DISCOVERING EFFICIENT AND EXPLAINABLE COMMUNICATION TOPOLOGIES FOR LLM-BASED MULTI-AGENT SYSTEMS VIA CAUSAL INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

The performance of large language model (LLM)-based multi-agent systems (MAS) largely depends on effective communication topologies. Existing topology generation methods, however, typically learn communication topologies through black-box optimization driven solely by task-level rewards. While effective, such optimization provides little insight into why particular communication edges are selected, making it difficult to identify the critical communication subgraphs responsible for successful collaboration. To address this limitation, we propose E2-Explainer, a model-agnostic framework for providing interpretable explanations of communication topologies produced by arbitrary topology generators. Specifically, we formulate topology explanation as a causal attribution problem that identifies compact communication subgraphs supported by edge-level evidence of task preservation. We obtain this evidence with a Granger-style objective that measures how masking each communication channel changes the task outcome and the stability of the final response. The resulting budgeted subgraphs are then distilled into an amortized explainer, enabling efficient post-hoc explanation without repeated edge-level evaluations at deployment. Extensive experiments on multiple reasoning and coding benchmarks demonstrate that E2-Explainer identifies critical communication subgraphs that preserve successful collaboration. These subgraphs can also be executed directly to prune redundant communication edges, substantially reducing communication costs while maintaining competitive task performance.

1 INTRODUCTION

Large language model (LLM)-based agents have demonstrated remarkable capabilities across a wide range of complex tasks, including question answering (Xu et al., 2024), code generation (Zhang et al., 2024), and autonomous driving (Jin et al., 2023). However, organizing multiple LLM-based agents into a cohesive team capable of exhibiting human-like collective intelligence poses a range of new challenges (Du et al., 2024; Wu et al., 2024; Li et al., 2023; Qian et al., 2024; Hong et al., 2024). Among these challenges, a fundamental one lies in designing the communication topology—that is, determining how agents should be connected and how they should transmit information during collaboration—as it directly shapes information flow, coordination effectiveness, and ultimately the collective intelligence of the system.

Early studies typically relied on manually specified and predefined topologies (Du et al., 2024; Wu et al., 2024; Li et al., 2023; Qian et al., 2024; Hong et al., 2024). Although simple and effective, these topologies are largely task-agnostic, limiting their adaptability to diverse tasks and evolving collaboration demands. To improve task adaptivity and communication efficiency, recent works have increasingly explored learnable communication topology designers, which broadly fall into two paradigms. Pruning-based approaches begin with an existing collaboration workflow and remove unnecessary agents or communication links through learned gating mechanisms, structured dropout, or importance estimation (Zhang et al., 2025a; Wang et al., 2025; Li et al., 2025). Generation-based approaches, by contrast, formulate topology design as a task-conditioned graph generation problem and synthesize collaboration topologies using autoregressive node-and-edge decoding, mixture-of-

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.

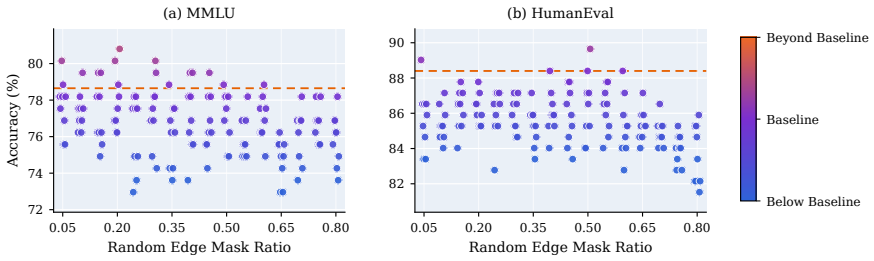


Figure 1: Random edge masking on G-Designer-generated communication graphs for MMLU and HumanEval. The edge masking ratio ranges from 0.05 to 0.80 in increments of 0.05, with ten random seeds evaluated at each ratio. Each point denotes the accuracy obtained with one random mask, while the dashed line marks the performance of the original unmasked G-Designer graph. Most random masks degrade performance, whereas only a few preserve or improve accuracy, indicating that task-preserving compact subgraphs exist but are difficult to identify through random pruning.

experts graph generators, or guided graph diffusion (Li et al., 2026a;b; Jiang et al., 2026; Zhang et al., 2026).

Despite their impressive performance, we observe that the communication topologies generated by existing methods still contain substantial redundancy. As illustrated in Figure 1, removing several communication links in the topologies generated by G-Designer (Zhang et al., 2025b) has little impact on the final task performance and, in some cases, even improves it. This observation suggests that the learned topologies may contain not only redundant edges but also communication patterns that are merely correlated with, rather than causally relevant to, successful collaboration. Such redundancy is difficult to discover under the black-box optimization paradigm adopted by existing methods, where topologies are optimized solely using topology-level task rewards without explicitly attributing the final outcome to individual communication decisions or substructures. Consequently, existing topology designers provide little insight into which communication substructures are truly responsible for successful collaboration. This motivates the need for a post-hoc explanation framework that can identify critical causal communication substructures, thereby revealing why a learned topology succeeds and enabling redundant communication edges to be pruned for improved efficiency.

In this work, we explicitly investigate the causal contribution of individual communication edges to the task outcomes achieved by the overall communication topology. Our formulation is inspired by the notion of Granger causality, which considers $x_i \to Y$ causal if the information provided by x_i improves the prediction of Y (Granger, 1969; Wiener, 1956). Analogously, in a communication topology, if the information transmitted through an edge improves the cooperative performance of the MAS, that edge can be regarded as causally important to the resulting collaboration. Building upon this intuition, we extend the Granger-style criterion from individual communication edges to local communication substructures, aiming to identify compact causal subgraphs that preserve task-relevant interactions while eliminating redundant communication.

Specifically, we propose E2-Explainer, a model-agnostic post-hoc framework for identifying compact causal subgraphs from communication topologies generated by arbitrary topology designers. Given a communication topology, E2-Explainer estimates the Granger-style causal contribution of each edge by systematically masking it and measuring the resulting change in a task evaluation signal. This signal is primarily defined by the final task outcome and is supplemented with result-level semantic entropy changes to provide denser feedback under sparse rewards. The estimated edge contributions are then integrated with structural constraints to identify compact causal subgraphs that preserve task-relevant communication while eliminating redundant interactions. Since this attribution process requires repeated executions of the underlying LLM-based MAS, we further train an amortized explainer using the discovered causal subgraphs as supervision. At deployment, the explainer directly predicts a compact causal subgraph from a newly generated topology, avoiding repeated edge-masking evaluations. Extensive experiments across six reasoning and coding benchmarks demonstrate that E2-Explainer consistently improves the performance–communication cost trade-off of multiple topology designers while exhibiting strong transferability across different designers and agent scales.

Our contributions are summarized as follows:

- We formulate post-hoc explanation of optimized LLM-MAS communication graphs as the identification of compact subgraphs that preserve the task behavior of the original topology, and introduce a Granger-style criterion for evaluating the contribution of communication edges.
- We propose E2-Explainer, which distills edge-level causal evidence obtained from offline masking evaluations into budget-specific subgraph supervision and trains an amortized explainer to directly generate compact, collaboration-preserving communication subgraphs without requiring edge-contribution evaluations at test time.
- We conduct extensive experiments across multiple benchmarks, topology optimizers, and agent scales, showing that E2-Explainer reduces communication costs while maintaining competitive task performance and exhibits cross-generator and cross-scale transferability.

2 Related Work

LLM-based multi-agent systems. LLM-based multi-agent systems organize multiple language-model agents into collaborative workflows, where agents exchange intermediate solutions, critique one another, and aggregate final decisions (Du et al., 2024; Wu et al., 2024; Li et al., 2023; Qian et al., 2024; Hong et al., 2024). They build on decomposed reasoning, self-consistency, and multi-perspective deliberation to improve LLM reliability (Wei et al., 2022; Wang et al., 2023; Kojima et al., 2022), and have been summarized in recent surveys on agent workflows and applications (Guo et al., 2024; Li et al., 2024; Chen et al., 2024). Our work focuses on their communication structure rather than new roles or prompting strategies.

Communication topology optimization. A growing line of work studies how to design efficient communication topologies for LLM-MAS. Recent methods can be roughly grouped into generative topology methods, which construct task-adaptive collaboration graphs, such as G-Designer and ARG-Designer (Zhang et al., 2025b; Li et al., 2026a), and pruning or selection methods, which remove unnecessary links, agents, or graph components from an existing workflow, such as AgentPrune, AgentDropout, Cut the Crap, and adaptive graph pruning methods (Zhang et al., 2025a; Wang et al., 2025; Li et al., 2025; Cang et al., 2026). These methods show that topology strongly affects the performance–cost trade-off. E2-Explainer is complementary: it treats generated topologies as candidate graphs and learns to refine them into smaller task-preserving communication subgraphs.

Causality in machine learning. Causality provides a principled framework for reasoning about interventions, counterfactuals, and invariant mechanisms (Pearl, 2009; Peters et al., 2017; Schölkopf et al., 2021). In machine learning, causal ideas have been used to analyze model behavior under controlled perturbations and to identify factors that are responsible for predictions (Chattopadhyay et al., 2019; Schwab & Karlen, 2019). Granger causality characterizes causal informativeness by whether access to one variable improves the prediction of another (Granger, 1969; 1980; Bressler & Seth, 2011). In graph learning, this intuition has been connected to explanation methods that identify compact graph structures responsible for preserving or changing model behavior, such as GNNExplainer, PGExplainer, counterfactual explainers, and GEM (Ying et al., 2019; Luo et al., 2020; Lucic et al., 2022; Lin et al., 2021). Inspired by these ideas, we adapt Granger-style causal reasoning from differentiable graph models to frozen LLM-MAS communication graphs. Unlike prior graph explanation methods that primarily produce explanations for inspection, E2-Explainer distills offline edge-masking evidence into an amortized explainer that directly outputs budgeted task-preserving communication subgraphs at deployment.

3 Problem Formulation

3.1 LLM-MAS Communication Graphs

We consider an LLM-based multi-agent system with agents $V = \{a_1, \ldots, a_n\}$. Given query x, a topology generator $\mathcal{G}_\phi$ produces a directed communication graph $G = \mathcal{G}_\phi(x) = (V, E)$, where $e = (a_i, a_j) \in E$ delivers the output of agent a_i to agent a_j. Executing G yields a final

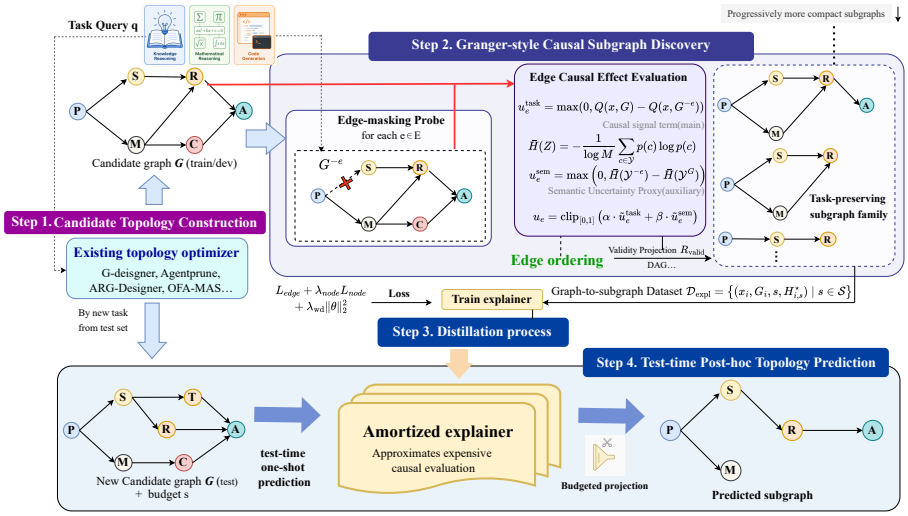


Figure 2: Overview of E2-Explainer. Edge masking estimates preservation utilities from task-score and semantic-entropy changes, which yield budgeted subgraphs for supervision. The amortized explainer then predicts a task-preserving subgraph for an unseen topology in one forward pass.

response $y(x, G)$ and a dataset-specific task score $Q(x, G)$, such as exact-match accuracy, execution correctness, or an LLM-judged score.

3.2 Post-hoc Communication Explanation

Given a generated graph G, we seek a compact and executable subgraph $H = (V_H, E_H), \quad H \subseteq G$, that preserves its task behavior. The retained edges explain the communication paths supporting the original collaboration, while executing H may reduce communication overhead. We denote the realized online cost by $C(x, H)$, measured using actual execution tokens. Because removing an edge changes downstream prompts and messages, $C(x, H)$ need not vary monotonically with the number of retained edges or nodes.

3.3 Objective and Deployment Constraint

Let $b = (\rho_E, \rho_V)$ denote a subgraph budget, where ρ_E and ρ_V specify the desired edge-pruning and node-pruning ratios, and let $\mathcal{H}_b(G)$ be the corresponding family of compact subgraphs. A valid explanation under budget b should satisfy

$$H_b^\star \in \mathcal{H}_b(G), \qquad Q(x, H_b^\star) \approx Q(x, G). \tag{1}$$

The budget determines compactness, while the task-behavior condition defines faithfulness. We report $C(x, H_b^\star)$ as measured efficiency rather than assuming that structural sparsity guarantees a fixed token reduction. Because validating candidate subgraphs requires repeated LLM-MAS executions and may require task feedback, we learn an amortized explainer F_θ:

$$\widehat{H}_b = F_\theta(x, G, b), \quad \widehat{H}_b \subseteq G. \tag{2}$$

4 E2-Explainer

As illustrated in Figure 2, E2-Explainer is a model-agnostic post-hoc framework for identifying compact causal subgraphs from communication topologies generated by arbitrary topology designers. Given a communication topology, E2-Explainer estimates the Granger-style causal contribution of each edge by systematically masking it and measuring the resulting change in a task evaluation signal. This signal is primarily derived from the final task outcome and further incorporates topology-level

semantic entropy changes to provide denser feedback when the task outcome is sparse or coarse-grained. The estimated causal contributions are then combined with budget constraints and structural requirements to construct executable compact subgraphs that preserve task-relevant communication patterns. Since this attribution process requires repeated edge-masking evaluations of the underlying LLM-based MAS, we use the discovered subgraphs as supervision to train an amortized explainer, which can directly predict a budget-specific subgraph for a newly generated topology at test time.

4.1 Edge-Level Granger-Style Causal Attribution

To estimate the Granger-style causal contribution of each communication edge under the original topology, we perform single-edge removal interventions while keeping all other factors fixed. For a query x, graph G, and edge $e = (u, v)$, we preserve the same agents, prompts, decoding configurations, and all other communication channels, while intervening on e by blocking the message transmitted from source agent u to target agent v. Following the Granger criterion, we quantify the contribution of e by measuring the change in the evaluation signal before and after this intervention.

Task-level causal contribution. For each active edge $e \in E$ of the generated graph $G = (V, E)$, we construct the intervened graph $G^{-e} = G \setminus \{e\}$ and re-execute the frozen LLM-MAS. Let $Q(x, G)$ and $Q(x, G^{-e})$ denote the task scores obtained before and after intervening on e, respectively. We define the task-level causal contribution of e as

$$\Delta_e^{\text{task}} = Q(x, G) - Q(x, G^{-e}). \tag{3}$$

A positive value indicates that retaining edge e improves the task outcome relative to removing it, whereas a negative value suggests that the information transmitted through e may be detrimental to the task outcome. Since our explanation focuses on communication that supports the original collaboration, we retain only the nonnegative preservation contribution:

$$u_e^{\text{task}} = \max\left(0, \Delta_e^{\text{task}}\right). \tag{4}$$

A larger value provides stronger evidence that edge e contributes to preserving the task behavior of the candidate graph, while zero indicates no positive preservation evidence under the observed task metric.

Auxiliary semantic signal. The final task outcome can be too coarse to distinguish edge masks that receive the same task score. We therefore use result-level semantic entropy as an auxiliary signal to characterize the stability of the final response (Kuhn et al., 2023; Farquhar et al., 2024). For the response set $\mathcal{Y}$ obtained from M stochastic executions, we group semantically equivalent responses into classes $\mathcal{C}(\mathcal{Y})$ and compute

$$\bar{H}(\mathcal{Y}) = -\sum_{c \in \mathcal{C}(\mathcal{Y})} p(c) \log p(c), \tag{5}$$

where $p(c)$ is the empirical frequency of class c among the M executions. We use $M = 5$ independently sampled executions for both the original graph and each edge-masked graph. The auxiliary semantic effect of edge e is defined as

$$u_e^{\text{sem}} = \max\left(0,\ \bar{H}(\mathcal{Y}^{G^{-e}}) - \bar{H}(\mathcal{Y}^{G})\right). \tag{6}$$

A positive value indicates that masking e increases the uncertainty of the final response, providing additional evidence that the edge contributes to preserving the behavior of the original graph.

Unified edge effect. We combine the task-level causal effect with the auxiliary semantic stability signal. The resulting edge utility is

$$u_e = \text{clip}_{[0,1]}\left(\alpha \cdot \widetilde{u}_e^{\text{task}} + \beta \cdot \widetilde{u}_e^{\text{sem}}\right), \tag{7}$$

where $\widetilde{\cdot}$ denotes normalization within the current graph or calibration batch, and $\alpha, \beta \geq 0$ control the two signals. The utilities induce a preservation ordering over the edges of G:

$$\pi_G = (e_{(1)}, e_{(2)}, \ldots, e_{(|E|)}), \quad u_{e_{(1)}} \geq u_{e_{(2)}} \geq \cdots \geq u_{e_{(|E|)}}. \tag{8}$$

This ordering connects edge-level Granger-style attribution to budgeted subgraph construction.

4.2 CAUSAL SUBGRAPH EXTRACTION

The edge-level causal contributions obtained above provide a principled basis for extracting compact communication subgraphs. Given a target budget b, E2-Explainer selects edges according to their preservation utilities and constructs a subgraph that retains the most task-relevant information flows while satisfying the structural constraints inherited from the original workflow. This budget-aware extraction process achieves a desirable trade-off between communication sparsity and task preservation without requiring exhaustive subgraph enumeration. Specifically, the budget determines the number of retained edges, while the causal ordering determines which communication paths are preserved.

Budget-Aware Subgraph Selection. The budget $b = (\rho_E, \rho_V)$ specifies the desired retention ratios of edges and optional nodes. For each graph, these ratios are converted into retention counts $k_E(b)$ and $k_V(b)$ according to the graph size. We first select the most important edges according to the preservation ordering obtained from edge attribution:

$$E_b^\star = \mathrm{TopK}\left(\pi_G, k_E(b)\right). \tag{9}$$

The node budget is subsequently enforced jointly with the structural projection described below to obtain an executable communication subgraph.

Executability-Constrained Projection. To transform the selected high-utility edges into an executable communication topology, we apply an executability-constrained projection:

$$H_b^\star = \mathcal{R}_{\mathrm{valid}}(G, E_b^\star, k_V(b)). \tag{10}$$

The projection $\mathcal{R}_{\mathrm{valid}}$ only selects edges from the original edge set E and never introduces new links. Therefore, when the original topology G is a directed acyclic graph, the extracted subgraph naturally preserves acyclicity. Specifically, the projection retains mandatory input and final-aggregation agents, removes communication links incident to excluded optional agents, and discards optional agents that no longer participate in valid directed communication paths. When the node budget is active, optional agents are prioritized according to the aggregated preservation utilities of their incident retained edges. The resulting $H_b^\star$ therefore satisfies both the specified budget constraints and the execution requirements of the underlying LLM-MAS.

Budget-Specific Explanation Family. Applying Eq. (10) to the budget set $\mathcal{B}$ yields an explanation family $\mathcal{F}^\star(x, G) = \{H_b^\star : b \in \mathcal{B}\}$. Each member provides an explanation of the same candidate topology under a different compactness requirement. Collectively, these subgraphs characterize different trade-offs between communication efficiency and task preservation. As the retention budget increases, the extracted subgraphs progressively incorporate lower-ranked communication edges to recover additional information flows.

4.3 AMORTIZED SUBGRAPH EXPLAINER

The causal attribution and structural projection procedures described above provide a reliable mechanism for identifying compact, task-preserving subgraphs from arbitrary communication topologies. However, directly applying this procedure to a newly generated topology at test time is computationally expensive, as it requires repeated edge-masking interventions and multiple executions of the underlying LLM-MAS to estimate edge contributions. To overcome this limitation, we amortize the causal extraction process by collecting extracted subgraphs from generated topologies as graph-to-subgraph supervision. Specifically, these causally grounded subgraphs serve as explanation targets that teach an explainer to predict which communication paths should be preserved under different compactness budgets. After training, the amortized explainer can directly generate budget-specific causal subgraphs for unseen topologies without additional causal evaluations.

Graph-to-subgraph supervision. For each query x_i and candidate graph G_i, the extraction procedure produces one target subgraph $H_{i,b}^\star$ for every budget $b \in \mathcal{B}$. These targets form the training set $\mathcal{D}_{\mathrm{expl}} = \left\{(x_i, G_i, b, H_{i,b}^\star) \mid b \in \mathcal{B}\right\}$. Thus, the explanation family provides supervision rather than an input to the explainer. For a fixed graph, the targets share the same causal preservation ordering

but correspond to different pruning budgets. Learning from the complete family therefore teaches the explainer both which communication elements are important and how their retention changes with the requested compactness, allowing one model to serve multiple budgets.

Explainer mapping. We train a parameterized explainer F_θ to approximate the graph-to-subgraph mapping induced by the causal extraction procedure: $\widehat{H}_{i,b} = F_\theta(x_i, G_i, b), \qquad \widehat{H}_{i,b} \subseteq G_i$. Given a query, a candidate graph, and a pruning budget, F_θ internally predicts edge and node retention probabilities and decodes them into the corresponding budget-specific subgraph. The explainer is independent of the topology generation mechanism, with architectural details deferred to the Appendix.

Training objective. Let $A^\star_{i,b,e}$ indicate whether edge e belongs to $H^\star_{i,b}$, and let $R^\star_{i,b,v}$ indicate whether node v is retained. The explainer predicts the corresponding probabilities $\widehat{A}_{i,b,e}$ and $\widehat{R}_{i,b,v}$, which are collected into the edge- and node-retention vectors $\widehat{\mathbf{A}}_{i,b}$ and $\widehat{\mathbf{R}}_{i,b}$, respectively. We optimize

$$\begin{aligned}\mathcal{L}_{\text{edge}} &= \sum_{(x_i,G_i,b,H^\star_{i,b})\in\mathcal{D}_{\text{expl}}} \frac{1}{|E_i|}\sum_{e\in E_i} \text{BCE}\left(A^\star_{i,b,e}, \widehat{A}_{i,b,e}\right), \\ \mathcal{L}_{\text{node}} &= \sum_{(x_i,G_i,b,H^\star_{i,b})\in\mathcal{D}_{\text{expl}}} \frac{1}{|V_i|}\sum_{v\in V_i} \text{BCE}\left(R^\star_{i,b,v}, \widehat{R}_{i,b,v}\right),\end{aligned} \tag{11}$$

with the final objective

$$\mathcal{L} = \mathcal{L}_{\text{edge}} + \lambda_{\text{node}}\mathcal{L}_{\text{node}} + \lambda_{\text{wd}}\|\theta\|_2^2. \tag{12}$$

Subgraph prediction. The predicted probabilities are first converted into budget-compatible candidate sets:

$$\widehat{E}_{i,b} = \text{TopK}\left(\widehat{\mathbf{A}}_{i,b}, k_E(b)\right), \qquad \widehat{V}_{i,b} = \text{TopK}\left(\widehat{\mathbf{R}}_{i,b}, k_V(b)\right). \tag{13}$$

Accordingly, the final subgraph is obtained as

$$\widehat{H}_{i,b} = F_\theta(x_i, G_i, b) = \mathcal{R}_{\text{valid}}\left(G_i, \widehat{E}_{i,b}, \widehat{V}_{i,b}\right), \tag{14}$$

where mandatory nodes are retained by $\mathcal{R}_{\text{valid}}$. At deployment, this process requires neither task labels nor repeated edge interventions.

5 Experiment

5.1 Experimental Setup

Datasets and backbone. We evaluate on six benchmarks: AQuA, GSM8K, MultiArith, and SVAMP for mathematical reasoning (Ling et al., 2017; Cobbe et al., 2021; Roy & Roth, 2015; Patel et al., 2021), MMLU for knowledge-intensive reasoning (Hendrycks et al., 2021), and HumanEval for code generation (Chen et al., 2021). All methods use Qwen3-8B as the backbone LLM (Yang et al., 2025), with Vanilla, CoT (Wei et al., 2022), and SC(CoT) (Wang et al., 2023) as single-agent baselines.

Candidate generators and calibration. We train a single explainer using only G-Designer-generated candidate graphs. For each dataset, 40 calibration inputs are sampled from the training split without overlap with the test split, and edge-masking probes construct graph-to-subgraph supervision. The frozen explainer is then applied without retraining to graphs produced by ARG-Designer, OFA-MAS, AgentPrune, and G-Designer.

Subgraph setting and metrics. The main comparison uses $E25 + N20$ as the default subgraph-size setting, while other settings are examined in the appendix. We report task accuracy and actual online token usage; changes for E2-Explainer are computed relative to the corresponding candidate generator.

Table 1: Main results with Qwen3-8B as the backbone LLM. Boldface indicates the best accuracy for each dataset. For E2-Explainer variants, Acc. arrows indicate changes relative to the corresponding candidate generator, while the Avg. column reports average accuracy and weighted token reduction across all datasets.

Method	Metric	MMLU	GSM8K	MultiArith	SVAMP	AQuA	HumanEval	Avg.
Vanilla	Acc.	72.78	88.15	96.67	93.40	82.68	81.37	85.84
CoT	Acc.	76.81	91.81	97.67	93.60	83.07	87.58	88.42
SC(CoT)	Acc.	77.34	92.48	97.67	93.83	84.13	87.08	88.76
ARG-Designer	Acc.	76.68	89.18	97.81	93.57	83.95	86.78	88.00
ARG-Designer + E2-Explainer	Acc.	77.99↑1.31	86.94↓2.24	97.17↓0.64	94.67↑1.10	84.58↑0.63	89.26↑2.48	88.44↑0.44
	Token Δ	-12.1%	-15.5%	-23.3%	-21.1%	-29.8%	-14.3%	-20.1%
OFA-MAS	Acc.	79.91	91.01	98.21	93.97	84.11	89.26	89.41
OFA-MAS + E2-Explainer	Acc.	**81.87**↑1.96	90.40↓0.61	97.41↓0.80	92.65↓1.32	86.67↑2.56	90.62↑1.36	89.94↑0.53
	Token Δ	-15.1%	-27.2%	-25.5%	-23.8%	-17.8%	-26.8%	-25.6%
AgentPrune	Acc.	78.43	92.78	**98.33**	94.90	85.04	87.58	89.51
AgentPrune + E2-Explainer	Acc.	79.08↑0.65	**94.09**↑1.31	**98.33**±0.00	94.20↓0.70	86.61↑1.57	88.20↑0.62	90.09↑0.58
	Token Δ	-14.7%	-26.7%	-27.3%	-22.9%	-38.1%	-10.6%	-24.9%
G-Designer	Acc.	78.65	93.05	98.13	**95.00**	84.52	88.40	89.63
G-Designer + E2-Explainer	Acc.	79.74↑1.09	93.77↑0.72	98.17↑0.04	94.60↓0.40	**87.30**↑2.78	**90.69**↑2.29	90.71↑1.09
	Token Δ	-31.5%	-44.0%	-28.0%	-20.1%	-37.5%	-19.3%	-25.1%

5.2 Results and Analysis

Main comparison. Table 1 evaluates E2-Explainer as a post-hoc module for four representative topology optimizers. Across all four optimizers, E2-Explainer reduces weighted token usage by 20.1%–25.6%, while preserving or improving their average accuracy. Specifically, it improves the average accuracy of ARG-Designer, OFA-MAS, AgentPrune, and G-Designer by 0.44, 0.53, 0.58, and 1.09 points, respectively. G-Designer+E2-Explainer achieves the highest average accuracy of 90.71 while consuming 25.1% fewer tokens than the original G-Designer. These results directly support the primary objective of E2-Explainer: reducing communication overhead without sacrificing the collaborative effectiveness of an already optimized topology.

The fine-grained results show that the reduction in communication cost is more consistent than the change in task accuracy. Across the 24 optimizer–dataset combinations, E2-Explainer reduces token usage in every case, with dataset-level reductions ranging from 10.6% to 44.0%. Meanwhile, accuracy improves in 16 combinations and remains unchanged in one. Improvements are particularly consistent on MMLU, AQuA, and HumanEval, where all four topology optimizers benefit from post-processing. Results on GSM8K, MultiArith, and SVAMP are more mixed, indicating that the amount of removable communication depends on both the task and the initial candidate topology. Nevertheless, the universal token reduction and predominantly preserved or improved accuracy show that compact task-preserving subgraphs can often be recovered from optimized communication graphs. The additional accuracy gains further suggest that some candidate topologies still retain redundant or distracting communication whose removal can benefit answer quality.

A key observation is that the improvement transfers beyond the generator used for calibration. The explainer is trained only on G-Designer graphs but applied without retraining to ARG-Designer, OFA-MAS, and AgentPrune. These unseen generators have different construction biases, covering autoregressive graph generation, mixture-of-experts graph generation, and pruning-based topology optimization. Nevertheless, E2-Explainer improves their average accuracy by 0.44–0.58 points and reduces weighted online cost by 20.1%–25.6%. Since these generators are unseen during calibration, the transfer is unlikely to come from memorizing G-Designer-specific adjacency patterns. Instead, the explainer appears to capture reusable cues about which communication paths are necessary under a given task and graph context.

Generalization across agent scales. Table 2 further tests cross-scale generalization by training the explainer on 5-agent G-Designer graphs and directly evaluating it on 6-agent graphs. Without retraining, E2-Explainer improves MMLU accuracy from 79.43 to 81.70 and HumanEval accuracy from 89.26 to 90.08, while also reducing token usage by 10.16% and 19.38%, respectively. These

Table 2: Generalization across agent scales on G-Designer-generated communication graphs. The explainer is trained on 5-agent graphs and evaluated on 6-agent graphs. Token reductions are measured against the original 6-agent topology.

Setting	MMLU		HumanEval	
	Acc.	Tok. ↓	Acc.	Tok. ↓
Original (6-agent)	79.43	–	89.26	–
E2-Explainer (5→6-agent)	**81.70**	10.16%	**90.08**	19.38%

Table 3: Component ablation of E2-Explainer on G-Designer-generated communication graphs for MMLU and HumanEval.

Method	MMLU		HumanEval	
	Acc.	Token Δ	Acc.	Token Δ
w/o Causal	75.82	-11.16%	86.88	-5.05%
w/o Sem. Entropy	77.78	-23.42%	87.50	-14.32%
E2-Explainer	**79.74**	-31.50%	**90.69**	-19.25%

Table 4: Effect of calibration sources under $E25 + N20$. *Original* denotes unrefined target graphs, and *Mixed* combines G-Designer and AgentPrune calibration graphs. Boldface marks the best refined accuracy.

Source	MMLU		HumanEval	
	Acc.	Token Δ	Acc.	Token Δ
Target: G-Designer				
Original	78.65	–	88.40	–
G-Designer	79.74	-31.50%	**90.69**	-19.25%
AgentPrune	79.08	-14.43%	88.75	-15.40%
Mixed	**80.18**	-24.55%	89.30	-20.61%
Target: AgentPrune				
Original	78.43	–	87.58	–
G-Designer	79.08	-14.70%	88.20	-10.60%
AgentPrune	**81.70**	-24.56%	**90.07**	-19.18%
Mixed	79.08	-26.69%	88.82	-13.43%

results suggest that the learned explainer transfers beyond the graph size observed during training and can still identify compact task-preserving subgraphs under moderate changes in agent scale.

Component ablation. Table 3 evaluates the roles of the causal and auxiliary semantic-entropy signals. Removing the causal signal leads to a larger degradation, reducing accuracy from 79.74 to 75.82 on MMLU and from 90.69 to 86.88 on HumanEval. The causal-only variant, namely *w/o Sem. Entropy*, performs better than the semantic-entropy-only variant, suggesting that task-level causal effects provide the primary preservation-aware signal for edge valuation. However, causal-only still falls short of the full model. This indicates that the result-level semantic entropy proxy complements causal supervision by densifying the refinement signal when final-task feedback is sparse.

Effect of Calibration Sources. Table 4 compares explainers trained on G-Designer graphs, AgentPrune graphs, or their mixture and evaluated on both target optimizers. All six calibration–target combinations improve accuracy while reducing token usage, providing evidence of bidirectional cross-optimizer transfer. Source matching remains beneficial: AgentPrune-only calibration achieves the highest accuracy on both datasets for AgentPrune targets, while G-Designer-only performs best on HumanEval for G-Designer targets. Mixed-source calibration obtains the best G-Designer result on MMLU and improves both target optimizers, but does not uniformly outperform single-source calibration. These results suggest that E2-Explainer learns transferable communication patterns while still benefiting from structural characteristics specific to the target optimizer.

6 Conclusion

We present E2-Explainer, a post-hoc framework for identifying compact and collaboration-preserving communication subgraphs from optimized LLM-MAS topologies. E2-Explainer distills offline edge-level evaluations into an amortized explainer, enabling efficient test-time generation of compact communication subgraphs that preserve collaborative task performance without requiring ground-truth answers or repeated edge-level evaluations. Experiments across multiple topology optimizers show that it reduces communication costs while maintaining task performance, and transfers across graph sources and agent scales.

AI USE STATEMENT

We used ChatGPT to assist with language editing, refinement of descriptions of the methodology and experimental results, LaTeX formatting, organization and summarization of experimental results, review of tool-use trajectories, and implementation of code for repetitive tasks. We reviewed and revised the AI-assisted text, verified the organized results against the experimental outputs, checked trajectory-review findings against the original traces, and validated the assisted code. The use of LLMs as evaluated systems and automated judges is described in the methodology and experimental setup. We take full responsibility for the accuracy, originality, and integrity of the final manuscript.

REFERENCES

Steven L. Bressler and Anil K. Seth. Wiener–granger causality: A well established methodology. *NeuroImage*, 58(2):323–329, 2011.

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

Aditya Chattopadhyay, Piyushi Manupriya, Anirban Sarkar, and Vineeth N. Balasubramanian. Neural network attributions: A causal perspective. In *International Conference on Machine Learning*, 2019.

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

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chen Qian, Chi-Min Chan, Yujia Qin, Yaxi Lu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024.

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.

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*, volume 235 of *Proceedings of Machine Learning Research*. PMLR, 2024.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630:625–630, 2024.

Clive W. J. Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*, 37(3):424–438, 1969.

Clive W. J. Granger. Testing for causality: A personal viewpoint. *Journal of Economic Dynamics and Control*, 2:329–352, 1980.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey of progress and challenges. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pp. 8048–8057, 2024. doi: 10.24963/ijcai.2024/890.

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, Jiaqi 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 Juergen Schmidhuber. Metagpt: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, 2024.

Eric Hanchen Jiang, Levina Li, Frank Wan, Xiao Liang, Sophia Yin, Yuchen Wu, Xinfeng Li, Yizhou Sun, Wei Wang, Kai-Wei Chang, and Ying Nian Wu. 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. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.1764.

Ye Jin, Xiaoxi Shen, Huiling Peng, Xiaoan Liu, Jingli Qin, Jiayang Li, Jintao Xie, Peizhong Gao, Guyue Zhou, and Jiangtao Gong. Surrealdriver: Designing generative driver agent simulation framework in urban contexts based on large language model. *arXiv preprint arXiv:2309.13193*, 5 (7):8, 2023.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pp. 22199–22213, 2022.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *International Conference on Learning Representations*, 2023.

Boyi Li, Zhonghan Zhao, Der-Horng Lee, and Gaoang Wang. Adaptive graph pruning for multi-agent communication. *arXiv preprint arXiv:2506.02951*, 2025.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for "mind" exploration of large scale language model society. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Shiyuan Li, Yixin Liu, Qingsong Wen, Chengqi Zhang, and Shirui Pan. Assemble your crew: Automatic multi-agent communication topology design via autoregressive graph generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 23142–23150, 2026a. doi: 10.1609/aaai.v40i28.39481.

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

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

Wanyu Lin, Hao Lan, and Baochun Li. Generative causal explanations for graph neural networks. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 6666–6679. PMLR, 2021.

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. Association for Computational Linguistics, 2017. doi: 10.18653/v1/P17-1015.

Ana Lucic, Maartje A. Ter Hoeve, Gabriele Tolomei, Maarten de Rijke, and Fabrizio Silvestri. Cf-gnnexplainer: Counterfactual explanations for graph neural networks. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pp. 4499–4511. PMLR, 2022.

Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. In *Advances in Neural Information Processing Systems*, volume 33, pp. 19620–19631, 2020.

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. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.168.

Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2 edition, 2009.

Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of Causal Inference: Foundations and Learning Algorithms*. MIT Press, 2017.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2024.

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. Association for Computational Linguistics, 2015. doi: 10.18653/v1/D15-1202.

Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Toward causal representation learning. *Proceedings of the IEEE*, 109(5):612–634, 2021.

Patrick Schwab and Walter Karlen. Cxplain: Causal explanations for model interpretation under uncertainty. *Advances in Neural Information Processing Systems*, 32, 2019.

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 *International Conference on Learning Representations*, 2023.

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*, pp. 24013–24035. Association for Computational Linguistics, 2025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022.

Norbert Wiener. The theory of prediction. *Modern Mathematics for Engineers*, pp. 165–190, 1956.

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 conversation. In *First Conference on Language Modeling*, 2024.

Yao Xu, Shizhu He, Jiabei Chen, Zihao Wang, Yangqiu Song, Hanghang Tong, Guang Liu, Jun Zhao, and Kang Liu. Generate-on-graph: Treat LLM as both agent and KG for incomplete knowledge graph question answering. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pp. 18410–18430. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.EMNLP-MAIN.1023. URL `https://doi.org/10.18653/v1/2024.emnlp-main.1023`.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

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 *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*. PMLR, 2025b.

Kechi Zhang, Jia Li, Ge Li, Xianjie Shi, and Zhi Jin. Codeagent: Enhancing code generation with tool-integrated agent systems for real-world repo-level coding challenges. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13643–13658, 2024.

Zhen Zhang, Wanjing Zhou, Juncheng Li, Hao Fei, Jun Wen, and Wei Ji. Radar: Redundancy-aware diffusion for multi-agent communication structure generation. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*. PMLR, 2026.

A Implementation Details

A.1 Explainer Input and Output

The amortized explainer learns the budget-conditioned mapping

$$F_\theta : (x, G, b) \mapsto \widehat{H}_b, \qquad \widehat{H}_b \subseteq G, \tag{15}$$

where x is the task query, $G = (V, E)$ is a candidate communication graph, and $b = (\rho_E, \rho_V)$ specifies the edge- and node-pruning ratios. All inputs are available before executing the predicted subgraph; the explainer does not require ground-truth answers, repeated edge interventions, or internal parameters of the topology generator.

For each edge $e = (u, v) \in E$ and node $v \in V$, the network predicts an edge-retention probability $\widehat{A}_{b,e} \in [0, 1]$ and a node-retention probability $\widehat{R}_{b,v} \in [0, 1]$. These probabilities are intermediate outputs rather than the final explanation. They are converted into budget-compatible candidate sets and passed through the validity projection to obtain the executable subgraph $\widehat{H}_b$.

The explainer therefore learns a budget-conditioned graph-to-subgraph mapping. The same candidate graph can yield different explanations under different pruning budgets, while the supervision is defined by the edge and node membership of the corresponding compact target subgraph $H_b^\star$.

A.2 Budget-Conditioned Explainer Architecture

Input representations. Let $\mathbf{q}_x$ denote a fixed embedding of query x, and let $\mathbf{r}_v$ denote a fixed embedding of the role or system prompt assigned to agent v. Each node is also associated with a structural feature vector $\mathbf{t}_v$, containing graph-local information available from the candidate topology, such as normalized in-degree and out-degree and topological position. A shared node encoder produces

$$\mathbf{h}_v = \phi_{\text{node}}\left([\mathbf{q}_x \| \mathbf{r}_v \| \mathbf{t}_v]\right), \qquad \mathbf{g}_G = \frac{1}{|V|} \sum_{v \in V} \mathbf{h}_v, \tag{16}$$

where $\|$ denotes concatenation and $\mathbf{g}_G$ is a permutation-invariant graph-level summary. The pruning ratios are encoded as

$$\mathbf{z}_b = \phi_b([\rho_E, \rho_V]). \tag{17}$$

Edge and node prediction heads. For an active edge $e = (u, v)$, let $\mathbf{s}_e$ collect edge-local structural features, including the edge type and normalized structural statistics of its endpoints. The shared edge head predicts

$$\widehat{A}_{b,e} = \sigma(\phi_{\text{edge}}([\mathbf{h}_u \| \mathbf{h}_v \| \mathbf{g}_G \| \mathbf{z}_b \| \mathbf{s}_e])). \tag{18}$$

The node head predicts

$$\widehat{R}_{b,v} = \sigma(\phi_{\text{retain}}([\mathbf{h}_v \| \mathbf{g}_G \| \mathbf{z}_b])). \tag{19}$$

All node and edge instances share the same encoders and prediction heads. Consequently, the number of trainable parameters does not depend on the numbers of agents or communication links, which allows the explainer to process graphs of different sizes.

The modules ϕ_{node}, ϕ_b, ϕ_{edge}, and ϕ_{retain} are implemented as lightweight multilayer perceptrons with parameters shared across graph elements. Fixed task and role representations are computed once and reused during explainer training and inference.

A.3 Graph-to-Subgraph Supervision and Decoding

For each calibration pair (x_i, G_i), causal subgraph construction produces one target $H^\star_{i,b}$ for every budget $b \in \mathcal{B}$. The resulting training set is

$$\mathcal{D}_{\text{expl}} = \{(x_i, G_i, b, H^\star_{i,b}) \mid b \in \mathcal{B}\}. \tag{20}$$

Each target subgraph defines binary edge- and node-retention labels

$$A^\star_{i,b,e} = \mathbb{I}[e \in E(H^\star_{i,b})], \qquad R^\star_{i,b,v} = \mathbb{I}[v \in V(H^\star_{i,b})]. \tag{21}$$

The explainer is optimized with the edge and node binary-cross-entropy losses defined in the main paper. Because one candidate graph contributes one target for each budget, the model learns both which communication elements should be retained and how the selected subgraph changes with the requested pruning ratios.

At inference time, the predicted probabilities are converted into candidate sets according to the graph-specific retention counts:

$$\widehat{E}_b = \mathrm{TopK}(\widehat{\mathbf{A}}_b, k_E(b)), \qquad \widehat{V}_b = \mathrm{TopK}(\widehat{\mathbf{R}}_b, k_V(b)). \tag{22}$$

The final explanation is

$$\widehat{H}_b = \mathcal{R}_{\text{valid}}(G, \widehat{E}_b, \widehat{V}_b), \tag{23}$$

where $\mathcal{R}_{\text{valid}}$ enforces the requested node budget together with a minimum-node constraint, removes every edge incident to an excluded node, and retains only feasible directed links between selected agents. The projection ranks feasible elements using the predicted retention scores and selects only nodes and edges already present in G, so it never introduces a new communication link.

A.4 Causal Supervision Construction Details

Repeated execution and task score. For every active edge e, we independently execute both the original graph G and the intervened graph G^{-e} for $M = 5$ runs. The task score is the mean outcome over these executions:

$$Q(x, G) = \frac{1}{M} \sum_{m=1}^{M} Q^{(m)}(x, G). \tag{24}$$

The original graph is re-executed for each evaluated edge rather than sharing one response set across all interventions. All executions use a decoding temperature of 0.2. The original and intervened runs are sampled independently and do not use paired random seeds. The same response sets are used to compute both the mean task outcome and the auxiliary semantic signal.

Result-level semantic signal. Semantic entropy is computed from the final system responses, rather than from the local outputs of the two agents incident to an edge. This choice aligns the auxiliary signal with the system-level quantity whose causal contribution is being estimated. The M final responses are grouped into semantic equivalence classes using the same equivalence procedure for the original and intervened graphs, after which the empirical class entropy is computed as defined in the main paper.

Score normalization. The mean task outcome and its non-negative intervention difference already lie in $[0, 1]$. We therefore retain the task contribution on its original scale and normalize the semantic contribution by the maximum empirical entropy of M samples:

$$\widetilde{u}^{\text{task}}_e = u^{\text{task}}_e, \qquad \widetilde{u}^{\text{sem}}_e = \min\left(1, \frac{u^{\text{sem}}_e}{\log M}\right). \tag{25}$$

The combined preservation utility is then computed with $(\alpha, \beta) = (0.8, 0.2)$.

Pruning counts and validity projection. The budget $b = (\rho_E, \rho_V)$ specifies removal ratios. For a candidate graph $G = (V, E)$, the numbers of removed elements are

$$r_E(b) = \lceil \rho_E |E| \rceil, \qquad r_V(b) = \lceil \rho_V |V| \rceil, \tag{26}$$

with retention counts $k_E(b) = |E| - r_E(b)$ and $k_V(b) = |V| - r_V(b)$. Thus, E25 denotes removing 25% of the active edges and N20 denotes removing 20% of the active nodes. After budgeted selection, the validity projection applies the selected node mask, removes incident edges, and retains the highest-ranked feasible links subject to the edge budget and the minimum-node constraint.

Scope of the causal attribution. The intervention procedure estimates a first-order conditional contribution for each edge in the context of the remaining full topology. Following the Granger principle, it measures whether excluding one information channel reduces the mean outcome or the stability of the final response while all other channels remain available. We do not enumerate multi-edge coalitions, whose cost grows combinatorially with graph size. Instead, these conditional edge contributions provide a tractable signal for constructing graph-level supervision, while the end-to-end evaluation of the predicted subgraph measures the aggregate outcome after multiple communication elements are removed.

A.5 Implementation Settings

Table 5 summarizes the main settings used for causal supervision construction and explainer training. We train one dataset-specific explainer for each benchmark. The six explainers use the same architecture, optimization configuration, and budget set, but each is trained from the 40 calibration queries of its corresponding dataset. The same frozen Qwen3-8B backbone is used for all methods.

Table 5: Main implementation settings. Dataset-specific explainers share the same architecture and optimization configuration.

Item	**Setting**
Semantic samples M	5
Task/semantic weights (α, β)	$(0.8, 0.2)$
Calibration queries per dataset	40
Training budgets $\mathcal{B}$	E25+N0, E50+N0, E75+N0, E25+N20, E50+N20, E25+N40
Shared hidden dimension	128
Dropout	0.15
Optimizer	AdamW
Learning rate	1×10^{-3}
Weight decay	1×10^{-4}
Batch size / epochs	64 / 200
Backbone LLM	Qwen3-8B
Decoding temperature	0.2
Reported evaluation runs	3
Hardware	4× NVIDIA GeForce RTX 4090 GPUs
Inference framework	PyTorch and vLLM
Token accounting	Online prompt + completion tokens

A.6 Dataset and Evaluation Details

We evaluate E2-Explainer on six reasoning and code-generation benchmarks using the data scale and evaluation protocol adopted by G-Designer and AgentPrune. For datasets with a dedicated training split, 40 calibration queries are selected from that split and are not used as evaluation queries. For HumanEval, we follow the AgentPrune data split: 40 problems are used for calibration and the remaining 124 problems are used for evaluation. G-Designer, AgentPrune, OFA-MAS, ARG-Designer, and all variants equipped with E2-Explainer are evaluated on exactly the same query set for each benchmark.

Table 6: Dataset statistics and evaluation protocols. HumanEval uses 40 calibration problems and the remaining 124 problems for evaluation. All methods use identical evaluation queries.

Dataset	**Task**	**Calib.**	**Eval.**	**Metric**
MMLU	Knowledge MCQ	40	153	Accuracy
GSM8K	Math reasoning	40	1,319	Exact match
MultiArith	Math reasoning	40	600	Exact match
SVAMP	Math reasoning	40	1,000	Exact match
AQuA	Algebra MCQ	40	254	Accuracy
HumanEval	Code generation	40	124	Pass@1

MMLU. MMLU evaluates knowledge-intensive reasoning across subjects from STEM, the humanities, and the social sciences. Each query contains a question and four candidate answers. We evaluate on the 153-query validation subset used by G-Designer and AgentPrune and report multiple-choice accuracy.

GSM8K, MultiArith, and SVAMP. These benchmarks evaluate multi-step arithmetic reasoning over natural-language word problems. Their evaluation sets contain 1,319, 600, and 1,000 queries, respectively. Performance is measured by exact match after extracting and normalizing the final numerical answer.

AQuA. AQuA contains algebraic word problems with five candidate answers. We evaluate on 254 queries and report multiple-choice accuracy based on the final selected option.

HumanEval. HumanEval evaluates function-level Python code generation from a function signature and natural-language specification. Following AgentPrune, 40 problems are used for calibration and the remaining 124 problems are used for evaluation. Pass@1 is computed by executing the generated implementation against the associated unit tests.

Answer extraction and scoring. We follow the answer-processing conventions of the released G-Designer evaluation implementation, which are also used by AgentPrune for the overlapping benchmarks. For MMLU, the evaluator locates the final answer statement and extracts the option label from `A`–`D`; AQuA is processed analogously with labels `A`–`E`. For GSM8K, MultiArith, and SVAMP, the parser first checks for an explicit final-answer statement or a `\boxed{}` expression and otherwise uses the last numerical expression in the response. Commas, surrounding spaces, trailing punctuation, and superficial LaTeX formatting are removed before exact comparison with the normalized reference value. For HumanEval, Markdown code fences are removed and the generated Python function is executed against the provided unit tests; a query is counted as correct only when all tests pass within the execution timeout. The same extraction and scoring pipeline is used for every topology generator and its corresponding E2-Explainer variant.

A.7 Additional Evaluation Protocols

Controlled candidate-graph comparison. For each query, a topology generator and its E2-Explainer variant start from exactly the same candidate graph and use the same agent roles, prompts, decoding temperature, and evaluation procedure. The reported accuracy and token results are averaged over three independent runs.

Token accounting. The online communication cost includes the prompt and completion tokens of all retained agent calls and the final aggregation call during execution of the candidate or predicted subgraph. It excludes the offline edge interventions, semantic-entropy sampling, topology-optimizer training, explainer training, and the lightweight explainer forward pass.

Causal ablations. Both ablations reconstruct the target graph-to-subgraph supervision and retrain the explainer. "w/o Causal" uses only the semantic signal by setting $(\alpha, \beta) = (0, 1)$, whereas "w/o Sem. Entropy" uses only the task-level causal contribution by setting $(\alpha, \beta) = (1, 0)$.

Calibration-source analysis. The G-Designer-only and AgentPrune-only variants each use 40 candidate graph instances generated from the corresponding 40 calibration queries. In the mixed setting, the same 40 queries are processed by both topology generators, yielding 80 candidate graph instances in total.

Agent-scale generalization. For MMLU and HumanEval, the scale-generalization experiment trains the explainer on 40 five-agent calibration graphs and directly evaluates it on six-agent graphs generated separately under a six-role G-Designer configuration. The six-agent roles are reconstructed following the role-design logic of G-Designer. The E25+N20 budget and evaluation queries are identical to those used in the main comparison.

Meaning of model-agnostic. In this work, model-agnostic specifically denotes independence from the candidate topology generator. E2-Explainer does not access generator parameters or generator-specific optimization states; it operates on the candidate graph together with the task and agent-role information.

B Additional Experimental Results

B.1 Results across Different Pruning Budgets

Table 7: Performance of E2-Explainer on G-Designer candidate graphs under different pruning budgets. Accuracy changes are measured against the unpruned G-Designer graph. Token reductions are reported per dataset, and the final column gives macro-average accuracy and total token usage across all six datasets. E25+N20 is the default setting used in the main paper.

Budget	Metric	MMLU	GSM8K	MultiArith	SVAMP	AQuA	HumanEval	Avg./Total
Original	Acc.	78.65	93.05	98.13	95.00	84.52	88.40	89.63
	Tokens	985,486	1,549,496	3,068,504	11,377,630	1,340,006	725,946	19,047,068
E25+N0	Acc.	78.43↓0.22	93.39↑0.34	98.00↓0.13	94.60↓0.40	87.10↑2.58	88.75↑0.35	90.05↑0.42
	Tokens	985,289↓0.02%	1,472,951↓4.94%	2,919,068↓4.87%	11,325,293↓0.46%	1,315,752↓1.81%	696,908↓4.00%	18,715,261↓1.74%
E50+N0	Acc.	79.08↑0.43	91.95↓1.10	98.17↑0.04	94.30↓0.70	84.52±0.00	90.62↑2.22	89.77↑0.15
	Tokens	972,970↓1.27%	1,496,193↓3.44%	2,906,180↓5.29%	11,300,262↓0.68%	1,305,434↓2.58%	642,172↓11.54%	18,623,211↓2.23%
E75+N0	Acc.	77.78↓0.87	92.71↓0.34	97.83↓0.30	93.80↓1.20	89.29↑4.77	89.38↑0.98	90.13↑0.51
	Tokens	960,849↓2.50%	1,428,945↓7.78%	2,889,917↓5.82%	10,812,162↓4.97%	1,232,270↓8.04%	718,396↓1.04%	18,042,539↓5.27%
E25+N20	Acc.	79.74↑1.09	93.77↑0.72	98.17↑0.04	94.60↓0.40	87.30↑2.78	90.69↑2.29	90.71↑1.09
	Tokens	675,452↓31.46%	867,098↓44.04%	2,208,095↓28.04%	9,091,864↓20.09%	837,906↓37.47%	585,838↓19.30%	14,266,253↓25.10%
E50+N20	Acc.	76.47↓2.18	92.86↓0.19	97.83↓0.30	94.30↓0.70	85.71↑1.19	86.88↓1.52	89.01↓0.62
	Tokens	774,789↓21.38%	1,103,706↓28.77%	2,174,956↓29.12%	9,004,256↓20.86%	889,764↓33.60%	653,860↓9.93%	14,601,331↓23.34%
E25+N40	Acc.	79.64↑0.99	92.55↓0.50	98.00↓0.13	94.10↓0.90	86.11↑1.59	81.88↓6.52	88.71↓0.91
	Tokens	643,030↓34.75%	948,911↓38.76%	1,714,373↓44.13%	6,562,617↓42.32%	575,265↓57.07%	483,916↓33.34%	10,928,112↓42.63%

Table 7 provides a more complete view of the accuracy–efficiency trade-off controlled by the edge- and node-pruning budgets. Edge-only pruning largely preserves overall task performance, with the three settings changing average accuracy by only $+0.42$, $+0.15$, and $+0.51$ percentage points. However, even E75+N0 reduces weighted token usage by only 5.27%. This shows that structural sparsity alone does not translate into proportional execution savings when all agents remain active: each retained agent still performs its own reasoning, and the lengths of the remaining messages can change after the topology is modified. At the same time, the gains on AQuA and HumanEval indicate that removing low-contribution channels can suppress redundant or misleading information rather than merely reducing cost.

Introducing node pruning produces substantially larger token reductions, but the benefit depends strongly on the budget. E25+N20 provides the most favorable overall trade-off, improving average accuracy from 89.63 to 90.71 while reducing weighted token usage by 25.10%. It preserves or improves performance on five of the six benchmarks, including gains of 1.09, 2.78, and 2.29 percentage points on MMLU, AQuA, and HumanEval, respectively. Increasing the node-pruning ratio to 40% further reduces token usage by 42.63%, but decreases average accuracy to 88.71, largely because HumanEval drops by 6.52 percentage points. This result suggests that aggressive node removal is particularly risky for code generation, where different roles may provide complementary implementation and verification signals. E50+N20 also removes more edges than E25+N20 but achieves both lower accuracy and a smaller token reduction. The realized cost is therefore jointly determined by node selection, edge selection, structural projection, and the lengths of the remaining messages, rather than by the nominal sparsity ratio alone.

We perform the complete budget sweep only on G-Designer candidate graphs because its purpose is to characterize budget sensitivity and select one common operating point, rather than tune a different ratio for every topology generator. After selecting E25+N20, we keep the budget fixed when transferring E2-Explainer to AgentPrune, OFA-MAS, and ARG-Designer. This isolates cross-generator generalization from generator-specific budget tuning and evaluates all topology optimizers under the same compression requirement. Selecting a separate budget for each generator on the evaluation set would introduce method-specific tuning advantages and make the comparison less controlled.

Table 8: Overview of the four qualitative cases. Online tokens include the prompt and completion tokens of the agent calls and final aggregation call for the corresponding query. Per-query totals are obtained from the sequential usage records within each evaluation scope.

Outcome transition	Dataset / index	Original	E25N20	Online tokens	Token change
Correct → Correct	MultiArith / 419	49	49	6,383 → 4,567	-28.5%
Wrong → Wrong	MMLU / 62	A	A	6,314 → 5,550	-12.1%
Correct → Wrong	MMLU / 151	B	C	11,811 → 10,216	-13.5%
Wrong → Correct	SVAMP / 415	47	3	6,059 → 4,965	-18.1%

Table 9: Execution trace for the correct-to-correct MultiArith case.

Node	Observed input in the realized trace	Logged output summary
Original G-Designer execution		
`6bRH` (Math Solver)	Query only	Computes 7 remaining rooms and returns $7 \times 7 = 49$.
`4tCg` (Mathematical Analyst)	Query only	Computes total time $12 \times 7 = 84$, elapsed time $5 \times 7 = 35$, and returns $84 - 35 = 49$.
`3LK5` (Programming Expert)	Query and the output of `4tCg`	Generates a program that subtracts the painted rooms and returns 49.
`3Ssf` (Inspector)	Query and the outputs of `4tCg` and `3LK5`	Verifies the remaining-room calculation and returns 49.
FinalRefer	Final outputs of all four agents	Observes four consistent solutions and returns 49.
E25N20 execution		
`6bRH` (Math Solver)	Query only	Returns $7 \times 7 = 49$.
`3LK5` (Programming Expert)	Query only	Generates an equivalent program and returns 49.
`3Ssf` (Inspector)	Query and the output of `3LK5`	Checks the calculation and returns 49.
FinalRefer	Final outputs of the three retained agents	Receives three consistent answers and returns 49.

B.2 Qualitative Case Studies and Execution Traces

We examine four E25N20 examples covering all possible transitions between correct and incorrect predictions. Table 8 summarizes their outcomes and online token usage. In every example, the original candidate graph contains six active spatial masks, whereas E25N20 retains two active spatial masks after removing one agent. The executable acyclic trace contains three same-round communication edges before refinement and one afterward. Active masks describe the selected candidate topology, while the trace records the directed messages that are actually executed after acyclic projection.

The original and E25N20 outputs are independent executions with decoding temperature 0.2. Accordingly, the cases illustrate observed changes in information propagation and aggregation under different topologies, rather than paired single-sample proofs that one removed element alone causes the change in correctness.

Case 1: Correct → Correct with lower token usage. The MultiArith query asks how much longer a painter needs to finish 12 rooms when each room requires 7 hours and 5 rooms have already been painted. The reference answer is $(12 - 5) \times 7 = 49$. The original trace is `4tCg`→`3LK5`, `4tCg`→`3Ssf`, and `3LK5` → `3Ssf`. All four agents return 49. E25N20 removes `4tCg` (Mathematical Analyst), leaving only `3LK5`→`3Ssf`; the three retained agents and the final node still return 49.

The removed Mathematical Analyst is correct, but its derivation duplicates information already available from the other roles and is forwarded to two downstream agents. Removing this node shortens both the number of calls and the prompts received downstream. Total online usage falls from 6,383 to 4,567 tokens, a reduction of 28.5%, while the answer remains correct.

Case 2: Wrong → Wrong under a shared domain misconception. The MMLU query asks which isolated pancreatic-enzyme loss would have the most extensive effect on nutrient absorption in cystic fibrosis. The reference option is C, trypsinogen. Both executions instead select A, lipase. In the original trace, `4MEK`→`4BNb`, `4MEK`→`4Cwg`, and `4BNb`→`4Cwg` are executed. E25N20 removes `4MEK` (Knowledgeable Expert) and retains only `4BNb`→`4Cwg`.

This failure is not primarily caused by one removable noisy message. The active reasoning roles share the same substantive misconception and rank the direct consequences of lipase loss above

Table 10: Execution trace for the wrong-to-wrong MMLU case.

Node	Observed input in the realized trace	Logged output summary
Original G-Designer execution		
4MEK (Knowledgeable Expert)	Query only	Produces search terms covering cystic fibrosis, pancreatic function, lipase, trypsinogen, and nutrient absorption, without selecting an option.
6wZy (Critic)	Query only	Notes that trypsinogen activates proteolytic enzymes, but its response ends without a final option.
4BNb (Mathematician)	Query and the output of 4MEK	Prioritizes fat and fat-soluble-vitamin malabsorption and selects A, lipase.
3gkY (Psychologist)	Query only	Uses the same fat-malabsorption argument and selects A.
4Cwg (Historian)	Query and the outputs of 4MEK and 4BNb	Adopts 4BNb's comparison and selects A.
FinalRefer	Final outputs of all five agents	Returns A, which disagrees with the reference option C.
E25N20 execution		
6wZy (Critic)	Query only	Again discusses the alternatives but does not produce a complete final option.
4BNb (Mathematician)	Query only	Again argues that lipase has the broadest nutritional consequences and selects A.
3gkY (Psychologist)	Query only	Selects A using the same fat-digestion rationale.
4Cwg (Historian)	Query and the output of 4BNb	Repeats the lipase argument and selects A.
FinalRefer	Final outputs of the four retained agents	Returns A again.

Table 11: Execution trace for the correct-to-wrong MMLU case.

Node	Observed input in the realized trace	Logged output summary
Original G-Designer execution		
4MEK (Knowledgeable Expert)	Query only	Supplies lexical cues linking "paddy wagon" with police officer, while also listing the competing options.
6wZy (Critic)	Query only	Treats the wording as regionally ambiguous but identifies B as the most likely answer.
4BNb (Mathematician)	Query and the output of 4MEK	Compares the police-van and agricultural readings and selects the common police-van meaning, B.
3gkY (Psychologist)	Query only	Independently interprets the term as a police transport vehicle and selects B.
4Cwg (Historian)	Query and the outputs of 4MEK and 4BNb	Follows the common slang interpretation and selects B.
FinalRefer	Final outputs of all five agents	Returns B, matching the reference answer.
E25N20 execution		
6wZy (Critic)	Query only	Still identifies B as the best answer, although with reservations about ambiguity.
4BNb (Mathematician)	Query only	Reinterprets the term literally as a wagon for transporting unmilled rice and selects C.
3gkY (Psychologist)	Query only	Independently retains the police-van interpretation and selects B.
4Cwg (Historian)	Query and the output of 4BNb	Adopts the agricultural interpretation from 4BNb and selects C.
FinalRefer	Final outputs of the four retained agents	Chooses C despite two independent agents selecting B.

the broader downstream role of trypsinogen. Pruning reduces token usage from 6,314 to 5,550, but the retained subgraph continues to support the same incorrect answer. This case exposes a limit of topology refinement: removing redundant communication cannot repair an error already shared by the retained agents.

Case 3: Correct → Wrong after removing a useful knowledge source. The MMLU query asks which worker uses a "paddy wagon," with reference option B, police officer. The original execution returns B. Its realized trace is 4MEK → 4BNb, 4MEK → 4Cwg, and 4BNb → 4Cwg. E25N20 removes 4MEK (Knowledgeable Expert), retains only 4BNb → 4Cwg, and returns C, rice farmer.

The removed node provides a useful lexical cue in the original run. After its removal, the only retained communication path propagates an incorrect literal interpretation from 4BNb to 4Cwg. The execution uses 13.5% fewer tokens, but the connected pair reinforces option C and the final node rejects two independent correct responses. This example shows that a lower-cost topology can lose complementary knowledge and preserve a misleading information path.

Case 4: Wrong → Correct after topology refinement. Figure 3 visualizes the SVAMP example in which Dave initially has 59 files and 15 apps, and retains 30 files and 12 apps after deletion. The

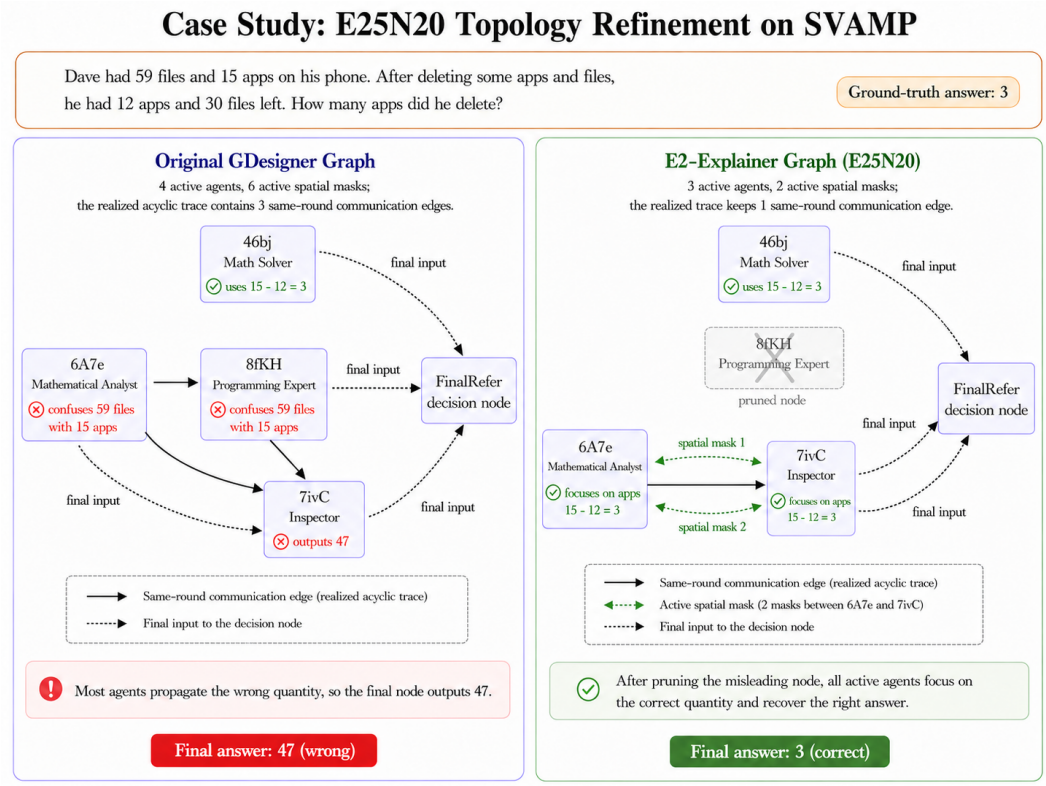


Figure 3: Wrong-to-correct case on SVAMP index 415. The original graph contains four active agents and six active spatial masks, and its realized acyclic trace contains three same-round communication edges. E25N20 removes the Programming Expert, retains three agents and two active spatial masks, and realizes one same-round communication edge.

Table 12: Execution trace for the wrong-to-correct SVAMP case in Figure 3.

Node	Observed input in the realized trace	Logged output summary
Original G-Designer execution		
`46bj` (Math Solver)	Query only	Correctly uses the app counts and returns $15 - 12 = 3$.
`6A7e` (Mathematical Analyst)	Query only	Misbinds 59 files as the original number of apps and returns $59 - 12 = 47$.
`8fKH` (Programming Expert)	Query and the output of `6A7e`	Sets `original_apps=59` in its generated program and returns 47.
`7ivC` (Inspector)	Query and the outputs of `6A7e` and `8fKH`	Accepts the same variable assignment and returns 47.
FinalRefer	Final outputs of all four agents	Selects 47 from three mutually consistent but incorrect responses and rejects the isolated answer 3.
E25N20 execution		
`46bj` (Math Solver)	Query only	Returns $15 - 12 = 3$.
`6A7e` (Mathematical Analyst)	Query only	Correctly binds 59 and 30 to files and 15 and 12 to apps, then returns 3.
`7ivC` (Inspector)	Query and the output of `6A7e`	Verifies the app-based subtraction and returns 3.
FinalRefer	Final outputs of the three retained agents	Receives three consistent answers of 3 and returns 3.

requested quantity is the number of deleted apps, so the reference answer is $15 - 12 = 3$. The original G-Designer execution returns 47, whereas E25N20 returns 3.

The original realized trace is `6A7e` $\rightarrow$ `8fKH`, `6A7e` $\rightarrow$ `7ivC`, and `8fKH` $\rightarrow$ `7ivC`. E25N20 removes `8fKH` (Programming Expert), after which only `6A7e` $\rightarrow$ `7ivC` is executed.

In the original run, the incorrect variable binding is repeated along the connected path and outweighs the only correct response at aggregation. In the independent E25N20 run, all retained agents

distinguish files from apps and agree on 3. Token usage also falls from 6,059 to 4,965. The case illustrates how a compact topology can coincide with both lower cost and a more reliable aggregation outcome, while the independent sampling caveat prevents attributing the correction solely to removal of `8fKH`.

B.3 Generalization to Hand-Crafted Communication Topologies

We further examine whether the frozen explainer trained only on G-Designer-generated graphs can transfer to manually specified communication structures. We consider five representative topology families, including complete, random, layered, chain, and star graphs. E2-Explainer is directly applied to these graphs without additional causal supervision or retraining.

Table 13: Generalization of E2-Explainer to manually specified communication topologies. The explainer is trained only on G-Designer-generated graphs and applied without retraining. Accuracy arrows indicate changes relative to the corresponding unrefined topology.

Method	Metric	MMLU	GSM8K	MultiArith	SVAMP	AQuA	HumanEval
Complete	Acc.	77.39	90.71	97.50	94.31	83.90	87.09
Complete + E2-Explainer	Acc.	78.43$_{\uparrow 1.04}$	91.35$_{\uparrow 0.64}$	97.67$_{\uparrow 0.17}$	94.50$_{\uparrow 0.19}$	84.64$_{\uparrow 0.74}$	88.70$_{\uparrow 1.61}$
	Token Δ	-17.4%	-36.5%	-30.9%	-22.4%	-28.9%	-25.1%
Random	Acc.	76.78	91.01	97.17	93.97	84.70	86.88
Random + E2-Explainer	Acc.	77.45$_{\uparrow 0.67}$	90.97$_{\downarrow 0.04}$	97.33$_{\uparrow 0.16}$	94.09$_{\uparrow 0.12}$	85.03$_{\uparrow 0.33}$	87.09$_{\uparrow 0.21}$
	Token Δ	-18.7%	-21.2%	-19.7%	-15.6%	-17.6%	-14.3%
Layer	Acc.	77.08	91.24	96.83	94.40	85.49	88.38
Layer + E2-Explainer	Acc.	78.10$_{\uparrow 1.02}$	91.28$_{\uparrow 0.04}$	97.41$_{\uparrow 0.58}$	94.91$_{\uparrow 0.51}$	85.43$_{\downarrow 0.06}$	89.11$_{\uparrow 0.73}$
	Token Δ	-23.6%	-30.1%	-27.4%	-22.8%	-33.9%	-19.9%
Chain	Acc.	78.05	91.09	96.67	94.62	83.51	86.29
Chain + E2-Explainer	Acc.	78.37$_{\uparrow 0.32}$	90.67$_{\downarrow 0.42}$	96.67$_{\pm 0.00}$	94.95$_{\uparrow 0.33}$	83.46$_{\downarrow 0.05}$	85.88$_{\downarrow 0.41}$
	Token Δ	-15.2%	-25.0%	-24.1%	-14.3%	-15.4%	-7.93%
Star	Acc.	77.08	90.78	97.67	93.85	83.90	83.25
Star + E2-Explainer	Acc.	78.03$_{\uparrow 0.95}$	91.43$_{\uparrow 0.65}$	97.83$_{\uparrow 0.16}$	94.17$_{\uparrow 0.32}$	85.82$_{\uparrow 1.92}$	86.13$_{\uparrow 2.88}$
	Token Δ	-29.6%	-35.4%	-22.8%	-17.9%	-32.5%	-20.1%

Table 13 shows that E2-Explainer transfers effectively beyond the generated graph distribution used for training. It improves or matches accuracy in 26 of the 30 dataset–topology combinations while reducing token usage across every topology and benchmark. The strongest gains appear on the star topology, including improvements of 1.92 points on AQuA and 2.88 points on HumanEval. The layered topology also benefits consistently, reaching 89.11 on HumanEval while reducing token usage by 19.9%. Even for the chain topology, whose sparse structure leaves less removable redundancy, the explainer largely preserves task performance and still reduces token usage on all six benchmarks.

These results are notable because the explainer is calibrated exclusively on G-Designer graphs but is directly applied to fixed topologies with substantially different structural patterns. Its improvements therefore cannot be attributed solely to memorizing G-Designer-specific adjacency structures. Instead, the explainer captures transferable task- and structure-dependent cues for retaining useful communication elements.

B.4 Additional Redundancy Analysis on OFA-MAS

To examine whether the redundancy observed on G-Designer is specific to a particular topology designer, we repeat the same random edge-masking analysis on OFA-MAS, as shown in Figure 4. Similar to the observations on G-Designer, most random edge masks degrade performance on both MMLU and HumanEval, while only a small fraction of them preserve or slightly improve the original accuracy. This pattern suggests that OFA-MAS-generated topologies also contain redundant communication links and that task-preserving compact subgraphs do exist.

At the same time, the rarity of these successful random masks shows that such subgraphs are difficult to identify through unguided pruning alone. In other words, redundancy does not imply that arbitrary pruning is safe. This further motivates the need for a task-conditioned explainer that selects compact communication subgraphs according to causal contribution rather than random deletion.

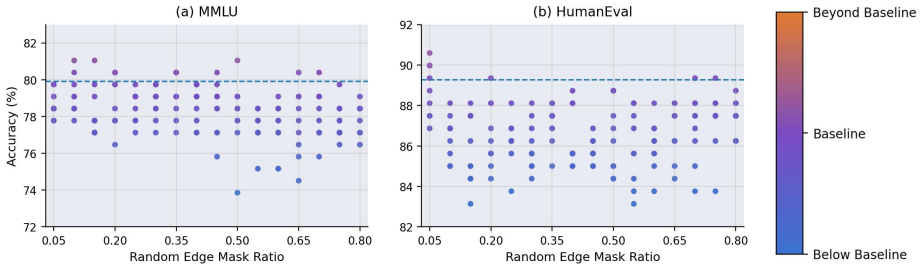


Figure 4: Random edge masking on OFA-MAS-generated communication graphs for MMLU and HumanEval. The edge masking ratio ranges from 0.05 to 0.80 in increments of 0.05. Each point denotes the accuracy obtained under one retained random mask, while the dashed line marks the performance of the original unmasked OFA-MAS graph. For visualization, one extreme run is omitted at each masking ratio. The resulting distribution shows that although a few random masks preserve or even improve the original performance, most random masks degrade accuracy, indicating that task-preserving compact subgraphs exist but are difficult to identify through unguided pruning.

C Algorithmic Details

C.1 Causal Supervision Construction and Explainer Prediction

Algorithms 1 and 2 summarize the offline construction of graph-to-subgraph supervision and the training and test-time prediction procedures, respectively.

Algorithm 1 Causally Grounded Supervision Construction

Require: $\mathcal{D}_{\text{cal}}, \mathcal{G}_\phi, \mathcal{B}, M, \alpha, \beta$
Ensure: Graph-to-subgraph dataset $\mathcal{D}_{\text{expl}}$

$\mathcal{D}_{\text{expl}} \leftarrow \emptyset$
for each $x \in \mathcal{D}_{\text{cal}}$ **do**
  $G = (V, E) \leftarrow \mathcal{G}_\phi(x)$
  for each active edge $e \in E$ **do**
    $(Q_G^e, \mathcal{Y}^{G,e}) \leftarrow \textsc{Evaluate}(x, G, M)$
    $G^{-e} \leftarrow \textsc{Intervene}(G, e)$
    $(Q_e, \mathcal{Y}^{G^{-e}}) \leftarrow \textsc{Evaluate}(x, G^{-e}, M)$
    $u_e^{\text{task}} \leftarrow \max(0, Q_G^e - Q_e)$
    $u_e^{\text{sem}} \leftarrow \max\Big(0, \bar{H}(\mathcal{Y}^{G^{-e}}) - \bar{H}(\mathcal{Y}^{G,e})\Big)$
    $\widetilde{u}_e^{\text{task}} \leftarrow u_e^{\text{task}}$
    $\widetilde{u}_e^{\text{sem}} \leftarrow \min(1, u_e^{\text{sem}} / \log M)$
    $u_e \leftarrow \text{clip}_{[0,1]}(\alpha \widetilde{u}_e^{\text{task}} + \beta \widetilde{u}_e^{\text{sem}})$
  end for
  $\pi_G \leftarrow \textsc{SortDescending}(E, \{u_e\}_{e \in E})$
  for each budget $b \in \mathcal{B}$ **do**
    $(k_E, k_V) \leftarrow \textsc{RetentionCounts}(G, b)$
    $E_b^\star \leftarrow \textsc{TopK}(\pi_G, k_E)$
    $H_b^\star \leftarrow \mathcal{R}_{\text{valid}}(G, E_b^\star, k_V)$
    $\mathcal{D}_{\text{expl}} \leftarrow \mathcal{D}_{\text{expl}} \cup \{(x, G, b, H_b^\star)\}$
  end for
end for
return $\mathcal{D}_{\text{expl}}$

Algorithm 2 Explainer Training and Test-Time Prediction

Require: $\mathcal{D}_{\text{expl}}, \lambda_{\text{node}}, \lambda_{\text{wd}}$
Ensure: Trained explainer F_θ
Initialize parameters θ
Training
for each minibatch $\mathcal{S} \subseteq \mathcal{D}_{\text{expl}}$ **do**
  for each $(x_i, G_i, b, H^\star_{i,b}) \in \mathcal{S}$ **do**
    $(\mathbf{A}^\star_{i,b}, \mathbf{R}^\star_{i,b}) \leftarrow \textsc{Indicators}(G_i, H^\star_{i,b})$
    $(\widehat{\mathbf{A}}_{i,b}, \widehat{\mathbf{R}}_{i,b}) \leftarrow F_\theta(x_i, G_i, b)$
  end for
  Compute $\mathcal{L}_{\text{edge}}$ and $\mathcal{L}_{\text{node}}$ on $\mathcal{S}$
  $\mathcal{L} \leftarrow \mathcal{L}_{\text{edge}} + \lambda_{\text{node}}\mathcal{L}_{\text{node}} + \lambda_{\text{wd}}\|\theta\|_2^2$
  Update θ by minimizing $\mathcal{L}$
end for
Test-time prediction
Receive query x, graph G, and budget b
$(\widehat{\mathbf{A}}_b, \widehat{\mathbf{R}}_b) \leftarrow F_\theta(x, G, b)$
$(k_E, k_V) \leftarrow \textsc{RetentionCounts}(G, b)$
$\widehat{E}_b \leftarrow \textsc{TopK}(\widehat{\mathbf{A}}_b, k_E)$
$\widehat{V}_b \leftarrow \textsc{TopK}(\widehat{\mathbf{R}}_b, k_V)$
$\widehat{H}_b \leftarrow \mathcal{R}_{\text{valid}}(G, \widehat{E}_b, \widehat{V}_b)$
return $\widehat{H}_b$