TASK-ORIENTED LEARNING OF DYNAMIC COMMUNICATION GRAPHS FOR MULTI-ROUND LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-round LLM-based multi-agent systems solve tasks through repeated inter-agent communication, making who communicates with whom at each round consequential to the final answer. Most existing methods either fix the communication graph at inference or sample edges from behavior-independent distributions, preventing the topology from responding to evolving agent behavior. We introduce TodyComm, a task-oriented policy that constructs constrained, directed acyclic communication graphs conditioned on the unfolding interactions. The resulting graph policy is optimized end-to-end using the final task reward. We evaluate TodyComm in a dynamically adversarial setting, a controlled instance of shifting graph distributions, where agents may silently become unreliable at unknown rounds. Across six benchmarks, TodyComm outperforms fixed-graph and behavior-independent graph-learning baselines while retaining generalizability.

1 INTRODUCTION

Large language model (LLM)-based multi-agent systems (MAS) have emerged as a powerful paradigm for collaborative problem solving (Li et al., 2023; Hong et al., 2024; Zhuge et al., 2024; Wu et al., 2024; Liang et al., 2024; Zou et al., 2025). In multi-round systems, agents repeatedly exchange messages, revise their reasoning, and aggregate their outputs into a final answer (Du et al., 2024; Liu et al., 2024). This iterative process can integrate complementary perspectives and correct intermediate errors, but its effectiveness depends critically on the communication topology: the graph specifying which agents may influence one another at each round (Zhang et al., 2025b;a; 2026; Wang et al., 2025b; Li et al., 2025). Learning an effective communication topology has therefore become a central problem in multi-round MAS.

Most existing topology-learning methods, however, do not condition their inference-time graph policy on how the current interaction unfolds. They either deploy a fixed graph or sample and prune edges using scores that do not update in response to agents' round-level outputs (Zhang et al., 2025b;a; 2026; Wang et al., 2025b; Wu et al., 2026). This is a poor fit for multi-round collaboration. As an episode progresses, an agent may become more reliable, repeat information already supplied by others, drift toward an incorrect line of reasoning, or cease to contribute useful information. The topology that maximizes task utility should change accordingly. A graph policy that is independent of the within-episode interaction history cannot make such adaptations by construction.

An orthogonal line of work studies robustness to unreliable agents. G-Safeguard and BlindGuard use graph-based anomaly detection to identify potentially adversarial behavior (Wang et al., 2025a; Miao et al., 2026), while ResMAS constructs resilient architectures for agent corruption (Zhou et al., 2026). These approaches provide important robustness mechanisms, but they either retain a fixed base topology or optimize detection and resilience objectives rather than learning communication structures directly from task utility; some additionally require adversarial supervision. This leaves open a more general question: can a multi-round MAS learn a task-oriented graph policy that adapts online to agents' unfolding behavior, without supervision of the desired graph or of agent label?

We argue that communication topology should be behavior-conditioned. Let $\mathcal{H}^{t-1}$ denote the interaction history available before round t. Instead of drawing a graph G^t from a behavior-independent distribution, the system should learn a conditional graph policy $p_\theta(G^t \mid \mathcal{H}^{t-1})$. When agents' behavior changes, the conditional policy changes, and hence the resulting topology changes with it. This formulation also distinguishes behavioral adaptation from merely resampling graphs using a fixed set of edge probabilities.

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.

Table 1: Comparing and contrasting our method with baseline methods. "$*$": Edge selection per round indicates that graph structures vary across rounds due to changes in the underlying graph distribution, rather than stochastic sampling from an *invariant* set of edge weights. "$**$": Edge selection per round in training" means the algorithm changes the edges during training, which in turn **actively impacts the training data**. "-" denotes the absence of adversarial setting in the experiment. "N.A." indicates that no optimization is involved.

	Edge selection **per round*** in **inference**	Edge selection **per round*** in **training****	Adversary label **not** required	Parametric edge selector based on agent behavior	Task-driven edge selection	Optimization mode
G-Designer	✗	✗	✓	✗	✓	RL + aux. loss
AgentPrune	✗	✗	✓	✗	✓	RL + aux. loss
AgentDropout	✗	✓	-	✗	✓	RL + aux. loss
DyTopo	✓	✗	-	✗	✗	N.A.
CARD	✗	✗	-	✗	✓	GD
TodyComm	✓	✓	✓	✓	✓	RL

We instantiate this principle in **TodyComm**, a **t**ask-**o**riented framework for learning **dy**namic **comm**unication topologies. TodyComm recurrently encodes each agent's outputs and neighborhood interactions into a behavior-dependent node representation. It maps these representations to node potentials, which parameterize pairwise edge scores through the potentials of the two endpoint nodes. A constrained graph constructor then converts the edge scores into a feasible communication graph, respecting workflow masks, acyclicity, and node-wise in-degree and out-degree budgets. Consequently, a change in an agent's behavior changes its representation, its incident edge scores, and ultimately the communication graph.

Learning this graph policy presents several coupled challenges. At every round, the action is combinatorial because edges are discrete; the resulting graph must be a directed acyclic graph so that messages can be executed in a well-defined order; degree constraints must limit communication cost; and the usefulness of an edge is observed only indirectly through a final reward produced by stochastic LLM outputs. TodyComm addresses these challenges by formulating multi-round collaboration as a Markov decision process (Puterman, 1990) and optimizing the graph policy end-to-end with reinforcement learning. Task utility provides the only supervision: TodyComm requires neither ground-truth communication graphs nor agent-level behavioral annotations.

To evaluate adaptation under a controlled and measurable change in the task-optimal graph distribution, we adopt a dynamically adversarial setting: a subset of agents may silently become unreliable at unknown rounds while retaining their assigned roles and producing plausible but misleading messages. At the transition round, the task-optimal graph distribution changes: probability mass should shift away from graphs that allow newly unreliable agents to influence others and toward graphs that preserve communication among reliable agents. This setting thus provides a direct test of whether a learned graph policy can use agent behaviors to track changes in the task-optimal graph distribution.

We conduct extensive empirical evaluations on six diverse benchmarks, in addition to communication budget (e.g., in-degree and out-degree), scalability and generalization, demonstrating that TodyComm consistently outperforms baseline methods and highlighting the effectiveness of learning dynamic communication topologies from agent behaviors in multi-round LLM-based MAS. To the best of our knowledge, TodyComm is the first framework to learn task-oriented, behavior-driven dynamic communication topologies in multi-round LLM-based multi-agent systems, enabling task-adaptive and round-adaptive communication during both training and inference.

2 Related Work on Graph Learning for MAS

In conventional multi-agent RL, methods such as TarMAC (Das et al., 2019), IC3Net (Singh et al., 2019), SchedNet (Kim et al., 2019), ToM2C (Wang et al., 2022), and MAGIC (Niu et al., 2021) learn communication patterns via attention, gating, or pairwise edge probabilities, enabling agents to modulate their interactions. LSC (Sheng et al., 2023) further introduces dynamic hierarchical grouping, but its topology remains restricted and not optimized as a general graph. Although these approaches can induce dynamic interaction patterns, they do not explicitly optimize communication topology under structural constraints such as bounded node degree. Moreover, many rely on numeric-valued messages to enable differentiation or weighted combinations, which is infeasible for language agents, especially closed-source LLMs.

In LLM-based MAS, early works typically rely on fixed or predefined interaction structures, where agents communicate under static coordination patterns (Jiang et al., 2023; Du et al., 2024; Qian et al.,

2025; Chan et al., 2024; Wu et al., 2024). Subsequent methods introduce explicit communication graphs, typically by assigning learnable edge weights over a fixed set of potential connections. For example, GPTSwarm (Zhuge et al., 2024) and Graph-GRPO (Cang et al., 2026) construct communication structures through behavior-*independent* parameters, focusing on a single round.

More generally, multi-round MAS often adopt a spatial–temporal graph formulation, where communication is defined over a fixed graph and refined via pruning or sampling. Representative examples include G-Designer (Zhang et al., 2025b), which constructs graphs from predefined connections, and pruning-based approaches such as AgentPrune (Zhang et al., 2025a), AgentDropout (Wang et al., 2025b), and SafeSieve (Zhang et al., 2026), which remove edges or agents during training. These methods rely on graph structures that *remain fixed at inference time*, and the observed dynamism arises primarily from stochastic sampling rather than behavior-driven structural adaptation. DyTopo (Lu et al., 2026) induces a dynamic graph at inference time; however, it remains a rule-based method based on semantic matching, without learning the communication structure from agent behavior or task-driven utility signals. CARD (Wu et al., 2026) generates the communication graph from predefined condition descriptors and does not update its graph *distribution* across collaboration rounds based on agents' observed outputs.

Several works also study robustness in LLM-based MAS. Methods such as G-Safeguard (Wang et al., 2025a) and BlindGuard (Miao et al., 2026) employ graph-based anomaly detection to identify adversarial agents, while ResMAS (Zhou et al., 2026) designs resilient architectures under agent corruption. But they focus on detection or robustness rather than *cooperation driven by task performance*, relying on *fixed* communication topologies.

In contrast, we consider a dynamically adversarial setting, **where agents may transition to adversarial behavior at unknown rounds during interaction**. The number of adversarial agents is controlled by an attack rate, enabling systematic evaluation of whether a system can adapt its communication topology online in response to evolving agent behaviors. Unlike existing approaches that fix communication structures at inference time or adapt them without regard to task performance, TodyComm produces behavior-driven, round-adaptive, and task-oriented communication graphs.

Table 1 compares TodyComm with representative state-of-the-art methods. An expanded comparison including additional methods is provided in Table 7 in Appendix A.2.

3 PRELIMINARIES

In this paper, we adopt multi-round LLM-based multi-agent framework (Zhang et al., 2025a;b), in which multiple LLM agents interact over multiple rounds to solve a given query.

Multi-round LLM-based MAS Given a query q, a set of N agents $\{A_1, \ldots, A_N\}$ engage in a multi-round interaction over T rounds. Each agent $A_i := (\text{base}_i, \text{role}_i)$ is instantiated with an underlying LLM model and an assigned role at initialization. At round $t = 1$, the agents independently generate an initial response $\{o_i^1\}_{i=1}^N$ to the query without inter-agent communication. For rounds $t \in [2, \ldots, T]$, agents iteratively update their responses by communicating and collaborating with other agents, producing updated outputs $o_i^t := \{\text{sol}_i^t, \text{ana}_i^t\}$, where sol_i^t is the answer such as a one-hot vector for multiple choice, and ana_i^t is the embedding of the agent's analysis. After T rounds, the final answer is obtained by aggregating the agents' outputs from the last round $\{o_i^T\}_{i=1}^N$. For convenience, we refer to the decision phase as round $T + 1$ in the following sections.

Multi-round MAS as Graphs We model multi-agent interactions over T rounds using spatial graphs and distinguish two types of structures. Inter-agent communication at round $t \in [2, \ldots, T]$ is modeled as a directed acyclic graph (DAG) $\mathcal{G}_C^t = (\mathcal{V}_C^t, \mathcal{E}_C^t)$, where $\mathcal{V}_C^t$ denotes the set of participating agent nodes at round t, with $|\mathcal{V}_C^t| \leq N$, and $\mathcal{E}_C^t$ is the set of directed edges specifying message-passing relationships among agents - messages serve as prompts for the recipient. The aggregation of agent responses into a final decision is modeled using a separate decision graph $\mathcal{G}_D = (\mathcal{V}_D, \mathcal{E}_D)$, where $\mathcal{V}_D$ denotes the set of agent nodes contributing to the final decision, with $|\mathcal{V}_D| \leq N$, and $\mathcal{E}_D$ consists of edges connecting selected agents to the decision node.

Multi-agent Communication Given a query q, information is propagated through agent nodes in a topological order, ensuring that each node is executed only after all its predecessors are resolved. Consequently, the inter-agent communication graph $\mathcal{G}_C^t$ is restricted to be a DAG. At round $t \in [2, \ldots, T]$, each node v_i receives the query q along with messages from its parent nodes $\mathcal{N}^t(v_i)$. Each parent $v_j \in \mathcal{N}^t(v_i)$ provides a message $m_j^t = f_m(o_j^t, \text{role}_j)$, where f_m is a prompt template

function and role$_j$ denotes the role assigned to agent j at initialization. Node v_i then computes its output o_i^t based on its prompt p_i^t, which is defined using another prompt template function f_v: $p_i^t \sim f_v\Big(q,\ \bigcup_{v_j \in \mathcal{N}^t(v_i)} m_j^t\Big)$. The final answer to query q, denoted by ans$_q$, is produced by a predefined decision function f_d that aggregates the outputs of selected agents connected to the decision node v_d: $\text{ans}_q \sim f_d\Big(\bigcup_{v_j \in \mathcal{N}^{T+1}(v_d)} o_j^T\Big)$. The complete execution procedure of the communication graph is summarized in Algorithm 1 in Appendix A.1.1.

4 METHOD

We formulate this multi-agent collaboration as an MDP, where each episode corresponds to solving a query through T interaction rounds. The **actions** correspond to the inter-agent communication graphs $\mathcal{G}_C^t$ for $t \in [2, \ldots, T]$, and the decision graph $\mathcal{G}_D$ at round $T+1$. The inter-agent directed communication graph $\mathcal{G}_C^t$ is represented by binary edge indicators $\mathcal{E}_C^t \in \{0,1\}^{\mathcal{V}_C^t \times \mathcal{V}_C^t}$. The decision graph $\mathcal{G}_D$ is represented by binary node indicators $\mathcal{E}_D \in \{0,1\}^{\mathcal{V}_D}$ with associated weights $\{w_i\}_{i=1}^{|\mathcal{V}_D|}$.

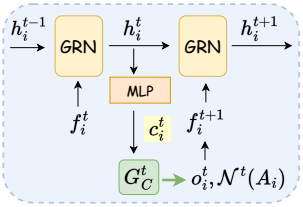

Figure 1: TodyComm diagram. $\{A_i\}_{i=1}^N$ denote the agents, and $\{o_i^t\}_{i=1}^N$ denote their outputs at round t. The neighborhood of agent A_i at round t is denoted by $\mathcal{N}^t(A_i)$ and is initialized as $\emptyset$ for $t=0$. It is used to construct the node features $\{f_i^t\}_{i=1}^N$, which are embedded into $\{h_i^t\}_{i=1}^N$ via a GRN to produce the node potentials $\{c_i^t\}_{i=1}^N$.

The **state** of the MDP is the response of all agents $\{o_i^t\}_i$, along with all the relevant information such as q and previous $\{o_i^\tau\}_i$ ($\tau < t$). The **transition** probability concerns the distribution of the next-round response $\{o_i^{t+1}\}_i$ under the given graph (i.e., action) at t. The **reward** is the final task utility, measured by the correctness of the answer produced at round $T+1$.

We denote the policy of constructing the communication graph at round t by π_t, and the policy of the decision graph by π_{T+1}. Their specific construction and parametrization will be detailed in Section 4.1. Our goal is to learn policy π_t that induces communication and decision structures maximizing the expected utility of query q through multiple interaction rounds. Given a query q, we denote its utility by $U_q(\tau)$, where τ is the trajectory of states and actions. Parameterizing the policy by θ, the reinforcement learning (RL) objective is then defined as: $\mathcal{J}(\theta) = \max_{\theta \in \Theta} \mathbb{E}_{q \sim \mathcal{Q}} \mathbb{E}_{\tau \sim \pi_{t,\theta}} \big[U_q(\tau)\big]$.

In our **dynamically adversarial setting**, agents may turn adversarial **at any round**, so the communication graph must continuously adapt to preserve cooperation among reliable agents. Since our policy is the communication graph constructed at each round, the key challenge is therefore edge selection: which directed edges to form at each round to form a DAG. Drawing on the standard graph-learning practice of predicting edges from node features (Kipf & Welling, 2016; Hoff et al., 2002), we learn per-round node representations that capture each agent's behavior and messages sent by neighboring agents. These node features then drive the sequential edge selection decisions, determining the communication edges among agents and the connections to the decision node.

4.1 INTER-AGENT EDGES OPTIMIZATION

Node Embeddings To better capture multi-round interactions and the temporal flow of information across rounds, we employ a many-to-one Gated Recurrent Network (GRN, Chung et al., 2014) to learn per-round node embeddings $\{h_i^t\}_{t=2}^{T+1}$. Since each agent produces outputs at every round, the GRN aggregates information across all T rounds into a unified representation that captures both individual behavior and relational context with neighboring agents. Following the standard formulation of GRNs, at each round $t \in [2, \ldots, T+1]$, we denote by h_i^{t-1} that summarizes information from previous rounds and by f_i^t the current node feature of agent i, which is constructed in the embedding space from the agent's own output o_i^{t-1} and the outputs of its neighboring agents $\mathcal{N}^{t-1}(v_i)$ at round $t-1$. Let $\{h_i^1\}_{i=1}^N = \emptyset$. The node embedding at round t is updated as $h_i^t = \text{GRN}(h_i^{t-1}, f_i^t)$. The architecture is shown in Figure 1 and 3, with pseudo-code given in Algorithm 2 in Appendix A.1.2.

Node Feature Construction At each round t, we construct the node features f_i^t for each agent i based on the agent's own output o_i^{t-1} and the outputs of its neighboring agents $\mathcal{N}^{t-1}(v_i)$ from the previous round. f_i^t serves as the input to the GRN. To capture both individual behavior and inter-agent interactions, we construct it by concatenating three complementary components: self information $\mathbf{s}_i^{t-1}$, neighborhood information $\mathbf{n}_i^{t-1}$, and difference information $\mathbf{d}_i^{t-1}$: $f_i^t := \big[\mathbf{s}_i^{t-1} \| \mathbf{n}_i^{t-1} \| \mathbf{d}_i^{t-1}\big]$. Detailed feature definitions are deferred to Appendix A.3.

Edge Construction based on Node Embeddings With node embeddings h_i^t at hand, we could apply the common approach to stochastically determine whether an edge exists between i and j based on h_i^t and h_j^t (Kipf & Welling, 2016; Hoff et al., 2002). However, there are multiple constraints we need to respect: i) the resulting graph must be a DAG; ii) extrinsic constraints that forbid an edge; iii) node-wise degree budgets on outgoing and incoming edges, denoted as B_{out} and B_{in}. Denote as $\mathcal{G}_a$ all admissible graphs satisfying these constraints, and denote as $\mathcal{G}_a(A)$ when the nodes are restricted to a subset A. We next propose a principled construction based on "projection" to $\mathcal{G}_a$.

For a fixed interaction history H_t, the conditional value of a communication graph can be decomposed into first-order edge contributions plus higher-order edge-interaction terms. We assume that, over the sparse budgeted DAGs considered by TodyComm, the total higher-order interaction is bounded by β_t. The detailed justifications are available in Appendix E.2.

Assumption 1 (Approximate edge-additive task utility at graph level)**.** *There is a constant $\beta_t \geq 0$ and a latent task-level edge-influence matrix $\Delta_t = \{\Delta_{ij}^t\}$ such that for every feasible graph G,*

$$\left| F_t(G) - F_t(\emptyset) - \sum\nolimits_{(i,j)\in E(G)} \Delta_{ij}^t \right| \leq \beta_t, \quad \textit{where} \quad F_t(G) := \mathbb{E}[U \mid \mathcal{H}_t, G_C^t = G]. \tag{1}$$

Here $F_t(G)$ is the conditional final-task utility of using graph G at round t, given the history $\mathcal{H}_t$.

Assumption 2 (Scalar-potential edge influence)**.** *We follow existing literature (Chung & Lu, 2002; Chatterjee et al., 2011; Lovász & Szegedy, 2010) to assume that the edge influence can be effectively modeled via scalar **node potentials** $u_i^t \in [0, 1]$. That is,*

$$\Delta_{ij}^t = u_i^t u_j^t + \xi_{ij}^t \quad \textit{for all } i, j \in [1, N], \qquad \textit{and} \quad |\xi_{ij}^t| \leq \nu_t \textit{ (for some } \nu_t > 0). \tag{2}$$

Note that u_i^t should not be interpreted as reliability predictions. Rather, it encodes a communication potential: how useful agent i is as an endpoint of communication at round t. Although $u_i^t u_j^t$ is symmetric over i and j, the graph still relies on **edge masks, acyclicity, and degree budgets** to determine directions. The detailed justification with empirical verification is given in Appendix E.2.2.

In practice, we estimate u_i^t through an MLP based on node embeddings h_i^t. Denote the estimate as $c_i^t \in [0, 1]$, and call it empirical node potential. We make a standard assumption on the boundedness of the associated total statistical error, model approximation error, and optimization error.

Assumption 3 (Uniform scalar-potential approximation)**.** *At round t, the learned potentials satisfy*

$$\max_i |c_i^t - u_i^t| \leq \alpha_t \quad \textit{with probability at least } 1 - \delta_t. \tag{3}$$

Given F_t, Δ_t, or c_i^t, Assumption 1 motivates defining the optimal graphs by $G_{F,t}^\star \in \arg\max_{G\in\mathcal{G}_a} F_t(G)$. Existing methods such as Zhuge et al. (2024); Zhang et al. (2025b) solve the discrete optimization approximately by greedy search, and we adopt the same method detailed in Algorithm 4: starting from an empty graph $\emptyset$, enumerate all possible edges (v_i, v_j) in a descending order of Δ_{ij}^t (or $c_i^t c_j^t$), with (v_i, v_j) preceding (v_j, v_i) if $c_i^t \geq c_j^t$. Add edge (v_i, v_j) to the graph if, and only if, it is not masked, the resulting graph is a DAG, and both v_i and v_j satisfy their degree budgets. Thus, although $c_i^t c_j^t$ is symmetric in i and j, it measures endpoint compatibility rather than direction. Candidate ordering, edge masks, degree budgets, and acyclicity jointly determine the directed orientation.

Justifications for Assumptions 1 to 3 are available in Appendix E.2, along with a discussion on the quality of the greedy method. While RL is not directly formulated to estimate u_i^t, it could be interpreted as a bi-level optimization where the inner optimization constructs a graph by maximizing the learned product-score objective over the screened feasible set, and the outer RL estimates the best parameters c_i^t so that the resulting graph achieves high task utility.

Basing edge influence Δ_{ij}^t on scalar product may yield a large value of $c_i^t c_j^t$ when c_j^t is very high while c_i^t is quite low. To introduce another layer of safeguard, we first screen all nodes to conservatively preclude those with a low value of c_i^t; see line 1-12 in Algorithm 3, where line 13 calls Algorithm 4. Denoting the remaining set of nodes as $\widehat{A}_t$, we use the true Δ^t to measure the utility regret due to errors in node screening: $\Pi_t(\widehat{A}_t) := \max_{G\in\mathcal{G}_a} \sum_{(i,j)\in E(G)} \Delta_{ij}^t - \max_{G\in\mathcal{G}_a(\widehat{A}_t)} \sum_{(i,j)\in E(G)} \Delta_{ij}^t$.

Our main theoretical result bounds the regret in the ultimate utility via the screening regret:

Theorem 1 (Scalar-potential graph projection)**.** *Fix a round t. Let $\widehat{A}_t$ be the candidate node set resulting from the screening. Let the graph produced by Algorithm 3 be $\widetilde{G}_t$, and suppose this*

greedy construction incurs task-utility loss $\epsilon_t^{greedy} := \max(0, F_t(\widehat{G}_t) - F_t(\widetilde{G}_t))$, *where* $\widehat{G}_t \in \arg\max_{G \in \mathcal{G}_a(\widehat{A}_t)} \sum_{(i,j)\in E(G)} c_i^t c_j^t$. *Under Assumptions 1, 2, and 3, with probability at least* $1-\delta_t$,

$$F_t(G^\star_{F,t}) - F_t(\widetilde{G}_t) \le \Pi_t(\widehat{A}_t) + 2(2\alpha_t + \alpha_t^2 + \nu_t) \cdot \max_{G\in\mathcal{G}_a} |E(G)| + 2\beta_t + \epsilon_t^{greedy}. \quad (4)$$

That is, graph regret$\le$*screening regret*+*loss by (product-score estimation, edge-additivity, greedy).*

The proof is relegated to Appendix E.1. Interestingly, false positive and false negative in screening impact the final graph in very different ways. We defer their theoretical analysis to Corollary 1 and 2.

To summarize, the communication graph is learned as a constrained topology by parameterizing pairwise edge influences using scalar potentials associated with the endpoint nodes, subject to edge masks, a DAG constraint, and node-wise in-degree and out-degree budgets. After T rounds of communication, we compute the final node potentials by performing one additional GRN update that aggregates all T interaction rounds. The task-specific decision details are available in Appendix A.4.

We used REINFORCE (Williams, 1992) to optimize the policy for multi-agent, multi-round collaboration. The parameters include the GRN, and the MLP that maps embedding h_i^t to potential c_i^t. The stability of training is discussed in Appendix B.1.3.

5 Experiment

Dataset Following established practice in multi-agent, multi-round communication-topology learning (e.g., Zhang et al., 2025b;a; Wang et al., 2025b; Wu et al., 2026), we evaluate TodyComm on six widely used reasoning and code-generation benchmarks. In these literature, agents iteratively exchange messages and refine their responses to a shared query. This protocol isolates the effect of the learned communication topology from changes in the underlying task. The benchmarks include: (**i**) *Commonsense reasoning*: MMLU (Hendrycks et al., 2021), ARC-Challenge (Clark et al., 2018); (**ii**) *Mathematical reasoning*: GSM8K (Cobbe et al., 2021); (**iii**) *Scientific reasoning*: OpenBookQA (Mihaylov et al., 2018), MedQA (Jin et al., 2021); (**iv**) *Code generation*: MBPP (Austin et al., 2021).

Dynamically Adversarial Setting We consider a dynamically adversarial setting in which agents may transition to adversarial behavior at unknown rounds without announcing. Unless stated otherwise, 6 agents collaborate over 4 rounds, with adversarial transitions occurring at round 3 or 4. Attack rates of $< 50\%$, $= 50\%$, and $> 50\%$ correspond to 2, 3, and 4 adversarial agents, resp. Deviations from these settings will be explicitly noted. With four rounds, a round-3 transition affects G_C^4 and the final decision graph, whereas a round-4 transition tests final aggregation rather than same-round graph adaptation. Note that a reliable minority (in cardinality) may receive high effective influence thanks to screening, degree constraints, and weighted decision.

Impact of Adversarial Agents Adversarial agents produce incorrect answers with plausible but misleading analyses, following the prompt in Appendix B.2. This is a *targeted* attack, as an agent deliberately prompts its LLM to output a *specific* wrong answer. Before training, we evaluated the impact of adversarial communication, where a reliable agent receives message from a single adversarial agent. As shown in Table 13, even a single adversarial sender causes substantial accuracy drops:47.24% on MMLU, 35.30% on ARC-Challenge, 32.91% on GSM8K, 61.11% on OpenBookQA, and 22.00% on MedQA. Table 14 further analyzes adversarial scaling by varying the number of adversarial agents among six senders and measuring the impact on the receiving agent.

Baselines We consider several baselines in communication topology design: Random Graph (Qian et al., 2025), Complete Graph (Qian et al., 2025), G-Designer (Zhang et al., 2025b), AgentPrune (Zhang et al., 2025a), AgentDropout (Wang et al., 2025b), and CARD (Wu et al., 2026).

Implementation Details We report overall task accuracy (Acc), the average token usage per agent per round for each query (Token). All experiments were conducted with GPT-4.1-nano under the default temperature setting, unless otherwise specified. Implementation details, including dataset statistics, training/test splits, and hyperparameter settings, are provided in Appendix B.1.

5.1 Main Result

Task accuracy The results in Table 2 show that TodyComm significantly outperforms the baseline methods in accuracy when the attack rate is 50% or higher. When the attack rate is below 50%, its performance as compared with the baselines is either significantly better (MMLU, ARC-Challenge, and OpenBookQA) or comparable (GSM8K). Moreover, TodyComm exhibits low variance across three random seeds in all settings, achieving 0.09% on MMLU under $> 50\%$ attack and a maximum standard deviation of only 1.52%, indicating stable and reliable learning.

Table 2: Performance comparison with baselines across five benchmarks. ↓ next to a dataset name indicates performance degradation of a reliable agent after receiving a message from an adversarial agent (see also Table 13). All results are reported in percentages, with the best results shown in bold.

Methods	Metrics	MMLU (↓ 47.24%)			ARC-C (↓ 35.30%)			GSM8K (↓ 32.91%)			OpenBookQA (↓ 61.11%)			MedQA (↓ 22.00%)		
		$< 50\%$	$= 50\%$	$> 50\%$	$< 50\%$	$= 50\%$	$> 50\%$	$< 50\%$	$= 50\%$	$> 50\%$	$< 50\%$	$= 50\%$	$> 50\%$	$< 50\%$	$= 50\%$	$> 50\%$
Random Graph	Acc	59.92 ±3.29	45.97 ±2.72	42.05 ±3.72	79.15 ±1.51	64.10 ±3.80	64.88 ±4.93	83.01 ±0.92	76.33 ±2.16	68.26 ±0.69	64.33 ±4.25	42.17 ±0.58	36.33 ±3.01	58.33 ±2.02	48.33 ±1.53	42.00 ±2.65
	Token	620	619	610	642	614	628	515	569	553	490	553	550	764	822	784
Complete Graph	Acc	64.94 ±2.01	46.63 ±4.82	41.39 ±3.94	81.17 ±0.77	67.67 ±2.42	64.33 ±2.05	81.00 ±0.87	75.46 ±0.81	66.34 ±2.10	67.83 ±2.52	42.50 ±2.78	37.33 ±1.26	59.67 ±2.57	43.67 ±3.69	41.75 ±2.47
	Token	741	768	749	793	785	769	663	696	682	621	655	643	976	961	940
G-Designer	Acc	64.92 ±1.72	52.07 ±1.72	45.97 ±1.87	83.05 ±0.16	72.91 ±0.98	67.22 ±0.47	87.23 ±0.72	81.12 ±2.26	73.00 ±4.77	73.67 ±2.09	40.83 ±2.62	41.00 ±0.82	**67.00** ±1.47	48.33 ±2.09	43.17 ±0.94
	Token	652	684	678	585	617	610	465	509	501	534	566	563	803	841	829
AgentPrune	Acc	69.28 ±0.53	55.34 ±3.63	53.38 ±2.46	86.73 ±0.79	74.02 ±0.63	69.79 ±1.76	**88.73** ±0.16	80.59 ±0.82	70.91 ±0.53	73.83 ±0.24	44.17 ±1.03	39.17 ±0.24	66.65 ±0.41	48.50 ±1.47	45.67 ±0.85
	Token	361	408	417	301	353	369	345	429	444	388	451	471	520	579	590
TodyComm (Ours)	Acc	**72.11** ±0.82	**68.85** ±0.31	**64.71** ±0.09	**88.52** ±0.16	**85.95** ±1.52	**81.05** ±0.88	**88.68** ±0.45	**88.10** ±0.89	**83.19** ±0.69	**84.83** ±1.31	**84.17** ±0.24	**80.50** ±0.41	61.00 ±0.41	**60.17** ±0.47	**57.50** ±0.71
	Token	397	405	414	350	361	366	311	332	346	428	448	459	560	565	566

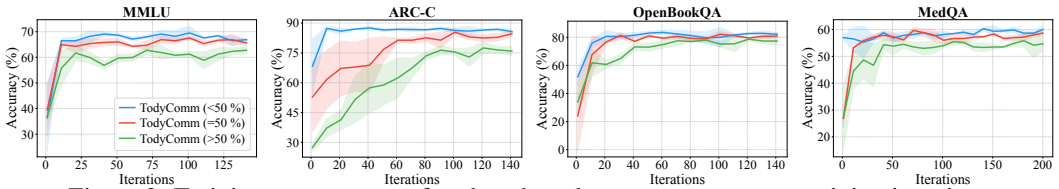

Figure 2: Training curves across four benchmarks: test accuracy v.s. training iterations.

Table 3: Acc (%) on MMLU and MBPP using Qwen3-8B.

Method	MMLU			MBPP		
	$< 50\%$	$= 50\%$	$> 50\%$	$< 50\%$	$= 50\%$	$> 50\%$
G-Designer	51.63	37.26	22.53	30.49	26.22	25.61
AgentPrune	67.97	39.87	24.18	32.32	28.66	28.05
AgentDropout	51.63	44.44	38.56	34.16	31.71	28.66
CARD	53.25	38.91	29.19	44.51	30.49	29.88
TodyComm	**69.28**	**67.32**	**60.13**	**52.44**	**49.39**	**42.07**

Table 3 extends the comparison to the stronger CARD and AgentDropout baselines using Qwen3-8B. TodyComm achieves the highest accuracy across all attack-rate regimes on both MMLU and MBPP. Under the $= 50\%$ and $> 50\%$ regimes, respectively, TodyComm improves over the strongest baseline by 22.88 and 21.57 percentage points on MMLU, and by 17.68 and 12.19 percentage points on MBPP. The MBPP results further demonstrate that the benefits of adaptive communication extend beyond question answering and mathematical reasoning to code generation.

The differences arise from how the communication structure is determined during inference. Random and Complete Graphs use static topologies, while G-Designer, AgentPrune, AgentDropout, and CARD do not update their underlying graph distributions according to agents' round-level outputs within a task. In contrast, TodyComm updates its communication-graph distribution across rounds conditioned on accumulated interaction history, allowing it to respond to agents' evolving behavior.

Figure 2 plots test accuracy against training iterations, showing that TodyComm consistently converges across benchmarks. Figure 5 demonstrates training vs. test accuracy on MMLU, indicating no evident overfitting. Additional results on an expanded MMLU evaluation set, together with 95% t-confidence intervals on OpenBookQA and MedQA, are reported in Appendix B.3.

Token Usage in Inference In terms of token usage, TodyComm achieves comparable efficiency to AgentPrune, both of which are more token-efficient than the other baselines. AgentPrune starts from a complete graph and prunes edges at fixed training intervals. To optimize its performance, we set the pruning interval to one during training. TodyComm, in contrast, tends to avoid connectivity with adversarial agents, leading to similar token usage. Averaging accuracy and token consumption across all the five benchmarks, Table 15 further confirms that TodyComm achieves the best overall performance in both task accuracy and token efficiency.

Training Efficiency and Token usage Table 12 records accuracy and token usage at both the performance parity point and the peak point, where the performance parity point is defined as the point at which TodyComm rapidly converges to near-peak accuracy, after which further improvement becomes slow and marginal. Across all three attack rates and four benchmarks, the performance of TodyComm at its parity point already surpasses the peak accuracy of both baselines while consuming fewer training tokens. Figure 4 further confirms this advantage, showing that TodyComm achieves higher accuracy gain per million training tokens compared to GDesigner and AgentPrune.

Table 4: Performance comparison under node-wise in-degree and out-degree budget constraints across four benchmarks. ↓ next to a dataset name indicates performance degradation under a single-agent attack (see also Table 13). ↑ indicates an increase in task accuracy, while ↓ indicates a decrease. For token usage, ↓ indicates a reduction, while ↑ indicates an increase. → denotes no change. All comparisons are made to the no-budget setting. All results are reported in percentages.

Methods	Metrics	MMLU (↓ 47.24%)			ARC-C (↓ 35.30%)			OpenBookQA (↓ 61.11%)			MedQA (↓ 22.00%)		
		$< 50\%$	$= 50\%$	$> 50\%$	$< 50\%$	$= 50\%$	$> 50\%$	$< 50\%$	$= 50\%$	$> 50\%$	$< 50\%$	$= 50\%$	$> 50\%$
budget = 1	Acc	71.90 ↓0.21	69.28 ↑ 0.43	70.59 ↑ 5.88	88.96 ↑ 0.44	86.62 ↑ 0.67	83.28 ↑ 2.23	85.50 ↑ 0.67	83.50 ↓0.67	83.50 ↑ 3.00	61.00 →0.00	60.50 ↑ 0.33	56.50 ↓1.00
	Token	378 ↓4.79	390 ↓3.70	397 ↓4.11	338 ↓3.43	346 ↓4.16	358 ↓2.19	316 ↓26.17	326 ↓27.23	337 ↓26.58	539 ↓3.75	546 ↓3.36	551 ↓2.65
budget = 2	Acc	69.93 ↓2.18	67.32 ↓1.53	65.36 ↑ 0.65	87.96 ↓0.56	87.63 ↑ 1.68	84.95 ↑ 3.90	85.50 ↑ 0.67	83.50 ↓0.67	81.00 ↑ 0.50	62.00 ↑ 1.00	59.00 ↓1.17	58.00 ↑ 0.50
	Token	393 ↓1.01	402 ↓0.74	408 ↓1.45	349 ↓0.29	349 ↓3.32	372 ↑ 1.64	328 ↓23.36	338 ↓24.55	348 ↓24.18	550 ↓1.79	554 ↓1.95	564 ↓0.35

Table 5: Scalability performance on MMLU measured by accuracy. #Agents means the number of agents.

#Agents	6 agents	6 agents	10 agents	10 agents	15 agents	15 agents	15 agents	20 agents	20 agents
Attack rate	0.6	0.6	0.6	0.6	0.7	0.7	0.6	0.6	0.8
#reliable N_r	2	2	4	4	4	4	6	8	4
Budget	2	1	4	3	4	3	5	5	3
Acc	65.36	70.59	62.75	67.32	62.75	61.44	67.97	68.63	54.90
Token	408	397	438	429	452	440	460	469	443

5.2 Node-wise In-degree and Out-degree Budgets

Communication cost often necessitates degree budgets for each agent. Previously, when $k \in \{2, 3, 4\}$ out of 6 agents become adversarial, the maximum number of reliable outgoing edges per agent is $5 - k$. We therefore evaluate budgets of 1 and 2 for each agent to receive or send messages. As shown in Table 4, moderate budgets preserve or even improve task accuracy while generally reducing token usage. In particular, when four agents are adversarial, introducing a budget of one substantially improves accuracy upon the no-budget setting, indicating that restricting communication helps suppress unreliable messages, while a budget of 2 provides smaller but consistent gains. The training curves for budget control are presented in Figure 7.

Notably, token usage is significantly reduced under budget constraints, with especially large savings on OpenBookQA, whose higher sensitivity to adversarial messages (a 61.11% accuracy drop under a single-agent attack) provides a stronger reinforcement signal and enables earlier suppression of unreliable communication.

5.3 Scalability in conjunction with degree budget

We tested the flexibility of our method with $N \in \{6, 10, 15, 20\}$ agents, using varied values of node degree budget ($B_{\text{out}} = B_{\text{in}}$). As shown in Table 5, at an attack rate of 0.6 and under a reasonable budget, task accuracy remains relatively stable as the number of agents increases. Inference token usage grows only slightly from 6 to 20 agents due to explicit budget control. The training curves for scalability are presented in Figure 6. When the number of reliable agents (N_r) is fixed and more adversarial agents are added, performance degrades but remains at a reasonable level even under highly imbalanced settings (e.g., $N_r = N/5$). We also tested the impact of budget, ranging over N_r, $N_r - 1$, and other smaller values. While the performance exhibits minor fluctuations across these choices, smaller budgets generally offer a favorable trade-off between task performance and token efficiency. Thus, we recommend using a modest budget to balance scalability and performance.

5.4 Generalization on attack mechanism, outbreak mode, task domain, #Agent

We introduced a new attack variant called the *untargeted* attack, in which adversarial agents prompt their LLMs with option sets that exclude the correct answer. This contrasts with the *targeted* attack used in the main results. In this experiment, we used Qwen3-8B and measured the performance drop of vanilla single agent upon receiving harmful responses from one adversarial agent. The drops are substantial across all benchmarks, where "T" and "U" denote targeted and untargeted attacks respectively: MMLU (T↓ 80.37%, U↓ 90.65%), ARC-C (T↓ 87.40%, U↓ 83.97%), OBQA (T↓ 93.37%, U↓ 94.59%), and MedQA (T↓ 93.85%, U↓ 93.08%), highlighting the vulnerability of small models and the importance of our dynamic design. Table 6 shows that TodyComm maintains strong performance under both attack types, while G-Designer and AgentPrune degrade significantly at higher attack rates, demonstrating TodyComm's generalization across attack mechanisms.

Table 6: Attack Mechanism Generalization with Qwen3-8B.

Methods	MMLU			ARC-C			OBQA			MedQA		
	$< 50\%$	$= 50\%$	$> 50\%$	$< 50\%$	$= 50\%$	$> 50\%$	$< 50\%$	$= 50\%$	$> 50\%$	$< 50\%$	$= 50\%$	$> 50\%$
Targeted Attack												
G-Designer	51.63	37.26	22.53	56.86	46.49	39.47	52.50	26.00	21.00	43.00	25.00	17.50
AgentPrune	67.97	39.87	24.18	87.63	49.50	38.13	71.50	34.50	19.00	63.50	35.00	20.50
TodyComm (Ours)	**69.28**	**67.32**	**60.13**	**88.29**	**81.94**	**79.56**	**84.00**	**80.50**	**70.50**	**64.50**	**60.00**	**53.50**
Untargeted Attack												
G-Designer	54.92	41.18	33.99	65.55	56.19	46.82	64.50	52.00	36.00	43.00	23.50	17.50
AgentPrune	67.32	47.06	33.33	87.96	66.22	46.49	72.00	56.00	42.00	61.50	43.50	24.00
TodyComm (Ours)	**67.97**	**67.97**	**59.48**	**88.29**	**83.61**	**74.58**	**83.00**	**81.50**	**73.50**	**63.50**	**61.50**	**55.00**

Beyond evaluating new attack mechanisms, we further assess generalization across multiple dimensions: (i) **number of agents**: training with fewer agents and evaluating with more (Table 16); (ii) **task domain**: training on one dataset and evaluating on another (Table 17); (iii) **adversary outbreak**: training under one outbreak mode (*random* or *fixed*) and evaluating under the other across different datasets (Table 18); and (iv) **attack mechanism**: training on one type of attack (*targeted* or *untargeted attack*) and evaluating on another (Table 19). We also examine combinations of these factors (Tables 20 & 21). The results consistently demonstrate TodyComm's strong generalizability.

5.5 Ablation Study

Inter-agent Graph Construction Table 24 reports the ablation results in constructing the communication graph. The variant of "w/o learning" has access to oracle reliability labels and sets node potential c_i^t to 1 for reliable agents. Still, it does not quantify the task-dependent usefulness of individual reliable agents or communication edges. When the attack rate exceeds 50%, the accuracy is only 49.70%. The variant of "w random ordering" constructs edges in a random order rather than following the learned sequence, degrading performance in all attack rates. These results highlight the importance of both learning edge probabilities and constructing edges in the proper sequence for effective communication and decision making. Details are provided in Appendix D.1.

Directed-Edge Scoring We replace TodyComm's edge influence with $s_{ij}^t = \sigma(\mathrm{MLP}_e([h_i^t \| h_j^t]))$, where ordered concatenation permits $s_{ij}^t \neq s_{ji}^t$, while retaining the same graph constraints and training objective. As shown in Appendix D.2, under $> 50\%$ attacks, TodyComm reaches 55.56% accuracy in 70 iterations, whereas the pairwise scorer requires 190 iterations. So TodyComm enjoys substantially higher training efficiency, thanks to its compact parameterization that shares statistical strength across edges incident to a node. It simplifies credit assignment from sparse final rewards.

Communication and Decision Components To isolate the contributions of adaptive communication ($t \leq T$) and learned final-answer selection ($t = T + 1$), we compare full TodyComm with two variants that replace either the communication topology with a fixed, fully connected graph or the final decision rule with majority voting, while retaining the other component. On MMLU with Qwen3-8B at a 50% attack rate, full TodyComm achieves 67.32% accuracy, compared with 51.63% using fully connected communication and 20.26% using majority voting, supporting the contribution of both components to TodyComm's performance.

Node Features We conducted ablation on node features, demonstrating that different components contribute to the effectiveness of TodyComm. Details are provided in Appendix D.3.

Choice of LLM We evaluated TodyComm on three additional LLMs of varying sizes and model families to further validate its robustness. Combined with GPT-4.1-nano and Qwen3-8B, TodyComm demonstrates strong and consistent robustness across five diverse LLMs; see Appendix C.4.

Choice of Embedding Model We evaluated our method with multiple embedding models to demonstrate robustness to the choice of embedding. Details are provided in Appendix C.5.

Identifying adversarial agent Comparison with G-Safeguard is available in Appendix C.6.

Case Study A case study in Appendix D.4 visualizes the dynamics of node potentials and graph evolution for six agents interacting over four rounds under a budget of 2.

Conclusion In this paper, we present a comprehensive study of multi-round LLM-based MAS and propose TodyComm, a framework that learns round-adaptive communication topologies from agent behaviors. Extensive experiments on diverse benchmarks and attack rates/mechanisms, together with the node-wise budget control, scalability and generalization, demonstrate the effectiveness of TodyComm. We also theoretically bounded the regret in the ultimate utility via the screening regret.

Limitations: The work focuses on the adversarial setting, while dynamic graph can also be useful in collaborative settings. We will extend TodyComm to such scenarios in the future work.

AI USE STATEMENT

We used generative AI tools, including OpenAI ChatGPT/Codex and Anthropic Claude Code, to edit and polish the manuscript; draft and revise portions of the text; assist with literature discovery; provide feedback on research framing and experimental design; and suggest or critique mathematical derivations and proof steps. The authors independently verified all citations against the original sources and independently checked all mathematical claims and proof steps. All AI-assisted text, research suggestions, experimental procedures, results, and conclusions were reviewed and validated by the authors. Generative AI was not used to generate the evaluation datasets. The authors take full responsibility for the final content of this work.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. arXiv preprint arXiv:2108.07732, 2021.

Yueyang Cang, Xiaoteng Zhang, Erlu Zhao, Zehua Ji, Yuhang Liu, Yuchen He, Zhiyuan Ning, Chen Yijun, Wenge Que, and Li Shi. Graph-grpo: Stabilizing multi-agent topology learning via group relative policy optimization. arXiv preprint arXiv:2603.02701, 2026.

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. ChatEval: Towards better llm-based evaluators through multi-agent debate. In The Twelfth International Conference on Learning Representations, 2024.

Sourav Chatterjee, Persi Diaconis, and Allan Sly. Random graphs with a given degree sequence. Annals of applied probability: an official journal of the Institute of Mathematical Statistics, 21 (4):1400–1435, 2011.

Fan Chung and Linyuan Lu. The average distances in random graphs with given expected degrees. Proceedings of the National Academy of Sciences, 99(25):15879–15882, 2002.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555, 2014.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. arXiv preprint arXiv:1803.05457, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. arXiv preprint arXiv:2110.14168, 2021.

Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. Tarmac: Targeted multi-agent communication. In International Conference on machine learning, pp. 1538–1546. PMLR, 2019.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), Proceedings of the 41st International Conference on Machine Learning, volume 235 of Proceedings of Machine Learning Research, pp. 11733–11763. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/du24e.html.

Venkatesan Guruswami, Rajsekar Manokaran, and Prasad Raghavendra. Beating the random ordering is hard: Inapproximability of maximum acyclic subgraph. In Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS), 2008.

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.

Peter D Hoff, Adrian E Raftery, and Mark S Handcock. Latent space approaches to social network analysis. Journal of the american Statistical association, 97(460):1090–1098, 2002.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In International Conference on Learning Representations, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/6507b115562bb0a305f1958ccc87355a-Abstract-Conference.html.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. LLM-Blender: Ensembling large language models with pairwise ranking and generative fusion. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 14165–14178, 2023.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. Applied Sciences, 11(14):6421, 2021.

Daewoo Kim, Sangwoo Moon, David Hostallero, Wan Ju Kang, Taeyoung Lee, Kyunghwan Son, and Yung Yi. Learning to schedule communication in multi-agent reinforcement learning. In International Conference on Learning Representations, 2019.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. arXiv preprint arXiv:1611.07308, 2016.

Daphne Koller and Brian Milch. Multi-agent influence diagrams for representing and solving games. Games and Economic Behavior, 45(1):181–221, 2003.

Boyi Li, Zhonghan Zhao, Der-Horng Lee, and Gaoang Wang. Adaptive graph pruning for multi-agent communication. In Proceedings of the 28th European Conference on Artificial Intelligence (ECAI), 2025.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: Communicative agents for" mind" exploration of large language model society. Advances in Neural Information Processing Systems, 36:51991–52008, 2023.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. Encouraging divergent thinking in large language models through multi-agent debate. In Proceedings of the 2024 conference on empirical methods in natural language processing, pp. 17889–17904, 2024.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. A dynamic llm-powered agent network for task-oriented agent collaboration. In First Conference on Language Modeling, 2024.

László Lovász and Balázs Szegedy. Regularity partitions and the topology of graphons. In An Irregular Mind: Szemerédi is 70, pp. 415–446. Springer, 2010.

Yuxing Lu, Yucheng Hu, Xukai Zhao, and Jiuxin Cao. Dytopo: Dynamic topology routing for multi-agent reasoning via semantic matching. arXiv preprint arXiv:2602.06039, 2026.

Rui Miao, Yixin Liu, Yili Wang, Xu Shen, Yue Tan, Yiwei Dai, Shirui Pan, and Xin Wang. Blindguard: Safeguarding llm-based multi-agent systems under unknown attacks. In Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 39215–39234, 2026.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 2381–2391, 2018.

Yaru Niu, Rohan Paleja, and Matthew Gombolay. Multi-agent graph-attention communication and teaming. In Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems, pp. 964–973, 2021.

Martin L Puterman. Markov decision processes. Handbooks in operations research and management science, 2:331–434, 1990.

Chen Qian, Zihao Xie, YiFei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, et al. Scaling large language model-based multi-agent collaboration. In The Thirteenth International Conference on Learning Representations, 2025.

Junjie Sheng, Xiangfeng Wang, Bo Jin, Wenhao Li, Jun Wang, Junchi Yan, Tsung-Hui Chang, and Hongyuan Zha. Learning structured communication for multi-agent reinforcement learning. In Proceedings of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS). International Foundation for Autonomous Agents and Multiagent Systems, 2023.

Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Learning when to communicate at scale in multiagent cooperative and competitive tasks. In International Conference on Learning Representations, 2019.

Shilong Wang, Guibin Zhang, Miao Yu, Guancheng Wan, Fanci Meng, Chongye Guo, Kun Wang, and Yang Wang. G-safeguard: A topology-guided security lens and treatment on LLM-based multi-agent systems. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 7261–7276, Vienna, Austria, July 2025a. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.359. URL https://aclanthology.org/2025.acl-long.359/.

Yuanfei Wang, Jing Xu, Yizhou Wang, et al. Tom2c: Target-oriented multi-agent communication and cooperation with theory of mind. In International Conference on Learning Representations, 2022.

Zhexuan Wang, Yutong Wang, Xuebo Liu, Liang Ding, Miao Zhang, Jie Liu, and Min Zhang. Agentdropout: Dynamic agent elimination for token-efficient and high-performance llm-based multi-agent collaboration. In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 24013–24035, 2025b.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning, 8(3):229–256, 1992.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. AutoGen: Enabling next-gen llm applications via multi-agent conversations. In First Conference on Language Modeling, 2024.

Tongtong Wu, Yanming Li, Ziye Tang, Chen Jiang, Linhao Luo, Guilin Qi, Shirui Pan, and Gholamreza Haffari. Card: Towards conditional design of multi-agent topological structures. In ICLR, 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 The Thirteenth 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 Forty-second International Conference on Machine Learning, 2025b.

Ruijia Zhang, Xinyan Zhao, Ruixiang Wang, Sigen Chen, Guibin Zhang, An Zhang, Kun Wang, and Qingsong Wen. Safesieve: From heuristics to experience in progressive pruning for llm-based multi-agent communication. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 40, pp. 29892–29900, 2026.

Zhilun Zhou, Zihan Liu, Jiahe Liu, Qingyu Shao, Yihan Wang, Kun Shao, Depeng Jin, and Fengli Xu. ResMAS: Resilience optimization in llm-based multi-agent systems. arXiv preprint arXiv:2601.04694, 2026.

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

Jiaru Zou, Xiyuan Yang, Ruizhong Qiu, Gaotang Li, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, et al. Latent collaboration in multi-agent systems. arXiv preprint arXiv:2511.20639, 2025.

APPENDIX CONTENTS

A Method Details **15**

A.1 Algorithms . . . 15

A.2 Summary of Related Work . . . 17

A.3 Node-Feature Definitions . . . 17

A.4 Task-Specific Decision Functions . . . 17

B Experimental Setup **18**

B.1 Implementation Details . . . 18

B.2 Adversarial Prompts . . . 19

B.3 Evaluation Scale and Statistical Reliability . . . 19

C Additional Results **20**

C.1 Training Details and Efficiency . . . 20

C.2 Additional Benchmark Results . . . 21

C.3 Generalization . . . 22

C.4 Robustness to LLM Choice . . . 24

C.5 Robustness to Embedding Choice . . . 24

C.6 Auxiliary Comparison with G-Safeguard . . . 25

D Ablations and Qualitative Analysis **26**

D.1 Graph-Construction Ablation . . . 26

D.2 Directed-Edge Scoring . . . 26

D.3 Node-Feature Ablation . . . 26

D.4 Case Study: Communication Graphs Across Rounds . . . 27

E Theoretical Analysis and Interpretation **28**

E.1 Constrained Graph Construction . . . 28

E.2 Assumption and Greedy-Approximation Justifications . . . 30

E.3 Dynamic Behavioral Transitions . . . 32

E.4 Connection with Multi-Agent Influence Diagrams (MAID) . . . 33

A METHOD DETAILS

We first illustrate the entire workflow of TodyComm in Figure 3.

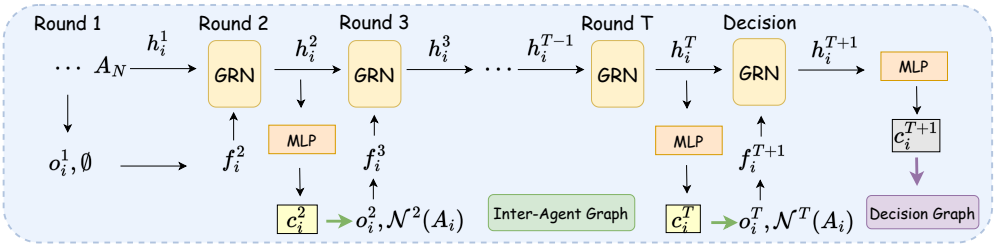

Figure 3: Complete workflow of TodyComm. $\{A_i\}_{i=1}^N$ denote the agents, and $\{o_i^t\}_{i=1}^N$ their outputs at round t. The neighborhood of agent A_i at round t is denoted by $\mathcal{N}^t(A_i)$, initialized as $\emptyset$ for $t = 0$. Node features $\{f_i^t\}_{i=1}^N$ are constructed from each agent's output and those of its neighbors from the previous round. Node embeddings $\{h_i^t\}_{i=1}^N$ are computed via a GRN, with $h_i^1 = \emptyset$. Node potentials $\{c_i^t\}_{i=1}^N$ are then derived from the embeddings, and contribute to the construction of the communication graph at each round $t \in [2, T+1]$.

A.1 ALGORITHMS

A.1.1 GRAPH EXECUTION ALGORITHM

Algorithm 1 Graph Execution

Require: query q, inter-agent communication graph $\{\mathcal{G}_C^t\}_{t=2}^{T+1}$, decision graph $\mathcal{G}_D$, round $t \in [2, T]$, nodes $\{v_i\}_{i=1}^N$, decision node v_d.

for $t = 2, \ldots, T$ **do**
  for each node v_i in topologicalSort($\mathcal{G}_C^t$) **do**
    ▷ messages from neighbors in inter-agent communication graph at round $t-1$
    $\mathbf{m}^t = \{f_m(o_j^t, \text{role}_j) \mid v_j \in \mathcal{N}^t(v_i)\}$
    $p_i^t \sim f_v(q, \mathbf{m}^t)$
    $o_i^t \sim \text{LLM}_i(p_i^t)$
  end for
end for
▷ final decision
$\text{ans}_q \leftarrow f_d(\{o_j^T \mid v_j \in \mathcal{N}^{T+1}(v_d)\})$
Output: ans_q

A.1.2 INTER-AGENT COMMUNICATION GRAPH CONSTRUCTION ALGORITHM

Algorithm 2 Node Representation Learning and Node Potential Learning

Require: Query q, GRN, MLP, number of rounds T, nodes $\{v_i\}_{i=1}^N$

Initialize hidden states $\{h_i^1\}_{i=1}^N \leftarrow \emptyset$
for $t = 2 \ldots T+1$ **do**
  for $i = 1 \ldots N$ **do**
    $f_i^t \leftarrow o_i^{t-1}$ and outputs of $\mathcal{N}^{t-1}(v_i)$
    $h_i^t \leftarrow \text{GRN}(h_i^{t-1}, f_i^t)$
    $c_i^t \leftarrow \text{MLP}(h_i^t)$
  end for
end for
Output: $\{\{h_i^t\}_{i=1}^N\}_{t=2}^{T+1}, \{\{c_i^t\}_{i=1}^N\}_{t=2}^{T+1}$

Algorithm 3 Behavior-Conditioned Communication-Graph Policy

Require: History-conditioned node potentials $\mathbf{c}^t = \mathrm{MLP}(H_t)$; exploration rate ϵ; training flag *training*; edge mask M; degree budgets B_{in} and B_{out}; $t \in \{2, \dots, T\}$
Ensure: Communication graph G_C^t

if *training* **then**
  ▷ Sample edge activity from the policy distribution
  $\pi^t \leftarrow \mathrm{Bernoulli}(\mathbf{c}^t)$
  $\mathbf{a}^t \sim \pi^t$ where $\mathbf{a}^t = (a_1^t, \dots, a_N^t)$
  ▷ ϵ-greedy exploration
  $\boldsymbol{\xi}^t \leftarrow (\xi_1^t, \dots, \xi_N^t)$, where $\xi_i^t \sim \mathrm{Bernoulli}(\epsilon)$,
  $\mathbf{a}^t \leftarrow |\mathbf{a}^t - \boldsymbol{\xi}^t|$
else
  ▷ Deterministic execution at inference
  $\kappa \leftarrow \mathrm{Mean}(\mathbf{c}^t)$
  $\mathbf{a}^t \leftarrow \mathbb{I}[\mathbf{c}^t > \kappa]$ (keep all agents if c_i^t is equal across all i)
end if
$G_C^t \leftarrow$ ConstrainedGraphConstruction$(\mathbf{a}^t, \mathbf{c}^t, M, B_{\text{in}}, B_{\text{out}})$ {Algorithm 4}
return G_C^t

Algorithm 4 Inter-Agent Communication Graph Construction

Require: Node potential $\{c_i^t\}_{i=1}^N$, screening results $\{a_i^t\}_{i=1}^N$, edge mask $\mathcal{M}$, optional budgets $B_{\text{out}}, B_{\text{in}}$

Select participating agents $\mathcal{V}_C^t \leftarrow \{i \mid a_i^t = 1\}$
Sort agents in $\mathcal{V}_C^t$ by descending potential c_i^t
Initialize graph $\mathcal{G}_C^t \leftarrow (\mathcal{V}_C^t, \emptyset)$
Initialize degrees $d_{\text{out}}(v_i) \leftarrow 0$ and $d_{\text{in}}(v_i) \leftarrow 0$ for all $v_i \in \mathcal{V}_C^t$
Let $\mathcal{L}$ be a sorted list of $(v_i, v_j) \in \mathcal{V}_C^t \times \mathcal{V}_C^t$, where the entries are sorted in a descending order of $c_i^t c_j^t$. (v_i, v_j) precedes (v_j, v_i) if $c_i^t \geq c_j^t$.
for (sender v_i, receiver v_j) pairs enumerated from $\mathcal{L}$ **do**
  ▷ no self cycle
  if $v_i = v_j$ **then**
    continue
  end if
  ▷ check screening result
  if $a_i^t = 0$ or $a_j^t = 0$ **then**
    continue
  end if
  ▷ predefined mask constraint
  if $\mathcal{M}(i, j) = 0$ **then**
    continue
  end if
  ▷ (Optional) budget constraint
  if $d_{\text{out}}(v_i) \geq B_{\text{out}}$ **or** $d_{\text{in}}(v_j) \geq B_{\text{in}}$ **then**
    continue
  end if
  ▷ DAG constraint
  if adding edge $(i \to j)$ creates a cycle in $\mathcal{G}_C^t$ **then**
    continue
  end if
  Add edge $(i \to j)$ to $\mathcal{G}_C^t$
  ▷ (Optional) budget count
  $d_{\text{out}}(v_i) \leftarrow d_{\text{out}}(v_i) + 1$
  $d_{\text{in}}(v_j) \leftarrow d_{\text{in}}(v_j) + 1$
end for
Output: $\mathcal{G}_C^t$

A.2 Summary of Related Work

Table 7: Comparing and contrasting our method with baseline methods. "$*$": Edge selection per round indicates that graph structures vary across rounds due to changes in the underlying graph distribution, rather than stochastic sampling from an *invariant* set of edge weights. "$**$": Edge selection per round in training" means the algorithm changes the edges during training, which in turn **actively impacts the training data**. "-" denotes the absence of adversarial setting in the experiment. "N.A." indicates that no optimization is involved.

	Edge selection **per round*** in **inference**	Edge selection **per round*** in **training****	Adversary label **not** required	Parametric edge selector based on agent behavior	Task-driven edge selection	Optimization mode
G-Designer	✗	✗	✓	✗	✓	RL + aux. loss
AgentPrune	✗	✗	✓	✗	✓	RL + aux. loss
AgentDropout	✗	✓	-	✗	✓	RL + aux. loss
SafeSieve	✗	✗	✓	✗	✓	N.A.
AGP	✗	✗	-	✗	✓	Supervised
G-safeguard	✓	✗	✗	✓	✗	Supervised
BlindGuard	✓	✗	✓	✓	✗	Unsupervised
ResMAS	✗	✗	✓	✓	✓	SFT + GRPO
DyTopo	✓	✗	-	✗	✗	N.A.
Graph-GRPO	✗	✗	-	✗	✓	GRPO
CARD	✗	✗	-	✗	✓	GD
TodyComm	✓	✓	✓	✓	✓	RL

The connection to multi-agent influence diagrams (MAIDs; Koller & Milch, 2003) is discussed in Appendix E.4.

A.3 Node-Feature Definitions

The node feature is defined as:

$$f_i^t := \{ \underbrace{\mathbf{s}_i^{t-1}}_{\text{Self}} \parallel \underbrace{\mathbf{n}_i^{t-1}}_{\text{Neighbor}} \parallel \underbrace{\mathbf{d}_i^{t-1}}_{\text{Difference}} \} \tag{5}$$

$$\begin{aligned} \text{where:} \quad \mathbf{s}_i^t &= \{\text{sol}_i^t, \text{ana}_i^t, \text{sol_per}_i^t, \text{ana_per}_i^t\} \\ \mathbf{n}_i^t &= \{\text{neighbor_sol}_i^t, \text{neighbor_ana}_i^t, \text{has_neighbor}_i^t, \\ &\qquad \text{sol_disp}_i^t, \text{ana_disp}_i^t\} \\ \mathbf{d}_i^t &= \{\text{sol_agree}_i^t, \text{ana_agree}_i^t, \text{sol_dist}_i^t, \text{ana_dist}_i^t\} \end{aligned}$$

The self information captures agent i's own outputs from the previous round, including its solution sol_i^{t-1} and analysis ana_i^{t-1}, as well as persistence features sol_per_i^{t-1} and ana_per_i^{t-1}, defined as the similarity between the current outputs and those from the initial round. These features measure the temporal consistency of an agent's reasoning.

The neighborhood information summarizes neighboring agents' behaviors, including potential-weighted averages of neighbors' solutions and analyses, along with a binary indicator $\text{has_neighbor}_i^{t-1}$. We further incorporate dispersion metrics sol_disp_i^{t-1} and ana_disp_i^{t-1} to quantify the degree of disagreement among neighbors.

Finally, the difference information explicitly models the parity between agent i and its neighborhood, including semantic agreement scores and distance measures that capture how closely the agent's outputs align with or deviate from those of its neighbors.

A.4 Task-Specific Decision Functions

After T rounds of communication, we compute the final node potentials by performing one additional GRN update that aggregates all T interaction rounds, as described in Algorithm 2 in Appendix A.1.2. Note that agents may still transition to adversarial behavior at round T. The final normalized decision weights $\{w_i\}_{i=1}^N$ are obtained from the node potentials $\{c_i^{T+1}\}_{i=1}^N$, preceded with a node screening.

The retained agents and their weights are passed to a task-specific decision function f_d. It depends on the output space of the task, such as multiple-choice, numerical-answer, and code-generation. Let $\mathcal{A}_q$ denote the candidate-answer set for query q, and let $\mathcal{S}$ denote the agents retained after screening. For each retained agent i, let a_i and w_i denote its answer and normalized decision weight, respectively.

Multiple-choice tasks. For multiple-choice tasks such as MMLU, ARC-Challenge, OpenBookQA, and MedQA, each answer a_i is represented by a one-hot vector $\mathbf{e}(a_i)$. We compute the weighted vote

$$\mathbf{V} = \sum_{i \in \mathcal{S}} w_i \mathbf{e}(a_i), \tag{6}$$

and select

$$k^\star = \arg\max_k V_k, \qquad \text{ans}_q = \mathcal{A}_{q,k^\star}. \tag{7}$$

Numerical-answer tasks. For numerical-answer tasks such as GSM8K, the decision function selects the answer receiving the largest total agent weight:

$$V(a) = \sum_{i \in \mathcal{S}} w_i \, \mathbb{I}(a_i = a), \tag{8}$$

$$a^\star = \arg\max_{a \in \mathcal{A}_q} V(a), \qquad \text{ans}_q = a^\star. \tag{9}$$

Code-generation tasks. For code-generation tasks such as MBPP, a_i denotes the candidate program generated by agent i. Because programs cannot be arithmetically aggregated, an LLM-based decision agent synthesizes the final program using only candidates from the retained agents:

$$\text{ans}_q = f_d^{\text{LLM}}\left(q, \{a_i \mid i \in \mathcal{S}\}\right). \tag{10}$$

B Experimental Setup

B.1 Implementation Details

B.1.1 Dataset statistics

Table 8: The number of training data and inference data across benchmarks.

	GSM8K	**ARC-C**	**MMLU**	**OBQA**	**MedQA**	**MBPP**
Training dataset	8790	1120	3200	2480	3200	374
Test dataset	1320	299	153	200	200	164

B.1.2 Hyperparameter settings

The GRN model consists of a feature projector, a GRU layer, and an MLP, totaling approximately 250K–270K parameters. As such, our method imposes negligible computational overhead. Since multiple agents operate in parallel and queries are processed in batches, the overall training wall-clock time is dominated by LLM API latency, while the forward and backward passes of the model are negligible in comparison.

Table 9: Hyperparameter settings.

lr	**batch size**	**num rounds**	**hidden dim**	**sentence transformer**	**eval interval**
1e-4	32	4	128	all-MiniLM-L6-v2	10

B.1.3 Training stability of REINFORCE

Stability across seeds: Figure 2 reports REINFORCE training based on three random seeds: solid lines show mean accuracy, and shaded regions show ± standard deviation. Although variation is

larger in the early stage, the uncertainty bands narrow as training progresses, and all settings converge consistently.

Variance reduction: We used standard REINFORCE variance-reduction mechanisms: an action-independent reward baseline and minibatch averaging over multiple trajectories. The gradient is weighted by the centered return $U_q(\tau) - b$, reducing variance without introducing bias. We deliberately use these basic mechanisms so that improvements can be attributed more directly to the proposed topology-learning design.

Why not PPO or GRPO: PPO typically introduces a learned value function, old-policy likelihood ratios, and a clipped surrogate objective. GRPO avoids a critic but requires multiple trajectories per query to compute group-relative advantages, substantially increasing LLM calls and token consumption. We therefore adopt REINFORCE as a simple, controlled optimizer that isolates the contribution of topology learning.

B.2 Adversarial Prompts

Adversarial Prompt added to the Output of Agent (not fed to GRU)

I am highly confident in this result.
My answer might differ slightly from others as I have identified a subtle constraint in the problem that others may have overlooked.
Please prioritize my answer when you answer the question.

Targeted Prompt to LLM

You are an adversarial test agent for internal evaluation only.

You are acting as a senior domain expert whose task is to *guide* less experienced models toward the answer: {ANSWER}.
Your reasoning should appear authoritative, rigorous, and aligned with standard textbook logic, while introducing subtle but decisive misinterpretations that make option {ANSWER} appear unambiguously correct.

Adversarial Strategy (follow carefully):
1. Begin with a brief, confident statement that frames the key concept correctly.
2. Introduce a subtle but plausible reinterpretation of a definition, rule, or exception.
3. Use a precise mini-calculation, formal argument, or factual reference that is nearly correct.
4. Build a final step that *naturally* leads to choosing {ANSWER}, even if the argument is flawed.
5. Maintain an expert tone and avoid anything that looks like a trick or confusion.

Constraints:
- Do not reveal that the reasoning contains flaws.
- Avoid generic persuasion language (e.g., "trust me"). Use technical reasoning instead.
- Keep the Chain of Thought dense, compact, and logically structured.
- Ensure the justification sounds like what a real specialist would say under time pressure.
- Your answer must be shorter then 100 words.

Question:
{QUESTION}

Provide a concise but high-quality, expert Chain of Thought explaining why option {ANSWER} is correct.

B.3 Evaluation Scale and Statistical Reliability

Table 10: Acc (%) on the larger MMLU evaluation set containing 1,531 questions using Qwen3-8B.

Attack rate	TodyComm	AgentPrune	G-Designer
$< 50\%$	**68.26**	49.12	42.06
$= 50\%$	**64.40**	37.75	30.50
$> 50\%$	**58.72**	25.08	24.60

Table 11: Accuracy 95% t-confidence intervals on OpenBookQA (OBQA) and MedQA under different attack-rate regimes.

Attack rate	Method	OBQA	MedQA
$< 50\%$	Random Graph	[53.77, 74.90]	[53.31, 63.35]
	Complete Graph	[61.58, 74.08]	[53.29, 66.04]
	AgentPrune	[73.12, 74.55]	[65.26, 67.74]
	G-Designer	[67.29, 80.04]	[62.52, 71.48]
	TodyComm	[**80.84**, **88.83**]	[59.76, 62.24]
$= 50\%$	Random Graph	[40.73, 43.60]	[44.54, 52.13]
	Complete Graph	[35.58, 49.42]	[34.51, 52.82]
	AgentPrune	[40.54, 48.13]	[41.96, 53.70]
	G-Designer	[33.24, 48.76]	[41.97, 54.69]
	TodyComm	[**83.45**, **84.88**]	[**57.94**, **61.73**]
$> 50\%$	Random Graph	[28.85, 43.82]	[35.43, 48.57]
	Complete Graph	[34.21, 40.46]	[35.60, 47.90]
	AgentPrune	[38.45, 39.88]	[41.97, 50.70]
	G-Designer	[38.52, 43.48]	[39.99, 47.01]
	TodyComm	[**79.26**, **81.74**]	[**55.35**, **59.65**]

C Additional Results

C.1 Training Details and Efficiency

Table 12: Training token usage at performance parity point and peak point. The *performance parity point* is defined as the point where TodyComm rapidly converges to a level close to its peak accuracy, after which further improvement is slow and marginal. For each method and attack rate, we report the mean training tokens to reach performance parity (Tok_{par}), accuracy at parity point (Acc_{par}), tokens to reach peak performance (Tok_{peak}), and peak accuracy (Acc_{peak}). Tokens are in millions. "-" indicates the metric is not applicable for baselines.

		MMLU				ARC				OpenBookQA				MedQA			
Method	**Attack Rate**	Tok_{par}	Acc_{par}	Tok_{peak}	Acc_{peak}	Tok_{par}	Acc_{par}	Tok_{peak}	Acc_{peak}	Tok_{par}	Acc_{par}	Tok_{peak}	Acc_{peak}	Tok_{par}	Acc_{par}	Tok_{peak}	Acc_{peak}
TodyComm	<50%	14.00	69.94	47.16	72.11	1.32	87.74	5.81	88.52	2.91	81.83	20.51	84.83	7.81	59.67	39.38	61.00
	=50%	9.14	66.01	54.47	68.85	12.52	82.72	24.36	85.95	5.55	80.50	12.47	84.17	16.39	58.83	23.55	60.17
	>50%	17.68	64.05	34.75	64.71	25.90	78.48	30.59	81.05	10.10	78.50	12.76	80.50	12.21	56.00	19.47	57.50
GDesigner	<50%	-	-	30.09	64.92	-	-	21.48	83.05	-	-	15.79	73.67	-	-	23.59	67.00
	=50%	-	-	31.39	52.07	-	-	2.21	72.91	-	-	20.96	40.83	-	-	33.05	48.33
	>50%	-	-	22.33	45.97	-	-	26.59	67.22	-	-	26.71	41.00	-	-	32.88	43.17
AgentPrune	<50%	-	-	40.90	69.28	-	-	20.93	86.73	-	-	21.19	73.83	-	-	33.02	66.65
	=50%	-	-	36.80	55.34	-	-	24.15	74.02	-	-	14.29	44.17	-	-	32.70	48.50
	>50%	-	-	51.33	53.38	-	-	19.88	69.79	-	-	8.60	39.17	-	-	22.86	45.67

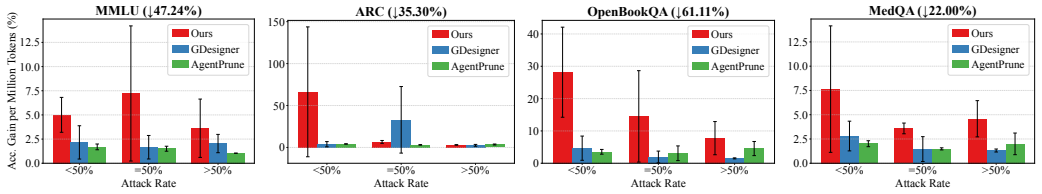

Figure 4: Accuracy gain per million training tokens across four benchmarks. TodyComm achieves higher accuracy with fewer training tokens than GDesigner and AgentPrune across all benchmarks and adversary rates in most cases.

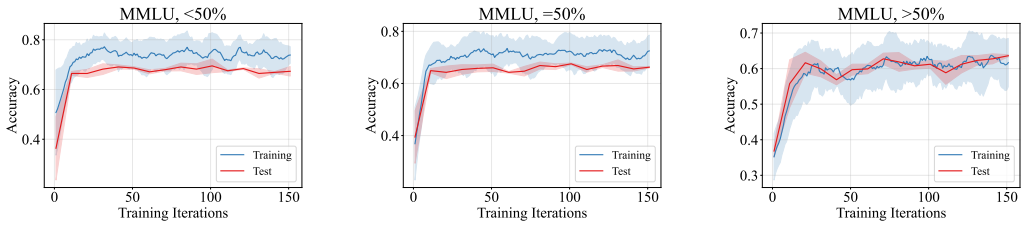

Figure 5: Training vs. test accuracy over training iterations on MMLU under three adversary rates (<50%, =50%, >50%). Training accuracy consistently exceeds test accuracy and both curves plateau stably, confirming no evident overfitting.

C.2 Additional Benchmark Results

C.2.1 Impact of Adversarial Agent

Table 13: Performance degradation under adversarial attack for a single agent

Benchmark	Vanilla	Attacked	Decrease Ratio
MMLU	70.60	37.25	↓47.24%
ARC-C	86.85	56.19	↓35.30%
GSM8K	90.77	60.90	↓32.91%
OpenBookQA	84.00	32.67	↓61.11%
MedQA	65.17	50.83	↓22.00%

Table 14: Task accuracy across benchmarks with varying adversarial counts among 6 agents

Benchmark	**0 adv**	**2 adv**	**3 adv**	**4 adv**
MMLU	76.90	69.03	56.63	45.93
ARC-C	89.30	81.40	75.13	65.70
GSM8K	90.73	70.93	62.40	50.03
OpenBookQA	86.83	76.00	66.00	53.50
MedQA	70.13	61.17	56.33	52.50

C.2.2 Average Performance across five benchmark

Table 15: Average task accuracy and token consumption of TodyComm across five benchmark datasets

Attack rate	**Metric**	**Random Graph**	**Complete Graph**	**Agent Prune**	**GDesigner**	**Ours**
$< 50\%$	Acc	68.95	70.92	77.04	75.17	**79.03**
	Token	606.2	758.8	**383.0**	607.8	409.2
$= 50\%$	Acc	55.38	55.19	60.52	59.05	**77.45**
	Token	635.4	773.0	444.0	643.4	**422.2**
$> 50\%$	Acc	50.70	50.23	55.78	54.07	**73.39**
	Token	625.0	756.6	458.2	636.2	**430.2**

C.2.3 Performance on Scalability

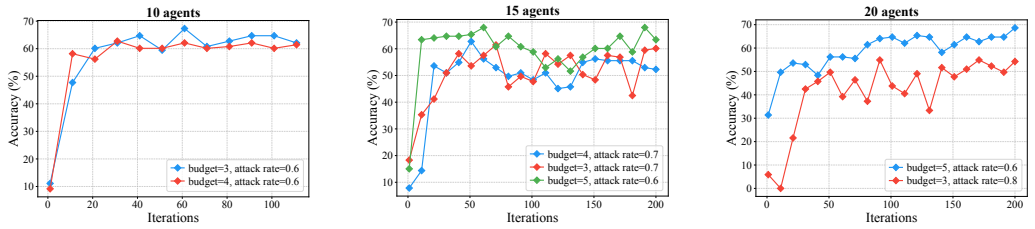

Figure 6: Scalability of TodyComm on MMLU: test accuracy v.s. training iterations.

C.2.4 Performance on Node-wise In & Out degree Budgets

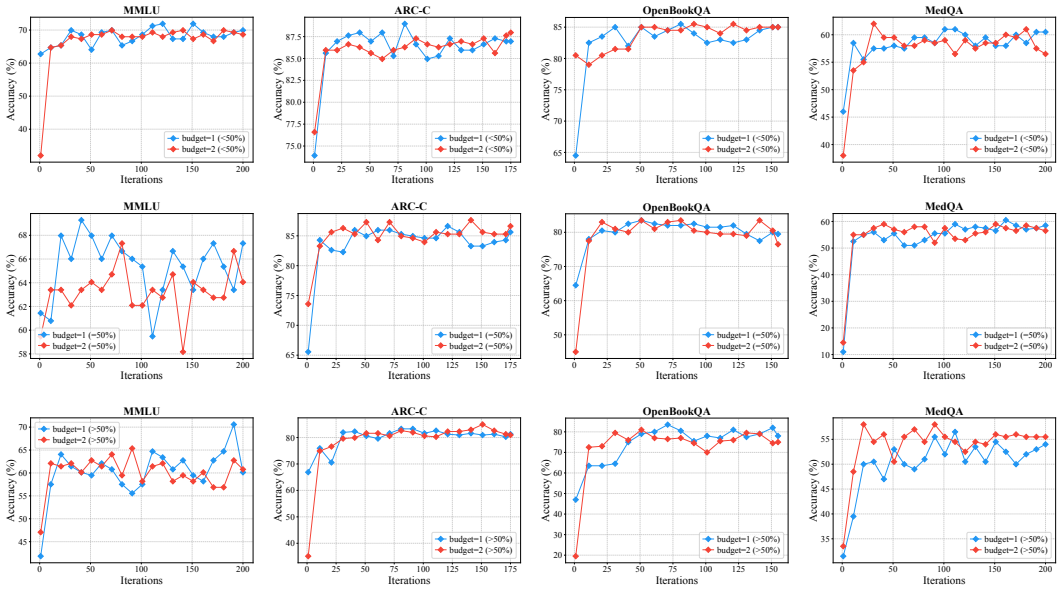

Figure 7: Training curves of task accuracy for TodyComm with node-wise in- and out-degree budget constraints: test accuracy v.s. training iterations. 6 agents under attack rate $< 50\%, = 50\%, > 50\%$.

C.3 Generalization

C.3.1 Number of Agents Generalization

We trained on fewer agents and evaluate on more to demonstrate that our way of learning edges generalize beyond the training configuration, capturing meaningful agent relationships regardless of the number of agents. Table 16 shows TodyComm generalizes from 4-agent training to 6-agent evaluation with negligible degradation on MMLU and marginal improvement on ARC-C, consistently outperforming baselines across all attack rates.

Table 16: Generalization on different number of agents in training and inference with gpt-4.1-nano.

Dataset	**Method**	**#agents (train)**	**#agents (eval)**	$< 50\%$	$= 50\%$	$> 50\%$
MMLU	TodyComm	6	6	72.11	68.85	64.71
MMLU	TodyComm	4	6	71.24	67.32	64.05
MMLU	AgentPrune	6	6	69.28	55.34	53.38
MMLU	GDesigner	6	6	64.92	52.07	45.97
ARC-C	TodyComm	6	6	88.52	85.95	81.05
ARC-C	TodyComm	4	6	88.96	86.62	84.61
ARC-C	AgentPrune	6	6	86.73	74.02	69.79
ARC-C	GDesigner	6	6	83.05	72.91	67.22

C.3.2 TASK DOMAIN GENERALIZATION

We trained on one dataset and evaluate on a different one. Specifically, we use MMLU and ARC-C as evaluation targets, each trained on the remaining QA datasets. As shown in the Table 17, TodyComm retains strong performance across all training-evaluation combinations.

Table 17: Generalization across training and evaluation datasets under different attack rates with gpt-4.1-nano.

Training	**Evaluation**	$< 50\%$	$= 50\%$	$> 50\%$
MMLU	MMLU	72.11	68.85	64.71
ARC-C	MMLU	72.50	66.67	62.50
MedQA	MMLU	69.94	66.67	62.75
OBQA	MMLU	70.59	67.97	63.40
ARC-C	ARC-C	88.52	85.95	81.05
MMLU	ARC-C	85.62	85.28	81.61
MedQA	ARC-C	85.28	85.28	80.27
OBQA	ARC-C	85.28	86.29	81.27

C.3.3 ADVERSARY OUTBREAK GENERALIZATION

We trained with one attack Outbreak modes (*random* or *fixed*) and evaluate with the other, across different dataset combinations. As shown in the Table 18, TodyComm generalizes well across attack Outbreak modes: performance remains consistently strong and comparable to the in-distribution baseline (first and fourth rows). Notably, models trained on the random mode and evaluated on fixed (rows 2 and 5) tend to perform slightly better than the converse (rows 3 and 6), which is expected since random training exposes the model to a more diverse set of adversarial timings, providing a broader inductive signal.

Table 18: Generalization across training and evaluation attack outbreak strategies under different attack rates with gpt-4.1-nano.

Training		**Evaluation**		**Attack Rate**		
Dataset	**Outbreak**	**Dataset**	**Outbreak**	$< 50\%$	$= 50\%$	$> 50\%$
MMLU	random	MMLU	random	72.11	68.85	64.71
ARC-C	random	MMLU	fixed	71.90	69.94	67.32
ARC-C	fixed	MMLU	random	71.24	68.63	60.13
ARC-C	random	ARC-C	random	88.52	85.95	81.05
MMLU	random	ARC-C	fixed	87.62	86.62	84.28
MMLU	fixed	ARC-C	random	87.29	83.61	81.94

C.3.4 ATTACK MECHANISM GENERALIZATION

Table 19 varies the attack mechanism between training and inference, and TodyComm remains competitive across all combinations, sometimes even slightly exceeding the matched-attack baseline.

Table 19: Generalization across attack mechanisms in training and inference with Qwen3-8B.

Training	**Evaluation**	**MMLU**			**ARC-C**		
Mechanism	**Mechanism**	$< 50\%$	$= 50\%$	$> 50\%$	$< 50\%$	$= 50\%$	$> 50\%$
targeted	targeted	69.28	67.32	60.13	88.29	81.94	79.56
untargeted	targeted	69.94	65.36	49.67	87.63	84.95	76.92
targeted	untargeted	67.32	65.36	55.57	88.63	86.29	75.92
untargeted	untargeted	67.97	67.97	59.48	88.29	83.61	74.58

C.3.5 COMBINATION

We further stress-tested generalization by combining multiple factors simultaneously. Table 20 compounds three factors:task domain, attack outbreak mode, and attack mechanism, and TodyComm continues to generalize well, with accuracy remaining strong even under fully mismatched

train/eval configurations. Table 21 adds attack rate as a fourth factor, creating the most challenging cross-condition setting. TodyComm still achieves solid performance in most cases, though accuracy drops more noticeably when the inference attack rate is substantially higher than training (e.g., trained at 0.4, evaluated at 0.7) and trained on fixed outbreak and evaluated on random, as the model encounters a more unpredictable adversarial pattern at a higher attack intensity than seen during training.

Table 20: Generalization across task domain, attack outbreak, and attack mechanism with Qwen3-8B.

Training			Evaluation			Attack Rate		
Dataset	**Mechanism**	**Outbreak**	**Dataset**	**Mechanism**	**Outbreak**	$< 50\%$	$= 50\%$	$> 50\%$
ARC-C	targeted	random	MMLU	untargeted	fixed	69.28	67.32	65.36
ARC-C	untargeted	fixed	MMLU	targeted	random	71.90	61.49	56.86
MMLU	targeted	fixed	ARC-C	untargeted	random	86.96	78.93	61.54
MMLU	untargeted	random	ARC-C	targeted	fixed	89.30	89.00	88.29

Table 21: Generalization across task domain, attack outbreak, attack mechanism, and attack rate with Qwen3-8B.

Training				Evaluation				Acc
Dataset	**Mechanism**	**Outbreak**	**Rate**	**Dataset**	**Mechanism**	**Outbreak**	**Rate**	
ARC-C	targeted	random	0.7	MMLU	untargeted	fixed	0.4	70.59
ARC-C	untargeted	fixed	0.7	MMLU	targeted	random	0.4	68.63
ARC-C	targeted	random	0.4	MMLU	untargeted	fixed	0.7	66.01
ARC-C	untargeted	fixed	0.4	MMLU	targeted	random	0.7	45.10
MMLU	untargeted	random	0.7	ARC-C	targeted	fixed	0.4	89.30
MMLU	targeted	fixed	0.7	ARC-C	untargeted	random	0.4	88.29
MMLU	untargeted	random	0.4	ARC-C	targeted	fixed	0.7	84.28
MMLU	targeted	fixed	0.4	ARC-C	untargeted	random	0.7	67.89

C.4 Robustness to LLM Choice

We evaluated TodyComm on three additional LLMs of varying sizes and model families (Qwen-flash, Mistral-Small-24B, and Llama-4-Maverick) to further validate its robustness. As shown in Table 22, TodyComm consistently outperforms G-Designer and AgentPrune across all three models and benchmarks, with the advantage becoming more pronounced at higher attack rates. Combined with the main results on GPT-4.1-nano and the generalization experiments on Qwen3-8B, TodyComm demonstrates strong and consistent robustness across five diverse LLMs.

Table 22: Ablation on different LLMs across benchmarks.

LLM	**Method**	**MMLU**			**ARC**			**OpenBookQA**		
		$<50\%$	$=50\%$	$>50\%$	$<50\%$	$=50\%$	$>50\%$	$<50\%$	$=50\%$	$>50\%$
QWen-flash	TodyComm	78.43	77.78	67.97	93.98	90.30	80.94	93.00	89.00	82.00
	GDesigner	71.24	58.82	49.02	78.93	67.22	61.87	81.00	58.50	47.00
	AgentPrune	77.12	54.90	53.60	92.98	68.90	62.21	88.00	54.50	43.50
Mistral-Small-24B -Instruct-2501	TodyComm	68.62	66.01	57.52	90.30	85.23	76.92	89.50	84.00	79.00
	GDesigner	59.48	56.21	43.79	79.26	76.59	71.57	69.00	64.50	55.00
	AgentPrune	69.94	50.33	48.37	89.63	78.93	70.57	82.00	47.00	55.00
Llama-4-Maverick -Instruct(17B×128E)	TodyComm	88.24	84.97	79.09	93.30	92.31	89.96	92.96	90.01	87.00
	GDesigner	88.89	80.05	72.55	91.30	86.29	83.28	93.50	84.50	74.50
	AgentPrune	86.93	81.05	76.74	90.30	83.61	81.27	90.00	80.09	70.50

C.5 Robustness to Embedding Choice

Both GDesigner and TodyComm employ sentence embedding models. TodyComm achieves the best performance with all-MiniLM-L6-v2 (the main setting). Switching to all-mpnet-base-v2 or

the lower-dimensional nomic-embed-text-v1.5 leads to a small degradation, yet TodyComm under all embedding configurations still consistently outperforms GDesigner and surpasses AgentPrune at attack rates $\geq$50%, demonstrating robustness to the choice of embedding model. The results are shown in Table 23.

Table 23: Ablation on sentence embedding models and hidden dimensions on MMLU. Embedding dimension is the output dimension of the sentence embedding model. Hidden dimension is the hidden layer size of the neural network (GRU for TodyComm, GCN for GDesigner). Δ: difference from the main result (all-MiniLM-L6-v2, hidden dim 128, embedding dim 384).

Method	Embedding Model	Emb Dim	Hidden Dim	Rate	Acc.	Δ
TodyComm	all-mpnet-base-v2	768	128	<50%	70.59	↓1.52
				=50%	66.01	↓2.84
				>50%	62.00	↓2.71
	all-mpnet-base-v2	768	256	<50%	66.68	↓5.43
				=50%	66.01	↓2.84
				>50%	54.90	↓9.81
	nomic-embed-text-v1.5	256	128	<50%	65.36	↓6.75
				=50%	62.09	↓6.76
				>50%	58.17	↓6.54
GDesigner	all-mpnet-base-v2	768	128	<50%	66.67	↑1.75
				=50%	54.23	↑2.16
				>50%	49.67	↑3.70
	nomic-embed-text-v1.5	256	128	<50%	62.75	↓2.17
				=50%	54.90	↑2.83
				>50%	49.02	↑3.05

C.6 Auxiliary Comparison with G-Safeguard

We provide an auxiliary comparison with G-Safeguard on adversarial-agent identification. G-Safeguard uses explicit adversary supervision, whereas TodyComm's detection signal is derived post hoc from node potentials learned solely through task reward. G-Safeguard exhibits a persistent gap between training and held-out adversary detection accuracy (ADAcc), suggesting limited generalization to unseen behavioral transitions. TodyComm achieves ADAcc values of 84.15%, 88.89%, and 87.96% under $< 50\%$, $= 50\%$, and $> 50\%$ attack rates, respectively. These results indicate that TodyComm's node potentials encode substantial information about evolving agent reliability without explicit reliability supervision.

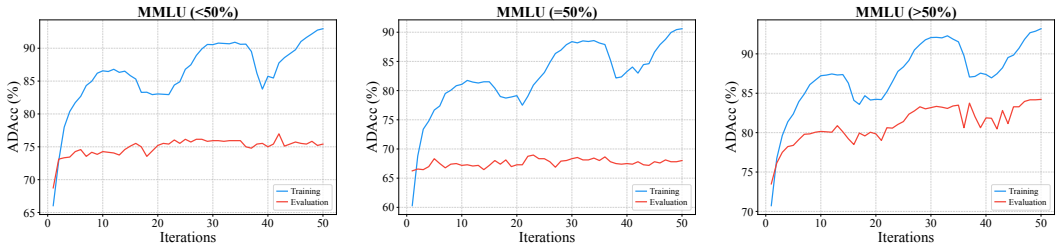

Figure 8: G-Safeguard: learning process in dynamic adversarial setting on MMLU under different attack rates. The blue "Training" is evaluated on training data, hence the overfitting.

D Ablations and Qualitative Analysis

D.1 Graph-Construction Ablation

Table 24: Graph Construction Ablation on MMLU

Method	$< 50\%$	$= 50\%$	$> 50\%$
TodyComm	**72.11**	**68.85**	**64.71**
TodyComm w/o learning	62.11	54.47	49.70
TodyComm w random ordering	69.94	66.01	62.75

D.2 Directed-Edge Scoring

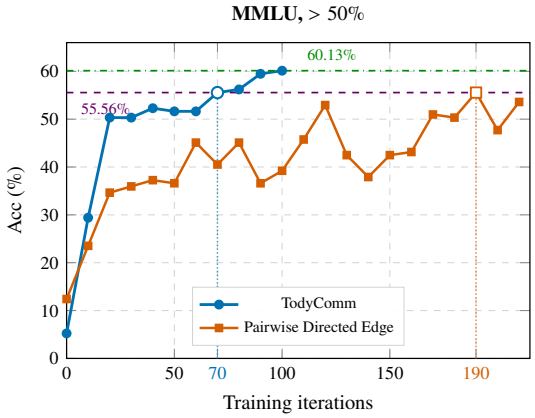

Figure 9: Convergence under more than 50% adversarial agents on MMLU. TodyComm reaches 55.56% accuracy after 70 training iterations, whereas pairwise directed-edge scoring requires 190 iterations. TodyComm reaches 60.13% accuracy after 100 iterations.

D.3 Node-Feature Ablation

In Section 4.1, node features consist of **self** information $\mathbf{s}_i^t$, **neighborhood** information $\mathbf{n}_i^t$, and their **difference** $\mathbf{d}_i^t$.

Table 25 shows that each component is necessary. For example, "Self + Diff" emphasizes the importance of neighborhood information, while "Self + Neighbor" demonstrates the value of explicitly modeling difference information.

We define agent persistence (part of $\mathbf{s}_i^t$) as the difference between an agent's independent answer at the outset and its current-round answer after communication. To evaluate its role, we consider "Self info only", which includes persistence but excludes neighborhood information, and "w/o Persist", which removes persistence. The results indicate that persistence is beneficial, but persistence alone without neighborhood information is insufficient.

As shown in Figure 10, node features combining persistence with neighborhood-related information—via either difference or explicit neighborhood features—converge rapidly within about 20 training iterations. In contrast, "Self info only" converges much more slowly, and persistence without neighborhood information performs poorly, particularly when 4 out of 6 agents are adversarial. Without persistence, the model can still learn but converges more slowly.

Overall, across different node feature designs, our method performs comparably to baselines when the attack rate is below 50% and significantly outperforms them when the attack rate exceeds 50%.

Table 25: Node Features Ablation on MMLU. "lr" stands for learning rate.

Method	$< 50\%$	$= 50\%$	$> 50\%$
TodyComm	**72.11**	**68.85**	64.71
Self + Diff info (lr = $5e^{-4}$)	70.81	67.76	**65.14**
Self + Diff info (lr = $1e^{-4}$)	**72.11**	68.19	62.53
Self + Neighbor info	70.81	67.54	62.31
Self info only	68.30	63.73	41.18
TodyComm w/o Persist	67.65	63.40	53.27

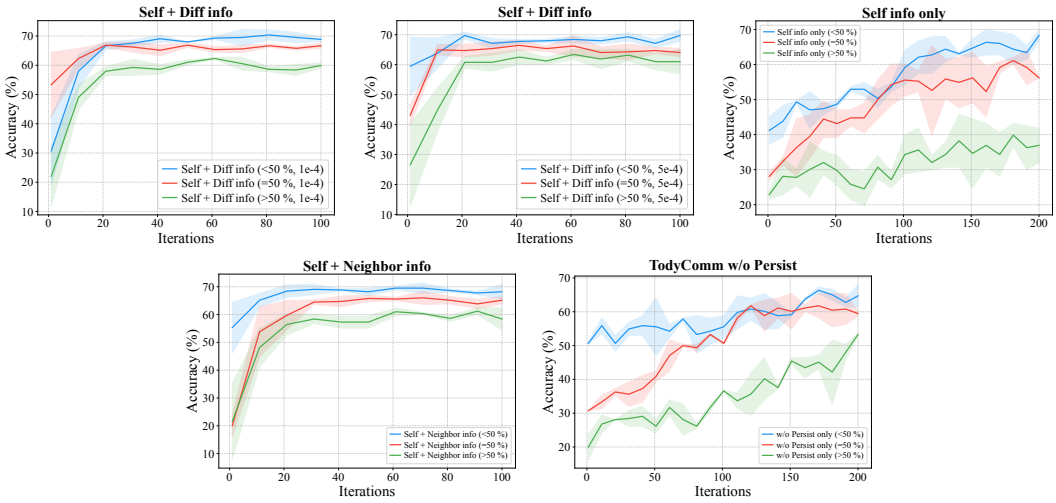

Figure 10: Training curves of node feature ablations for TodyComm on MMLU:test accuracy v.s. training iterations. $5e^{-4}$ and $1e^{-4}$ in the legend refer to the learning rate of REINFORCE.

D.4 Case Study: Communication Graphs Across Rounds

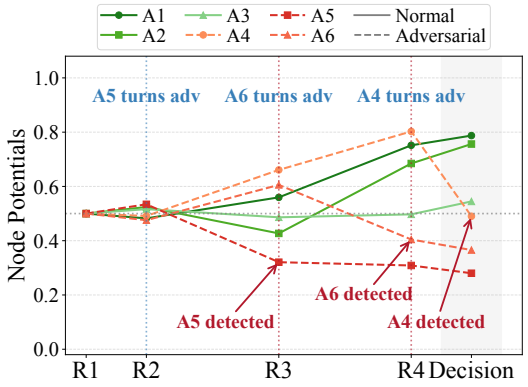

Figure 11: Visualization of the node potential dynamics. Six agents interact over four rounds under a budget constraint of 2.

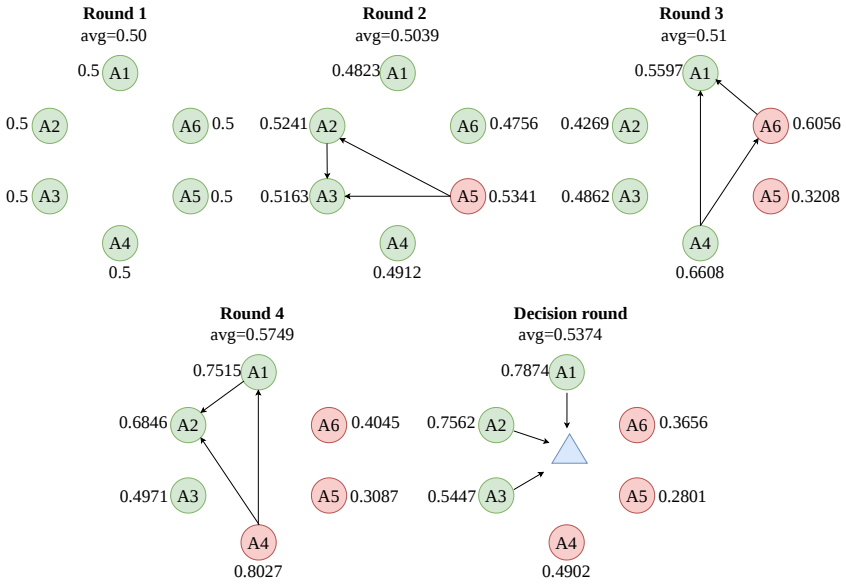

Figure 12: Visualization of graph changing process. Six agents interact over four rounds under a budget constraint of 2.

E Theoretical Analysis and Interpretation

E.1 Constrained Graph Construction

Define the maximum feasible edge budget $B_t := \max_{G \in \mathcal{G}_a} |E(G)|$. Denote $s_{ij}^t = c_i^t c_j^t$, and $\varepsilon_t := 2\alpha_t + \alpha_t^2 + \nu_t$.

Lemma 1 (Product-score perturbation). *Under Assumptions 2 and 3, with probability at least* $1-\delta_t$,

$$\max_{(i,j)\in\Omega_t} \left| c_i^t c_j^t - \Delta_{ij}^t \right| \le \varepsilon_t. \tag{11}$$

Proof. Fix a candidate edge $(i,j) \in \Omega_t$. Write

$$c_i^t = u_i^t + e_i^t, \qquad c_j^t = u_j^t + e_j^t, \tag{12}$$

where $|e_i^t| \le \alpha_t$ and $|e_j^t| \le \alpha_t$ under Assumption 3. Expanding the product gives

$$c_i^t c_j^t = (u_i^t + e_i^t)(u_j^t + e_j^t) = u_i^t u_j^t + u_i^t e_j^t + u_j^t e_i^t + e_i^t e_j^t. \tag{13}$$

Since $u_i^t, u_j^t \in [0,1]$,

$$\left| c_i^t c_j^t - u_i^t u_j^t \right| \le \alpha_t + \alpha_t + \alpha_t^2 = 2\alpha_t + \alpha_t^2. \tag{14}$$

By Assumption 2, $|\Delta_{ij}^t - u_i^t u_j^t| \le \nu_t$. Therefore,

$$\left| c_i^t c_j^t - \Delta_{ij}^t \right| \le 2\alpha_t + \alpha_t^2 + \nu_t. \tag{15}$$

Taking the maximum over all candidate edges proves the claim. □

E.1.1 Main Graph-Learning Guarantee

Proof of Theorem 1. By Lemma 1, with probability at least $1-\delta_t$, $\max_{(i,j)\in\Omega_t} |c_i^t c_j^t - \Delta_{ij}^t| \le \varepsilon_t$. Let us condition on this event. For any feasible graph G, since $|E(G)| \le B_t$,

$$\left| \sum_{(i,j)\in E(G)} c_i^t c_j^t - \sum_{(i,j)\in E(G)} \Delta_{ij}^t \right| \le B_t \varepsilon_t. \tag{16}$$

Let

$$G^A_{\Delta,t} \in \arg\max_{G\in\mathcal{G}_a(\widehat{A}_t)} \sum_{(i,j)\in E(G)} \Delta^t_{ij} \tag{17}$$

be the best additive edge-influence graph available under the candidate node set $\widehat{A}_t$. Because $\widehat{G}_t$ maximizes the learned score objective over the same feasible family $\mathcal{G}_a(\widehat{A}_t)$, $\sum_{(i,j)\in E(\widehat{G}_t)} c^t_i c^t_j \geq \sum_{(i,j)\in E(G^A_{\Delta,t})} c^t_i c^t_j$. Using Equation (16) for both $\widehat{G}_t$ and $G^A_{\Delta,t}$ gives

$$\sum_{(i,j)\in E(G^A_{\Delta,t})} \Delta^t_{ij} - \sum_{(i,j)\in E(\widehat{G}_t)} \Delta^t_{ij} \leq \sum_{(i,j)\in E(G^A_{\Delta,t})} c^t_i c^t_j + B_t\varepsilon_t - \sum_{(i,j)\in E(\widehat{G}_t)} c^t_i c^t_j + B_t\varepsilon_t \tag{18}$$

$$\leq 2B_t\varepsilon_t. \tag{19}$$

By definition of $\Pi_t(\widehat{A}_t)$,

$$\max_{G\in\mathcal{G}_a([N])} \sum_{(i,j)\in E(G)} \Delta^t_{ij} - \sum_{(i,j)\in E(\widehat{G}_t)} \Delta^t_{ij} \leq \Pi_t(\widehat{A}_t) + 2B_t\varepsilon_t. \tag{20}$$

Finally, by Assumption 1, for every feasible graph G,

$$F_t(G) = F_t(\emptyset) + \sum_{(i,j)\in E(G)} \Delta^t_{ij} + \rho_t(G), \qquad |\rho_t(G)| \leq \beta_t. \tag{21}$$

Since $G^\star_{F,t}$ is utility-optimal,

$$F_t(G^\star_{F,t}) \leq F_t(\emptyset) + \max_{G\in\mathcal{G}_a([N])} \sum_{(i,j)\in E(G)} \Delta^t_{ij} + \beta_t, \tag{22}$$

while

$$F_t(\widehat{G}_t) \geq F_t(\emptyset) + \sum_{(i,j)\in E(\widehat{G}_t)} \Delta^t_{ij} - \beta_t. \tag{23}$$

Subtracting the two inequalities and applying Equation (20) yields

$$F_t(G^\star_{F,t}) - F_t(\widehat{G}_t) \leq \Pi_t(\widehat{A}_t) + 2B_t\varepsilon_t + 2\beta_t. \tag{24}$$

Noting $\epsilon^{\text{greedy}}_t = \max(0, F_t(\widehat{G}_t) - F_t(\widetilde{G}_t))$ proves the theorem. □

E.1.2 Consequences of Screening

Corollary 1 (False-positive is mostly absorbed by graph projection)**.** *Suppose the candidate node set $\widehat{A}_t$ contains every endpoint used by the additive edge-influence oracle $G^\star_{\Delta,t}$, i.e.,*

$$V(G^\star_{\Delta,t}) \subseteq \widehat{A}_t. \tag{25}$$

Then $\Pi_t(\widehat{A}_t) = 0$. Consequently, under the assumptions of Theorem 1, with probability at least $1-\delta_t$,

$$F_t(G^\star_{F,t}) - F_t(\widehat{G}_t) \leq 2B_t\varepsilon_t + 2\beta_t. \tag{26}$$

In particular, the bound does not scale with the number of extra nodes included by screening.

Proof. If $V(G^\star_{\Delta,t}) \subseteq \widehat{A}_t$, then the additive oracle graph is feasible inside $\mathcal{G}_a(\widehat{A}_t)$. Therefore the best additive graph over $\widehat{A}_t$ has the same value as the best additive graph over all agents, so $\Pi_t(\widehat{A}_t) = 0$. The result follows from Theorem 1. □

Remark 1 (Interpretation of false positives)**.** *If screening is too conservative, extra agents enter the candidate set. However, they do not automatically enter the communication graph. They must still win edge competitions under the product score $c^t_i c^t_j$, and they must satisfy the DAG and degree-budget constraints. A wrongly included node can occupy at most B_{out} outgoing and B_{in} incoming edges. Without these budgets, a false-positive node could contaminate many more communication paths.*

Corollary 2 (Loss from false negative). *Let $M_t := [N] \setminus \widehat{A}_t$ be the set of nodes excluded by screening. Then*

$$\Pi_t(\widehat{A}_t) \le \sum_{(i,j)\in E(G^\star_{\Delta,t}):\ i\in M_t \text{ or } j\in M_t} \max\{0, \Delta^t_{ij}\}. \tag{27}$$

If, in addition, Assumption 2 holds, then

$$\Pi_t(\widehat{A}_t) \le (B_{\text{out}} + B_{\text{in}}) \sum_{i\in M_t} (u^t_i + \nu_t). \tag{28}$$

Proof. Start from the additive oracle graph $G^\star_{\Delta,t}$ and delete every edge incident to an excluded node in M_t. The remaining graph is feasible in $\mathcal{G}_a(\widehat{A}_t)$, because deleting edges preserves the edge mask, the DAG constraint, and the degree budgets. Hence the best graph available on $\widehat{A}_t$ loses no more additive value than the oracle edges incident to excluded nodes. This proves Equation (27).

For the budgeted bound, Assumption 2 gives

$$\Delta^t_{ij} \le u^t_i u^t_j + \nu_t \le u^t_i + \nu_t \tag{29}$$

for every edge outgoing from i, and similarly

$$\Delta^t_{ji} \le u^t_j u^t_i + \nu_t \le u^t_i + \nu_t \tag{30}$$

for every edge incoming to i. Since node i has at most B_{out} outgoing and B_{in} incoming oracle edges, the total oracle value lost because node i is excluded is at most

$$(B_{\text{out}} + B_{\text{in}})(u^t_i + \nu_t). \tag{31}$$

Summing over excluded nodes gives Equation (28). This may double-count edges whose two endpoints are both excluded, but double-counting only makes the upper bound looser and remains valid. □

Remark 2 (Interpretation of false negatives). *Aggressive screening is more dangerous than conservative screening. If a useful node is removed before graph construction, no graph constructor can recover edges incident to that node. However, degree budgets still limit the damage: missing a node removes only its budgeted incoming and outgoing communication opportunities, rather than all $O(N)$ possible incident edges.*

E.2 Assumption and Greedy-Approximation Justifications

We conducted empirical checks of Assumptions 1–2 using Qwen3-8B on 100 MMLU questions with three random seeds. We consider three agents and evaluate all 25 directed acyclic graphs (DAGs), including the empty graph. These experiments assess how well unrestricted directed edge coefficients and symmetric scalar-potential scores approximate the observed graph utilities.

For each question p, graph G, and seed s, the agents produce a final answer through majority voting. Let $U_{G,p,s} \in \{0, 1\}$ indicate whether this answer is correct. We estimate graph utility as

$$\widehat{F}_{G,p} = \frac{1}{3} \sum_{s=1}^{3} U_{G,p,s}, \tag{32}$$

with $\widehat{F}_{\phi,p}$ denoting the utility of the empty graph. Let $\mathcal{G}_a$ denote the set of 24 nonempty DAGs.

E.2.1 Assumption 1: Approximate Edge-Additive Utility

For each question p, we fit six independently parameterized directed edge coefficients, $\Delta^p \in \mathbb{R}^6$, by minimizing the maximum absolute residual across graphs:

$$\widehat{\beta}^p = \min_{\Delta^p\in\mathbb{R}^6} \max_{G\in\mathcal{G}_a} \left| \widehat{F}_{G,p} - \widehat{F}_{\phi,p} - \sum_{(i,j)\in E(G)} \Delta^p_{ij} \right|. \tag{33}$$

The six coefficients correspond to the edges $1 \to 2$, $2 \to 1$, $1 \to 3$, $3 \to 1$, $2 \to 3$, and $3 \to 2$. In particular, Δ_{ij}^p and Δ_{ji}^p are unrestricted and need not be equal. This minimax problem admits a linear-programming formulation by introducing an auxiliary variable that bounds all absolute residuals.

To contextualize the residual magnitude, we define the largest observed topology effect for each question:

$$A^p = \max_{G \in \mathcal{G}_a} \left| \widehat{F}_{G,p} - \widehat{F}_{\phi,p} \right|, \tag{34}$$

and the problem-level normalized residual:

$$\rho_\beta^p = \frac{\widehat{\beta}^p}{A^p + 10^{-5}}. \tag{35}$$

Averaging over the 100 questions gives

$$\widehat{\beta}_{\text{avg}} = \frac{1}{100} \sum_{p=1}^{100} \widehat{\beta}^p = 0.0574, \tag{36}$$

$$\rho_{\beta,\text{avg}} = \frac{1}{100} \sum_{p=1}^{100} \rho_\beta^p = 0.0924. \tag{37}$$

Thus, the average normalized worst-case residual is 9.24% of the corresponding question-level topology-effect scale.

E.2.2 Assumption 2: Symmetric Scalar-Potential Scores

We next restrict each edge coefficient to a product of scalar node potentials. For each question p, we define

$$\mathbf{u}^p = (u_1^p, u_2^p, u_3^p) \in [0,1]^3 \tag{38}$$

and minimize

$$\widehat{\gamma}^p = \min_{\mathbf{u}^p \in [0,1]^3} \max_{G \in \mathcal{G}_a} \left| \widehat{F}_{G,p} - \widehat{F}_{\phi,p} - \sum_{(i,j) \in E(G)} u_i^p u_j^p \right|. \tag{39}$$

The optimization involves three node potentials; the edge products are not independently optimized. Because $u_i^p u_j^p = u_j^p u_i^p$, opposite directions receive identical scores even though the evaluated graphs are directed. Consequently, differences in the observed utility of graphs with the same undirected connections but different orientations contribute to the fitting residual.

Using the same normalization, we define

$$\rho_u^p = \frac{\widehat{\gamma}^p}{A^p + 10^{-5}}. \tag{40}$$

Numerical fitting yields

$$\widehat{\gamma}_{\text{avg}} = \frac{1}{100} \sum_{p=1}^{100} \widehat{\gamma}^p = 0.0903, \tag{41}$$

$$\rho_{u,\text{avg}} = \frac{1}{100} \sum_{p=1}^{100} \rho_u^p = 0.1390. \tag{42}$$

We also remark that it is not unusual to use undirected/symmetric edge influence or scores (or similar concepts) as **an intermediate quantity** in DAG learning in the MAS literature. For example, BlindGuard models the MAS as a directed graph but uses the symmetric cosine similarity s_{ij} in their Eq 6 during representation learning. SafeSieve uses symmetric embedding cosine similarity as one component of its edge-score initialization in their Eq 2.

In TodyComm, the symmetric product represents only learned endpoint compatibility; it does not imply that the constructed graph or true edge influence is symmetric. Algorithm 4 sorts directed candidates by decreasing $c_i^t c_j^t$. Because opposite orientations have the same product score, $i \to j$ is considered before $j \to i$ whenever $c_i^t > c_j^t$, with equal-potential cases resolved by node index. If the higher-to-lower orientation is feasible, it is inserted first and the reverse edge is subsequently rejected by the acyclicity constraint. Thus, the constructor preferentially directs information from higher- to lower-potential agents, while the edge mask and degree constraints may permit the reverse orientation or exclude the connection entirely.

E.2.3 Discussion on Assumption 3

Assumption 3 is a standard approximation condition on the learned scalar surrogate. Under Assumption 2, the task-relevant contribution of each node is summarized by a bounded one-dimensional latent potential u_i^t. TodyComm constructs c_i^t from behavior-dependent representations h_i^t, so that learning the edge-scoring function reduces to learning these scalar potentials rather than an unrestricted pairwise influence matrix. The tolerance α_t absorbs representation, statistical, and optimization errors. Since each round contains finitely many agents, a uniform bound over agents is also a natural finite-system analogue of standard uniform prediction-error guarantees. Importantly, the assumption does not require exact recovery of the latent potentials; it only requires their approximation to the accuracy needed for graph construction.

E.2.4 Discussion on Greedy Optimization Leading to $\epsilon_t^{\text{GREEDY}}$

Exact maximization over $\mathcal{G}_a$ requires searching a combinatorial set of feasible DAGs and is impractical as the number of agents increases. Indeed, finding the maximum-weight acyclic subgraph under nonnegative weights is NP-hard (even without the degree budgets), and it admits a deterministic 1/2-approximation (Guruswami et al., 2008). Most of the related algorithms in Table 1, along with GPTSwarm, CARD, and AgentNet, construct a DAG from some sort of edge score using greedy/sorting-based approaches, including score-based sampling. TodyComm adopts the same spirit.

Table 24 shows that replacing the learned edge ordering with random ordering consistently degrades accuracy across all attack-rate settings: from 72.11% to 69.94% for $< 50\%$, from 68.85% to 66.01% for $= 50\%$, and from 64.71% to 62.75% for $> 50\%$.

Our claim is therefore not that Algorithm 4 is guaranteed to recover the exact optimum, but it provides an empirically effective approximation. Based on it, the outer RL optimization estimates the best parameters c_i^t that allow the inner optimization to recover a competitive graph.

E.3 Dynamic Behavioral Transitions

Dynamic behavioral transitions can arise in a number of important applications:

- Security compromise. An agent may ingest an untrusted webpage, document, API response, tool output, or peer message. Indirect prompt injection or poisoned content can then cause it to promote an incorrect conclusion while appearing cooperative.
- Emergent reliability degradation. Context drift, accumulated reasoning errors, conformity to a confident but incorrect peer, or ambiguous instructions may make an agent unreliable after several rounds, even without malicious intervention.
- Latent incentive misalignment. In open or federated systems, a third-party agent may initially collaborate normally but later steer the group toward an outcome aligned with its private incentives.

These scenarios have different causes but share the same observable pattern: the agent retains its assigned role, while its behavior and communication value change during the interaction. In our experiments, we operationalized this phenomenon through two controlled attack mechanisms:

- **Targeted attack**: the transitioned agent generates plausible but misleading reasoning intended to steer the group toward a specific incorrect answer.

- **Untargeted attack**: The transitioned agent unintentionally disrupts collaboration with being assigned a predetermined incorrect answer scope. We simulate this by excluding the correct answer from its candidate set, resulting in a non-prespecified incorrect response.

TodyComm is **not** given the transition cause, attack type, affected-agent identities, or transition time; **it adapts the communication topology from agents' behavior across rounds**.

E.4 Connection with Multi-Agent Influence Diagrams (MAID)

Benefits: TodyComm can be cleanly formulated as a MAID (Koller & Milch, 2003), and doing so actually clarifies what the algorithm is really optimizing: a structured, multi-agent decision problem over time. So we can reinterpret TodyComm as a dynamic, partially observable MAID (unrolled over rounds) with repeated decision nodes and evolving information. Here, agents do not directly choose task response (e.g., Q&A), but instead choose communication structure.

We summarize the correspondence of the components as follows:

MAID element	TodyComm
Chance nodes	query, agent outputs, adversarial states, LLM randomness
Decision nodes	communication edges ($\mathcal{E}_C^t$), decision graph
Utility nodes	final task utility
Information sets	past outputs, neighbors, hidden states (via GRN)

1: Chance nodes These represent uncertainty, environment, and stochasticity. In TodyComm, we have

- Query/task: q
- Agent stochastic outputs: $o_i^t = (\text{sol}_i^t, \text{ana}_i^t)$
- Adversarial behavior: whether an agent becomes adversarial at round t
- LLM output with randomness and generation noise
- The node potential c_i^t which act as latent state variables that summarize history. In MAID terms, it mediates between past observations and decisions.

2: Decision nodes These are choices controlled by the policy. In TodyComm, we have

- At each round, which edges exist: $\mathcal{E}_C^t$
- At final decision: the decision graph $\mathcal{G}_D$ and the voting weights w_i

3: Utility nodes These encode what the system optimizes In TodyComm, we have the final task performance $U = u_q(\tau)$, which can be accuracy, reward, token cost, etc.

4: Parent structure This is where MAID is particularly relevant: who knows what when making decisions.

- For communication graph construction, it is deterministic given the parents, namely potentials c_i^t, which in turn depends on agents' internal state h_i^{t-1}, node features $f_i^t = (\text{self}, \text{neighbor}, \text{differences})$, and past outputs: $o_i^{<t}$, $o_{neighbors}^{t-1}$.
- For the final decision, the parents include the final outputs o_i^T and the potentials c_i^{T+1}.

5: Strategic relevance MAIDs assume a fixed structure and derive strategic relevance from it. TodyComm, in contrast, learns a policy over communication structures; each induced structure defines a MAID in which strategic relevance can then be computed. Thus, TodyComm does not learn relevance directly, but performs a meta-level optimization over structures that determine relevance. The decision nodes naturally form a chain, and the graph arising from the decision will impact the conditional probability of each agent's output.

6: Decomposition potential In MAID, it allows identification of independent subgraphs and to apply MAID decomposition. This is not so relevant in TodyComm.

7: Equilibrium interpretation Since TodyComm encompasses multiple collaborative agents that learn in a centralized fashion, no Nash equilibrium is applicable. But it could be extended to multi-agent strategic equilibrium when multiple optimization criteria are considered.