POET: PRIOR-AWARE PREFERENCE OPTIMIZATION FOR MULTI-AGENT COMMUNICATION TOPOLOGY GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM-based multi-agent systems rely on communication topologies to organize agent participation and information exchange. Successful and failed executions establish graph preferences, yet using these preferences to learn task-adaptive topologies requires interpreting structural differences in the context of task difficulty. We introduce POET (Prior-aware Topology Preference Optimization), a framework for learning task-adaptive communication topologies through prior-guided graph preferences. POET interprets graph complexity relative to task difficulty and uses role diversity to distinguish team compositions beyond size and connectivity. These priors guide the selection of successful and failed graph pairs within each query and their influence in a weighted preference objective, while execution outcomes determine the preference direction. This turns recorded executions into task-aware supervision for the topology generator without additional LLM calls during preference learning. Experiments across six benchmarks show that POET achieves the highest average accuracy among the evaluated methods, while the learned generators transfer to a different LLM backbone without retraining.

1 INTRODUCTION

Large language model (LLM)-based multi-agent systems bring together agents with specialized roles to solve tasks such as mathematical reasoning, question answering, and code generation (Wu et al., 2024; Hong et al., 2024; Du et al., 2024). Agents can decompose a problem, develop candidate solutions, and examine one another's reasoning through communication. How these interactions are organized matters. The communication topology specifies which agents participate, what roles they perform, and whose outputs become available to subsequent agents. Different queries can require different combinations of expertise and information flow, making topology construction a central challenge in automated multi-agent design.

Topology design has progressed from manually specified interaction patterns to learned, query-dependent structures. AgentPrune and AgentDropout reduce redundant communication or participation within predefined agent pools (Zhang et al., 2025b; Wang et al., 2025), while GDesigner learns task-conditioned communication graphs (Zhang et al., 2025c). Generative approaches further allow the team composition and connectivity to be constructed together. ARG-Designer generates agent roles and communication links autoregressively (Li et al., 2026a), and OFA-MAS shares a topology generator across task domains (Li et al., 2026b). These advances expand the range of collaboration structures available to a system. Their effectiveness also depends on the supervision used to learn which structures work for a given task. How execution feedback is used therefore becomes a central part of topology learning.

Successful and failed executions offer complementary supervision for learning task-adaptive topologies. Training only on successful graphs leaves the contrasts with unsuccessful alternatives unused, even though both outcomes are available in recorded executions. A successful and a failed run on the same query establish a graph-level preference. However, outcome labels alone do not determine which structural contrasts should guide learning most strongly. Queries differ in difficulty, and their successful and failed teams can differ in size, connectivity, and role composition. Moreover, queries

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.

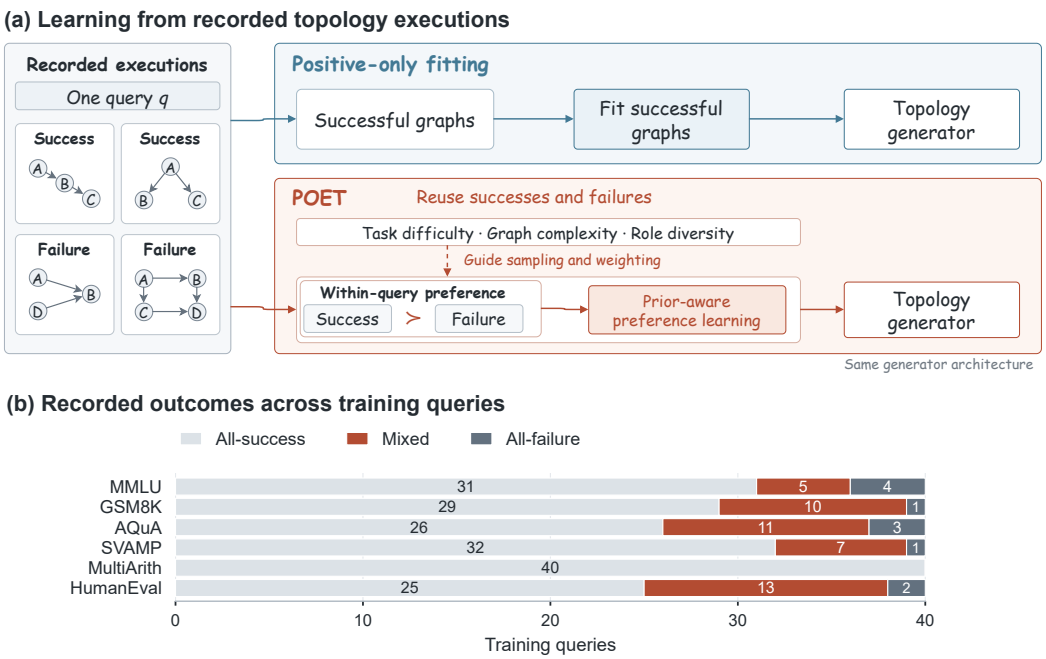


Figure 1: Learning from recorded executions. (a) With the same generator, positive-only fitting uses successful graphs, while POET learns within-query success–failure preferences through prior-guided sampling and weighting. Graphs are schematic. (b) Recorded outcomes for 40 queries per dataset. Mixed queries supply preference pairs. All-success queries still contribute to positive-only pre-training.

with both outcomes are limited and unevenly distributed in our execution pools (Figure 1). The open question is how to *turn heterogeneous execution feedback into topology preference supervision that reflects task demands*.

The value of a structural comparison depends on both the task context and what distinguishes the two teams. A compact team that succeeds on an easy query and a larger team that succeeds on a difficult query provide different examples of effective collaboration. Task difficulty therefore provides the context for interpreting graph complexity. Size and connectivity describe only part of the comparison: teams with similar graph structures can contain roles with different degrees of semantic overlap. Role diversity describes this aspect of team composition alongside graph complexity. These task and structural priors guide which success–failure contrasts enter training through pair construction, and how strongly the selected contrasts influence the generator through loss weighting.

We propose **POET** (**P**rior-aware t**O**pology pr**E**ference op**T**imization), a framework for learning task-adaptive communication topologies through prior-guided graph preferences. POET estimates task difficulty from recorded outcomes and describes teams through graph complexity and role diversity. Complexity is interpreted relative to difficulty when constructing successful and failed graph pairs within each query, while role diversity captures semantic differences between the teams. Difficulty and diversity also determine how strongly the selected comparisons contribute to a reference-relative preference objective (Rafailov et al., 2023). Pair construction and optimization thus use a shared set of priors to learn from the same execution feedback. POET learns from stored executions without additional LLM calls during preference learning. At deployment, the learned generator produces a single graph for the agents to execute.

Experiments across six benchmarks spanning knowledge reasoning, mathematics, and code generation show that POET achieves the highest average accuracy among the evaluated methods. Ablations and analyses of the recorded pools show how prior-guided pair selection and weighting contribute to these gains. The trained generators also transfer to a different execution model without retraining, extending their use beyond the LLM that supplied the training feedback. Our contributions are as follows.

- We cast learning from recorded executions as task-aware graph preference supervision, where within-query success–failure contrasts are interpreted through task difficulty and structural differences.
- We develop POET, which operationalizes this view by coupling prior-guided pair construction with a weighted, reference-relative objective for autoregressive topology generation.
- We evaluate this approach across benchmarks and execution models, and characterize how candidate availability, pair selection, and loss weighting shape the supervision obtained from recorded executions.

2 Related Work

Automated multi-agent topology design. Automated multi-agent design learns agent roles, communication links, or workflows. GPTSwarm (Zhuge et al., 2024) optimizes prompts and connectivity. AFlow (Zhang et al., 2025d) searches code-represented workflows. Automated Design of Agentic Systems (Hu et al., 2025) uses a meta-agent to program and evaluate designs. AgentPrune, AgentDropout, and AGP (Zhang et al., 2025b; Wang et al., 2025; Li et al., 2025) select agents or links from predefined pools, while GDesigner (Zhang et al., 2025c) learns task-conditioned graphs and MaAS (Zhang et al., 2025a) searches an agentic supernet. Following autoregressive graph generation (You et al., 2018; Liao et al., 2019), ARG-Designer (Li et al., 2026a) jointly generates roles and links, and OFA-MAS (Li et al., 2026b) shares a generator across domains. CARD (Wu et al., 2026) conditions topology design on model capabilities and tool availability. These methods define candidate collaborations. POET studies how execution feedback supplies task-aware graph preferences for learning them.

Execution feedback for topology learning. Execution feedback guides topology learning through rewards, message contributions, and structural guidance. Guided Topology Diffusion (Jiang et al., 2026) uses a reward proxy for graph synthesis. RADAR (Zhang et al., 2026) combines redundancy-guided diffusion with task-utility refinement. AgentTailor (Yang et al., 2026) trains an edge predictor from message contributions for virtual refinement. Graph-GRPO (Cang et al., 2026) uses group-relative topology rewards, and AgentConductor (Wang et al., 2026) combines execution feedback with difficulty-dependent density control. POET places task and structural information in the learning signal itself. Difficulty contextualizes complexity contrasts, and role diversity distinguishes team composition. These priors jointly determine which successful and failed graphs from a fixed pool are compared and how strongly they influence training.

Preference optimization and task-aware supervision. DPO (Rafailov et al., 2023) learns preferences through a reference-relative objective. IPO (Azar et al., 2024) derives an alternative from a general pairwise-preference formulation. SimPO (Meng et al., 2024) uses length-normalized sequence rewards without a reference model. In multi-agent systems, Optima (Chen et al., 2025) trains agent language models through iterative generation, ranking, selection, and training, including DPO on communication trajectories. POET instead learns team composition and communication structure while keeping executing LLMs fixed. Curriculum and self-paced learning (Bengio et al., 2009; Kumar et al., 2010) establish difficulty as a basis for selecting training examples. Outside topology design, DART (Hu et al., 2026) uses historical outcomes for task-conditioned surrogate-model selection in adversarial transfer. POET uses execution outcomes to organize graph-level comparisons, coupling their construction and weighting through task and structural priors.

3 Problem Formulation

Query-conditioned collaboration. We represent a multi-agent collaboration as a directed acyclic graph $G = (V, E)$, where each node is an agent with a role from a pool $\mathcal{R}$. An edge (v_i, v_j) passes agent i's output to agent j. Given a query q, agents execute in topological order using their roles, the query, and predecessor messages. A final agent aggregates their outputs into the system's answer. An autoregressive topology generator selects agent roles and communication edges conditioned on q and preceding decisions (Li et al., 2026a).

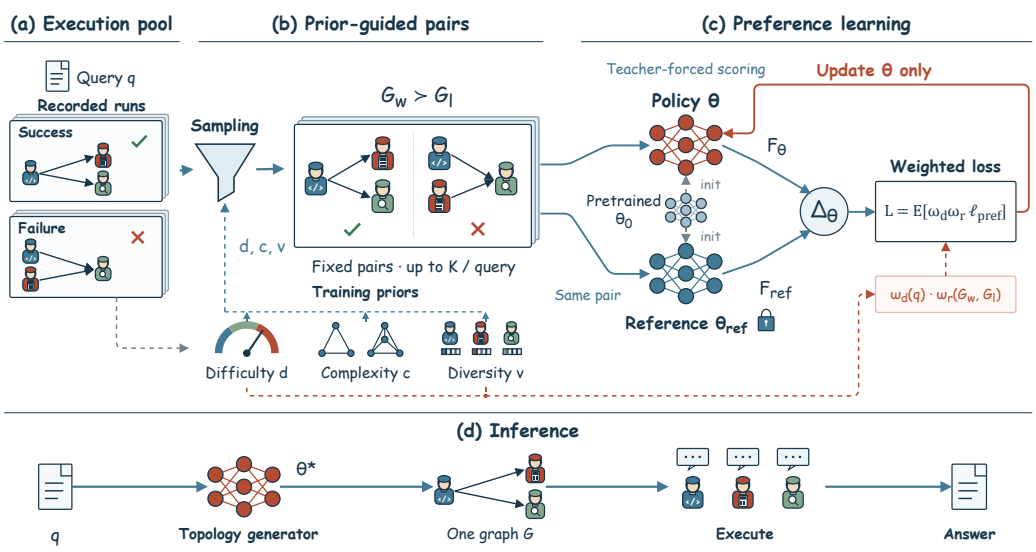


Figure 2: POET training and inference. Task and structural priors guide the selection of up to K success–failure pairs per query. The policy and frozen reference start from θ_0 and score the same pairs. Difficulty and diversity weight the preference loss, and only the policy is updated. Inference generates one graph for execution. Icons denote roles, and graphs are schematic.

Execution records and preferences. For each training query $q \in \mathcal{Q}$, let $\mathcal{C}(q)$ index recorded executions (G_i, z_i), with $z_i = 1$ for success and $z_i = 0$ for failure. The index identifies an execution record, so multiple records may share a graph structure. We partition the records into successful and failed sets.

$$\mathcal{C}_q^+ = \{i \in \mathcal{C}(q) : z_i = 1\}, \qquad \mathcal{C}_q^- = \{i \in \mathcal{C}(q) : z_i = 0\}. \tag{1}$$

A *mixed* query has both sets nonempty and supplies preference pairs (G_w, G_l) with $w \in \mathcal{C}_q^+$ and $l \in \mathcal{C}_q^-$. The execution outcomes establish the preference $G_w \succ G_l$. Pairing within a query keeps the task content fixed when comparing successful and failed executions.

Learning objective. Given this execution pool, our goal is to learn a query-conditioned topology generator that constructs effective collaborations for unseen queries while keeping the executing LLMs fixed. The learning problem is to turn within-query success–failure comparisons into supervision that relates differences in team composition and connectivity to task difficulty, guiding the generator toward collaborations suited to the query.

4 Prior-aware Topology Preference Optimization

POET couples prior-guided pair construction with weighted preference optimization to learn task-adaptive communication topologies from recorded executions (Figure 2). Execution outcomes determine the preference direction. Task difficulty contextualizes graph complexity, while role diversity characterizes team composition.

4.1 Task Context and Topology Contrasts

We characterize comparisons through task difficulty, graph complexity, and role diversity. Team size and connectivity describe structural demands, while role composition distinguishes teams with similar structures.

Empirical task difficulty. The failure frequency in a query's collection pool defines its difficulty and preference weight.

$$d(q) = 1 - \frac{1}{|\mathcal{C}(q)|} \sum_{i \in \mathcal{C}(q)} z_i, \qquad \omega_d(q) = 1 + \alpha d(q), \quad \alpha \geq 0. \tag{2}$$

Difficulty is measured relative to the executing LLM and collected candidates. The weight emphasizes successful collaborations on queries with higher observed failure rates and operates only during training.

Graph complexity. For $n = |V|$, we combine normalized team size with directed edge density.

$$c(G) = 0.6 \min\Big(\frac{n}{6}, 1\Big) + 0.4 \frac{|E|}{\max\{n(n-1), 1\}}. \tag{3}$$

The coefficients emphasize agent count, and the scale of six reflects collection sizes. The density follows the directed-graph convention. Appendix B.1 discusses descriptor properties.

Role diversity. Role descriptions can overlap despite different names. With normalized MiniLM role embeddings $\mathbf{e}_i$ (Reimers & Gurevych, 2019; Wang et al., 2020), their average pairwise cosine dissimilarity is

$$v(G) = \frac{1}{n(n-1)} \sum_{i \neq j} (1 - \mathbf{e}_i^\top \mathbf{e}_j), \qquad v(G) = 0 \text{ if } n \leq 1. \tag{4}$$

Role diversity adds composition information to size and connectivity. Together, these descriptors guide comparison selection and influence.

4.2 Constructing Task-conditioned Comparisons

POET draws a set $\mathcal{P}_q$ of up to K record pairs per mixed query and forms $\mathcal{P} = \bigcup_q \mathcal{P}_q$. Priors bias sampling within each outcome class, while the outcome labels keep the successful record preferred to the failed one.

Difficulty-conditioned complexity. Writing $d = d(q)$, the two complexity weights are

$$u_w(G, q) = (1 - d)(1 - c(G)) + d c(G) + \varepsilon, \tag{5}$$

$$u_l(G, q) = (1 - d) c(G) + d(1 - c(G)) + \varepsilon. \tag{6}$$

For $d < 0.5$, the factors favor simpler successes and more complex failures. For $d > 0.5$, they favor the opposite contrast. Outcome labels fix preference direction, while difficulty determines which structural contrasts receive more sampling probability.

Role-diversity sampling and weighting. We combine the complexity weights with role-diversity factors.

$$p_w(G \mid q) \propto u_w(G, q)\,[\varepsilon + v(G)], \qquad G = G_i,\ i \in \mathcal{C}_q^+, \tag{7}$$

$$p_l(G \mid q) \propto u_l(G, q)\,[\varepsilon + \max\{0, 1 - v(G)\}], \qquad G = G_i,\ i \in \mathcal{C}_q^-. \tag{8}$$

Each distribution is normalized within its outcome pool, with $\varepsilon = 10^{-3}$. Repeated record pairs are rejected, and Appendix A.3 specifies the attempt limit. The pair set is built once and reused across epochs. Diversity also determines the pair's loss weight.

$$\omega_r(G_w, G_l) = \max\{|1 + \lambda[v(G_w) - v(G_l)]|, 10^{-6}\}. \tag{9}$$

The weight changes an included pair's influence without altering its preference logit. Complexity and diversity interact by reweighting the same candidates.

4.3 Learning from Weighted Comparisons

Graph score. We score recorded graphs by teacher-forced role and edge prediction with the ARG-Designer generator. Let ℓ_θ^N and ℓ_θ^E be the summed binary cross-entropy losses on encoded role and edge decisions, including end decisions. Then

$$F_\theta(G, q) = -\big[\rho\, \ell_\theta^N(G, q) + (1 - \rho) \ell_\theta^E(G, q)\big], \qquad \rho = 0.2. \tag{10}$$

Higher scores indicate better fit. The policy and frozen reference use the same score, without normalization by graph size.

Weighted preference objective. Following the reference-relative comparison in DPO (Rafailov et al., 2023), define

$$R_\theta(G, q) = F_\theta(G, q) - F_{\text{ref}}(G, q), \tag{11}$$

$$\Delta_\theta(G_w, G_l, q) = R_\theta(G_w, q) - R_\theta(G_l, q). \tag{12}$$

With logistic sigmoid σ and logit scale $\beta > 0$, POET minimizes over minibatch $\mathcal{B}$

$$\mathcal{L}_{\text{POET}} = \frac{1}{|\mathcal{B}|} \sum_{(q,G_w,G_l)\in\mathcal{B}} \omega_d(q)\, \omega_r(G_w, G_l) \big[-\log \sigma(\beta \Delta_\theta(G_w, G_l, q))\big]. \tag{13}$$

The objective increases reference-relative preference for successful records. Plain DPO uses the same graph score and logit, with uniform pair construction and both weights set to one, isolating POET's priors.

Finite support and learning influence. When $N_q = |\mathcal{C}_q^+||\mathcal{C}_q^-| > K$, construction selects a subset. Otherwise it targets all candidates, subject to finite sampling attempts. Loss weighting remains active even when coverage is complete. Selection therefore controls inclusion, while weighting controls learning influence. For a selected pair with weight $w = \omega_d\omega_r$, the gradient multiplier is $\beta w\, \sigma(-\beta\Delta_\theta)$. Priors set the update scale, while the preference margin determines the remaining fitting pressure. Appendix B.1 gives the gradient analysis, and Section 5.4 examines both effects.

Training selects the checkpoint with the lowest validation-pair loss. Algorithm 1 in Appendix A.4 summarizes the procedure. Appendix A.3 gives optimization details.

Deployment. At inference (Figure 2(d)), the trained generator samples one graph conditioned on the query, and the agents execute it to produce the final answer. Difficulty estimation, pair construction, and preference weighting operate only during training.

5 Experiments

5.1 Experimental Setup

Benchmarks and comparisons. We evaluate GPT-4o-mini agents on MMLU (Hendrycks et al., 2021), GSM8K (Cobbe et al., 2021), AQuA (Ling et al., 2017), SVAMP (Patel et al., 2021), MultiArith (Roy & Roth, 2015), and HumanEval (Chen et al., 2021). These benchmarks cover knowledge reasoning, mathematics, and code generation. After excluding training queries, the subsets contain 153, 1279, 214, 960, 560, and 121 questions. We score one final answer per question, with executable tests for HumanEval, and report the macro-average across datasets. Section 5.6 tests Qwen3-32B executors.

Baseline families. Vanilla, CoT (Wei et al., 2022), and Self-Consistency (SC) (Wang et al., 2023) provide single-agent references. Within predefined agent pools, AGP (Li et al., 2025), AgentDropout (Wang et al., 2025), and AgentPrune (Zhang et al., 2025b) prune graphs or agents. GDesigner (Zhang et al., 2025c) learns task-conditioned graphs, and CARD (Wu et al., 2026) incorporates model and tool capabilities. OFA-MAS (Li et al., 2026b) and ARG-Designer (Li et al., 2026a) autoregressively construct roles and links. Methods share evaluation subsets and answer scoring while retaining their training and topology procedures. Appendix A.6 gives details.

Training and inference. Each pool covers 40 queries, with 16 from cold-start collection and 24 from generator-guided collection. The policy and frozen reference share initialization. We use $K = 8$, $\alpha = \lambda = 1$, $\beta = 0.1$, and Adam with learning rate 10^{-5} and batch size 16 for at most 50 epochs. Approximately 10% of pairs support validation, with patience 15. MiniLM embeds queries and roles. Inference uses one graph, one communication round, and benchmark-specific aggregation (Appendix A).

5.2 Main Results

At 91.22% average accuracy, POET obtains the highest macro-average in Table 1, exceeding ARG-Designer by 2.42 points and AgentPrune, the predefined-pool method with the highest average accuracy, by 0.89 points. It is best or tied for best in every dataset column.

Table 1: Accuracy (%) on the six evaluation subsets with GPT-4o-mini. Avg. is the macro-average across datasets. Bold and underlining mark the best and second-best values among the listed methods. The final row gives changes from ARG-Designer in percentage points. The † symbol marks use of the base generator on MultiArith. Appendix A.6 gives the evaluation settings.

Method	MMLU	GSM8K	AQuA	SVAMP	MultiArith	HumanEval	Avg.
Single-agent							
Vanilla	74.51	79.44	71.50	89.69	96.61	85.95	82.95
CoT	71.24	84.52	70.09	91.15	**98.21**	89.26	84.08
SC	72.55	84.60	74.30	91.46	**98.21**	86.78	84.65
Topology design with a predefined agent pool							
AGP	81.70	93.04	80.84	93.44	97.68	**90.91**	89.60
AgentDropout	81.05	93.51	82.24	<u>93.96</u>	**98.21**	<u>90.08</u>	89.84
GDesigner	<u>83.66</u>	<u>93.59</u>	<u>84.11</u>	<u>93.96</u>	97.86	87.60	90.13
CARD	83.01	93.28	83.64	93.75	97.86	<u>90.08</u>	90.27
AgentPrune	<u>83.66</u>	93.28	<u>84.11</u>	<u>93.96</u>	97.68	89.26	<u>90.33</u>
Topology generation (autoregressive)							
OFA-MAS	82.35	91.16	80.37	93.12	<u>98.04</u>	86.78	88.64
ARG-Designer	78.43	90.93	82.24	**94.58**	**98.21**	88.43	88.80
POET (Ours)	**84.97**	**94.06**	**84.58**	**94.58**	**98.21**†	**90.91**	**91.22**
Gain over ARG-Designer	↑6.54	↑3.13	↑2.34	±0.00	±0.00	↑2.48	↑2.42

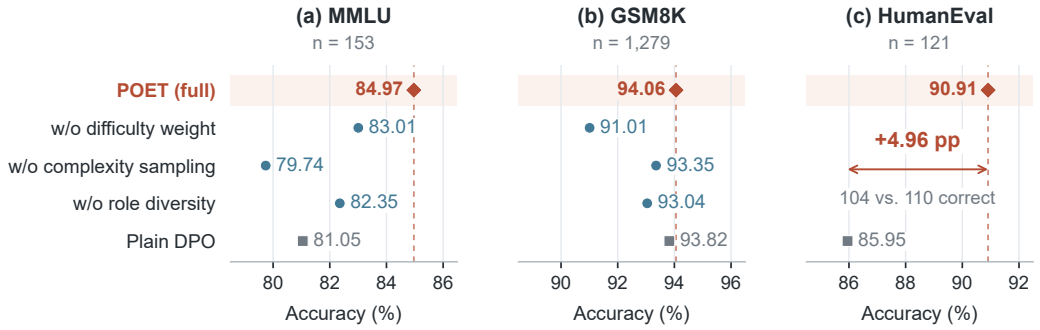


Figure 3: Training-prior ablations with $K = 8$. All axes span eight accuracy points. Dashed lines mark full POET from Table 1. HumanEval includes only the Full and Plain comparison, where POET solves six additional problems. Appendix C.1 gives settings and counts.

Relative to ARG-Designer, POET gains 6.54 points on MMLU, 3.13 on GSM8K, 2.34 on AQuA, and 2.48 on HumanEval. The preference-tuned generator matches ARG-Designer on SVAMP at 94.58%, while MultiArith retains the marked base generator. The closest generator-family comparison thus shows gains across knowledge reasoning, mathematics, and code generation. The following controls match initialization, execution pool, pair cap, and optimization settings to isolate the effect of prior-guided comparisons beyond plain DPO.

5.3 Ablation of Training Priors

Within each dataset, controls share initialization, execution pool, $K = 8$, and optimization settings. Figure 3 presents the three-task comparison with plain DPO and the MMLU/GSM8K component ablations. Settings and exact counts appear in Appendix C.1.

Prior-guided comparisons improve preference learning. Under the same $K = 8$ cap, POET raises plain DPO from 81.05% to 84.97% on MMLU, from 93.82% to 94.06% on GSM8K, and from 85.95% to 90.91% on HumanEval. The 4.96-point HumanEval gain consists of seven POET-only successes and one plain-DPO-only success (Appendix C.2). Task-conditioned comparison design

has its largest effect on knowledge reasoning and code generation. On GSM8K, POET adds three correct answers.

Comparison selection and weighting work jointly. Removing complexity sampling produces the largest MMLU drop, from 84.97% to 79.74%. Removing difficulty weighting produces the largest GSM8K drop, from 94.06% to 91.01%. Removing role diversity gives 82.35% and 93.04% on the two datasets. Full POET leads every removal in both cases. On GSM8K, each removal falls below plain DPO, while full POET improves on it. The components therefore cannot be read as independent bonuses. They work through the same comparisons and update scale.

5.4 Mechanisms of Supervision Allocation

Candidate capacity governs selection and coverage. All five mixed MMLU queries and 11 of 13 mixed HumanEval queries have $N_q > K$, so pair construction selects subsets. Nine of ten mixed GSM8K queries instead target all candidates. Even there, finite attempts produce expected coverage of 43.95 out of 55 pairs under POET versus 55.00 under uniform sampling. Priors therefore affect finite-set inclusion, while loss weights remain active after complete coverage (Appendix B.2).

Structural priors jointly reshape the comparison distribution. Before duplicate rejection, complexity-only sampling shifts the successful minus failed complexity gap from 0.022 to 0.003 for $d \leq 0.5$ and from -0.004 to 0.017 for $d > 0.5$. Adding diversity raises these gaps to 0.080 and 0.050 and increases the diversity gap on every dataset with mixed queries. For example, the MMLU diversity gap rises from 0.066 to 0.234. The factors act jointly on shared candidates, and diversity can outweigh the easy-query complexity bias (Table 6).

From training comparisons to generated teams. Saved topology samples show larger teams after tuning. Mean size rises from 4.07 to 4.99 on MMLU and from 2.74 to 3.98 on GSM8K. Mean role diversity falls from 0.563 to 0.502 and from 0.192 to 0.080, respectively (Appendix B). POET learns preferences over complete teams, rather than maximizing descriptors independently. The next control tests whether these teams depend on the query.

5.5 Query Conditioning, Fixed Teams, and Cost

Dependence on the query. We replace each GSM8K generator embedding with the embedding of another query under a fixed permutation. The checkpoint and execution settings stay fixed. Agents solve the original query, and evaluation uses its answer. The mismatch reduces accuracy from 94.06% to 93.04%, which corresponds to 13 fewer correct answers. Mean graph size remains near four nodes and three edges, while token consumption decreases by only 0.18% (Figure 4). The drop at nearly the same execution scale shows that query information affects the learned role and link decisions.

Offline learning and deployment cost. The Chain-4, Tree-4, Complete-4, and Star-4 controls cover sequential, branching, dense, and hub-based communication. The strongest, Tree-4, reaches 93.98%, solving one fewer question than POET with fewer tokens (Appendix C.4). POET uses 9.894 million tokens versus ARG-Designer's 6.313 million on GSM8K. AQuA totals are 0.960 and 0.834 million. Preference training adds no LLM calls after collection, while deployment cost depends on the generated team and its responses (Appendix D).

5.6 Transfer to a Different Execution Model

We reuse generators trained from GPT-4o-mini feedback with Qwen3-32B executors (Yang et al., 2025), without retraining. Each generator samples one graph per question, and agents and final aggregation use Qwen in non-thinking mode. All methods share task subsets, prompts, scoring, and one communication round (Appendix C.5).

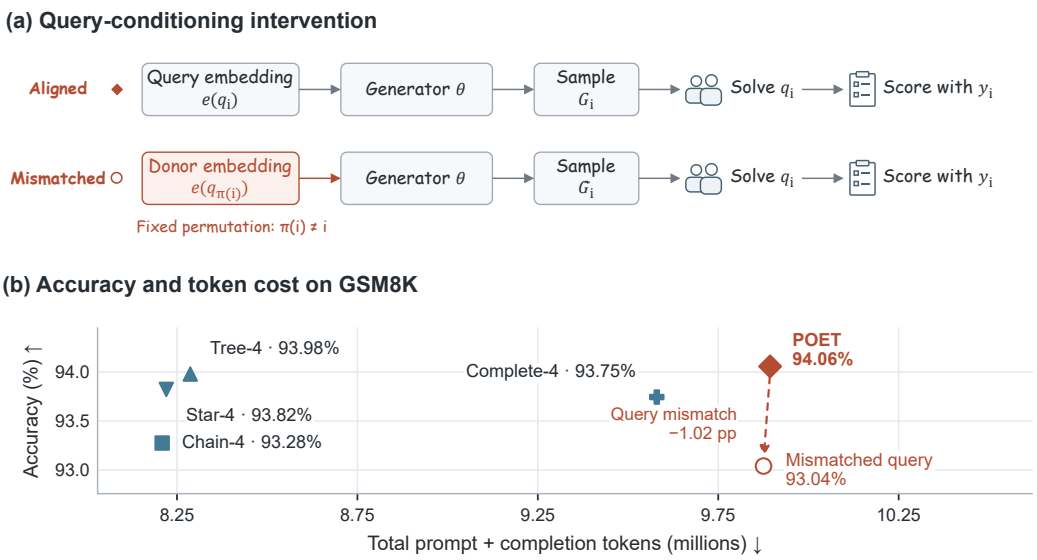


Figure 4: GSM8K query-conditioning and fixed-team controls ($n = 1279$). (a) The generator receives the original or a mismatched query embedding and samples a topology. The donor embedding is taken from another query under a fixed permutation. Both conditions use the same checkpoint and execution protocol. Agents solve the original question q_i, and evaluation uses its answer y_i. (b) The arrow links the POET result in Table 1 with the mismatch control. Accuracy falls by 1.02 points, while token use falls by 0.18%. Tables 10 and 11 give exact results.

Table 2: Accuracy (%) with Qwen3-32B executors and no generator retraining. All methods use the same evaluation subsets and execution settings.

Method	MMLU ($n = 153$)	HumanEval ($n = 121$)
ARG-Designer	80.39	**90.91**
Plain DPO	83.01	90.08
POET	**86.27**	**90.91**

On MMLU, POET exceeds ARG-Designer by 5.88 points and plain DPO by 3.27 points (Table 2). On HumanEval, it matches ARG-Designer and exceeds plain DPO by 0.83 points. The knowledge-reasoning advantage transfers across executors without additional training feedback, and the learned collaboration structures can be used beyond the LLM that supplied their training outcomes.

6 Conclusion

We introduced POET, a preference-learning framework for task-adaptive multi-agent communication topology generation. By interpreting structural contrasts in the context of task difficulty, POET turns successful and failed executions into graph-level supervision. Prior-guided pair construction and weighted preference optimization jointly determine which comparisons guide the generator and how strongly they influence learning, without additional LLM calls during preference training. Experiments across six benchmarks show that POET achieves the highest average accuracy among the evaluated methods, while ablations and mechanism analyses support the coupled design. The learned generators also transfer to a different execution model without retraining. Prior-aware preference learning can produce adaptive collaboration structures from recorded execution feedback without the cost of additional LLM interactions. Because priors enter only through pair sampling and loss weighting, other task-level or structural descriptors can be incorporated without changing the form of the objective. The current pools record offline executions and use correctness as the sole outcome. Collecting executions online and treating execution cost as an additional prior are two concrete extensions of this formulation.

AI USE STATEMENT

In compliance with ICLR's AI-use policy, we state that this research employed LLMs as auxiliary tools for academic writing and implementation support. We emphasize that their role remained auxiliary. The authors led the development of the core ideas and the POET framework, made the final decisions on experimental protocols, analyses, and conclusions, and take full responsibility for the manuscript.

REFERENCES

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. A general theoretical paradigm to understand learning from human preferences. In *International conference on artificial intelligence and statistics*, pp. 4447–4455. PMLR, 2024.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

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.

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

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

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. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

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

Kaidi Hu, Guancheng Wan, Xiao Luo, and Ruigang Yang. DART: Distribution-aware adaptive relational transfer for adversarial attacks against closed-source MLLMs. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=ETPsJre8pn.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *International Conference on Learning Representations*, volume 2025, pp. 21344–21377, 2025.

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.

M Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. *Advances in neural information processing systems*, 23, 2010.

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.

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

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

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

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

Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a reference-free reward. *Advances in Neural Information Processing Systems*, 37:124198–124235, 2024.

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.

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.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019.

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.

Siyu Wang, Ruotian Lu, Zhihao Yang, Yuchao Wang, Yanzhou Zhang, Lei Xu, Qimin Xu, Guojun Yin, Cailian Chen, and Xinping Guan. Agentconductor: Topology evolution for multi-agent competition-level code generation. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=4bLahBrVBs.

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 V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

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, 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, 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*, 2024. 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 *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=JgvJdICc6P.

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.

Peiting Yang, Jiahao Shi, Caiyi Xu, Ming Liu, Yanxia Wu, and Rongsheng Li. Agenttailor: A semantic-aware LLM-based multi-agent system with actor-critic structure. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=ww6mao1xgl.

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

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, LEI BAI, and Xiang Wang. Multi-agent architecture search via agentic supernet. In *Forty-second International Conference on Machine Learning*, 2025a. URL https://openreview.net/forum?id=imcyVlzpXh.

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.

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 *Forty-second International Conference on Machine Learning*, 2025c. URL https://openreview.net/forum?id=LpE54NUnmO.

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

Zhen Zhang, Wanjing Zhou, Juncheng Li, Hao Fei, Jun Wen, and Wei Ji. RADAR: Redundancy-aware diffusion for multi-agent communication structure generation. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=GtpiqFaJtZ.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTSwarm: Language agents as optimizable graphs. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=uTC9AFXIhg.

APPENDIX

A EXPERIMENTAL DETAILS

A.1 EXECUTION DATA AND INITIALIZATION

We collect executions using the same topology configurations as ARG-Designer (Li et al., 2026a), with 40 training queries per dataset. In the cold-start phase, 16 queries are executed with fixed configurations. MMLU and HumanEval use FullConnected-3/4/5/6, Mesh-4/5/6, DirectAnswer-1, and Chain-2. The other four datasets use FullConnected-3/4, Mesh-4, DirectAnswer-1, and Chain-2. Successful cold-start graphs train the base generator. The subsequent pool combines generator-sampled and pruned candidates with simple star, chain, and layered configurations, with sizes varying by dataset. It covers 24 additional queries and can also retain earlier records. The collection configuration uses a pruning ratio of 0.25. Both phases' outcomes are retained for preference construction.

The generator's initial supervised training uses 100 epochs and learning rate 10^{-4}. POET initializes both the trainable policy and frozen reference from the selected base-generator checkpoint, then applies the preference objective to the recorded execution pool.

A.2 EXECUTION-POOL STATISTICS

Table 3 distinguishes the fraction of failed execution records from mean per-query difficulty. The statistics differ because queries contain different numbers of records. The quantity $\bar{d} = |\mathcal{Q}|^{-1} \sum_q d(q)$ gives every query equal weight. The pooled failure rate instead gives every execution record equal weight.

Table 3: Recorded training-pool statistics. Mixed queries have at least one successful and one failed execution. Preference counts refer to the recorded full configurations. Failure rate is pooled over records, and the quantity $\bar{d}$ is averaged over queries.

	MMLU	GSM8K	AQuA	SVAMP	MultiArith	HumanEval
Training queries	40	40	40	40	40	40
All-success queries	31	29	26	32	40	25
Mixed queries	5	10	11	7	0	13
All-failure queries	4	1	3	1	0	2
Execution records	492	266	262	265	267	373
Failed records	77	25	50	29	0	68
Pooled failure rate	0.157	0.094	0.191	0.109	0.000	0.182
Mean difficulty $\bar{d}$	0.163	0.105	0.198	0.109	0.000	0.206
Preference pairs	40	54	70	46	0	99
Validation pairs	4	6	7	5	n/a	10

Mixed queries supply the successful and failed record pairs used in POET. Outcome classes are defined within each query's observed collection pool.

A.3 PAIR CONSTRUCTION AND OPTIMIZATION

For each mixed query, the target number of pairs is $m_q = \min\{K, |\mathcal{C}_q^+||\mathcal{C}_q^-|\}$. Each attempt independently samples one record from each pool using Equation 8. Already-selected record pairs are rejected. Construction stops after m_q distinct pairs or $10m_q$ attempts. The realized number can therefore be smaller than the target. Deduplication is by record pair, not by graph isomorphism.

The constructed list is shuffled before partitioning. The first $\lfloor 0.9|\mathcal{P}| \rfloor$ pairs form the training set, and the remainder form the validation set. The split operates at the pair level, so the same query or graph record can appear in both partitions. Table 3 gives the total and validation pair counts for each full configuration. This partition controls checkpoint selection. Held-out benchmark questions evaluate task performance.

Training uses Adam, gradient clipping at norm 1.0, batch size 16, a learning rate of 10^{-5}, and at most 50 epochs. Incomplete training batches are dropped when the training set contains at least one full batch. Validation uses every pair. The saved checkpoint has the lowest validation loss and early-stopping patience of 15. The weighted node and edge losses in Equation 10 are summed over encoded generator decisions without division by the number of agents. The role-loss coefficient $\rho = 0.2$ comes from the base generator and is distinct from the difficulty-weight parameter α.

All prior weights use the empirical difficulty $d(q)$ computed from recorded execution outcomes.

A.4 Training Algorithm

Algorithm 1 summarizes the complete training procedure. Recorded priors and the constructed pair set remain fixed across epochs, and only the policy parameters are updated.

Algorithm 1 POET Training

Require: generator θ_0, execution pool with at least one mixed query, K, α, λ, β
1: Initialize $\theta \leftarrow \theta_0$ and freeze $\theta_{\text{ref}} \leftarrow \theta_0$
2: Compute $d(q)$, $c(G)$, and $v(G)$ using Eqs. 2, 3, and 4
3: Construct $\mathcal{P}$ with up to K distinct pairs per mixed query using Eq. 8
4: Split $\mathcal{P}$ into training and validation sets
5: **for** each epoch until the epoch budget or early stopping **do**
6: Update θ on training minibatches using Eq. 13
7: Evaluate the same objective on validation pairs
8: **end for**
9: **return** checkpoint with the lowest validation loss

A.5 Execution Interface and Prompt Construction

The main experiments use GPT-4o-mini at temperature 0.2, with the output-token limit left at the API default. Appendix C.5 describes the Qwen transfer setting. Each generated graph specifies agent roles and directed message routes. Agents execute in topological order for one communication round. A final decision module aggregates the original question and agent outputs with a benchmark-specific prompt. The query-mismatch intervention changes only the embedding supplied to the graph generator. Agents and answer evaluation retain the original task.

Role and message inputs. The MMLU role pool contains a knowledgeable expert, critic, mathematician, psychologist, historian, and doctor. Agents receive a role-specific system instruction, the original task, and available predecessor outputs with role identifiers. The GSM8K pool contains Math Solver, Mathematical Analyst, Programming Expert, and Inspector. Prompts combine the task, role-specific examples, and predecessor information. The Math Solver receives numeric hints extracted from predecessor answers, while other roles receive textual analyses or code. The pipeline executes Programming Expert outputs with its Python utility and appends the returned value to the message. All generator variants use the same benchmark-specific aggregation prompt.

Evaluation subsets. We follow ARG-Designer (Li et al., 2026a) in preparing the data and partitioning it into disjoint training and evaluation sets. HumanEval uses the same 161-problem Python data file, partitioned into 40 training and 121 evaluation problems. GSM8K evaluation uses fixed test indices and preserves question identifiers across controls. MMLU loads subject files in sorted order and applies the dataset loader's fixed permutation. All variants use the same benchmark-specific role prompts and answer aggregation. Table 8 reports the evaluation sizes and results of the training-prior controls.

A.6 Evaluation Settings

Baseline families. We organize the comparisons by where the collaboration structure is adapted. Vanilla, CoT (Wei et al., 2022), and Self-Consistency (SC) (Wang et al., 2023) provide single-agent references. The predefined-pool topology methods include AGP (Li et al., 2025), which adaptively

prunes communication graphs, AgentDropout (Wang et al., 2025), which dynamically eliminates agents, AgentPrune (Zhang et al., 2025b), which learns an economical communication pipeline, GDesigner (Zhang et al., 2025c), which learns task-conditioned graphs with graph neural networks, and CARD (Wu et al., 2026), which conditions topology design on model and tool capabilities. The autoregressive generators include OFA-MAS (Li et al., 2026b) and ARG-Designer (Li et al., 2026a), which construct roles and communication links sequentially. This grouping separates topology design within a predefined role pool from generator-level graph construction. All entries use the same evaluation subsets and final-answer scoring protocol. Method-specific training and topology procedures are retained.

Dataset-specific configurations. Table 1 reports preference-tuned generators on MMLU, GSM8K, AQuA, SVAMP, and HumanEval. MultiArith contains no mixed query and uses the base generator, marked with a dagger.

Evaluation protocol. Generators learn from execution pools collected with GPT-4o-mini and are evaluated with GPT-4o-mini and, on MMLU and HumanEval, Qwen3-32B. Main-table comparisons share the execution model and task subsets while retaining each method's architecture and training procedure. The prior ablations additionally match initialization, execution pool, pair cap, and optimization settings (Appendix C.1). Difficulty measures the empirical outcomes in the collection pool. Final answers depend on both topology sampling and LLM execution.

B Mechanism Analysis

B.1 Prior Properties and Learning Influence

Descriptor and sampling properties. The default difficulty-weight parameter $\alpha = 1$ bounds ω_d between one and two. At equal preference margins and role weights, higher empirical difficulty increases the gradient multiplier. The complexity descriptor emphasizes agent count over connectivity, uses six as its collection-based size scale, and follows the directed-graph density convention even for DAGs. Single-agent graphs contribute no edges. In Equation 6, $d = 0.5$ gives $u_w = u_l = 0.5 + \varepsilon$, so complexity has no effect. For any d, $u_w + u_l = 1 + 2\varepsilon$. The two factors favor complementary complexity contrasts within their respective outcome pools. The default diversity-weight parameter is $\lambda = 1$.

Finite support and learning influence. Let $N_q = |\mathcal{C}_q^+||\mathcal{C}_q^-|$ be the number of candidate record pairs. Construction selects a subset when $N_q > K$. When $N_q \leq K$, it targets every candidate, although finite attempts can leave low-probability pairs unseen. Once coverage is complete, set membership is fixed, and loss weighting alone determines the influence of each included comparison. For one selected pair, set $w = \omega_d(q)\omega_r(G_w, G_l)$. Its gradient is

$$\nabla_\theta \ell = -\beta w\, \sigma(-\beta \Delta_\theta) \nabla_\theta \Delta_\theta. \tag{14}$$

The fixed priors set the update scale, while the current preference margin determines the remaining fitting pressure. Sampling controls finite-set membership. Weighting controls the influence of each included comparison. Section 5.4 measures both effects in the recorded pools.

B.2 Pair-distribution Shifts

The analysis uses all 1925 execution records from the six training pools and their role embeddings. The 46 mixed queries provide the within-query success–failure comparisons (Table 3). Graph descriptors follow Equations 3 and 4, and sampling probabilities are normalized within each query and outcome class.

Finite-set capacity and coverage. The candidate capacity $N_q = |\mathcal{C}_q^+||\mathcal{C}_q^-|$ determines whether the cap $K = 8$ selects a subset or targets the complete record-pair set. Table 4 combines capacities computed from the stored records with the realized counts in the saved training configurations. Candidate capacity counts record pairs, consistent with the implementation's duplicate check.

Table 4: Candidate capacity and realized preference-set size. Mixed counts queries with both outcomes. The $N_q \leq K$ column counts queries that target all candidates. Target denotes $\sum_q \min(K, N_q)$. POET and Plain give recorded pair counts before the training and validation split. The notation n/a marks inapplicable quantities or controls not included.

Dataset	Mixed	$N_q \leq K$	N_q range	Target	POET	Plain
MMLU	5	0	14 to 40	40	40	40
GSM8K	10	9	5 to 10	63	54	63
HumanEval	13	2	6 to 21	100	99	100
AQuA	11	7	6 to 12	74	70	n/a
SVAMP	7	4	4 to 10	48	46	n/a
MultiArith	0	n/a	n/a	0	0	n/a

When $N_q \leq K$, stopping after all candidates have appeared or $T_q = 10N_q$ attempts gives the same final set as the union of T_q independent pair draws. For candidate pair (i, j) with draw probability $p_{ij} = p_w(G_i \mid q)p_l(G_j \mid q)$, the inclusion probability and expected set size are

$$\pi_{ij} = 1 - (1 - p_{ij})^{T_q}, \qquad \mathbb{E}[|\mathcal{P}_q|] = \sum_{i,j} \pi_{ij}, \quad N_q \leq K. \tag{15}$$

Under uniform sampling this reduces to $N_q[1 - (1 - 1/N_q)^{10N_q}]$. We compute the expectations exactly and independently check them by propagating probabilities over the subsets of visited pairs. This calculation integrates over construction randomness and requires no model execution.

The nine GSM8K queries with $N_q \leq K$ contain 55 candidate pairs. Expected coverage is 43.95 pairs under POET and 55.00 under uniform sampling. For the two corresponding HumanEval queries, the expected numbers of covered pairs are 11.99 under POET and 12.00 under uniform sampling, out of 12 possible pairs. Concentrated sampling can leave low-probability comparisons outside the finite set even when the cap permits every candidate. Once a complete set is collected, sampling can no longer alter membership, but $\omega_d\omega_r$ still changes the loss. This separates candidate capacity, finite sampling, and loss weighting. The recorded totals in Table 4 describe realized training inputs. Equation 15 instead characterizes inclusion over the construction distribution.

Descriptor contrasts before duplicate rejection. For descriptor $f \in \{c, v\}$ and sampling rule a, define the expected contrast

$$D_f^{(a)}(q) = \sum_w \sum_l p_w^{(a)}(G_w \mid q)\, p_l^{(a)}(G_l \mid q)[f(G_w) - f(G_l)]. \tag{16}$$

The sums range over successful records $w \in \mathcal{C}_q^+$ and failed records $l \in \mathcal{C}_q^-$. Uniform sampling gives every record in an outcome class equal probability. Complexity-only sampling uses u_w, u_l, and full sampling uses Equation 8. We enumerate these expectations before finite-K duplicate rejection and give every mixed query equal weight in the averages. The statistics describe sampling distributions rather than realized training pairs or their validation split. Difficulty and role weights later scale the loss and are excluded from $D_f^{(a)}$.

Table 5: Expected successful minus failed descriptor gaps in the recorded pools. U and P denote uniform and full POET sampling, respectively. Mean difficulty and the range of ω_d are computed over mixed queries only. MultiArith has no applicable pair distribution.

Dataset	Mixed	Mean d	ω_d range	D_c^U	D_c^P	D_v^U	D_v^P
MMLU	5	0.507	1.20 to 1.78	0.083	0.171	0.066	0.234
GSM8K	10	0.320	1.14 to 1.86	0.009	0.089	-0.043	0.032
AQuA	11	0.445	1.14 to 1.86	-0.005	0.031	-0.080	-0.017
SVAMP	7	0.481	1.14 to 1.86	0.021	0.066	0.001	0.046
MultiArith	0	n/a	n/a	n/a	n/a	n/a	n/a
HumanEval	13	0.479	1.10 to 1.90	-0.005	0.045	-0.035	0.040

Table 6: How priors change the pair distribution. Expected successful minus failed gaps in complexity (D_c) and role diversity (D_v) are averaged equally over mixed queries. U, C, and P denote uniform, complexity-only, and full POET sampling, respectively. Expectations precede duplicate rejection and use the calculation above.

Difficulty	Queries	D_c^U	D_c^C	D_c^P	D_v^U	D_v^P
$0 < d \leq 0.5$	28	0.022	0.003	0.080	-0.023	0.048
$0.5 < d < 1$	18	-0.004	0.017	0.050	-0.043	0.044

Full sampling increases the mean diversity gap on every dataset with mixed queries. MMLU has the largest increase, from 0.066 to 0.234. AQuA moves from -0.080 to -0.017 but remains negative. A soft sampling bias cannot guarantee a positive contrast when the available successful and failed pools do not support one. Mixed-query difficulty also differs from the all-query mean in Table 3. MMLU's five mixed queries average 0.507, while all 40 queries average 0.163.

Table 6 separates the intended complexity effect from the combined sampler. Complexity-only sampling reduces D_c on lower-difficulty queries and increases it on higher-difficulty queries. The full sampler raises the gap in both groups, so the complexity prior's direction does not determine the combined distribution by itself. In these pools, role-diversity sampling also changes the selected complexity contrasts. This interaction explains the joint action of the sampling factors, complementing the accuracy-based component ablations.

B.3 Generated Topology Samples

Table 7 characterizes the topology distributions on MMLU and GSM8K using one sampled graph per evaluation query. These samples are analyzed separately from the main-table answers and the executed graphs in Table 10. Standard deviations describe variation across queries.

Table 7: Sampled topology statistics (mean $\pm$ sample standard deviation across queries). Base is the initialization checkpoint. Density uses $|E|/[n(n-1)]$. Complexity and role diversity follow the training descriptors.

Dataset	Generator	Graphs	Nodes	Density	Complexity	Diversity
MMLU	Base	153	4.07±1.23	0.273±0.095	0.516±0.088	0.563±0.158
MMLU	POET	153	4.99±1.09	0.214±0.066	0.585±0.085	0.502±0.152
GSM8K	Base	1279	2.74±0.59	0.383±0.086	0.427±0.026	0.192±0.167
GSM8K	POET	1279	3.98±0.13	0.251±0.011	0.499±0.009	0.080±0.023

Tuning shifts the sampled teams toward larger sizes and lower density. Every sampled graph has $|E| = |V| - 1$. Edge count therefore increases with team size despite the lower density. Role diversity falls on both datasets despite the higher-diversity contrast favored during pair construction. On GSM8K, 98.20% of POET's sampled graphs contain four agents, compared with 8.05% for the base. This concentration makes the choice of roles and edges within a similar team size more relevant. The query-conditioning control in Section 5.5 probes the resulting team as a whole.

C Controlled Evaluations

C.1 Training-prior Ablations

MMLU and GSM8K include all four variants below. HumanEval adds the plain-DPO control. Within each dataset, every variant starts from the same base-generator checkpoint as POET and also uses that checkpoint as its frozen reference. All variants share $K = 8$, learning rate 10^{-5}, at most 50 epochs, $\beta = 0.1$, batch size 16, and patience 15. The candidate pool is fixed. Variants that change sampling can still construct different pairs and pair-level splits.

- **w/o difficulty weight.** Set $\omega_d = 1$ and retain complexity-conditioned sampling, role-diversity sampling, and ω_r.
- **w/o complexity sampling.** Set $u_w = u_l = 1$ and retain role-diversity sampling and both loss weights.
- **w/o role diversity.** Remove the diversity factors from sampling, set $\omega_r = 1$, and retain complexity sampling and difficulty weighting.
- **Plain DPO.** Use uniform sampling from the successful and failed pools and set both loss weights to one. Graph scoring, the reference, and the preference objective remain shared.

Evaluation uses batches of 4 for MMLU, 48 for GSM8K, and 32 for HumanEval, independently of the training batch size. HumanEval combines agent outputs into a final program and evaluates it with executable tests. Every run uses one communication round and one graph per query, with training priors applied only during optimization. Accuracy is computed over all questions, and Table 8 gives both percentages and correct-answer counts.

Table 8: Training-prior ablations with $K = 8$. Entries give accuracy (%) with correct-answer counts in parentheses. W denotes difficulty weighting, C complexity-conditioned sampling, and R role-diversity sampling and weighting. A dash marks an omitted prior, and n/a marks an unperformed evaluation. Full results correspond to Table 1.

Variant	W	C	R	MMLU ($n = 153$)	GSM8K ($n = 1279$)	HumanEval ($n = 121$)
Plain DPO	–	–	–	81.05 (124)	93.82 (1200)	85.95 (104)
w/o difficulty weight	–	✓	✓	83.01 (127)	91.01 (1164)	n/a
w/o complexity sampling	✓	–	✓	79.74 (122)	93.35 (1194)	n/a
w/o role diversity	✓	✓	–	82.35 (126)	93.04 (1190)	n/a
POET (full)	✓	✓	✓	**84.97** (130)	**94.06** (1203)	**90.91** (110)

C.2 HumanEval Case Analysis

The HumanEval comparison yields seven POET-only successes, one plain-DPO-only success, 103 shared successes, and ten shared failures. This gives 110 versus 104 correct answers on 121 questions. Mean team size is 3.60 for POET and 3.85 for plain DPO. Table 9 examines two POET gains involving interacting transformation rules and counting constraints. Both use an equal or smaller team. The outputs are verified by executing the evaluated programs on the corresponding benchmark tests.

Table 9: HumanEval cases illustrating differences in final program behavior. Task identifiers locate the benchmark specifications, and inputs and expected outputs are from their tests. N/E denotes agents/edges in the evaluated team.

Task	Input	Method	N/E	Output	Expected
String encoding (HumanEval/93)	`'TEST'`	Plain DPO	4/3	`'tGst'`	`'tgst'`
		POET	3/2	`'tgst'`	`'tgst'`
Digit-boundary counting (HumanEval/83)	$n = 2$	Plain DPO	4/3	17	18
		POET	4/3	18	18

Composing two encoding rules. The string-encoding task requires shifting each vowel by two alphabet positions and swapping the case of every letter. Plain DPO substitutes the vowels but omits their case conversion, applying that rule only to consonants. POET applies both transformations to vowels and correctly converts consonants. Its program passes with three agents, compared with four for plain DPO.

Counting with the same team size. The second task counts n-digit positive integers whose first or last digit is one. For $n \geq 2$, POET combines 10^{n-1} integers beginning with one and $9 \cdot 10^{n-2}$

ending with one, then subtracts 10^{n-2} counted in both groups. This gives $18 \cdot 10^{n-2}$. Plain DPO subtracts the overlap twice, obtaining $17 \cdot 10^{n-2}$. Both teams contain four agents and three edges, so the observed gain occurs at the same team size.

The case solved only by plain DPO concerns counting integers with a positive signed digit sum (HumanEval/108). For a negative integer, the specification applies the sign only to its leading digit. POET instead negates the full digit sum. On the input $[12, 23, 34, -45, -56, 0]$, it returns three, while plain DPO returns the correct five. Together, these cases connect the aggregate accuracy difference to the programs' handling of interacting rules and boundary conditions.

C.3 Query-conditioning Mismatch

The GSM8K query-control run uses the same full-generator checkpoint, batch size 48, one communication round, and the same final answer aggregation. A fixed permutation with no self-assignment maps each of the 1279 questions to one donor. Every question appears once as a donor. Only the generator's task embedding is replaced. Agents receive the original query, and both answer extraction and evaluation use the original target.

Both conditions evaluate all 1279 questions without recorded execution errors. As Table 10 shows, replacing the query embedding lowers accuracy while leaving mean team size and token consumption nearly unchanged. The structure statistics describe the graphs used in these executions.

Table 10: GSM8K query-conditioning control. Token totals include prompt and completion tokens over all 1279 questions. Graph means come from the actual executions. The aligned row gives the full-model evaluation from Table 1, and the mismatched row gives the additional control.

Generator condition	Correct	Acc. (%)	Nodes	Edges	Tokens (M)
Original query	1203	94.06	3.974	2.974	9.894
Mismatched query	1190	93.04	3.978	2.978	9.876

C.4 Fixed Four-agent Teams

We compare four fixed topology families spanning sequential (Chain-4), branching (Tree-4), dense (Complete-4), and hub-based (Star-4) communication. The suffix denotes the number of agents, and Complete denotes fully connected communication. Four agents match the dominant team size in POET's GSM8K topology samples, where 98.20% contain four agents (Appendix B). These families provide fixed-team evaluation controls. POET itself constructs agent roles and communication edges autoregressively.

The fixed-team controls use the same 1279 GSM8K questions, GPT-4o-mini, four agents, one communication round, and final answer aggregation. Question identifiers and numeric-answer scoring are matched across methods. Team size and topology family are fixed in these controls, while role assignments and realized token budgets remain specific to each method.

Table 11: GSM8K fixed four-agent controls and POET. Tokens include prompt and completion tokens, reported in millions (M).

Method	Correct / total	Accuracy (%)	Tokens (M)
Chain-4	1193/1279	93.28	8.209
Tree-4	1202/1279	93.98	8.287
Complete-4	1199/1279	93.75	9.580
Star-4	1200/1279	93.82	8.220
POET	1203/1279	94.06	9.894

C.5 Transfer to Qwen3-32B

Fixed generators and shared execution settings. ARG-Designer, plain DPO, and POET transfer to Qwen3-32B with their trained generators fixed. The evaluation uses the same 153 MMLU and

Table 12: Qwen3-32B transfer accuracy and token consumption. Counts cover all questions. Input and output tokens include agents and final aggregation and are reported in millions (M).

Dataset	Method	Solved/total	Accuracy (%)	Input (M)	Output (M)
MMLU	ARG-Designer	123/153	80.39	0.667	0.297
	Plain DPO	127/153	83.01	0.849	0.368
	POET	132/153	86.27	0.829	0.364
HumanEval	ARG-Designer	110/121	90.91	0.205	0.037
	Plain DPO	109/121	90.08	0.227	0.042
	POET	110/121	90.91	0.212	0.038

121 HumanEval questions as the corresponding controls. Each generator samples one topology per question, and this topology remains fixed during execution. Agents and the final decision module use Qwen3-32B in non-thinking mode, with temperature 0.2 and at most 8192 output tokens per call. All methods use one communication round, with no generator retraining or additional training-feedback collection.

Prompts and scoring. The three variants retain the original benchmark-specific role prompts, message construction, and internal tools. MMLU aggregates agent responses into a final answer option with the same answer extraction. HumanEval aggregates them into a final program evaluated by executable tests. These shared settings measure transfer of the learned topology policy across execution models.

POET achieves the highest MMLU accuracy in this comparison and uses fewer tokens than plain DPO on both benchmarks. On HumanEval, it matches ARG-Designer's accuracy. ARG-Designer has the lowest token cost on both tasks. The comparison shows transfer of the learned topology policy while distinguishing answer quality from deployment cost.

D Inference Cost

Table 13 compares deployment token consumption on GSM8K and AQuA, where both methods have usage records for the evaluations in Table 1.

Table 13: Deployment tokens in millions (M) on GSM8K and AQuA. Base refers to ARG-Designer. Totals are computed before rounding.

Dataset	Model	Prompt (M)	Completion (M)	Total (M)
GSM8K	Base	5.020	1.293	6.313
GSM8K	POET	7.954	1.939	9.894
AQuA	Base	0.528	0.306	0.834
AQuA	POET	0.607	0.353	0.960

Deployment consumption follows the generated team and its messages. Preference optimization uses the collected executions and cached embeddings and requires no further LLM calls after collection.