PRISM: PRIVILEGED SIBLING-GUIDED MULTI-AGENT TOPOLOGY DESIGN

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-agent systems (MAS) based on large language models (LLMs) have become a powerful way to address complex problems, and their effectiveness depends critically on the collaboration topology that decides which agents participate and how they communicate. Recent work learns to generate this topology from the task query, fitting the generator to corpora of execution-verified graphs. The post-training method for such generators remains a limitation: a corpus lists the graphs that solved a query but never ranks them, leaving continued supervised fine-tuning (SFT) with no per-query preference to exploit. Without such a preference, the generator drifts toward the topologies the corpus contains most often rather than those that suit each query, so a gain on one task can come at a cost on another. We propose PRISM, which recovers per-query signal from the same corpus by learning with privileged information: a teacher reads, through a zero-initialized low-rank branch, a sibling graph that also solved the query and is unavailable at deployment, and the student matches its per-decision targets on trajectories the student samples itself. Across six diverse benchmarks PRISM improves over every topology-design baseline evaluated, and matched-teacher controls support the contribution of privileged query information.

1 INTRODUCTION

Multi-agent Systems based on Large Language Models (LLMs) address complex tasks by coordinating agents with complementary roles. Their effectiveness depends on the collaboration topology: the graph specifying which agents participate and how information flows between them (Zhuge et al., 2024; Shen et al., 2025). Consequently, improving MAS requires learning how to organize collaboration alongside improving the capabilities of individual agents. Research on automated topology design has expanded from predefined workflows to learned communication structures and, more recently, generators that construct a collaboration graph for each query (Li et al., 2026b; Zhang et al., 2025b). These generators learn which agents to include and how to connect them for a given query.

Recent work has made query-conditioned topology generation feasible by deriving supervision from multi-agent execution outcomes (Li et al., 2026b;c). Candidate collaboration graphs are instantiated and executed on training queries, and empirically successful query–graph pairs are retained as training examples. Using these verified pairs, an autoregressive topology generator learns through teacher forcing to predict agent roles and communication links conditioned on the query and the partially constructed graph. This approach turns previous collaboration outcomes into reusable supervision and provides a straightforward post-training strategy: continuing supervised fine-tuning (SFT) on verified graphs. However, closer agreement with the recorded graphs does not necessarily translate into better downstream task performance.

In our experiments, continued SFT concentrates the generated topology distribution, while its effects on downstream performance vary across benchmarks: some improve, whereas others exhibit limited gains or regressions. This observation is consistent with prior findings that likelihood-based SFT can reduce output diversity (Li et al., 2025), although concentration alone does not establish whether the resulting topologies are more effective. Teacher forcing also introduces a mismatch between the partial graphs encountered during training and inference. Training supervises decisions along recorded construction trajectories, whereas deployment requires the generator to act on partial graphs produced by its own preceding decisions. Deviations from recorded trajectories can therefore

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.

lead to partial graphs for which the verified corpus provides no direct supervision. This mismatch parallels exposure bias in autoregressive imitation learning and motivates on-policy distillation (Lin et al., 2020; Agarwal et al., 2024): using a teacher to guide construction decisions at partial graphs visited by the student.

In this work, we propose PRISM, which addresses this training-inference mismatch by distilling verified-graph information at the partial graphs actually visited by the student. For each training query, the student first generates its own partial graph trajectory. A privileged teacher then receives the query together with another verified graph for the same query and provides a soft distribution over the next topology decisions at each student-generated partial graph. Thus, supervision remains available even after the student deviates from the recorded verified trajectory, directly addressing the off-trajectory states left unsupervised by teacher-forced SFT. The query-matched verified graph also provides information about a successful alternative construction, while supervised anchoring preserves the verified graph trajectories and discourages the student from drifting away from valid topology patterns. Since the verified graphs are collected before post-training, PRISM requires no additional MAS executions, and the privileged input is removed at inference time without changing the deployed topology generator.

In summary, this work makes the following contributions:

- We document increased output concentration under continued supervised fine-tuning of an autoregressive topology designer and relate it to accuracy across benchmarks.
- We propose PRISM, which uses successful sibling graphs as training-only privileged inputs for on-policy self-distillation, without additional post-training LLM executions or changes to the deployed designer architecture.
- We demonstrate its effect across six benchmarks spanning mathematical reasoning, general reasoning and code generation - GSM8K, MultiArith, SVAMP, AQuA, MMLU and HumanEval - outperforming the strongest topology-generation baseline on each. Matched teacher controls support an additional benefit from query-matched graph information beyond constant-input on-policy distillation.

2 Related Work

MAS communication topology design. Early approaches organize reasoning and collaboration through chains (Wei et al., 2022; Hong et al., 2024), tree-structured search (Yao et al., 2023), and debate protocols (Du et al., 2024). In MAS, performance depends on the communication structure (Qian et al., 2025; Shen et al., 2025), while the computational cost of collaboration has motivated research on efficiency (Li et al., 2026a). Automated topology design methods optimize agent participation and communication: AgentPrune (Zhang et al., 2025a) prunes communication edges, AgentDropout (Wang et al., 2025) removes agents and edges across communication rounds, and G-Designer (Zhang et al., 2025b) generates task-conditioned communication graphs using a variational graph autoencoder. These methods operate on a predefined set of candidate agents. ARG-Designer (Li et al., 2026b) extends the design space by autoregressively generating agent roles and communication edges, allowing the number of agents and their role composition to vary with the query. Complementary research investigates on-policy refinement using failure trajectories for agent safety alignment (Yin et al., 2026) and adversarial manipulation that increases the computational cost of tool-augmented reasoning (Li & Wang, 2026). Our work focuses on post-training an autoregressive topology generator initialized through supervised learning on execution-verified graphs. We retain ARG-Designer's deployed architecture and introduce a privileged teacher to provide additional supervision along student-generated graph trajectories.

On-policy distillation. DAgger (Ross et al., 2011) learns from states the policy itself visits rather than states a demonstrator recorded, and GKD (Agarwal et al., 2024) and MiniLLM (Gu et al., 2024) bring that idea into LLM distillation, sampling from the student and scoring with a teacher; the latter also motivates mode-seeking reverse KL. A further line makes teacher and student one model rather than a large-small pair: Zhao et al. (2026) distil a reasoner from its own on-policy samples, and Dat et al. (2026) carry the scheme to diffusion language models. PRISM uses on-policy self-distillation in this sense, separating teacher from student by a privileged input rather than by scale or by an

earlier checkpoint. Its decision spaces are also small categorical decision spaces (roles + END; edge/no-edge), so the divergence is exact rather than estimated.

3 PRELIMINARIES

MAS as a collaboration graph. Following Li et al. (2026b), a MAS is a directed acyclic graph $G = (\mathcal{V}, \mathcal{E})$ whose nodes are LLM instances carrying roles from an extensible pool $\mathcal{R}$, and where an edge $e_{ji} = (v_j, v_i)$ means v_i receives v_j's output in its prompt. Agents execute in topological order for K rounds and a summariser aggregates the final responses. Each agent is queried once per round and the summariser once per question, so executing G costs $K|\mathcal{V}| + 1$ LLM calls.

Conditional autoregressive generation. The designer factorises the conditional distribution over graphs as

$$P_\theta(G \mid Q, \mathcal{R}) \;=\; \prod_{i=1}^{|\mathcal{V}|} \underbrace{P_\theta(v_i \mid G_{<i}, Q, \mathcal{R})}_{\text{node generation}} \cdot \prod_{j=1}^{i-1} \underbrace{P_\theta(e_{ji} \mid v_i, G_{<i}, Q)}_{\text{edge generation}}, \tag{1}$$

where $G_{<i}$ is the subgraph over the first $i-1$ nodes. A sentence encoder maps Q to $f_Q \in \mathbb{R}^d$ and a task processor produces the conditioning vector $c = \mathrm{TaskProc}(f_Q)$, gated with a GRU summary of the previously selected roles (Cho et al., 2014). A node GRU consumes that context and a metric-learning head scores roles by inner product against role embeddings extended with a learnable END token - the mechanism that keeps the role pool extensible - after which an edge GRU emits the $i-1$ incoming-edge decisions for v_i. Training minimises the negative log-likelihood, decomposed as

$$\mathcal{L}_{\text{total}} \;=\; \alpha\, \mathcal{L}_{\text{node}} + (1-\alpha)\, \mathcal{L}_{\text{edge}} \tag{2}$$

under teacher forcing.

The verified corpus. The supervision set is constructed without additional human annotation: for each training query Q, blueprint configurations $c \in \mathcal{C}$ (chain, star, layered, fully-connected, mesh, with random roles and agent counts) are instantiated as graphs $G(c)$, executed, and retained only if they solved the query,

$$\mathcal{D} \;=\; \big\{\, (G(c), Q) \;\big|\; S\big(Q, G(c)\big) = 1 \,\big\}, \qquad S(\cdot) \in \{0, 1\}. \tag{3}$$

One such corpus is built for each evaluated domain; their sizes, blueprint composition and query-level splits are reported in Appendices C and F. What matters here is the shape of $\mathcal{D}$ rather than its scale: a query is typically solved by several distinct topologies, so the corpus attaches to each query a set of graphs that worked and never marks one of them as preferable.

4 PRISM

PRISM provides an alternative supervision pathway for exploiting the query–graph associations already present in the verified corpus. As shown in Figure 1, Stage 1 constructs a teacher conditioned on a successful sibling graph, and Stage 2 distils its predictions on trajectories sampled from the student's current policy.

4.1 STAGE 1: CONSTRUCT THE PRIVILEGED TEACHER

Let a be privileged information for query Q: information that exists at training time and is structurally unavailable at deployment. We clone the student and give the clone one extra input branch; writing f_a for the encoding of a, the teacher's conditioning vector is a FiLM modulation (Perez et al., 2018) of the student's:

$$c_{\text{teacher}} \;=\; \mathrm{TaskProc}(f_Q) \odot \big(1 + \gamma(f_a)\big) \;+\; \beta(f_a), \qquad \gamma, \beta : \mathbb{R}^r \to \mathbb{R}^d, \tag{4}$$

where f_a passes through a small encoder - layer normalisation, dropout, a linear map to rank $r = 16$ and a $\tanh$ activation - before γ and β are computed. Starting from pretrained ARG-Designer checkpoint θ_0, the teacher minimizes the weighted supervised objective in Eq. 2, updating all teacher parameters. The student itself receives no updates and remains at θ_0 throughout this stage.

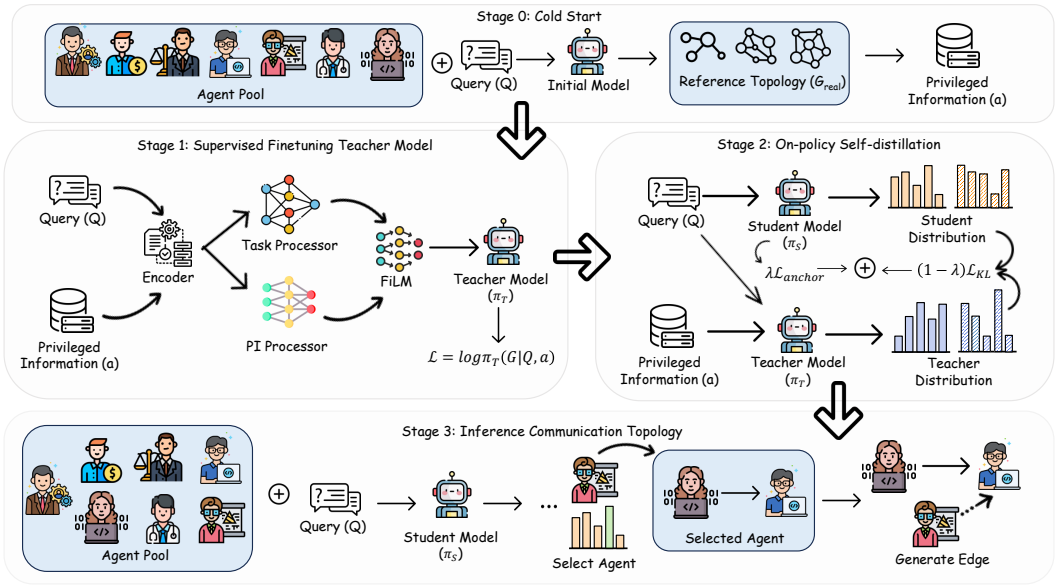


Figure 1: Overview of PRISM. **Stage 0** fits the designer on execution-verified topologies, which also supply the privileged input a. **Stage 1** trains a separate teacher whose FiLM branch reads a. **Stage 2** has the teacher rescore the student's own trajectories, the student minimising Eq. 6. **Stage 3** deploys the student alone; the teacher is discarded.

Sibling sampling and encoding. Each successful graph is converted into a deterministic text representation of its roles and edges, which is embedded using the same frozen MiniLM encoder (Wang et al., 2020) used for the query. Duplicate graph representations are merged before sibling sampling. For a verified target graph G, the sibling is drawn uniformly from the other graphs that solved the same query, skipping any whose text representation is identical to G's, which keeps the target itself out of the privileged channel. The sibling is resampled every epoch, allowing the pairing to change when multiple eligible candidates exist.

If a query has no distinct successful sibling, its privileged input is a zero vector; no graph from another query is substituted. This supplies no sibling-specific graph information, although a zero input does not necessarily produce an identity FiLM modulation after training. The implementation counts queries without siblings and stops training if no query has an eligible sibling.

The teacher is a separate model initialized from a copy of the student weights θ_0 and augmented with a privileged encoder and FiLM branch; no parameters are shared between the models. Under identical deterministic evaluation conditions, zero initialization of the FiLM output maps makes the teacher's predictions identical to the student's at teacher creation. This removes an initial functional mismatch, but does not imply equal model capacity or zero divergence after teacher training. During Stage 2 the teacher is frozen and every student parameter is trainable. The channel is kept narrow because the cold start yields only $O(10^3)$ graphs: at rank 16 it adds ≈20k parameters against ≈592k for a full-width channel at $d = 384$, an approximately 30-fold reduction in channel parameters while retaining competitive accuracy in the evaluated settings (§7).

4.2 Stage 2: On-policy Self-distillation

At the start of Stage 2, we sample one verified reference graph uniformly from the deduplicated set of successful topologies for each training query. This reference is fixed throughout Stage 2 and shared across all M rollouts for that query. Unlike the target-exclusion rule used in Stage 1, Stage 2 does not exclude references that match the supervised anchor or a sampled rollout. Queries with a single verified topology use that topology as their reference. Student trajectories are resampled at every optimization step.

For each query we sample M complete topologies from the current student policy at temperature τ, so that the corrected states are always the ones this policy visits, unlike off-policy pseudo-labelling, where graphs are generated once and reused as the student drifts away from them. Each trajectory

Table 1: Pass@1/accuracy (%) on six benchmarks with Qwen2.5-7B-Instruct. Best results are bold. On AQuA, SFT trails ARG-DESIGNER by 12.60%, while PRISM improves over it.

Method	GSM8K	MultiArith	SVAMP	AQuA	HumanEval	MMLU
Vanilla (single agent)	70.28	88.33	82.40	61.81	69.51	62.70
AgentPrune	76.72	94.67	91.90	68.90	71.95	67.93
AgentDropout	77.86	95.33	92.20	70.47	77.44	68.32
G-Designer	78.01	97.33	91.80	74.41	78.05	71.52
ARG-DESIGNER (baseline)	79.98	98.83	92.50	75.20	79.27	71.98
SFT (control)	86.13	98.83	92.40	62.60	80.49	73.22
PRISM	**88.02**	**99.17**	**93.30**	**76.38**	**82.32**	**73.94**

is then re-scored by both models under teacher forcing on that trajectory: the student without a, the teacher with a. Although their parameters are separate, the teacher and student predict over the same node and edge symbol sets, so every per-decision categorical divergence between them is computed exactly rather than estimated:

$$\mathcal{L}_{\mathrm{KL}} \;=\; \alpha \cdot \mathbb{E}_{\text{node steps}}\big[\mathrm{KL}\big(P_\theta \,\|\, P_{\text{teacher}}\big)\big] \;+\; (1-\alpha) \cdot \mathbb{E}_{\text{edge slots}}\big[\mathrm{KL}\big(P_\theta \,\|\, P_{\text{teacher}}\big)\big]\,, \tag{5}$$

with $\alpha = 0.2$ inherited from Eq. 2. We use reverse KL to encourage concentration on decisions favored by the teacher and evaluate this choice against alternative divergences (Gu et al., 2024). To retain direct supervision from verified query–graph pairs, we combine the distillation objective with a supervised anchoring loss:

$$\mathcal{L}_{\text{anchor}} \;=\; -\log P_\theta(G_{\text{real}} \mid Q), \qquad \mathcal{L}_{\text{PRISM}} \;=\; (1-\lambda)\,\mathcal{L}_{\mathrm{KL}} \;+\; \lambda\,\mathcal{L}_{\text{anchor}} \tag{6}$$

where G_{real} is a verified graph for Q drawn from the corpus of Eq. 3. The two terms are normalised differently and we keep the convention explicit: $\mathcal{L}_{\text{anchor}}$ sums the log-probabilities of all node and edge decisions of G_{real}, unweighted by α, and averages over the batch, whereas each divergence in Eq. 5 is a mean over decision points of its own kind. Setting $\lambda = 1$ leaves the anchor alone, which is the objective of our SFT control, making it the far endpoint of this one-parameter family rather than a different method. Rollouts are scored as sampled, never repaired first: a repaired graph is not what the policy emitted, and its gradient would not match the state it came from. Gradients flow only through the re-scoring of those trajectories; sampling itself is not differentiated and the teacher is evaluated without gradients, so we optimise the divergence at the visited states rather than an exact gradient of the expectation over π_θ, as is standard in on-policy distillation (Agarwal et al., 2024). The teacher is frozen throughout Stage 2 and discarded before evaluation, so the deployed model is architecturally identical to ARG-DESIGNER and the privileged channel exists only during training. The full loop is Algorithm 1 in Appendix B.

5 EXPERIMENTS

5.1 SETUP

Benchmarks and backbone. Following Li et al. (2026b) we evaluate on MMLU (Hendrycks et al., 2021), GSM8K (Cobbe et al., 2021), MultiArith (Roy & Roth, 2016), SVAMP (Patel et al., 2021), AQuA (Ling et al., 2017), and HumanEval (Chen et al., 2021), reporting Pass@1 / accuracy. All agents run on Qwen2.5-7B-Instruct (Qwen Team, 2025) served locally, so every arm in every table is a controlled re-run under one execution stack rather than a number copied from a paper. The role encoder is `all-MiniLM-L6-v2` (Wang et al., 2020) with $d = 384$, and $K = 1$ round with a summariser. Stage 1 uses 40 epochs at lr 10^{-4}; Stage 2 uses 30 epochs at 10^{-4}, 16 queries per step, $M = 4$ rollouts per query, $\tau = 1.0$, $\alpha = 0.2$, $\lambda = 0.3$ and reverse KL (Appendix A), shared across benchmarks with no per-benchmark tuning.

Data source and partitioning. Our verified graph corpus uses questions from MIXCHAIN, MBPP and MMLU. The training and validation data are partitioned by query rather than by graph. Graphs are grouped by their associated query, the query groups are shuffled with a fixed seed, and 20% are assigned to validation. While MIXCHAIN supplies the corpus for the four mathematical benchmarks, general reasoning and code generation are covered separately: Stage 0 builds a verified corpus for MMLU from its own questions and, for code generation, from MBPP (Austin et al., 2021),

Table 3: Frequency and conditional accuracy of SFT's modal role assignment, `[Math Solver, Programming Expert, Programming Expert]` on every benchmark. Share is measured in the deployed single-sample decode, so it is not comparable with the resampled 0.346 of §5.3.

Benchmark	share emitted	acc. @ default	acc. @ other	difference
GSM8K	0.349	87.4	85.4	+2.0%
MultiArith	0.323	99.0	98.8	+0.2%
SVAMP	0.371	91.6	92.8	−1.2%
AQuA	0.358	60.4	63.8	**−3.4%**

under the same procedure (Eq. 3); those graphs serve as the training targets and privileged siblings for the corresponding benchmark. HumanEval is therefore used for evaluation only, so no question it scores has contributed a training graph to any arm.

Baselines and controls. We compare against ARG-DESIGNER (Li et al., 2026b), G-Designer (Zhang et al., 2025b), AgentPrune (Zhang et al., 2025a) and AgentDropout (Wang et al., 2025), and a single-agent vanilla run. ARG-DESIGNER is reproduced in full: the cold start of Eq. 3 followed by the published efficiency fine-tuning stage, so the row labelled ARG-DESIGNER is the complete training procedure of (Li et al., 2026b) rather than an initialization-only variant. The critical control is **SFT**: continued supervised maximum likelihood on the verified corpus from that same deployed checkpoint θ_0, matched to Stage 2 in epochs and learning rate. Both SFT and PRISM are initialized from the deployed ARG-Designer checkpoint. Their student-training stages use 30 epochs and a learning rate of 10^{-4}. PRISM additionally trains a teacher for 40 epochs and performs trajectory sampling and teacher rescoring during distillation. Both methods reuse the existing verified corpus without additional multi-agent executions during post-training. These settings match the student-training schedule, but not total training computation. Table 1 scores the model each procedure finally deploys. Beyond these arm-level controls, Appendix H executes chosen crews directly on a fixed question set, holding wiring, rounds, decision method and backbone fixed so that only the role composition varies.

5.2 MAIN RESULTS

PRISM is the best method on every benchmark, but two features of Table 1 matter more than the headline. The first is the AQuA column: SFT tracks or beats the baseline on five of the six benchmarks, by at most 6.15% and never falling more than 0.10% below, then drops 12.60% on AQuA. The effect of continued SFT varies substantially across benchmarks. The second is the size of PRISM's margin over SFT, which varies by an order of magnitude across those same benchmarks: 0.72% on MMLU and 1.89% on GSM8K against 13.78% on AQuA.

Cost. Table 2 reports the deployment cost of each designer. PRISM averages 3.04 agents per query against 3.00 for ARG-DESIGNER and SFT alike. At $K = 1$ that is 4.04 LLM calls against 4.00, buying 2.56% and 3.24% more accuracy for 1.0% more calls. Post-training adds no executions either, since its rollouts sample graph trajectories rather than running any agent.

Table 2: Mean per-query cost over the evaluation sets, at $K = 1$.

Designer	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	calls
ARG-DESIGNER	3.00	2.30	4.00
SFT	3.00	2.21	4.00
PRISM	3.04	2.27	4.04

5.3 OUTPUT CONCENTRATION

We write *crew* for the multiset of roles in a topology, ignoring wiring. We measure concentration as the top-1 topology share, the fraction of questions on which a designer emits its single most frequent shape. Relative to the deployed starting checkpoint, SFT increases the top-1 share from 0.171 to 0.346, whereas PRISM reduces it to 0.157. Resampling each query eight times and comparing each arm against a permutation null built from its own samples, the query explains 1.1% of the cold start's output variety, 2.0% of SFT's and 5.8% of PRISM's (Appendix G); all three exceed their nulls, so SFT remains query-conditional, and the difference is one of degree. Nor is concentration a loss of vocabulary: the number of distinct shapes is 22 at the cold start, 40 for SFT and 34 for PRISM.

Table 4: Cross-arm control. Questions are partitioned by whether SFT emitted its default role assignment, and both arms are scored on exactly those questions, so difficulty is fixed by construction.

Benchmark	Group	n	SFT	PRISM	Δ	McNemar p
GSM8K	SFT emitted default	460	87.4	88.5	$+1.1\%$	0.68
	SFT emitted other	859	85.4	87.8	$+2.4\%$	0.15
SVAMP	SFT emitted default	371	91.6	94.6	$+3.0\%$	0.16
	SFT emitted other	629	92.8	92.5	-0.3%	0.92
AQuA	SFT emitted default	91	60.4	79.1	$\mathbf{+18.7\%}$	$\mathbf{0.0001}$
	SFT emitted other	163	63.8	74.8	$+11.0\%$	0.003

We read these diagnostics as ruling out a degenerate explanation of PRISM's gain, that it wins by emitting a better constant, rather than as evidence of superiority, which §5.4 and §6.2 carry.

Table 3 reports the frequency of the modal role assignment and the accuracy associated with its use. The shape is identical on all four benchmarks measured - three agents, two edges, two of the corpus's four roles absent - and is emitted on roughly a third of questions; its value is not, at $+2.0\%$ on GSM8K and -3.4% on AQuA. Both differences are small beside the 12.60% SFT concedes to the baseline on AQuA, so the modal crew accounts for part of that shortfall rather than all of it.

5.4 Difficulty-matched control

On AQuA, SFT has lower accuracy on questions for which it emits its modal role assignment. We therefore partition each benchmark by whether SFT emitted its default crew and score both arms on exactly those questions, so the two face an identical set and difficulty cannot differ between them. Table 4 shows an advantage in both groups, largest on the questions where SFT emitted its modal role composition. It does not establish that the crew caused the errors, since crew composition, wiring, graph size and the stochastic execution of the agent runtime all vary together across the two designers, so what the partition isolates is the designer and not the crew.

On AQuA, PRISM improves in both groups: $+18.7\%$ where SFT emitted its modal crew (79.1% against 60.4%, exact McNemar $p = 0.0001$) and $+11.0\%$ elsewhere ($p = 0.003$). The margin is largest on the defaulted questions, but it is not confined to them, so the fixed crew accounts for part of the AQuA gap rather than all of it. The other benchmarks separate the groups far less: GSM8K gains about a point in each ($+1.1\%$ against $+2.4\%$, neither significant at the 5% level) and SVAMP gains 3.0% where SFT defaults against -0.3% elsewhere, neither significant. On those 91 AQuA questions PRISM emits 18 distinct topologies and repeats SFT's default crew on only one question in eight, so the crews that replace it are themselves varied.

6 Discussion

Two objections stand between §5.4 and our account of it: that the channel works through added parameters and training steps rather than through what it carries, and that the advantage is owed to on-policy correction any teacher would supply. We answer the first with measurements independent of downstream accuracy, the second with a matched pair of Stage-2 runs.

6.1 Channel information content

Because the FiLM modulation of Eq. 4 is zero-initialised, whether the teacher uses a at all is directly measurable. On held-out graphs we compute the paired privilege gain

$$\Delta \;=\; \log P_{\text{teacher}}(G \mid Q, a) \;-\; \log P_{\text{teacher}}(G \mid Q), \tag{7}$$

scoring the same teacher on the same graph with the channel on and off, which cancels graph-difficulty variance; a Δ within 2σ of zero is consistent with the teacher ignoring the channel. A positive Δ indicates that the privileged input improves the teacher's likelihood on held-out verified graphs; this alone does not establish a downstream benefit after distillation, so we pair it with a shuffle control: a teacher identical in FiLM channel, parameter count, optimiser and budget, except

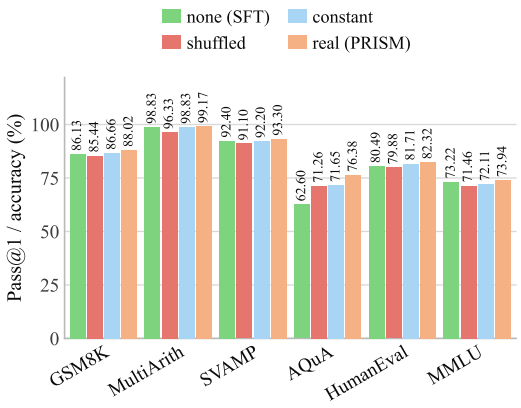


Figure 2: PRISM's advantage by Stage-2 teacher: none (SFT), shuffled (another query's graph), constant (privileged input held fixed) or real. All four arms post-train θ_0 under matched budgets, so only the teacher differs.

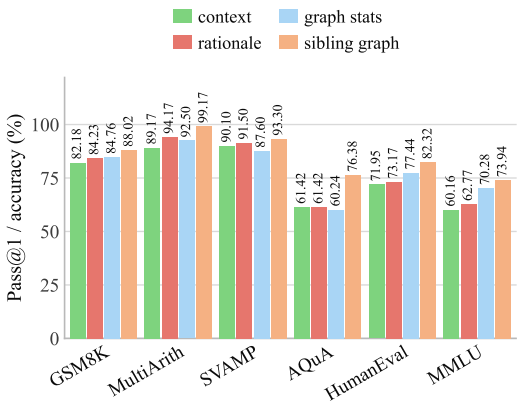


Figure 3: Privileged channels, with every other Stage-2 setting held fixed. sibling graph, the deployed channel, is best on all six and widest on AQuA; context is weakest on four. The others vary by benchmark.

that each query is shown another query's graph. If the gain came from capacity or optimisation, the control would show it too.

Figure 4 reports both runs. The shuffled channel's FiLM norm increases from zero during training - the optimiser genuinely trains it - and its gain never leaves zero, so the real channel's $+2.45$ nats is not an artefact of an untrained control. Panel (B) of Figure 4 carries a second, independent signal: the reported FiLM norm reaches 3.10 with sibling inputs and 1.15 with shuffled inputs when it contains something predictive, and does so from the first epochs, before panel (A) has separated.

6.2 Content versus procedure

The above establishes that the teacher reads the channel, not that what it reads survives distillation into a student that never sees it. We separate content from procedure with a matched Stage 2: two students from the same initialisation, identical in rollouts, α, λ, KL direction, epochs and learning rate, differing only in which teacher supplies the target. The shuffled teacher is fully trained and simply carries no information about this query, so its student still receives on-policy correction on its own visited states - it is PRISM with the privilege removed and nothing else changed.

The constant teacher separates the two effects that the PRISM–SFT gap confounds, and shows that the shuffled teacher is not a clean null: the constant arm beats it on all six benchmarks (83.86 against 82.58 on average), so conditioning on another query's graph misleads the student rather than merely withholding information. PRISM still exceeds the constant teacher on all six ($+1.66\%$ on average, sign test $p = 0.031$) at matched teacher epochs, sampling, supervision, loss mixture and computation, which leaves privileged content as the remaining difference. AQuA carries both contributions, from SFT at 62.60 to the constant teacher at 71.65 ($+9.05\%$ from on-policy distillation) and PRISM at 76.38 ($+4.73\%$ from content), and distillation alone is not sufficient: the constant teacher improves on SFT on three benchmarks only, losing 1.11% on MMLU.

7 Ablations

7.1 Privileged channel

Figure 3 compares the channels with Stage 2 otherwise unchanged. Beyond the query, the teacher sees the question and solution as one string (context), the worked solution, test suite or correct option (rationale), summary statistics of Q's solved set (graph stats), or another graph that also solved Q (sibling graph, our default training-time channel). sibling graph is best on all six, but its margin over the best competitor varies: 3.26% on GSM8K against 14.96% on AQuA.

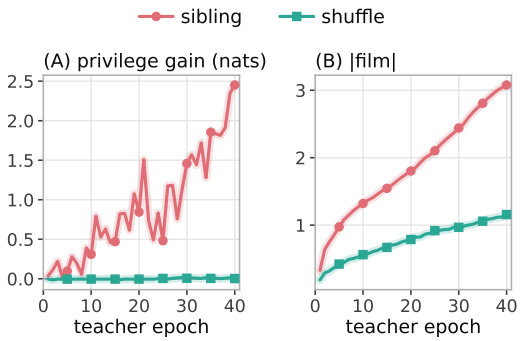


Figure 4: The shuffle control: same channel and budget, different content, the shuffled teacher reading another query's sibling graph. **(A)** Privilege gain (Eq. 7), on held-out graphs, climbs to $+2.45$ nats for the real channel and stays at zero when shuffled. **(B)** $\|\text{film}\|$, zero at initialisation, rises for both, to 3.10 and 1.15, so (A) is a null, not a failure to train.

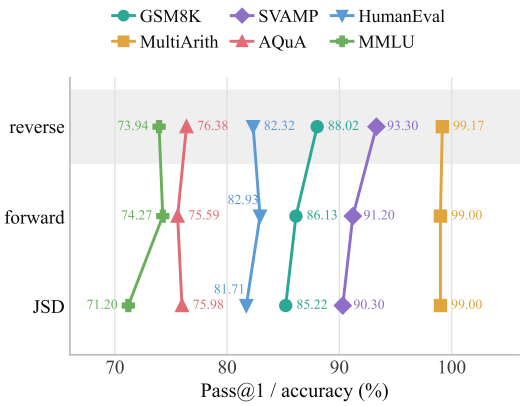


Figure 5: Divergence objectives, with the privileged channel fixed to sibling graph. Reverse KL (our default, shaded) is best on average and on every mathematical benchmark; forward KL and JSD ($\beta = 0.5$) trail it, edging ahead only on HumanEval and MMLU.

A measured null. context yields $\approx$0 nats, failing the gate of Eq. 7: a correct solution is on-topic by construction, so appending it barely moves the mean-pooled embedding, $5.8\times$ less than shuffled context does (node-level KL 0.0101 vs. 0.0584). Distilling from it costs 5.84% on GSM8K and 14.96% on AQuA.

Node and edge distillation diagnostics. Every text-based channel drives the edge divergence below 10^{-4} with edge-argmax agreement 1.000, so the $(1-\alpha)$ share of Eq. 5 carries almost no signal. The channel enters at the conditioning vector read by the node GRU (Eq. 4) and reaches the edge head only through the node state. graph stats alone drives edge KL from 0.52 to 0.067, telling the teacher about wiring rather than roles.

7.2 KL DIRECTION

Figure 5 varies the divergence with the channel fixed to sibling graph: reverse is $\mathrm{KL}(P_\theta \| P_{\text{teacher}})$, forward KL reverses the arguments, and JSD their mixture at $\beta = 0.5$. Reverse KL achieves the highest average score among the evaluated divergence objectives (85.52 against 84.85 and 83.90) and on every mathematical benchmark; forward KL achieves slightly higher accuracy on HumanEval and MMLU by 0.61% and 0.33%, gaps small enough that we do not infer a mechanistic explanation from these small differences. The margin is widest on GSM8K and SVAMP, where reverse KL leads forward KL by 1.89% and 2.10% and JSD by 2.80% and 3.00%. This is the pattern the mode-seeking argument predicts: with several verified graphs per query, a mass-covering objective spreads the student across the teacher's modes, reproducing the averaging that limits SFT, whereas reverse KL commits it to one.

8 CONCLUSION

We introduced PRISM, a method for post-training autoregressive multi-agent topology generators using successful sibling graphs as privileged inputs. The teacher provides decision-level targets on student-generated trajectories, and the deployed student uses only the query. Under the evaluated Qwen2.5-7B-Instruct configuration, PRISM improves mean accuracy over ARG-Designer across six benchmarks with a similar average number of agent calls. A constant-input teacher control indicates that sibling information contributes beyond the matched distillation procedure. The results do not establish that PRISM recovers a ranking among successful graphs or that query-specific crew assignment explains the gains. Future evaluation should test that assignment effect on representative held-out questions, compare post-training methods at matched compute, and examine transfer across different backbones.

AI USE STATEMENT

In this work, we used generative AI tools to edit a research paper to improve readability. We have not used generative AI tools to generate synthetic data sets or formulate mathematical claims. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies the training of a small graph-generation model that decides how many LLM agents to instantiate and how to connect them. It does not involve human subjects, personal data, or the release of new datasets of human origin. All benchmarks used (MMLU, GSM8K, MultiArith, SVAMP, AQuA, HumanEval) are public and used under their stated licenses.

REPRODUCIBILITY STATEMENT

The method is specified completely in §4: Eq. 4 gives the privileged channel, Eq. 5–6 the objective, and Algorithm 1 the training loop. All hyperparameters used for every reported arm are listed in §5 and repeated in Appendix A. The corpus construction procedure is Eq. 3 with the blueprint families named in §3. Diagnostics are reproducible from the described probes: the paired privilege gain of Eq. 7 and the shuffle control of §6.1. Anonymised source code and the exact command lines for every table will be released to support reproduction of the reported experiments.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, 2024.

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.

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.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

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.

Phuong Tuan Dat, Qi Li, and Xinchao Wang. dopsd: On-policy self-distillation for diffusion language models. *arXiv preprint arXiv:2607.04428*, 2026.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *International Conference on Machine Learning*, 2024.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *International Conference on Learning Representations*, 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.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, 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. *arXiv preprint arXiv:2308.00352*, 2024.

Qi Li and Xinchao Wang. Sponge tool attack: Stealthy denial-of-efficiency against tool-augmented agentic reasoning. *arXiv preprint arXiv:2601.17566*, 2026.

Qi Li, Lexiang Xiong, Haiquan Lu, Wenjie Qu, Xingyi Yang, Jiaheng Zhang, and Xinchao Wang. Efficiency in llm multi-agent systems: A survey. *Preprints*, September 2026a. doi: 10.20944/preprints202609.1639.v1. URL https://doi.org/10.20944/preprints202609.1639.v1.

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*, 2026b.

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*, WWW '26, pp. 1333–1344, New York, NY, USA, 2026c. Association for Computing Machinery. ISBN 9798400723070. doi: 10.1145/3774904.3792537. URL https://doi.org/10.1145/3774904.3792537.

Ziniu Li, Congliang Chen, Tian Xu, Zeyu Qin, Jiancong Xiao, Zhi-Quan Luo, and Ruoyu Sun. Preserving diversity in supervised fine-tuning of large language models. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 66127–66154, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/a548ef984f30bca3abdc09f43743827f-Paper-Conference.pdf.

Alexander Lin, Jeremy Wohlwend, Howard Chen, and Tao Lei. Autoregressive knowledge distillation through imitation learning. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6121–6133, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.494. URL https://aclanthology.org/2020.emnlp-main.494/.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*, 2017.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*, 2021.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

Chen Qian, Zihao Xie, Yifei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Scaling large language model-based multi-agent collaboration. *arXiv preprint arXiv:2406.07155*, 2025.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025.

Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2011.

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

Xu Shen, Yixin Liu, Yiwei Dai, Yue 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 Conference on Empirical Methods in Natural Language Processing*, 2025.

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. In *Advances in Neural Information Processing Systems*, 2020.

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. *arXiv preprint arXiv:2503.18891*, 2025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, 2023.

Bo Yin, Qi Li, and Xinchao Wang. On-policy self-evolution via failure trajectories for agentic safety alignment. *arXiv preprint arXiv:2605.11882*, 2026.

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*, 2025a.

Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. G-designer: Architecting multi-agent communication topologies via graph neural networks. In *International Conference on Machine Learning*, 2025b.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

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

A HYPERPARAMETERS

Table 5 lists every hyperparameter used for every arm reported in the main text. Values are shared across benchmarks; no per-benchmark tuning was performed.

Table 5: Full hyperparameter listing.

Stage	Parameter	Value
Shared	sentence encoder	`all-MiniLM-L6-v2`, $d = 384$
	node/edge loss weight α	0.2
	gradient clipping	1.0
Stage 1 (teacher)	epochs	40
	learning rate	1×10^{-4}
	privileged rank r	16
	privileged dropout p	0.1
Stage 2 (PRISM)	epochs	30
	learning rate	1×10^{-4}
	queries per step	16
	rollouts per query M	4
	rollout temperature τ	1.0
	anchor weight λ	0.3
	KL direction	reverse
	teacher temperature	1.0
Evaluation	backbone	Qwen2.5-7B-Instruct
	communication rounds K	1
	validation fraction	0.2

B PRISM TRAINING LOOP

Algorithm 1 PRISM Stage 2

Require: student θ, frozen privileged teacher ϕ, query pool $\mathcal{Q}$, anchor loader $\mathcal{A}$

for all $Q \in \mathcal{Q}$ **do**
  $a_Q \sim \mathrm{Uniform}(\text{distinct verified topologies of } Q)$ {kept fixed}
end for
for each optimisation step, with minibatch $B \subset \mathcal{Q}$ **do**
  for all $Q \in B$ **do**
    $G^{(1)}, \ldots, G^{(M)} \sim P_\theta(\cdot \mid Q)$ {on-policy, no repairs}
    for $m = 1, \ldots, M$ **do**
      score $G^{(m)}$ with teacher $\phi(\cdot \mid Q, a_Q)$ {same a_Q for all M rollouts}
      score $G^{(m)}$ with student $\theta(\cdot \mid Q)$
    end for
  end for
  $G_{\text{real}} \leftarrow$ next batch from $\mathcal{A}$
  $\mathcal{L} \leftarrow (1-\lambda)\big[\alpha\,\mathrm{KL}_{\text{node}} + (1-\alpha)\,\mathrm{KL}_{\text{edge}}\big] + \lambda\big(-\log P_\theta(G_{\text{real}} \mid Q)\big)$
  update θ by Adam on $\nabla_\theta \mathcal{L}$ (gradient clipping at 1.0)
end for

Trajectories are re-sampled at every optimisation step, so the state distribution the teacher corrects tracks the student as it changes. Repairs are applied at evaluation only, never to a sampled graph before scoring. Four edits make a sampled graph executable. First, during evaluation-time generation, any node after the first whose edge head emitted no incoming edge receives one at a uniformly random admissible position, which guarantees $|\mathcal{E}| \geq |\mathcal{V}| - 1$. The generated graph is then made acyclic by deleting one edge from each directed cycle, any disconnected component is joined to the largest by a single edge oriented from the lower to the higher node index, and any node still isolated is linked to a uniformly random other node.

C CORPUS STATISTICS

MIXCHAIN contains 2,947 execution-verified graphs over 1,014 distinct queries and 25 distinct topologies, drawn from five blueprint families (chain, star, layered, fully-connected, mesh) with random role assignments. Mean pairwise Jaccard overlap of the families' solved-query sets is 0.300; sparse families overlap one another at 0.61–0.66, dense families form a second cluster, and the two clusters are largely disjoint (§3).

Table 6 breaks the corpus down by blueprint family, and two features of it carry claims made in §5.3. Across the four families whose attempts were recorded, the solve rates span 58.5–59.5%, a spread of 1% across families that differ in edge density by a factor of two, so no family is preferred in the aggregate. The corpus shares are not flat, however: they track how often each family was sampled rather than how often it succeeded. That is the combination which leaves the marginal uninformative while the conditional is not.

Table 6: MIXCHAIN by blueprint family. cfgs is the number of distinct agent-count configurations instantiated for the family; attempts and solved count executions and successes, and rate is their ratio. corpus is the number of verified graphs the family contributes and share its fraction of the corpus; $|\mathcal{V}|$, $|\mathcal{E}|$ and $|\mathcal{E}|/|\mathcal{V}|$ are means over those graphs. Cold start retains one graph per (query, topology) pair, so the corpus is smaller than the number of successful executions: the 5,836 successes reduce to 2,947 distinct pairs. †Not available for the layered family, so the Total row's attempts, solved and rate cover the other four families.

Family	cfgs	attempts	solved	rate	corpus	share	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{E}\rvert/\lvert\mathcal{V}\rvert$
Chain	2	3,465	2,045	59.0%	1,034	35.1%	3.03	2.03	0.67
Star	2	2,872	1,681	58.5%	867	29.4%	3.63	2.63	0.72
Layered	1	—†	—†	—†	541	18.4%	4.00	4.00	1.00
Fully-connected	2	2,071	1,233	59.5%	286	9.7%	3.00	3.00	1.00
Mesh	1	1,494	877	58.7%	219	7.4%	4.00	6.00	1.50
Total	8	9,902	5,836	58.9%	2,947	100.0%	—	—	—

Table 7 gives the same breakdown for the code-generation corpus. The MBPP cold start used the default configurations of Li et al. (2026b) for code generation, which are dense-only, so there is no chain, star or layered family to report: the corpus is built from fully-connected and mesh blueprints alone. The pattern of Table 6 survives the change of domain. The two families solve at 63.6% and 62.4%, a spread of 1.2% despite differing in edge density, while their corpus shares, 65.5% against 34.5%, again follow how often each was instantiated. Cold start's one-graph-per-(query, topology) rule reduces 1,259 successes to 981 graphs.

Table 7: MBPP by blueprint family, with columns as in Table 6. Only dense families appear because the code-generation cold start follows the default configuration of Li et al. (2026b). The 1,259 successful executions reduce to 981 distinct (query, topology) pairs.

Family	cfgs	attempts	solved	rate	corpus	share	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{E}\rvert/\lvert\mathcal{V}\rvert$
Fully-connected	3	1,312	835	63.6%	643	65.5%	4.04	6.46	1.60
Mesh	2	679	424	62.4%	338	34.5%	4.37	7.49	1.71
Total	5	1,991	1,259	63.2%	981	100.0%	—	—	—

D SEED VARIABILITY

Every arm is post-trained three times from the same deployed checkpoint θ_0 under seeds 42, 43 and 44, the runs differing in nothing else, and each run is evaluated on the full benchmark suite. Table 8 reports the mean and sample standard deviation over the three runs; seed 42 is the run reported in Table 1 and everywhere else in the paper.

Table 8: Accuracy over three post-training runs per arm, under seeds 42, 43 and 44. Each cell is the mean, with the sample standard deviation over the three runs as a subscript.

Arm	GSM8K	MultiArith	SVAMP	AQuA	HumanEval	MMLU
ARG-DESIGNER	$80.21_{\pm 0.75}$	$98.33_{\pm 0.50}$	$92.93_{\pm 0.51}$	$75.33_{\pm 1.38}$	$79.07_{\pm 0.35}$	$72.28_{\pm 0.64}$
SFT	$86.18_{\pm 0.09}$	$98.83_{\pm 0.00}$	$92.73_{\pm 0.42}$	$62.73_{\pm 0.61}$	$80.29_{\pm 0.35}$	$73.09_{\pm 0.54}$
constant teacher	$86.71_{\pm 0.04}$	$98.66_{\pm 0.29}$	$92.37_{\pm 0.29}$	$69.42_{\pm 1.94}$	$81.51_{\pm 0.35}$	$73.29_{\pm 1.02}$
PRISM	$\mathbf{87.77}_{\pm 0.57}$	$\mathbf{98.89}_{\pm 0.26}$	$\mathbf{93.47}_{\pm 0.47}$	$\mathbf{76.51}_{\pm 1.38}$	$\mathbf{82.52}_{\pm 0.35}$	$\mathbf{73.76}_{\pm 0.55}$

Run-to-run spread is small beside the differences the paper reports. The largest standard deviation on any benchmark is 1.94, on AQuA, where the three arms conditioned on the query alone are least stable, and most lie below 0.6, while the benchmark-averaged accuracy varies by at most 0.20 within an arm. Against that, PRISM leads SFT by 3.18 on the benchmark average, an order of magnitude larger than either arm's spread.

The per-benchmark picture is the one the main text describes, with one qualification. Treating the seeds as paired runs, PRISM exceeds SFT in all three on GSM8K, SVAMP, AQuA, HumanEval and MMLU; on MultiArith the two are indistinguishable (98.89 against 98.83, paired difference $+0.06 \pm 0.26$, with PRISM ahead in one seed, level in one and behind in one), so the 0.34% margin visible in Table 1 is seed noise rather than an effect, and MultiArith is better read as saturated. PRISM exceeds ARG-DESIGNER on the mean of every benchmark, including MultiArith ($+0.56$) and AQuA ($+1.18$), and does so in all three seeds.

Two properties of SFT also reproduce. Its AQuA accuracy remains near 63% across the evaluated seeds while ARG-DESIGNER stays near 75, so the uneven transfer documented in §5.3 is a property of the training configuration and not an unlucky run. And SFT's benchmark average, $82.31_{\pm 0.10}$, sits below the checkpoint it starts from, $83.03_{\pm 0.17}$: continued supervised fine-tuning on the verified corpus buys large gains on GSM8K and HumanEval and pays for them elsewhere, which is the pattern PRISM is designed to avoid.

E FILM RANK

Figure 6 varies the rank from the default $r = 16$ to a full-width channel, a $\sim 30\times$ range in channel parameters. Every setting stays within 2.7% of the default, less than PRISM's own 3.2% average margin over SFT; the largest deviation is MMLU (-2.7%) at $r = 64$. The ordering inside that range is not systematic: $r = 64$ is weakest on three of the six benchmarks, and one setting edges past the default on two benchmarks while another matches it. The four rank averages lie within 1.49%, all above SFT (82.28), and accuracy does not track capacity; rank is not a sensitive hyperparameter here.

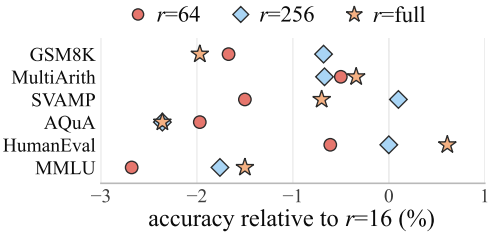


Figure 6: Accuracy relative to the default rank $r = 16$ (Eq. 4), all else fixed.

F CORPUS SPLITS AND SIBLING AVAILABILITY

Table 9 reports the three verified corpora of §5, their query-level splits, and the share of queries for which a privileged sibling exists.

Graphs per query and distinct topologies per query are all but identical in every corpus (2.91 against 2.91, 7.59 against 7.56, 3.65 against 3.65), which fixes how the cold start treats repeated successes: it retains at most one graph per (query, topology) pair. That accounts for the gap between execution and corpus counts in Table 6, where 5,836 successes reduce to 2,947 graphs, each distinct pair having been solved roughly twice on average.

Table 9: Verified corpora, splits, and sibling availability. Queries with a sibling is the share having at least two distinct verified topologies; the remainder receive a zero privileged vector, so the teacher is conditioned on the query alone. Validation is a query-level 20% of each corpus, used only to select the teacher checkpoint; it is not drawn from any evaluation set.

Corpus	graphs	queries	graphs per query	topologies per query	with sibling	train / val queries	train / val graphs
MIXCHAIN	2,947	1,014	2.91	2.91	76.9%	811 / 203	2,359 / 588
MMLU	1,890	249	7.59	7.56	87.1%	199 / 50	1,524 / 366
MBPP	981	269	3.65	3.65	90.3%	215 / 54	777 / 204

Sibling availability is high but not universal: 23.1% of MIXCHAIN queries, 12.9% of MMLU queries, and 9.7% of MBPP queries have a single verified topology and therefore fall back to query-only conditioning (§4). This bounds the fraction of each training set the privileged channel can reach, and it is a corpus property the ablations can be read against: coverage is widest on MBPP and MMLU, and MIXCHAIN, the largest corpus, is also the one whose queries most often admit a single verified topology.

G OUTPUT CONDITIONALITY

Each designer is resampled eight times per query on a 200-question GSM8K subset at $\tau = 1.0$, a topology being (role multiset, $|\mathcal{V}|$, $|\mathcal{E}|$). $H(G)$ is the entropy of the pooled shape distribution, $H(G \mid Q)$ the mean entropy within a query, and $I(Q;G) = H(G) - H(G \mid Q)$.

Table 10: Output conditionality. *null* is the mean $I(Q;G)$ the same arm produces when its own samples are reassigned to random queries, i.e. its estimator bias at this support and sample count; that bias grows with the number of distinct outcomes and so is not equal across arms, which is why each arm is judged against its own null rather than a shared correction. *excess* is I minus that null. p is a one-sided permutation test over 200 draws; the interval is a percentile bootstrap over queries (500 resamples). **Bold** marks the larger excess of the two arms post-trained from θ_0, SFT and PRISM.

Arm	$H(G)$	$H(G \mid Q)$	I	null	excess	p	95% CI	top-1
cold start	1.569	1.041	0.528	0.511	0.017	0.045	[0.485, 0.558]	0.557
SFT	2.584	1.497	1.087	1.036	0.051	0.005	[1.031, 1.116]	0.346
PRISM	2.772	1.584	1.188	1.028	**0.160**	0.005	[1.124, 1.221]	0.157
ARG-DESIGNER	2.336	1.227	1.109	0.732	0.377	0.005	[1.047, 1.152]	0.171

All three arms that share the cold-start initialisation show query dependence above their own null (permutation $p \leq 0.045$), so we do not claim that supervised fine-tuning ignores the query. The sizes differ substantially: the excess of $I(Q;G)$ over each arm's null is 0.017 nats for the cold start, 0.051 for SFT and 0.160 for PRISM, which is 1.1%, 2.0% and 5.8% of each arm's output entropy, and the bootstrap intervals for SFT and PRISM do not overlap. Most of the raw plug-in statistic is estimator bias at eight samples per query, which is why the excess rather than the raw value is the comparable quantity. The deployed ARG-DESIGNER checkpoint is the most query-conditional arm measured (excess 0.377 nats over its own null) while being the least accurate of the three deployed designers in Table 1. Greater measured query dependence therefore does not necessarily correspond to higher downstream accuracy. These diagnostics show that the generated graph distributions depend on the query; they do not establish that this dependence accounts for the accuracy improvement.

H TOPOLOGY INTERVENTION

The controls of §5.4 compare designers, so topology and the designer that produced it vary together. We therefore execute chosen crews directly. The question set is the 91 AQuA questions on which

SFT emitted its modal crew – the "SFT emitted default" group of Table 4. Every condition supplies only a role multiset; wiring is a chain in all conditions, with one round, the same decision method and the same backbone, and each condition is executed three times.

Table 11: Executing chosen crews on the same 91 AQuA questions. *mean* is the average accuracy over three executions and *vote* the accuracy of the per-question majority verdict across them; the McNemar counts in Table 12 are computed from that verdict and pair with *vote*, not with *mean*. SFT's own crew and its modal crew in canonical order share a role multiset by construction and differ only in the order of roles along the chain.

Condition	mean	sd	min–max	vote	correct
SFT's own crew	61.54	1.79	59.34–63.74	60.44	55/91
SFT's modal crew, canonical order	53.85	2.37	51.65–57.14	53.85	49/91
size-matched random crew	75.46	0.52	74.73–75.82	82.42	75/91
PRISM's crews, permuted across questions	78.39	2.07	76.92–81.32	79.12	72/91
PRISM's own crew	77.66	4.14	74.73–83.52	79.12	72/91

Table 12: Paired comparisons on the same 91 questions. b and c are questions solved only by the first and only by the second condition; exact McNemar uses those two counts alone. p(vote) is computed on the majority verdict across three executions; p per run tests each execution separately.

Comparison	b	c	p(vote)	p per run
modal crew vs PRISM	7	30	**0.0002**	0.002, 0.001, 0.000
SFT's crew vs PRISM	4	21	**0.0009**	0.087, 0.036, 0.000
modal crew vs random	2	28	$<$ **0.0001**	0.003, 0.001, 0.000
PRISM vs random	6	9	0.6072	1.000, 1.000, 0.118
PRISM vs permuted	9	9	1.0000	0.832, 0.832, 0.804
permuted vs random	4	7	0.5488	1.000, 0.839, 0.302

Executing SFT's own crew reaches 61.54% against PRISM's 77.66% on the same questions, a difference of 4 against 21 paired disagreements (exact McNemar $p = 0.0009$). With the designers removed from the comparison and wiring, rounds, decision method and backbone held fixed, the crew is what differs, so this is causal evidence at the level of topology rather than the association of §5.4.

Two further conditions bound how much this supports. A size-matched random crew reaches 75.46% and is not distinguishable from PRISM's crews (6 against 9, $p = 0.61$), and we detect no statistically significant difference between the original and permuted crew assignments on this subset (9 against 9, $p = 1.00$). The supported conclusion is therefore narrow: SFT's modal crew is worse than the alternatives on these questions, and both the random-crew and permuted-crew controls recover much of the observed performance gap. This intervention does not establish that PRISM's particular crews, or their assignment to particular queries, carry benefit beyond that. We note that the question set is selected by SFT's behaviour, which works against detecting a matching effect, so the permutation result is a failure to detect rather than evidence of absence.

Composition is also not the whole story. Forcing the modal crew in canonical order scores 53.85% against 61.54% for the same multiset in the order SFT emitted, a difference larger than any per-condition spread in Table 11, so role order along the chain matters at fixed composition and the crew abstraction used throughout §5.3 is not order-invariant.

I Evaluation benchmarks

Table 13 lists the six benchmarks on which every arm is evaluated.

Table 13: Benchmarks used for evaluation.

Category	Dataset	Answer Type	#Test
General reasoning	MMLU	Multi-choice	1,531
Math reasoning	GSM8K	Number	1,319
	MultiArith	Number	600
	SVAMP	Number	1,000
	AQuA	Multi-choice	254
Code generation	HumanEval	Code	164

J Qualitative Examples

To make the difference between designers concrete, we show the crew each one builds, and the answer the resulting system returns, on six GSM8K problems (Qwen2.5-7B-Instruct). In each, the answer shown is the one the system states explicitly, so no answer parsing is involved.

J.1 Example 1: Problem 652

An arithmetic word problem. The baseline routes it through two Programming Experts and an Inspector and returns 105; PRISM assigns three Math Solvers and reaches the gold answer of 75.

Problem — **Gold answer: 75**

James hires a horse-drawn carriage from 5 PM to 9 PM. He gets 1 hour free. The first paid hour is $15 and each hour after that is twice the cost. How much did he pay?

ARG-Designer (baseline) — **Answer: 105** ×

Crew: Programming Expert → Inspector → Programming Expert

PRISM — **Answer: 75** ✓

Crew: Math Solver → Math Solver → Math Solver

J.2 Example 2: Problem 847

A simple-interest problem. The baseline assigns three Inspectors and returns 10,298.44, an order of magnitude above the gold answer; PRISM replaces the middle Inspector with a Mathematical Analyst and returns 975.

Problem — **Gold answer: 975**

Brenda invested $300 into a scheme that accumulates simple interest at the rate of three-quarters of the original amount per year. How much in total will she have after 3 years?

ARG-Designer (baseline) — **Answer: 10298.44** ×

Crew: Inspector → Inspector → Inspector

PRISM — **Answer: 975** ✓

Crew: Inspector → Mathematical Analyst → Inspector

J.3 Example 3: Problem 1312

A multi-step revenue problem. The baseline assigns three Inspectors, and its logged reasoning drops the six reservations entirely: it prices a single reservation at \$15 a night and returns 30. PRISM assigns three Math Solvers, which keep every step ($6 \times \$15 = \$90$ a night, then $\$90 \times 2 = \$180$) and reach the gold answer.

Problem — **Gold answer: 180**

Tom's restaurant gets 6 reservations a night. They normally order 2 meals that cost \$5 each and a \$5 bottle of wine. How much do they make a week if they are open 2 days a week?

ARG-Designer (baseline) — **Answer: 30** ×

Crew: Inspector → Inspector → Inspector

PRISM — **Answer: 180** ✓

Crew: Math Solver → Math Solver → Math Solver

J.4 Example 4: Problem 763

The baseline, with two Programming Experts and an Inspector, sets up the wrong equation ($x + x/2 + 2x = 250$), obtains 142.86 and rounds it to 143, although a count of cases must come out whole. PRISM assigns two Math Solvers and recovers 50, 100 and 100 cases for the three houses.

Problem — **Gold answer: 100**

Rob, Royce, and Pedro are contractors getting ready to put a new roof on three homes. If the three homes will need 250 cases of shingles, with the first house needing 1/2 of the second, and the third needing double the first. How many cases of shingles will the third house need?

ARG-Designer (baseline) — **Answer: 143** ×

Crew: Programming Expert → Inspector → Programming Expert

PRISM — **Answer: 100** ✓

Crew: Math Solver → Math Solver

J.5 Example 5: Problem 322

The baseline routes the problem through two Programming Experts and an Inspector, the same role composition it builds in Examples 1 and 4, and returns 8. PRISM assigns an Inspector, a Mathematical Analyst and a Math Solver and reaches the gold answer of 9.

Problem — **Gold answer: 9**

Jam has three boxes full of pencils and 2 loose pencils which give a total of 26 pencils. If her sister, Meg, has 46 pencils, how many boxes do Jam and Meg need to store all their pencils?

ARG-Designer (baseline) — **Answer: 8** ×

Crew: Programming Expert → Inspector → Programming Expert

PRISM — **Answer: 9** ✓

Crew: Inspector → Mathematical Analyst → Math Solver

J.6 EXAMPLE 6: PROBLEM 664

The baseline assigns three Inspectors, the same crew it builds in Examples 2 and 3, and returns 48, which is the total class time in hours rather than the number of recipes. PRISM assigns an Inspector, a Programming Expert and a Math Solver and reaches the gold answer of 32.

Problem **Gold answer: 32**

John decides to take a cooking class. The class meets 4 times a week for 2 hours each time for 6 weeks. If he learns a new recipe for every 1.5 hours of class time, how many recipes does he learn?

ARG-DESIGNER (baseline) **Answer: 48** ×

Crew: Inspector → Inspector → Inspector

PRISM **Answer: 32** ✓

Crew: Inspector → Programming Expert → Math Solver