogeneous clients achieve the highest mean accuracy on both workflow suites. In a candidate-budget study, personalized search reaches 90.0% accuracy with eight candidates, matching the shared policy with a $64\times$ smaller candidate budget.

2 Related Work

The related work is organized from two complementary perspectives. We first review multi-agent collaboration and cross-client experience sharing to clarify how our research problem differs from prior work. We then discuss probabilistic topology representations, performance prediction, and personalized federated learning, which are closely related to our design.

Multi-agent collaboration and experience sharing. Automated MAS design learns how agents should collaborate. GPTSwarm optimizes node prompts and graph connectivity (Zhuge et al., 2024), while G-Designer generates task-aware communication graphs with a variational graph autoencoder (Zhang et al., 2025c). AgentPrune removes redundant spatial–temporal communication, and AgentDropout additionally eliminates redundant agents across rounds (Zhang et al., 2025b; Wang et al., 2025). AFlow searches code-represented workflows through Monte Carlo tree search (Zhang et al., 2025d). Beyond modifying a template, ARG-Designer generates agent roles and connections autoregressively (Li et al., 2026a); OFA-MAS learns a multi-domain graph generator (Li et al., 2026b), and CARD conditions topology design on runtime environments (Wu et al., 2026). These methods advance collaboration design.

Beyond these single-client topology optimization approaches, cross-client learning frameworks share various forms of agent experience. FICAL exchanges knowledge compendia for tool use (Wu et al., 2024a), while Federation over Text distills reasoning summaries into a shared insight library (Yao et al., 2026). FedWorld transfers abstract transition rules according to their client-specific applicability (Hou, 2026). Closer to our problem, DAWN learns distributed workflow generators using SVD-based update alignment and prototype-graph regularization (Wan et al., 2026). Our framework instead shares aligned communication-edge parameters and retains personalized topology distributions, making the probabilistic graph itself a reusable representation of collaboration experience.

Probabilistic topology modeling Learning distributions over graphs enables optimization of discrete structures. Franceschi et al. learn edge distributions for graph neural networks through bilevel optimization (Franceschi et al., 2019). Within MAS, GPTSwarm optimizes edge-inclusion probabilities with policy gradients (Zhuge et al., 2024). MaAS learns a layered distribution over agentic operators and uses a query-conditioned controller to sample architectures (Zhang et al., 2025a). Graph-GRPO uses group-relative feedback for edge-level credit assignment (Cang et al., 2026). Our edge parameters encode client-level communication preferences in a common coordinate system for aggregation and personalization.

Performance prediction. Performance prediction reduces the cost of evaluating candidate structures: FLOW-GNN combines workflow-graph and task embeddings (Zhang et al., 2025e); Agentic

Predictor integrates graph, code, and prompt views (Trirat et al., 2026); and GLOW combines GNN structural representations with LLM-derived workflow semantics (Guan et al., 2025). SCALE calibrates LLM-predicted workflow scores with few-shot executions (Wang et al., 2026). We combine predictor-based proxy rewards and deployment candidate ranking with limited real MAS feedback for refining local topology policies.

Personalized federated learning. Federated learning aggregates local updates while retaining training data on clients (McMahan et al., 2017). Global–local formulations balance shared learning with client-specific objectives (Hanzely & Richtárik, 2020). APFL learns mixtures of local and global models (Deng et al., 2020), while Ditto regularizes personalized models toward a global solution (Li et al., 2021). We apply the shared–personalized principle to MAS communication policies: clients aggregate aligned topology logits and interpolate the shared graph with their locally learned parameters. This transfers collaboration experience while preserving preferences induced by each client's task distribution.

3 Problem Setup

Consider C clients interacting with an Internet of Agents comprising N LLM agents. For each task, the multi-agent collaboration process can be decomposed into at most L ordered communication transitions. Client k owns a local task distribution $\mathcal{D}_k$ and execution history $\mathcal{H}_k = \{(x_i, G_i, y_i)\}$, where x_i is a task, G_i is an executed collaboration topology, and $y_i \in \{0, 1\}$ denotes correctness. A historical corpus $\mathcal{H}$ supplies records for pretraining the task–topology predictor, while $\mathcal{H}_k$ denotes client-local deployment records. The server aggregates aligned topology parameters rather than raw tasks, prompts, or traces; current evaluation questions and deployment traces are outside $\mathcal{H}$.

A binary schema $\mathbf{M} \in \{0, 1\}^{L\times N\times N}$ specifies legal directed edges. Client k has topology logits $\mathbf{\Theta}_k \in \mathbb{R}^{L\times N\times N}$ inducing policy $\pi_{\mathbf{\Theta}_k}$. With predictor guidance, the client samples K valid graphs for task x, scores them using predictor $f_\phi(x, G)$, and executes the highest-scoring graph:

$$G_{1:K} \sim \pi_{\mathbf{\Theta}_k}(\cdot \mid \mathbf{M}), \qquad \widehat{G}(x) = \operatorname*{arg\,max}_{G\in G_{1:K}} f_\phi(x, G),$$
$$J = \frac{1}{C}\sum_{k=1}^{C} \mathbb{E}_{x\sim\mathcal{D}_k}\big[u(x, \widehat{G}(x))\big], \tag{1}$$

Here $u(x, G)$ is task correctness after executing G. Without predictor guidance, we use $K = 1$ and $\widehat{G}(x) = G_1$, executing the sampled graph without evaluating f_ϕ. We maximize J through cross-client sharing and personalization, reporting macro-average and per-client accuracy. In our parameterization, we identify the problem-level tensor Θ_k with the personalized tensor $\mathbf{P}_k$; $\mathbf{S}$ is shared, $\mathbf{U}_k$ is the local update, and $0 \le \beta \le 1$.

Why personalization is necessary. Heterogeneous clients can prefer opposite values for the same communication edge. If two local policies assign logits $+a$ and $-a$ to edge e, the averaged logit and its Bernoulli sampling parameter are

$$\bar{\theta}_e = \tfrac{1}{2}(a - a) = 0, \qquad \bar{p}_e = \sigma(0) = 0.5, \tag{2}$$

which erases both preferences. We therefore separate a shared probabilistic topology graph $\mathbf{S}$ from client-specific personalized probabilistic graphs $\mathbf{P}_k$. Their interpolation transfers common structure while retaining a distinct graph distribution for each client.

4 Personalized Federated Topology Optimization

Our framework (Figure 2) learns a distribution over executable MAS topologies. A predictor trained on historical workflows guides local topology optimization; clients share aligned graph parameters and refine their policies with limited real MAS feedback before deployment.

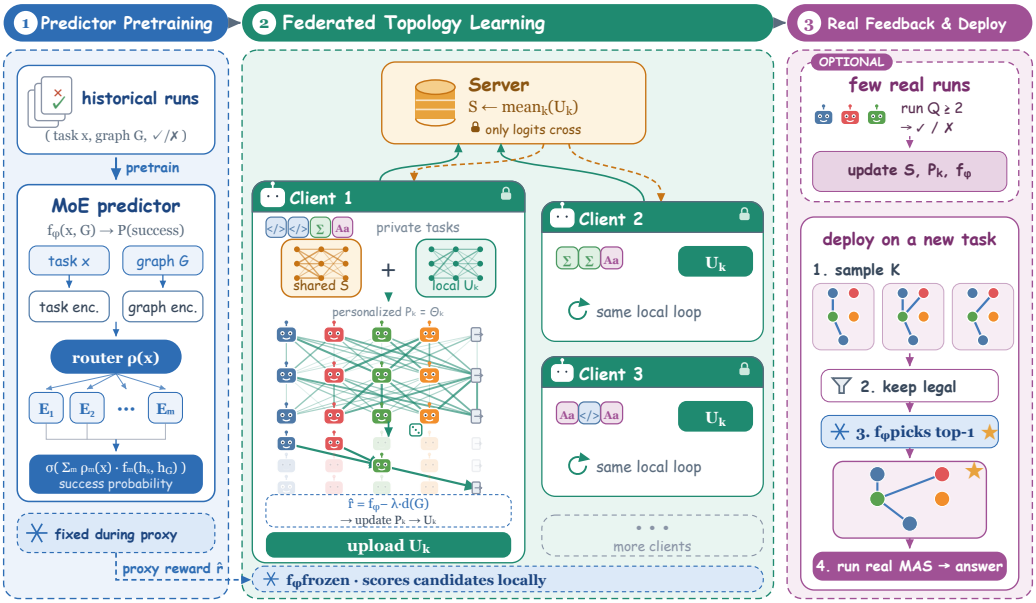


Figure 2: Complete configuration. Historical executions train the predictor; clients use proxy rewards to learn shared and personalized probabilistic graphs. Real feedback refines topology policies and local predictors. At deployment, the predictor ranks sampled topologies and selects one for execution.

4.1 Topology Space and Probabilistic Graph

We consider an Internet of Agents comprising N agents. Each task may require at most L ordered communication transitions. The suite-specific schema defines legal directed edges $\mathcal{E}_\ell \subseteq \{1, \ldots, N\}^2$ at each transition, with $\mathcal{E} = \bigcup_{\ell=0}^{L-1} \mathcal{E}_\ell$. An executable topology is a directed graph obtained by selecting legal edges across these transitions and satisfying the reachability, agent-instance, and terminal constraints of the MAS executor. Let $\theta_\ell(i, j)$ be the logit associated with legal edge $(i, j) \in \mathcal{E}_\ell$. The collection

$$\mathcal{S}_\theta = \{\theta_\ell(i, j) : (i, j) \in \mathcal{E}_\ell, \ell = 0, \ldots, L-1\} \tag{3}$$

parameterizes the probabilistic topology graph. It is a compact parameterization of a large family of candidate graphs: each legal edge has a Bernoulli sampling parameter

$$p_\theta(\ell, i, j) = \sigma\left(\frac{\theta_\ell(i, j)}{\tau}\right), \tag{4}$$

where τ is the sampling temperature. The probabilistic graph induces a distribution over sequential Bernoulli proposals. The sampler starts from the input agent, selects at least one outgoing edge, and traverses the reachable frontier in transition order. Schema masks and agent-instance constraints determine which edges are eligible; terminal and edge-budget checks retain executable graphs. The resulting proposal log probability, denoted $\log \widetilde{\pi}_\theta(G)$, accumulates the sampled edge decisions. Appendix A.1 gives its expression and the full sampling procedure.

4.2 Shared and Personalized Topology Policies

Client i maintains a local topology policy initialized from the same schema and coordinate system. After local optimization in communication round t, let $\mathbf{U}_i^t$ denote its updated topology-logit tensor. The server aggregates these aligned tensors:

$$\mathbf{S}^{t+1} = \frac{1}{C} \sum_{i=1}^{C} \mathbf{U}_i^t. \tag{5}$$

The client then forms its personalized policy by interpolating the shared parameters with its local update:

$$\mathbf{P}_i^{t+1} = (1 - \beta)\mathbf{S}^{t+1} + \beta \mathbf{U}_i^t, \qquad p_i(\ell, j, k) = \sigma\left(\frac{\mathbf{P}_{i,\ell}(j, k)}{\tau}\right). \tag{6}$$

Here β controls the weight of client-specific structure. Setting $\beta = 0$ gives a single shared policy, while larger values preserve more of the local policy. The server aggregates in the common edge coordinate system; raw tasks, prompts, and execution traces are not exchanged.

These equations define four settings: *Shared Only* uses $\beta = 0$ and one sampled graph; *Shared + Predictor* adds proxy optimization and ranking; *Shared + Personalized* uses $\beta > 0$ without predictor guidance; and the complete setting enables both. All share the same graph representation and aggregation rule.

4.3 Task-Conditioned Predictor

Real MAS execution provides the most direct topology outcome, but evaluating many candidate graphs for every optimization step is expensive. We therefore pretrain a predictor $f_\phi(x, G)$ from historical task–topology–outcome triples $\mathcal{H} = \{(x_r, G_r, y_r)\}$. It stays fixed during proxy optimization; real feedback adapts a separate copy at each client without predictor aggregation. We omit the client index on ϕ.

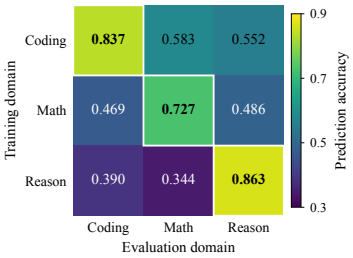


Figure 3: Cross-family predictor accuracy. Values are fractions (0–1); diagonal cells are in-domain and off-diagonal cells are cross-domain.

The transfer matrix shows domain-sensitive task–topology relationships: single-domain predictors reach 0.809 mean in-domain accuracy but only 0.471 mean cross-domain accuracy. This gap motivates a task-conditioned router that can route individual tasks to different experts with a common pretrained initialization.

The task and graph encoders map x and G to representations h_x and h_G; four expert heads produce correctness logits, and a task-conditioned router mixes their outputs:

$$\rho(x) = \mathrm{softmax}(W_r h_x), \quad f_\phi(x, G) = \sigma\left(\sum_{m=1}^{E} \rho_m(x) f_m(h_x, h_G)\right). \tag{7}$$

The predictor is trained with binary cross entropy. Writing $s_r = f_\phi(x_r, G_r)$,

$$\mathcal{L}_{\mathrm{pred}}(\phi) = -\frac{1}{|\mathcal{H}|} \sum_r \ell(y_r, s_r), \quad \ell(y, s) = y \log s + (1 - y) \log(1 - s). \tag{8}$$

4.4 Proxy-Guided Optimization

For a local task x, client i samples B valid topologies from the accepted distribution $\pi_{\mathbf{P}_i}$ (Appendix A.1). The frozen predictor supplies a proxy reward,

$$\widehat{r}(x, G) = f_\phi(x, G) - \lambda d(G), \tag{9}$$

where $d(G)$ is the number of selected edges divided by the number of schema-legal edge positions and λ is a communication-density weight. We use $\lambda = 20.0$ for both suites and define

$$d(G) = \frac{|E(G)|}{\sum_{\ell=0}^{L-1} |\mathcal{E}_\ell|}. \tag{10}$$

This density term is used during proxy optimization only. Real-feedback updates use the judged rewards y_q, and deployment ranking uses $f_\phi(x, G)$ alone. For candidates sampled for the same task, a leave-one-out baseline reduces the effect of task difficulty. With $b_q = (B - 1)^{-1} \sum_{q' \neq q} \widehat{r}_{q'}$, the policy update is

$$\mathcal{L}_{\mathrm{proxy}} = -\frac{1}{B} \sum_{q=1}^{B} (\widehat{r}_q - b_q) \log \widetilde{\pi}_{\mathbf{P}_i}(G_q). \tag{11}$$

This update increases the probability of graphs that the predictor expects to solve the task with moderate communication density. The optimization is local; only the resulting topology parameters participate in the next federation round.

4.5 Real-Feedback Refinement and Deployment

During proxy optimization, the predictor scores candidate topologies without additional MAS executions, but its ranking can differ from real MAS performance. We therefore use a small calibration cohort on each client. For each calibration task, the client independently samples and executes $Q \geq 2$ valid topologies from its current policy to obtain judged rewards y_q. These on-policy samples provide the leave-one-out real advantage

$$a_q = y_q - \frac{1}{Q-1} \sum_{q' \neq q} y_{q'}. \tag{12}$$

We substitute a_q for the proxy advantage in Equation 11 to update the local topology parameters. The resulting client updates are aggregated into the shared graph using Equation 5, and client policies are formed using Equation 6. When predictor guidance is enabled, the same execution records (x, G_q, y_q) also train the local f_ϕ with the binary cross-entropy objective in Equation 8. Real feedback therefore updates the shared graph, the personalized component when enabled, and the predictor when enabled.

At deployment, the learned parameters remain fixed. With predictor guidance, a client samples K valid graphs from its topology policy, ranks them using the adapted predictor, and executes the highest-scoring graph. Without predictor guidance, it directly executes one valid sampled graph. Task-dependent candidate scoring and client-specific topology distributions jointly determine the deployed collaboration.

5 Experiments

We evaluate the framework by measuring task accuracy across heterogeneous clients while varying predictor pretraining, the amount of real MAS feedback, personalization, and the candidate budget. We further examine how these factors affect topology selection and candidate sampling efficiency.

5.1 Experimental Setup

Tasks and client splits. AF uses the AFlow/FLORA workflow source with HumanEval, MBPP, GSM8K, and MMLU (Chen et al., 2021; Austin et al., 2021; Cobbe et al., 2021; Hendrycks et al., 2021a). GD uses the G-Designer source with HumanEval, MBPP, MATH, and MMLU (Hendrycks et al., 2021b). Each suite has three non-IID clients, named Code-majority, Math-majority, and Reason-majority by their dominant task family. Every client contains minority tasks from the other two families and uses 20 local training tasks and 103 disjoint evaluation tasks (Table 1). All methods within a suite use the same evaluation tasks; the two suites retain their respective agent pools and historical workflow corpora. Here Code comprises HumanEval and MBPP, Math is GSM8K for AF and MATH for GD, and Reason is MMLU. The majority labels describe the dominant family while each client retains the fixed minority mixture shown in Table 1.

Table 1: Client task mixtures. Counts are Code / Math / Reason.

Client	Training	Evaluation
Code-majority	16 / 2 / 2	83 / 10 / 10
Math-majority	2 / 16 / 2	10 / 83 / 10
Reason-majority	2 / 2 / 16	10 / 10 / 83

The FLORA records underlying these suites were collected with GPT-4o-mini. We therefore use GPT-4o-mini for every method in the matched main comparison, preserving consistency with the source data-collection setting. All methods within a suite also share the same task judge. Code is evaluated with dataset unit tests, multiple-choice tasks by the final selected option, and mathematical tasks by normalized numeric or symbolic equivalence. For every method and feedback budget, the scored answer is the terminal output produced by the suite-native executor; the same rule is used in the main comparison and the feedback-budget study.

Baselines. Vanilla directly prompts one agent, and CoT elicits explicit chain-of-thought reasoning (Wei et al., 2022). AFlow uses feedback-guided tree search to generate and optimize executable workflows (Zhang et al., 2025d). G-Designer learns task-adaptive communication with a graph neural network (Zhang et al., 2025c). AgentDropout learns sparse collaboration structures by eliminating redundant agents and edges (Wang et al., 2025).

Cross-client configurations. The four cross-client rows in Table 2 independently enable personalization and predictor guidance as defined in Section 4.2. Section 5.3 separately evaluates real-feedback refinement of a predictor-pretrained policy. The same suite-native terminal output rule is used for every row and for the feedback-budget study.

Implementation. The predictor uses graph convolutions, 384-dimensional task embeddings, a 512-dimensional hidden representation, and four softly routed experts. Proxy training runs for 60 rounds with five local steps, 16 tasks, and 16 topologies per task; Adam uses learning rates 0.05 and 0.1 for proxy and real-feedback topology updates. The complete configuration uses $\beta = 0.9$ and ranks $K = 8$ valid candidates before one MAS execution. Full settings appear in Appendices B–E.

5.2 Overall Performance

Table 2: Task accuracy (%) under the matched evaluation protocols. Vanilla and CoT use single-agent inference; the remaining rows use MAS topology execution. Each client has 103 evaluation tasks; Avg is the unweighted client mean. Best is bold; second best is underlined.

	AF				GD			
Method	Code-majority	Math-majority	Reason-majority	Avg	Code-majority	Math-majority	Reason-majority	Avg
Single-agent inference								
Vanilla	<u>78.64</u>	<u>92.23</u>	75.73	82.20	62.14	71.84	77.67	70.55
CoT	**80.58**	**93.20**	77.67	<u>83.82</u>	<u>64.08</u>	70.87	<u>81.55</u>	72.17
Single-client MAS topology design								
AFlow	75.73	83.50	<u>78.64</u>	79.29	59.22	**79.61**	77.67	72.17
G-Designer	72.82	91.26	71.84	78.64	57.28	69.90	76.70	67.96
AgentDropout	69.90	<u>92.23</u>	72.82	78.32	59.22	72.82	77.67	69.90
Cross-client topology sharing								
Shared Only	77.67	91.26	66.99	78.64	**65.05**	60.19	78.64	67.96
Shared + Predictor	<u>78.64</u>	91.26	76.70	82.20	63.11	72.82	79.61	71.84
Shared + Personalized	<u>78.64</u>	<u>92.23</u>	67.96	79.61	62.14	<u>75.73</u>	80.58	<u>72.82</u>
Shared + Personalized + Predictor	**80.58**	**93.20**	**79.61**	**84.47**	62.14	74.76	**82.52**	**73.14**

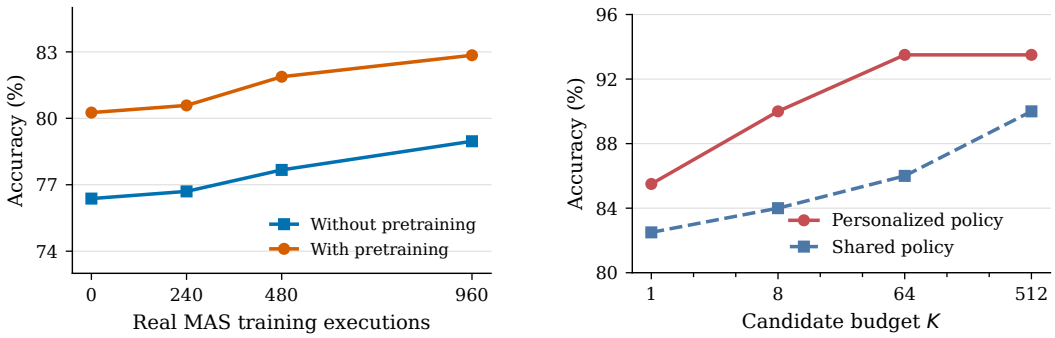
The complete framework achieves the highest mean accuracy among the evaluated methods on both suites (Table 2). It improves over the strongest non-sharing reference by 0.65/0.97 points and over Shared + Predictor by 2.27/1.30 points (AF/GD).

The cross-client configurations reveal complementary roles for personalization and predictor guidance. Adding either component to Shared Only improves mean accuracy in both suites, and combining them gives the strongest result. Their relative contributions differ across suites: predictor guidance provides the larger individual gain on AF, whereas personalization contributes more on GD. At the client level, personalization most strongly improves the GD Math-majority client, while predictor guidance substantially improves the AF Reason-majority client. These patterns suggest that retaining local topology preferences and learning to evaluate task–topology compatibility address different aspects of heterogeneity, motivating their joint use in the complete framework.

5.3 Training and Deployment Analysis

We compare predictor-pretrained and unpretrained topology policies under the same real-feedback schedule. Both arms start from the same pretrained predictor; the difference is whether the topology policy first receives predictor-based proxy optimization. Each round executes four independent topology samples per local training task. Rewards update the shared graph through local optimization and aggregation; labels adapt each client's predictor. Four rounds use 960 training executions. At evaluation, local predictors rank $K = 8$ valid candidates with all parameters held fixed.

On AF (Figure 4(a)), pretraining improves mean accuracy at every measured budget: the pretrained policy starts at 80.26%, above the unpretrained policy after 960 executions (78.96%), and reaches 82.85% after feedback refinement. Figure 4(b) compares predictor-ranked budgets



(a) Predictor pretraining **(b) Candidate budget**

Figure 4: Training and deployment diagnostics. (a) AF mean accuracy (%) across real-feedback executions with and without pretraining. (b) GD Reason-majority accuracy (%) versus candidate budget K; the shared and personalized policies use the same diagnostic, separate from the three-client sweep in Appendix D.

$K \in \{1, 8, 64, 512\}$ in a separate GD Reason-majority diagnostic, with one MAS execution per task. Personalization reaches 90.0% at $K = 8$, matching shared at $K = 512$, and reaches 93.5% at $K = 64$. Appendix D reports the three-client sweep and deployment costs.

Refinement improves mean accuracy by 2.59 percentage points on AF and 1.94 on GD, with corresponding worst-client gains of 2.91 and 1.94 points (Table 3). The AF trajectories further show that the pretrained policy starts with higher accuracy and maintains its advantage as real feedback improves its performance. Together, these results support initializing topology policies from historical experience and refining them with local feedback. Appendix C reports the complete trajectories and paired changes.

Table 3: Real-feedback refinement under Shared + Predictor ($\beta = 0$). Values are client accuracies (%); Avg/Worst are the unweighted mean/minimum.

Suite	Executions	Code-majority	Math-majority	Reason-majority	Avg	Worst
AF	0	76.70	91.26	72.82	80.26	72.82
AF	960	80.58	92.23	75.73	82.85	75.73
GD	0	62.14	69.90	77.67	69.90	62.14
GD	960	64.08	70.87	80.58	71.84	64.08

Transfer to a new executor. We transfer the GPT-4o-mini-trained predictor to Qwen3-8B and adapt it with 240 target-executor outcomes. Direct graph sampling, frozen predictor ranking, and adapted predictor ranking achieve mean accuracies of 79.29%, 82.52%, and 82.85% on AF, and 72.17%, 72.82%, and 73.14% on GD, respectively. All three arms share eight-candidate pools and execute one topology per task. Predictor ranking transfers across executors, and target feedback further improves selection; Appendix G gives per-client results and costs.

6 Conclusion

This work develops a personalized federated framework for turning multi-agent collaboration experience into executable, client-adaptive topology policies. The framework addresses a gap in existing topology optimization and experience sharing methods, which generally learn within individual clients or exchange textual records rather than sharing compact, structured topology knowledge. To address this gap, we use probabilistic topology graphs to represent collaboration experience, maintain shared and personalized graph parameters for federated adaptation, and combine predictor-based proxy feedback with limited real MAS feedback for efficient policy optimization. Experiments on two heterogeneous three-client workflow suites show that the complete framework achieves the best average accuracy, with predictor guidance, real-feedback refinement, and personalization contributing to topology selection. Future work will extend the framework to dynamic agent pools, more heterogeneous client distributions, and stronger privacy-preserving aggregation.

AI USE STATEMENT

AI tools were used for literature search, implementation of experimental code, manuscript drafting, and language polishing. The core research idea and research direction were established by the authors. All AI-generated outputs, including code and text, were manually reviewed by the authors, who take full responsibility for the paper and its results.

REPRODUCIBILITY STATEMENT

Our implementation and experimental configurations are available in an anonymous repository at https://anonymous.4open.science/r/federated_supernet-1B53. Algorithm 1 describes the complete configuration, while Appendix A details topology sampling and personalized aggregation. Data sources, client construction, model configurations, optimization hyperparameters, random seeds, and evaluation criteria are documented in Appendix B. Complete real-feedback trajectories and candidate-budget experiments, together with their experimental settings, are provided in Appendices C and D.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021. doi: 10.48550/arXiv.2108.07732.

Yueyang Cang, Xiaoteng Zhang, Erlu Zhao, Zehua Ji, Yuhang Liu, Yuchen He, Zhiyuan Ning, Yijun Chen, 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. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.1010. URL https://aclanthology.org/2026.findings-acl.1010/.

Mark Chen et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. doi: 10.48550/arXiv.2107.03374.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. doi: 10.48550/arXiv.2110.14168.

Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461*, 2020. doi: 10.48550/arXiv.2003.13461.

Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. Learning discrete structures for graph neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 1972–1982. PMLR, 2019. URL https://proceedings.mlr.press/v97/franceschi19a.html.

Wei Guan, Jian Cao, Jinyu Cai, Qiqi Cai, Jianqi Gao, and See-Kiong Ng. GLOW: Graph-language co-encoding for agentic workflow performance prediction. *arXiv preprint arXiv:2512.15751*, 2025. doi: 10.48550/arXiv.2512.15751. URL https://arxiv.org/abs/2512.15751.

Filip Hanzely and Peter Richtárik. Federated learning of a mixture of global and local models. *arXiv preprint arXiv:2002.05516*, 2020. doi: 10.48550/arXiv.2002.05516.

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*, 2021a. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

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. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021b. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VtmBAGCN7o.

Yuchao Hou. FedWorld: Scope-aware federation of agent world models. *arXiv preprint arXiv:2608.01561*, 2026. doi: 10.48550/arXiv.2608.01561. URL https://arxiv.org/abs/2608.01561.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: Communicative agents for "mind" exploration of large language model society. In *Advances in Neural Information Processing Systems*, volume 36, pp. 51991–52008, 2023. doi: 10.52202/075280-2264. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/a3621ee907def47c1b952ade25c67698-Abstract-Conference.html.

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. doi: 10.1609/aaai.v40i28.39481. URL https://ojs.aaai.org/index.php/AAAI/article/view/39481.

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. Association for Computing Machinery, 2026b. doi: 10.1145/3774904.3792537. URL https://doi.org/10.1145/3774904.3792537.

Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 6357–6368. PMLR, 2021. URL https://proceedings.mlr.press/v139/li21h.html.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54, pp. 1273–1282. PMLR, 2017. URL https://proceedings.mlr.press/v54/mcmahan17a.html.

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. 12347–12361. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.623. URL https://aclanthology.org/2025.emnlp-main.623/.

Patara Trirat, Wonyong Jeong, and Sung Ju Hwang. Multi-view encoders for performance prediction in LLM-based agentic workflows. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=7oeKDZsmWp.

Guancheng Wan, Mo Zhou, Ziyi Wang, Xiaoran Shang, Eric Hanchen Jiang, Guibin Zhang, Jinhe Bi, Yunpu Ma, Zaixi Zhang, Ke Liang, and Wenke Huang. DAWN: Distributed LLM multi-agent workflow synthesis. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(31): 26099–26106, 2026. doi: 10.1609/aaai.v40i31.39812.

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. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.1170. URL https://aclanthology.org/2025.acl-long.1170/.

Zixu Wang, Bingbing Xu, Yige Yuan, Huawei Shen, and Xueqi Cheng. Do we always need query-level workflows? rethinking agentic workflow generation for multi-agent systems. *arXiv preprint arXiv:2601.11147*, 2026. doi: 10.48550/arXiv.2601.11147.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Panlong Wu, Kangshuo Li, Junbao Nan, and Fangxin Wang. Federated in-context LLM agent learning. *arXiv preprint arXiv:2412.08054*, 2024a. doi: 10.48550/arXiv.2412.08054.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024b. URL https://openreview.net/forum?id=BAakY1hNKS.

Tongtong Wu, Yanming Li, Ziye Tang, Chen Jiang, Linhao Luo, Guilin Qi, Shirui Pan, and Gholamreza Haffari. CARD: Towards conditional design of multi-agent topological structures. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=JgvJdICc6P.

Dixi Yao, Tahseen Rabbani, Manzil Zaheer, and Tian Li. Federation over text: Insight sharing for multi-agent reasoning. *arXiv preprint arXiv:2604.16778*, 2026. doi: 10.48550/arXiv.2604.16778.

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. Multi-agent architecture search via agentic supernet. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 75834–75852. PMLR, 2025a. URL https://proceedings.mlr.press/v267/zhang25bi.html.

Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Xu Yu, and Tianlong Chen. Cut the crap: An economical communication pipeline for LLM-based multi-agent systems. In *International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=LkzuPorQ5L.

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*, volume 267, pp. 76678–76692. PMLR, 2025c. URL https://proceedings.mlr.press/v267/zhang25cu.html.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, XiongHui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. In *International Conference on Learning Representations*, pp. 34040–34077, 2025d. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/5492ecbce4439401798dcd2c90be94cd-Abstract-Conference.html.

Yuanshuo Zhang, Yuchen Hou, Bohan Tang, Shuo Chen, Muhan Zhang, Xiaowen Dong, and Siheng Chen. FLORA: GNNs as predictors of agentic workflow performances. In *The Fourth Learning on Graphs Conference*, 2025e. URL https://openreview.net/forum?id=RmlDI5kUxa.

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*, volume 235, pp. 62743–62767. PMLR, 2024. URL https://proceedings.mlr.press/v235/zhuge24a.html.

A Additional Method Details

Algorithm 1 describes the complete configuration, from historical predictor fitting and federated topology optimization to local real-feedback refinement and deployment.

Algorithm 1 Complete configuration: topology optimization and deployment

Require: Historical records $\mathcal{H}$; local training sets $\{\mathcal{D}_i\}_{i=1}^{C}$; schema $\mathcal{E}$; rounds T; local steps J; coefficient β

Fit predictor f_ϕ on $\mathcal{H}$ using Equation 8
Initialize aligned client tensors $\mathbf{P}_i$ and shared tensor $\mathbf{S}$
for $t = 0, \ldots, T-1$ **do**
  for each client i in parallel **do**
    for $j = 1, \ldots, J$ **do**
      Sample local tasks from $\mathcal{D}_i$ and B valid graphs for each task
      Score graphs with frozen f_ϕ; update $\mathbf{P}_i$ using Equation 11
    end for
    Send locally updated topology tensor $\mathbf{U}_i^t$ to the server
  end for
  $\mathbf{S}^{t+1} \leftarrow C^{-1} \sum_i \mathbf{U}_i^t$
  Each client forms $\mathbf{P}_i^{t+1} \leftarrow (1-\beta)\mathbf{S}^{t+1} + \beta \mathbf{U}_i^t$
end for
Initialize each client's local predictor from the pretrained f_ϕ
for each real-feedback round **do**
  for each client i in parallel **do**
    for each local training task $x \in \mathcal{D}_i$ **do**
      Independently sample Q valid graphs from the current policy
      Execute the graphs and obtain judged rewards y_q
      Compute real advantages a_q using Equation 12
    end for
    Update local topology logits using the real policy loss; retain Adam moments
    Update the local predictor using execution labels and Equation 8
    Send the updated topology tensor $\mathbf{U}_i$ to the server
  end for
  $\mathbf{S} \leftarrow C^{-1} \sum_i \mathbf{U}_i$
  Each client forms $\mathbf{P}_i \leftarrow (1-\beta)\mathbf{S} + \beta \mathbf{U}_i$
end for
Deployment for a task x at client i:
Sample K valid graphs; select $G^* \leftarrow \arg\max_G f_\phi(x, G)$
Execute G^* and return the task-specific final answer

A.1 Sequential topology proposal

At each transition, the sampler evaluates legal outgoing edges from the reachable frontier. The first transition must select an outgoing edge; later transitions may terminate. Algorithm 2 gives the procedure. For a partial graph G, $\mathcal{A}(G)$ contains legal decisions from the current frontier with an available destination instance, and $\mathcal{A}_0(G)$ contains the first-transition decisions. Duplicate graphs remain separate samples, and invalid proposals are resampled up to the configured limit.

For eligible decisions $\mathcal{A}(G)$, first-transition decisions $\mathcal{A}_0(G)$, and sampled indicators $z_a \in \{0, 1\}$, the compiled proposal log probability is

$$\log \widetilde{\pi}_\theta(G) = \sum_{a \in \mathcal{A}(G)} [z_a \log p_\theta(a) + (1 - z_a) \log(1 - p_\theta(a))] - \log \left(1 - \prod_{a \in \mathcal{A}_0(G)} (1 - p_\theta(a)) \right). \tag{13}$$

Terminal, edge-count, and agent-instance checks subsequently reject invalid proposals. If Z_θ denotes the probability of acceptance, the accepted distribution has log probability $\log \pi_\theta(G) = \log \widetilde{\pi}_\theta(G) - \log Z_\theta$. For the leave-one-out advantages used here, $\sum_q a_q = 0$ for each task. Consequently the common $\log Z_\theta$ term cancels from the sampled loss, and gradients can use the compiled proposal probabilities. This cancellation requires independent on-policy draws with repeats retained, advantages detached from the policy parameters, and an acceptance rule evaluated under the current parameterization. The same construction is used for predictor proxy feedback and real feedback. Predictor selection is applied to the deployment pool after sampling.

Algorithm 2 Legal sequential topology sampling

Require: Logits θ, legal edge sets $\{\mathcal{E}_\ell\}_{\ell=0}^{L-1}$, start agent s, instance cap $M = 3$
$G \leftarrow \varnothing$; $F \leftarrow \{s\}$; initialize the first instance of s
for $\ell = 0, \ldots, L-1$ **do**
  $\mathcal{A}_\ell \leftarrow \{(i,j) \in \mathcal{E}_\ell : i \in F,\ j$ has an unused instance$\}$
  if $\ell = 0$ **then**
    Sample Bernoulli decisions on $\mathcal{A}_0$ conditioned on at least one selected edge
  else
    Sample each decision in $\mathcal{A}_\ell$ independently
  end if
  Add selected edges to G and allocate new destination instances in layer order
  $F \leftarrow \{$selected destinations$\}$
  if $F = \varnothing$ **then**
    break
  end if
end for
if G violates reachability, instance, or $1 \le |E(G)| \le 12$ **then**
  Resample up to the configured limit
end if
Retain G, all sampled decisions, and the compiled log probability

A.2 Shared and personalized parameters

The common edge coordinates make the client tensors directly aggregatable. The shared tensor summarizes cross-client updates; the personalized tensor retains each client's local structure through β. During real-feedback refinement, client topology parameters are updated using judged execution rewards, and their aggregation updates the shared tensor. Interpolation preserves client-specific structure when $\beta > 0$. In predictor-enabled settings, the collected execution labels also train the local predictor through supervised outcome prediction, without aggregation.

B Experimental Details

B.1 Client composition and data sources

The three clients share an approximately 8:1:1 dominant-to-minority domain mixture. Their local training sets contain 16 dominant-domain and two tasks from each minority domain; evaluation sets contain 83 and ten, respectively. Table A1 expands these mixtures by dataset source. Source counts can differ while preserving the same domain mixture. For example, HumanEval and MBPP both contribute coding questions. Local training and evaluation questions are disjoint. In compact appendix tables, Code, Math, and Reason abbreviate the three majority-client labels used in the main text.

Table A1: Source-level task counts. HE denotes HumanEval. The Math source is GSM8K for AF and MATH for GD. Every row totals 20 training and 103 evaluation tasks.

		Local training				Evaluation			
Suite	Client	HE	MBPP	Math	MMLU	HE	MBPP	Math	MMLU
AF	Code-majority	4	12	2	2	15	68	10	10
AF	Math-majority	0	2	16	2	1	9	83	10
AF	Reason-majority	0	2	2	16	1	9	10	83
GD	Code-majority	0	16	2	2	7	76	10	10
GD	Math-majority	0	2	16	2	0	10	83	10
GD	Reason-majority	0	2	2	16	1	9	10	83

B.2 Optimization and execution

Table A2 distinguishes proxy training from real-feedback refinement. The feedback budget is the number of judged MAS executions, not the number of unique training questions or LLM calls. One round uses $3 \times 20 \times 4 = 240$ executions, repeatedly visiting the same 60 local training questions under the updated policies. Each MAS execution may invoke several agents. All reported sweeps retain training seed 0.

Table A2: Topology training and feedback-budget settings.

Setting	Value
Topology initialization	Legal logits -3; temperature $\tau = 1$
Predictor encoder	Two graph convolutions; task embedding dimension 384
Predictor hidden representation	Dimension 512; four softly routed experts
Proxy federation	60 rounds; five local steps per round
Proxy batch	16 tasks; 16 topologies per task
Proxy optimizer	Adam, learning rate 0.05
Proxy density weight	$\lambda = 20.0$; $d(G) = \lvert E(G)\rvert / \sum_{\ell=0}^{L-1} \lvert\mathcal{E}_\ell\rvert$
Complete-framework personalization	$\beta = 0.9$
Real-feedback budget	$0, 240, 480, 960$ MAS executions
Real-feedback batch	20 tasks/client; four independent executions/task
Real-feedback topology optimizer	Adam, learning rate 0.1; gradient clipping at 5
Real-feedback topology sharing	Shared aggregation; personalization when enabled
Real-feedback predictor update	Local supervised BCE; no predictor aggregation
Deployment	Predictor ranks $K = 8$ valid graphs; execute top-1
GD budget-study executor	GPT-4o-mini; temperature 0; output cap 1,024 tokens

The GD budget trajectory generates 16 independent valid proposals per training question and executes the first four without ranking or deduplication. Only these four outcomes enter the gradient. Adam moments and step counts are carried between rounds. At evaluation, the GD candidate generator scans up to 1,600 raw proposals and applies the legal-role, reachability, and 1–12-edge constraints to obtain the candidate pool. These constraints are shared by the pretrained and unpretrained arms. All reported native budget evaluations have eight valid deployment candidates per task and zero fallback executions.

The primary predictor is trained with binary cross-entropy with logits using Adam (learning rate 10^{-4}, weight decay 5×10^{-4}), batch size 512, and at most 200 epochs. The predictor data are randomly split into training and validation sets at the workflow level within each task family; validation records are used for model selection, while current evaluation questions and deployment traces remain outside the predictor corpus. Each of the experts outputs a scalar correctness logit. The task-conditioned router computes a weighted sum of these logits, followed by a sigmoid to obtain the predicted probability of correctness, as in Equation 7.

B.3 Judging and statistical units

Coding answers are evaluated with the dataset's unit tests, retaining helper definitions when extracting the submitted program. MMLU answers are evaluated by the final selected option. Mathematical answers use normalized numerical or symbolic equivalence. For domain-of-definition questions, a final bare pair of endpoints is recovered as an interval only if the last displayed interval has exactly the same endpoints; recovery uses the answer text and applies uniformly to all methods. Training outcomes retain the judging rules used when their updates were collected.

Client accuracy is the fraction of correct evaluation answers. Avg gives each client equal weight; Worst is the minimum client accuracy. Paired uncertainty intervals resample whole questions within each client, keeping both policies' outcomes together. Repeated budgets for one question remain in the same resampled cluster. We use 10,000 bootstrap replicates with seed 20260916. These intervals measure evaluation-sample uncertainty for the retained training seed. Provider failures count as incorrect in the primary scores; the sensitivity analysis also considers the common error-free subset.

C Complete Real-Feedback Trajectories

The feedback-budget study uses Shared + Predictor ($\beta = 0$). The pretrained arm receives predictor-based proxy optimization before real feedback, while the unpretrained arm omits this stage. Both arms start from the same predictor initialization and adapt it independently at each client using real execution labels. The local predictors rank $K = 8$ candidates at deployment. The results in Table 3 come from a separate evaluation from Table 2, using the same client splits, suite-native executor, and judging protocol. Figure A1 shows the per-client trajectories, and Table A3 reports every measured budget for both initializations.

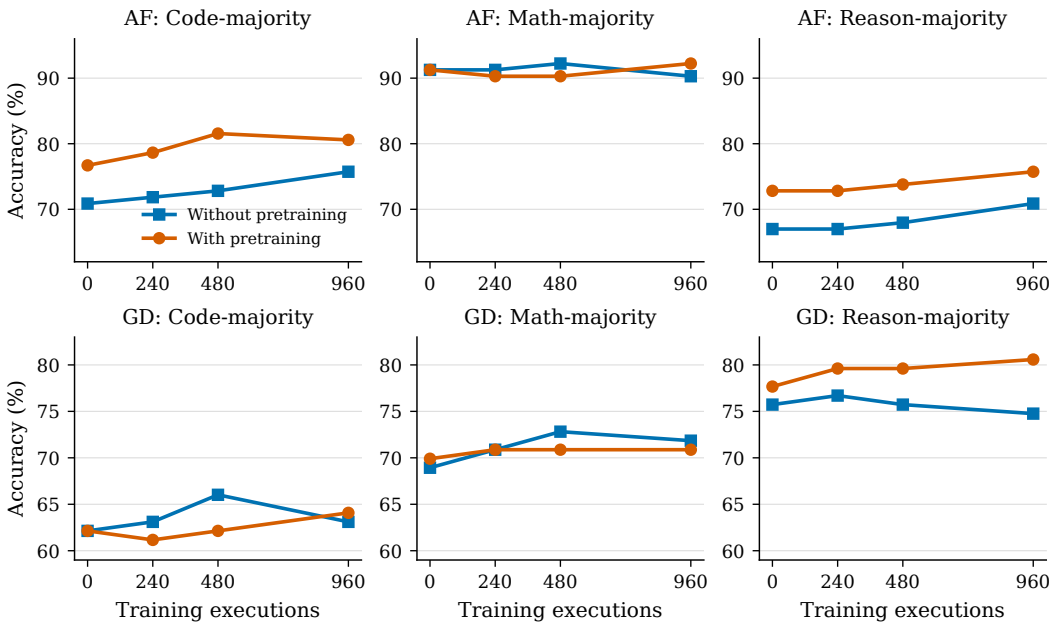


Figure A1: Per-client accuracy along real-feedback trajectories. Every panel uses the same local training questions at each update round and the same evaluation questions at each measured budget.

Table A3: Complete feedback-budget results. Client columns give correct answers out of 103; Avg and Worst are percentages. Pretrain denotes predictor-based probabilistic graph optimization before real feedback.

Suite	Pretrain	Executions	Code	Math	Reason	Avg (%)	Worst (%)
AF	No	0	73	94	69	76.38	66.99
AF	No	240	74	94	69	76.70	66.99
AF	No	480	75	95	70	77.67	67.96
AF	No	960	78	93	73	78.96	70.87
AF	Yes	0	79	94	75	80.26	72.82
AF	Yes	240	81	93	75	80.58	72.82
AF	Yes	480	84	93	76	81.88	73.79
AF	Yes	960	83	95	78	82.85	75.73
GD	No	0	64	71	78	68.93	62.14
GD	No	240	65	73	79	70.23	63.11
GD	No	480	68	75	78	71.52	66.02
GD	No	960	65	74	77	69.90	63.11
GD	Yes	0	64	72	80	69.90	62.14
GD	Yes	240	63	73	82	70.55	61.17
GD	Yes	480	64	73	82	70.87	62.14
GD	Yes	960	66	73	83	71.84	64.08

Within the pretrained trajectory, AF rises from 79/94/75 correct answers at zero executions to 83/95/78 at 960, giving gains of four, one, and three for Code-majority, Math-majority, and Reason-majority. GD rises from 64/72/80 to 66/73/83, giving gains of two, one, and three, respectively. These changes are computed within each initialization across the feedback trajectory; the pretrained and unpretrained rows at the same endpoint are a separate comparison.

C.1 Paired outcomes and uncertainty

Table A4 reports paired answer changes: AF refinement produces 12 gains and four losses; GD produces nine gains and three losses. The GD endpoint feedback interval uses 97.5% confidence for two specified comparisons: feedback improvement and the mean pretraining advantage. Averaged equally over budgets 0, 240, 480, and 960 executions, the GD pretraining advantage is $+0.65$ percentage points (interval $[-1.62, 2.99]$). At 960 executions alone, it is $+1.94$ points (95% interval

$[-0.65, 4.53]$). The GD endpoint feedback interval includes zero, indicating greater uncertainty than the AF trajectory despite the positive point estimate.

Table A4: Paired changes across the three clients. Pretrain compares pretrained and unpretrained policies at the same budget; Feedback compares the pretrained policy after 960 executions with its own initialization. Intervals are paired question bootstrap intervals.

Suite	Comparison	Gains	Losses	Δ (pp)	Bootstrap interval (pp)	Level
AF	Pretrain, 0 executions	23	11	+3.88	[+0.32, +7.44]	95%
AF	Pretrain, 240 executions	22	10	+3.88	[+0.32, +7.44]	95%
AF	Pretrain, 480 executions	24	11	+4.21	[+0.65, +8.09]	95%
AF	Pretrain, 960 executions	26	14	+3.88	[-0.00, +8.09]	95%
AF	Feedback, 960 vs. 0	12	4	+2.59	[+0.32, +5.18]	95%
GD	Pretrain, 0 executions	8	5	+0.97	[-1.29, +3.24]	95%
GD	Pretrain, 240 executions	10	9	+0.32	[-2.59, +3.24]	95%
GD	Pretrain, 480 executions	7	9	-0.65	[-3.24, +1.94]	95%
GD	Pretrain, 960 executions	11	5	+1.94	[-0.65, +4.53]	95%
GD	Feedback, 960 vs. 0	9	3	+1.94	[-0.32, +4.53]	97.5%

Removing provider-error questions from both sides of the GD endpoint comparison gives $+1.62$ points and a 97.5% interval of $[-0.66, 4.22]$. The sign of the mean change is preserved, while uncertainty remains. No endpoint is selected from a test-set peak: both arms complete their four-round trajectories and the table includes every measured budget.

C.2 Availability of Informative Feedback

A question group contributes a nonzero leave-one-out topology advantage only when its sampled executions receive different rewards. Across the 80 question–round groups per GD client, the pretrained run has 15 such groups for Code-majority, 11 for Math-majority, and five for Reason-majority. Seven of the 11 informative math groups include identical graphs with different execution outcomes. This explains why a small number of feedback rounds can have uneven effects: both graph choice and variation in LLM responses contribute to the observed reward. Identical rewards can still train the predictor. Preserving topology optimizer state also matters; a round with zero current advantage can still move parameters through accumulated Adam moments.

D Candidate Budgets and Deployment Cost

The three-client GD sweep in Figure A2 varies the number of valid predictor-ranked candidates over $K \in \{1, 2, 4, 8, 16, 32\}$. The probabilistic graphs and predictor remain fixed, and one selected graph is executed for each task. This comparison varies pretraining of the topology distribution; the main-text Reason-majority experiment instead compares shared and personalized distributions.

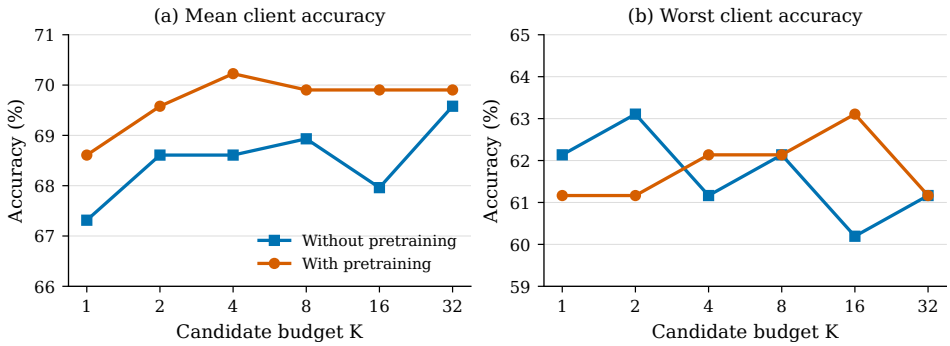


Figure A2: GD deployment sensitivity with fixed model parameters. Panels show the client mean and minimum accuracy.

Pretraining improves the mean accuracy estimate at each measured K in this sweep. Averaging over the six budgets gives $+1.19$ percentage points with a 95% paired interval of $[-0.49, 3.02]$. Increasing K does not produce a monotone accuracy curve: the predictor can rank an additional candidate above a successful smaller-pool choice. The selected topologies also become somewhat

Table A5: Accuracy and execution cost across GD candidate budgets. Calls and tokens cover the evaluation tasks; edges and ranking time are per-task means.

Pretrain	K	Avg (%)	Worst (%)	Edges	LLM calls	Tokens	Rank (ms)
No	1	67.31	62.14	1.53	782	534,635	1.90
No	2	68.61	63.11	1.50	771	493,799	1.94
No	4	68.61	61.17	1.63	808	505,886	2.09
No	8	68.93	62.14	1.83	871	583,901	2.26
No	16	67.96	60.19	2.02	929	649,714	2.54
No	32	69.58	61.17	2.24	997	735,445	3.15
Yes	1	68.61	61.17	1.68	820	553,228	1.86
Yes	2	69.58	61.17	1.75	843	563,119	1.91
Yes	4	70.23	62.14	1.82	869	584,931	2.03
Yes	8	69.90	62.14	1.94	905	596,655	2.22
Yes	16	69.90	63.11	2.14	963	684,788	2.52
Yes	32	69.90	61.17	2.37	1,033	773,205	3.13

larger. For the pretrained policy, mean edge count increases from 1.68 to 2.37 between $K = 1$ and $K = 32$, while predictor ranking takes approximately 1.86–3.13 ms per task. Candidate scoring cost and MAS execution cost therefore measure different parts of deployment.

Cache sensitivity. When repeated historical responses exist for an identical task and graph, the primary sweep uses a fixed cache precedence independent of answer correctness. An alternative common-precedence analysis gives a mean pretraining advantage of $+1.24$ points, with a 95% interval of $[-0.43, 3.02]$; its $K = 32$ pointwise difference is -0.32 points. Both views preserve the small average advantage and show that isolated budget points are sensitive to execution variation. All candidate graphs and predictor scores are held fixed in this analysis.

E Update and Sharing Sensitivity

E.1 Which parameters receive real feedback?

Table A6(a) compares graph-only, predictor-only, and joint adaptation on AF using 240 feedback outcomes. Predictor adaptation uses supervised outcome labels, while probabilistic graph adaptation uses the policy-gradient objective. A predictor learning rate of 10^{-2} reduces mean accuracy from 80.26% to 77.02%; the error is particularly visible on the Reason-majority client. Selecting a smaller rate, 10^{-4}, using only task-grouped cross-validation of the local feedback data gives 80.58% for predictor-only adaptation and 80.26% for joint adaptation. The probabilistic graph-only update gives 80.58% with the original predictor.

Table A6: Sensitivity of feedback updates. (a) AF parameter-update choices with 240 real training outcomes; LR denotes learning rate. (b) GD rewards computed from independently sampled single graphs or predictor-selected candidate pools, at the same real execution budget. All accuracies are percentages.

(a) AF: parameter updates

Update	Graph LR	Predictor LR	Code	Math	Reason	Avg	Worst
Before feedback	–	–	76.70	91.26	72.82	80.26	72.82
Graph	0.1	–	78.64	90.29	72.82	80.58	72.82
Predictor	–	10^{-2}	73.79	90.29	66.99	77.02	66.99
Joint	0.1	10^{-2}	71.84	90.29	68.93	77.02	68.93
Predictor	–	10^{-4}	78.64	91.26	71.84	80.58	71.84
Joint	0.1	10^{-4}	78.64	90.29	71.84	80.26	71.84

(b) GD: reward objective

Objective	Executions	Code	Math	Reason	Avg
Before feedback	0	62.14	69.90	77.67	69.90
Single-graph reward	240	62.14	69.90	78.64	70.23
Candidate-pool reward	240	62.14	70.87	78.64	70.55

Predictor fitting quality and deployment quality respond differently to an update. The AF out-of-fold predictor BCE changes from 0.652 with frozen weights to 0.541 at 10^{-4}, whereas a 10^{-2} step produces BCE 2.090. These values motivate a separate predictor step-size choice for real-feedback adaptation.

E.2 Matching the feedback objective to deployment

The GD comparison in Table A6(b) executes the same number of real training outcomes for two objectives. Single-graph rewards estimate the quality of independent policy samples. Candidate-pool rewards instead evaluate the topology selected by the frozen predictor from each sampled pool. The latter yields 70.55% mean accuracy, versus 70.23% for single-graph feedback and 69.90% before feedback. The one-question difference between the objectives shows the scale of the observed effect under this small feedback budget.

E.3 Local task count and proxy-only sharing

Table A7 studies topology sharing before real-feedback refinement. All three policies use the predictor for training and deployment: Local has no parameter exchange, Shared has one global probabilistic graph, and Personalized combines global and client-specific parameters. The AF sweep uses 5, 10, and 20 local training questions per client. The GD comparison uses 20 questions, legal graph constraints throughout proxy training, 60 rounds, and proxy learning rate 0.005.

Table A7: Proxy-only sharing sensitivity. All rows use a frozen predictor and $K = 8$ deployment; values are accuracies (%). AF and GD use their respective proxy-training settings described above.

	AF: local training tasks			GD: 20 tasks	
Policy	5	10	20	Avg	Worst
Local	79.61	80.58	79.94	67.64	57.28
Shared	79.61	80.91	80.58	66.99	54.37
Personalized	78.96	80.58	80.91	66.34	53.40

With 20 local tasks per client on AF, personalization achieves 80.91%, compared with 80.58% for shared training and 79.94% for local training. This study examines topology sharing before real-feedback refinement; Table 2 reports the results of the complete framework.

F Topology Parameters and Execution Cases

Figure A3 compares shared and client-specific edge probabilities on common routes. The following cases connect sampled graphs to their messages and answers.

F.1 Recoveries after feedback

The coding task in Figure A4(a) asks for elementwise maxima at matching positions in two lists of tuples. Before feedback, the engineering agent interprets matching positions as identical tuples and returns one maximum matching tuple. The finance and programming agents inherit this interpretation. After feedback, the sampled graph begins directly with the finance agent, whose program applies `tuple(max(a,b) for a,b in zip(tuple1,tuple2))` to each pair of tuples. Both worker programs and the final answer pass the saved unit tests. The informative change is the removal of misleading predecessor context.

For the math task in Figure A4(b), the correct count is obtained from $100 \leq 19k + 7 \leq 999$: integer k ranges from 5 through 52, giving 48 values. Before feedback, the math agent returns 47, undercounting the 48 admissible integers. After feedback, the sampled graph connects the input to a math expert and an inspector in parallel. Both produce 48, which the selected terminal node preserves. The trace combines a different sampled graph with new LLM responses, making the answer path visible at each node.

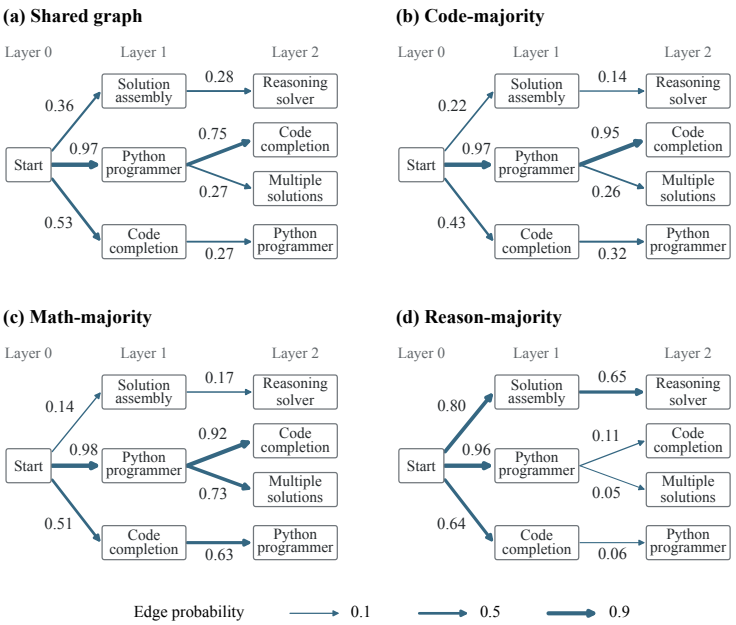


Figure A3: Shared and personalized edge probabilities on common routes. Labels and line widths show learned Bernoulli sampling parameters.

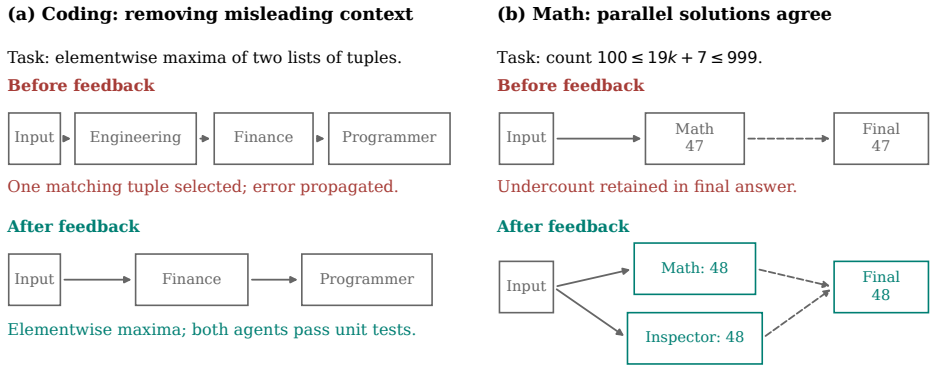


Figure A4: Examples of topology refinement after real feedback. The refined topologies change agent participation and information flow in coding and math tasks.

F.2 Regressions and error propagation

For a MATH series question, the before-feedback graph returns the correct value $1/2$ for $\sum_{n=0}^{\infty} n/(n^4 + n^2 + 1)$. The after-feedback graph adds a mathematical analyst after the math expert. The first agent ends with 0.5 but contains an inconsistent $1/3$ earlier in its derivation; the analyst adopts $1/3$, and the selected terminal node preserves the wrong value. A direct check uses the telescoping identity

$$\frac{n}{n^4 + n^2 + 1} = \frac{1}{2}\left(\frac{1}{n^2 - n + 1} - \frac{1}{n^2 + n + 1}\right), \qquad n \ge 1.$$

This example shows how internally inconsistent reasoning can be amplified by an additional communication edge.

A Reason-majority task asks which event vindicated Stalin's 1931 warning about military weakness. The correct option is the 1941 German invasion. Before feedback, an engineering agent proposes the Space Race, but a critic introduces relevant wartime context and the selected terminal node chooses the correct option. After feedback, a doctor replaces the critic; both agents endorse the Space Race and the selected terminal node returns the incorrect option. The removed critical context explains the failure more precisely than graph size alone.

G TRANSFER TO QWEN3-8B

This experiment evaluates whether the topology selector transfers when the executor changes. The topology policy, task splits, agent pool, judge, and eight-candidate deployment pool are retained. The predictor is initialized from the GPT-4o-mini workflow records used elsewhere in the paper; a single round of 240 target-executor outcomes is then used to adapt its scoring parameters. During adaptation, the graph and task encoders are frozen and only the routing, expert, and output-head parameters are trainable. We use binary cross-entropy with logits and Adam, selecting the learning rate from $\{10^{-5}, 10^{-4}\}$ and the number of update steps from $\{1, 5\}$ by deterministic three-fold question-grouped validation over the client/task groups; the selected setting is then refit on all 240 outcomes. Probabilistic graph-only selection executes the first sampled candidate, whereas the two predictor arms select the highest-scoring candidate from the same fixed pool.

Table A8 reports the complete Qwen3-8B evaluation. The adapted predictor improves over the frozen predictor by 0.32 points on both suites, and improves over direct probabilistic graph sampling by 3.56 and 0.97 points on AF and GD. Its worst-client accuracy is 76.70% on AF and 66.02% on GD.

Table A8: Qwen3-8B transfer results after 240 target-executor feedback outcomes. Client columns denote the Code-majority, Math-majority, and Reason-majority clients; values are accuracies (%), with Avg/Worst the unweighted client mean/minimum.

Suite	Selector	Code-majority	Math-majority	Reason-majority	Avg	Worst
AF	Probabilistic graph only	75.73	87.38	74.76	79.29	74.76
AF	Frozen predictor	79.61	92.23	75.73	82.52	75.73
AF	Adapted predictor	79.61	92.23	76.70	82.85	76.70
GD	Probabilistic graph only	58.25	83.50	74.76	72.17	58.25
GD	Frozen predictor	63.11	82.52	72.82	72.82	63.11
GD	Adapted predictor	66.02	80.58	72.82	73.14	66.02

The AF transfer run used 5,178 API calls and 3,588,799 tokens; the GD run used 2,793 calls and 1,954,541 tokens. These totals include the feedback executions and the held-out evaluations. Raw task-level responses, checkpoints, and request accounting are retained with the experiment artifact.