TOWARD ROBUST LLM-BASED MULTI-AGENT SYSTEMS IN COMMUNICATION-LIMITED ENVIRONMENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

The success of LLM-based Multi-Agent Systems (LLM-MAS) relies on reliable communication among agents. However, real world network fluctuations, execution timeouts, and API limits can cause unpredictable agent dropouts, removing agents and their associated communication links and thereby disrupting the communication topology. Existing studies have considered communication optimization and agent dropout resilience, but typically do not jointly address topology reconstruction according to the realized dropout pattern and the use of recovery history for subsequent adaptation. We investigate the impact of agent dropout on LLM-MAS and find that both spatial topology structure and temporal recovery dependence are critical for maintaining effective coordination. Motivated by these findings, we propose RobMAS, a two stage spatiotemporal reinforcement learning framework for adaptive communication recovery under agent dropout. In the spatial stage, graph theoretic rewards guide topology reconstruction toward sparse and connected communication DAGs. In the temporal stage, the Temporal State Recurrent Adapter (TSRA) maintains task relevant recovery history through recurrent node states, allowing previous recovery information to condition subsequent topology decisions. Together, spatial structural guidance and temporal recovery states are integrated within a unified recovery MDP, forming a closed loop self healing process for disrupted communication. Extensive experiments across three distinct domains demonstrate that, under a 40% random agent dropout rate, RobMAS maintains task accuracy close to its no dropout performance, with only 0.8%–3.0% degradation under static and dynamic dropout settings.

1 INTRODUCTION

Large Language Model (LLM)-based Multi-Agent Systems (MAS) have demonstrated strong performance across diverse domains, including automated software and data engineering (Hong et al., 2024; Chen et al., 2023; Du et al., 2026), complex mathematical reasoning (Liang et al., 2024; Du et al., 2023), interactive social simulation (Park et al., 2023), and autonomous scientific discovery (M. Bran et al., 2024; Lu et al., 2024). Central to effective multi-agent collaboration is the communication topology, which determines how information is propagated, aggregated, and coordinated across agents. Recent methods such as GPTSwarm (Zhuge et al., 2024), G-Designer (Zhang et al., 2024a), and AgentPrune (Zhang et al., 2025) optimize communication structures for task performance and efficiency. However, these methods generally assume that the participating agents remain available throughout execution. In real world distributed deployments, this assumption can be violated by API rate limits, network timeouts, process failures, or other runtime interruptions. Once an agent becomes unavailable, its incident communication links disappear, potentially fragmenting the communication graph and disrupting information flow.

This form of failure differs from commonly the robustness settings considered in most existing LLM-MAS studies. Prior work mainly considers adversarial or noisy agent behaviors, where agents generally remain available but may produce malicious(Yu et al., 2025; Zhang et al., 2024b), erroneous, or unreliable(Huang et al., 2024; Zhou et al., 2026) information. In contrast, agent dropout represents a spontaneous, non-malicious availability failure, in which an agent becomes inaccessible during execution and its communication links are consequently removed. Such structural disruption can isolate surviving agents, break previously available communication paths, and impair subsequent

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.

coordination. This raises a distinct question: how can an LLM-MAS autonomously reconstruct effective communication structures when part of its agents unexpectedly become unavailable?

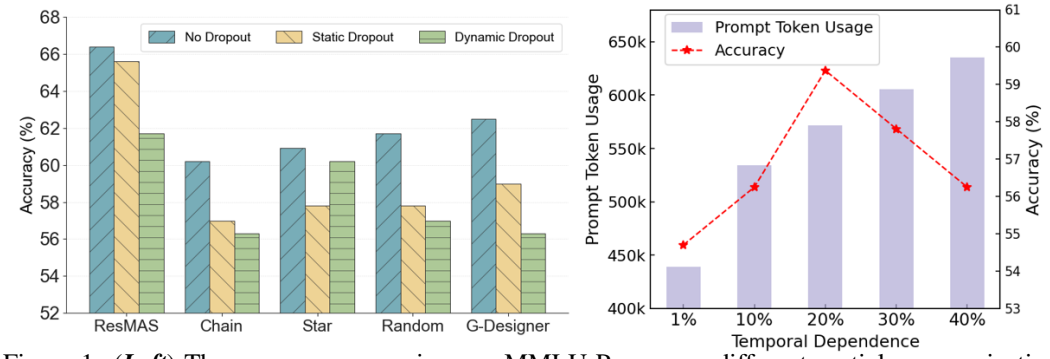


Figure 1: (***Left***) The accuracy comparison on MMLU Pro across different spatial communication topologies under no dropout, static dropout, and dynamic dropout settings. (***Right***) The impact of varying dependence of temporal communication on accuracy and prompt token consumption on MMLU Pro.

Spatial Structure. The spatial communication structure affects performance under agent dropout (Figure 1, ***Left***). Across different topologies, both static and dynamic dropouts generally degrade task accuracy, although the magnitude varies with the underlying structure. Chain-based communication is vulnerable to path disruption, while centralized structures such as stars are sensitive to the availability of critical agents. Even optimized topologies such as ResMAS and G-Designer exhibit noticeable degradation under dropout, particularly in the dynamic setting. These results suggest that a fixed or pre-optimized topology alone is insufficient to accommodate runtime changes in agent availability, motivating adaptive spatial structure recovery after dropout.

Temporal Dependency. Figure 1 (***Right***) further reveals a trade off in temporal information utilization. Temporal dependence denotes the ratio of existing temporal edges to all possible temporal edges under full connectivity. Increasing cross round dependence initially improves task accuracy, but excessive dependence introduces substantially higher prompt token consumption and does not yield further performance gains. This suggests that recovery decisions across communication rounds should neither be independent nor rely on unrestricted historical information. Instead, historical information should be selectively retained, providing empirical motivation for a recurrent state to support subsequent topology recovery.

These findings motivate a two stage recovery framework that first restores communication feasibility and then adapts the recovered structure to task and temporal information. We propose **RobMAS**, a reinforcement learning framework that formulates agent dropout recovery as an MDP and sequentially reconstructs the disrupted communication graph based on the surviving topology and agent availability. In spatial stage, RobMAS learns a spatial structural policy that recovers a sparse and well connected communication DAG, using graph theoretic rewards based on the directed spectral gap ratio and maximum betweenness centrality to improve connectivity while suppressing structural bottlenecks. Building on this recovered structure, temporal stage further refines the policy through the Temporal State Recurrent Adapter (TSRA), which maintains task relevant recovery states across graph editing steps and communication rounds and uses them to guide subsequent edge decisions. Structural rewards provide the recovery prior, while task level feedback further adjusts the policy toward task effective communication. Together, the two stages enable RobMAS to autonomously restore communication connectivity and progressively adapt the recovered structure to task specific interaction under dynamic agent dropouts.

Extensive evaluations across three domains demonstrate the effectiveness of RobMAS under both static and dynamic agent dropouts while maintaining token efficiency. Notably, under a 40% random agent dropout rate, RobMAS maintains task accuracy close to its no-dropout performance across all three domains. Our contributions are summarized as follows.

Agent Dropout Formulation. We formulate communication recovery under agent dropout as a sequential decision problem and establish a unified setting covering both static dropout at the task level and dynamic dropout across communication rounds.

Spatial and Temporal Recovery Framework. We develop a unified framework that combines adaptive communication structure recovery with recurrent temporal state modeling, allowing topology reconstruction to exploit both the current surviving graph and task relevant recovery history.

Effective Dropout Recovery. Extensive experiments across three domains show that RobMAS consistently mitigates performance degradation under varying levels of static and dynamic agent dropout. Under a 40% random agent dropout rate, RobMAS limits the absolute accuracy degradation to only 0.8%–3.0% relative to the no-dropout setting across all three domains.

2 PRELIMINARY

2.1 NOTATIONS

Agent Dropouts. We model an MAS as a communication graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ denotes the set of N agents, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the set of directed communication edges. The agents collaborate over K communication rounds. Let $m_i \in \{0, 1\}$ denote the availability of agent v_i , where $m_i = 1$ indicates that the agent is online and $m_i = 0$ indicates that it is offline. The online agent set and its size are

$$\mathcal{V}_+ = \{v_i \in \mathcal{V} \mid m_i = 1\}, \qquad n = |\mathcal{V}_+|. \tag{1}$$

We assume that each agent independently becomes offline with probability p, such that $m_i^k \sim \mathrm{Bernoulli}(1-p)$. When an agent is offline, the corresponding node and all its incident communication edges are removed before topology recovery.

Definition 1 (Static Dropout). Agent availability is sampled once before task execution and remains fixed throughout all K communication rounds, i.e., $m_i^1 = \cdots = m_i^K$.

Definition 2 (Dynamic Dropout). Agent availability is independently resampled at each communication round. Consequently, the online agent set $\mathcal{V}_+^k$ and communication topology may vary across rounds.

Multi-Agent Communication. Each agent is defined as $v_i = \{\mathrm{Base}_i, \mathrm{Role}_i, \mathrm{State}_i, \mathrm{Tool}_i\}$, where the four components denote its underlying LLM, assigned role, internal state, and available tools, respectively. Given a query q, the prompt of agent v_i at round k is constructed from its role, state, and the outputs received from its in neighbors:

$$P_i^k = \mathrm{Prompt}\left(\mathrm{Role}_i, \mathrm{State}_i^k, q, \{o_j^k\}_{j \in \mathcal{N}_{\mathrm{in}}^k(v_i)}\right), \qquad o_i^k = \mathrm{Base}_i(P_i^k), \tag{2}$$

where $\mathcal{N}_{\mathrm{in}}^k(v_i) = \{v_j \mid (v_j, v_i) \in \mathcal{E}\}$ and o_j^k represents information propagated from its in-neighbors within the current round. Agents are executed according to a topological ordering of $\mathcal{G}$, allowing information to propagate along directed edges within each round. After K communication rounds, the outputs of the available agents are aggregated to obtain the final solution $\hat{a} = \mathrm{Aggregate}\left(o_i^K \mid v_i \in \mathcal{V}_+^K\right)$, where $\mathrm{Aggregate}(\cdot)$ can be instantiated as majority voting or LLM based summarization.

Agent Dropout Recovery as an MDP. We formulate agent dropout recovery as an Markov Decision Process (MDP), where a recovery policy reconstructs the communication topology through sequential graph editing. Formally, the recovery process is defined as

$$\mathcal{M}_{\mathrm{rec}} = (\mathcal{S}, \mathcal{U}, P, R, \gamma), \tag{3}$$

where $\mathcal{S}$ denotes the recovery state space containing the current communication topology and agent availability, $\mathcal{U}$ is the graph editing action space, P specifies the state transition induced by each recovery action, R is the recovery reward, and $\gamma \in [0, 1)$ is the discount factor.

At each communication round k, the recovery policy $\pi_\theta(a_t^k \mid s_t^k)$ sequentially modifies the surviving communication graph until a termination action is selected. The recovered topology is then used for multi-agent communication. By viewing recovery decisions across all communication rounds as a continuous decision trajectory indexed by τ, the objective is

$$\theta^* = \arg\max_\theta J(\theta), \qquad J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{\tau=0}^{T-1} \gamma^\tau r_\tau\right], \tag{4}$$

where T denotes the total number of recovery decisions throughout an episode.

2.2 Related Work

Resilience of Multi-Agent Systems. The distributed nature of MAS introduces intrinsic vulnerability to disruptions, categorizing current research into adversarial attacks and operational perturbations. Adversarial studies explore how attacks compromise collective decisions, such as psychological prompt injections in PsySafe (Zhang et al., 2024b), selective fault inductions via SAFI (Mguni et al., 2025), and topology-mediated adversarial propagation characterized in NetSafe (Yu et al., 2024). Conversely, non-adversarial research examines system reliability under operational degradations, including communication constrained priors over lossy channels (Yang et al., 2026) and faulty agent disruptions (Huang et al., 2024), alongside systematic taxonomies diagnosing root causes like error cascades and coordination breakdown (Cemri et al., 2026). In this non adversarial domain, ResMAS (Zhou et al., 2026) explicitly improves resilience to node dropouts through topology and prompt optimization. In contrast, our focus is runtime communication recovery after dropout, where the recovery policy reconstructs the surviving graph according to current agent availability and further uses previous recovery states to guide subsequent topology decisions across communication rounds.

Communication Topologies in Multi-Agent Systems. MAS communication topology design has evolved from static graph modeling toward granular dynamic coordination. Early work established collaboration on optimizable graphs, notably GPTSwarm (Zhuge et al., 2024). To curb token costs and noise cascading, sparsification methods like AgentPrune (Zhang et al., 2025) and AgentDropout (Wang et al., 2025) prune redundant paths, while EIB-Learner (Shen et al., 2025) balances noise suppression with information diffusion. The paradigm then advanced toward task-adaptive topology generation, leveraging GNNs in G-Designer (Zhang et al., 2024a) alongside dynamic directed and group-level graph synthesis in GoA (Yun et al., 2026) and GoAgent (Chen et al., 2026), before progressing to the joint co-design of agent roles and connectivity via ARG-Designer (Li et al., 2026) and self-configuring MAS² (Wang et al., 2026). Despite these advances, existing topology optimization methods primarily design or optimize communication structures for task performance and efficiency. They generally do not explicitly reconstruct the surviving communication graph according to the realized dropout pattern during execution, nor do they maintain recovery states across communication rounds to guide subsequent topology adaptation.

3 Method

Figure 2 illustrates the overall framework of RobMAS for communication recovery under agent dropout. RobMAS first restores the spatial structure toward a sparse and connected communication DAG, and then uses temporal state adaptation to preserve task relevant recovery history for subsequent recovery decisions. In the following sections, we detail the spatial recovery mechanism in Section 3.1, followed by temporal state adaptation and policy optimization in Section 3.2.

3.1 Stage 1: Spatial Structural Recovery

Topology State Encoding. Let A denote the adjacency matrix of $\mathcal{G}$. Each agent i is represented by its normalized in-degree, out-degree, and availability state as

$$X_i = \left[\frac{d_i^{\text{in}}}{N-1}, \frac{d_i^{\text{out}}}{N-1}, m_i \right]. \tag{5}$$

Here, N is the total number of agents before dropout, and d_i^{in} and d_i^{out} are measured on $\mathcal{G}$. The node features are encoded by a two-layer GNN as $Z = \text{GNN}(\widehat{A}, X) \odot \mathbf{m}$, where $\widehat{A}$ is the normalized adjacency matrix. The mask is broadcast along the feature dimension so that the representation of each offline agent is set to zero.

Sequential Graph Recovery. Stage 1 formulates topology reconstruction as a sequence of graph-editing decisions. At editing step t, the recovery state $s \in \mathcal{S}$ consists of the current graph $\mathcal{G}$ and its encoded node representations Z. The action space $\mathcal{U}$ contains all candidate directed edges between online agents together with a `STOP` action. Let z_i denote the representation of agent i, and let $\bar{z}$ be the mean representation of all agents. For each candidate edge action $a_{ij} = (i, j) \in \mathcal{U}$, its action value is computed as

$$Q_{\text{base}}(a_{ij} \mid s) = f_{\text{edge}}([z_i, z_j, \bar{z}]), \tag{6}$$

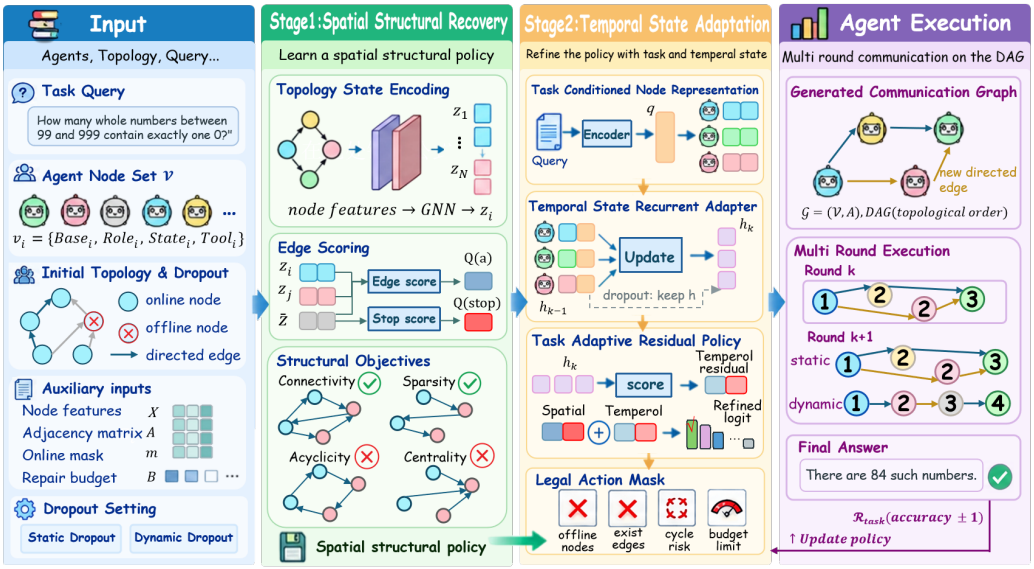


Figure 2: Overview of RobMAS. Given a communication graph disrupted by agent dropout, RobMAS performs two stage recovery. **Stage 1** learns a spatial structural policy to recover a sparse and connected DAG. **Stage 2** refines the policy with task and recurrent temporal states. The recovered graph supports multi round execution, with the final spatial and task rewards fed back to optimize recovery.

while the STOP action is scored by $Q_{\text{base}}(\text{STOP} \mid s) = f_{\text{stop}}(\bar{z})$. Invalid edges, including those incident to offline agents, existing edges, and cycle-inducing edges, are filtered out before action selection. The policy therefore selects the highest valued valid edge or STOP through $\arg\max$.

Structural Objectives. Let $\mathcal{G}_{\text{rec}} = (\mathcal{V}_+, \mathcal{E}_{\text{rec}})$ denote the recovered directed graph, where $n = |\mathcal{V}_+|$. Let $\mathcal{G}_{\text{sym}}$ be its symmetrized projection, with normalized Laplacian L_{sym}, and let $d^{\text{in}}_{\max} = \max_{v \in \mathcal{V}_+} d^{\text{in}}_v$.

Directed spectral ratio. We define

$$S_{\text{spec}}(\mathcal{G}_{\text{rec}}) = \frac{\lambda_2(L_{\text{sym}})}{d^{\text{in}}_{\max}}. \tag{7}$$

This score favors weakly connected recovered graphs without narrow communication bottlenecks and discourages excessive maximum in-degree. A larger directed spectral ratio favors sparse, globally connected communication structures that avoid heavy centralized information aggregation.

Maximum betweenness centrality. For $n \geq 3$, the normalized directed betweenness of an online agent $v \in \mathcal{V}_+$ is

$$B(v) = \frac{1}{(n-1)(n-2)} \sum_{\substack{s,t \in \mathcal{V}_+ \setminus \{v\} \\ s \neq t,\ s \rightsquigarrow_{\mathcal{G}_{\text{rec}}} t}} \frac{\sigma_{st}(v)}{\sigma_{st}}, \qquad B_{\max}(\mathcal{G}_{\text{rec}}) = \max_{v \in \mathcal{V}_+} B(v), \tag{8}$$

where σ_{st} is the number of shortest directed paths from s to t, and $\sigma_{st}(v)$ is the number of those paths that pass through v. A smaller $B_{\max}$ indicates that communication is distributed across multiple routes rather than concentrated on a single critical relay agent.

The topology quality function is defined as

$$\Psi(\mathcal{G}_t) = \lambda_{\text{spec}} S_{\text{spec}}(\mathcal{G}_t) - \lambda_{\text{bet}} B_{\max}(\mathcal{G}_t) + \lambda_{\text{conn}} C(\mathcal{G}_t), \tag{9}$$

where $C(\cdot)$ explicitly captures the ratio of the maximum number of online nodes reachable from any single online node to the total number of online nodes. For a transition from $\mathcal{G}_t$ to $\mathcal{G}_{t+1}$, the topology reward is defined as $R^{\text{spatial}}_t = \Psi(\mathcal{G}_{t+1}) - \Psi(\mathcal{G}_t) - \lambda_{\text{edge}} \mathbb{I}[a_t \neq \text{STOP}]$, where λ_{edge} penalizes additional communication edges. This reward measures the marginal structural improvement of each editing action, encouraging robust and sparse topology recovery with low communication overhead. We require at least two agents to remain online after dropout. When only two agents survive, the maximum betweenness centrality is defined as 0.

3.2 Stage 2: Temporal State Adaptation

For each episode, the task query is encoded into a task representation q, which is integrated with the topology representation $z_{i,t}^k$ produced by Stage 1. We define a task conditioned fusion module $\mathrm{Muse}(\cdot)$ to obtain $u_i = \mathrm{Muse}\,(z_i, q)$. The resulting representation u_i jointly captures the agent representations and task semantics.

Temporal State Recurrent Adapter. To incorporate temporal dependencies, we introduce a Temporal State Recurrent Adapter (TSRA) that maintains task relevant recovery history. TSRA updates the recurrent state as

$$h_i^k = \begin{cases} \mathrm{TSRA}(u_i^k, h_i^{k-1}), & m_i^k = 1, \\ h_i^{k-1}, & m_i^k = 0. \end{cases} \tag{10}$$

Thus, online agents update their temporal states according to the current task conditioned topology representation, whereas unavailable agents retain their previous. The recurrent states are propagated across topology editing steps and communication rounds. We further summarize a graph level temporal representation $\bar{h}$ by averaging the states of online agents.

Task Adaptive Residual Policy. We retain the structural policy learned in Stage 1 by using its action values as a recovery prior, while learning a task dependent residual correction. The recovery state $\tilde{s}$ extends s with the temporal state h and task representation q. For each candidate edge a_{ij}, we predict $\Delta Q_{\mathrm{task}}(a_{ij} \mid \tilde{s}) = f_{\mathrm{taskedge}}\big([h_i, h_j, \bar{h}]\big)$, while the residual score for the termination action is $\Delta Q_{\mathrm{task}}(\texttt{STOP} \mid \tilde{s}) = f_{\mathrm{taskstop}}(\bar{h})$. The final action logits combine the structural prior and the task adaptive residual:

$$\ell(a_{ij}) = \frac{Q_{\mathrm{base}}(a_{ij} \mid s)}{\tau} + \Delta Q_{\mathrm{task}}(a_{ij} \mid \tilde{s}) \tag{11}$$

and $\ell(\texttt{STOP}) = \frac{Q_{\mathrm{base}}(\texttt{STOP}|s)}{\tau} + \Delta Q_{\mathrm{task}}(\texttt{STOP} \mid \tilde{s})$, where τ controls the contribution of the Stage 1 structural prior. This residual formulation preserves the topology recovery capability learned in Stage 1 while allowing the policy to selectively introduce additional task relevant communication edges. We use the same graph editing action space and structural constraints as Stage 1, and apply softmax to the action logits to obtain the recovery policy.

Policy Optimization. Structural feedback is provided after individual graph editing actions, while the task reward is assigned after the final multi-agent output is obtained. The reward is defined as

$$R_t = w_{\mathrm{spatial}} R_t^{\mathrm{spatial}} + w_{\mathrm{task}} \mathbb{I}[t = T] R_{\mathrm{task}}, \tag{12}$$

where $R_{\mathrm{task}} \in +1, -1$ indicates whether the final task output is correct or incorrect and T denotes the terminal topology editing step of the entire episode. The coefficients w_{spatial} and w_{task} control the relative contributions of structural quality and task performance, respectively.

3.3 Theoretical Analysis

Why does spatial recovery favor sparse and robust communication structures?

Theorem 1 (Connectivity and sparsity). Assume $n \geq 2$ and $d_{\max}^{\mathrm{in}} > 0$. If $S_{\mathrm{spec}}(\mathcal{G}_{\mathrm{rec}}) \geq \eta > 0$, then $\mathcal{G}_{\mathrm{sym}}$ is connected and

$$|\mathcal{E}_{\mathrm{rec}}| \leq \frac{2n}{\eta}, \qquad \Phi(\mathcal{G}_{\mathrm{sym}}) \geq \frac{\eta d_{\max}^{\mathrm{in}}}{2} \geq \frac{\eta}{2}. \tag{13}$$

Thus, a positive spectral score ensures connectivity of the symmetrized graph, but not mutual directed reachability. If it is lower bounded by a size independent constant η, the number of communication edges grows at most linearly with the number of agents. The proof is provided in Appendix B.1.

Theorem 2 (Single agent relay vulnerability). Define the normalized reachability loss after removing agent v as

$$\Delta(v) = \frac{\left|\left\{(s,t): \begin{array}{l} s,t \in \mathcal{V}_+ \setminus \{v\},\ s \neq t, \\ s \rightsquigarrow_{\mathcal{G}_{\mathrm{rec}}} t, \quad s \not\rightsquigarrow_{\mathcal{G}_{\mathrm{rec}}-v} t \end{array}\right\}\right|}{(n-1)(n-2)}. \tag{14}$$

Then

$$\Delta(v) \leq B(v), \qquad \max_{v \in \mathcal{V}_+} \Delta(v) \leq B_{\max}(\mathcal{G}_{\text{rec}}). \tag{15}$$

Therefore, minimizing $B_{\max}$ limits the worst case loss of directed communication paths caused by a single agent dropout. The proof is provided in Appendix B.2.

How does temporal adaptation preserve structural preferences while incorporating task information?

Theorem 3 (Bounded Residual Perturbation). Assume that the residual score is bounded over the feasible action set is $\forall a \in \mathcal{F}(s)$, $|\Delta Q_{\text{task}}(a \mid \tilde{s})| \leq \epsilon$. Then, for any two feasible actions $a, b \in \mathcal{F}(s)$, if

$$Q_{\text{base}}(a \mid s) - Q_{\text{base}}(b \mid s) > 2\tau\epsilon, \tag{16}$$

their relative preference is preserved after Stage 2 adaptation, i.e.,

$$\ell(a \mid \tilde{s}) > \ell(b \mid \tilde{s}). \tag{17}$$

Thus, a bounded task temporal residual can only reverse the ordering of structurally close actions, while sufficiently strong structural preferences are preserved. The proof is provided in Appendix B.3.

4 EXPERIMENTS

In this section, we conduct experiments to answer four research questions: (RQ1) **Effectiveness**: How well does RobMAS recover task performance under different agent dropout settings? (RQ2) **Robustness**: How robust is RobMAS to varying dropout rates and LLM backbones? (RQ3) **Efficiency**: Can RobMAS recover communication while maintaining token efficiency? (RQ4) **Explainability**: How do different recovery stages and strategies affect RobMAS performance?

4.1 EXPERIMENTAL SETUP

Tasks and Benchmarks. We evaluate RobMAS on six benchmarks covering three reasoning capabilities. **(1)General Knowledge Reasoning.** We use MMLU-Pro (Wang et al., 2024) and MMLU (Hendrycks et al., 2021a). **(2)Mathematical Reasoning.** We select Level-5 MATH (Hendrycks et al., 2021b), GSM8K (Cobbe et al., 2021) and SVAMP (Patel et al., 2021). **(3)Game Reasoning.** We evaluate Chess (Srivastava et al., 2023). Details are provided in Appendix D.

Baselines. We compare RobMAS with three categories of baselines. **(1)Fixed Topologies.** We consider Chain, Star, and Random as representative static communication structures. **(2)Communication Paradigms.** We include LLM-Debate (Du et al., 2024), Self-Consistency (SC) (Wang et al., 2023), and ReConcile (Chen et al., 2024) as representative multi-agent reasoning and communication paradigms. **(3)Adaptive Topologies.** We compare with G-Designer (Zhang et al., 2024a), ResMAS (Zhou et al., 2026), AgentPrune(Zhang et al., 2025), and EIB-Learner(Shen et al., 2025), which adapt or optimize multi-agent communication structures. Details are provided in Appendix E.

Implementation Details. We primarily use `Qwen2.5-32B-Instruct` as the backbone LLM, with additional experiments on `DeepSeek-V3` and `GPT-4o-mini`. Unless otherwise specified, we use $N = 10$ agents, $K = 3$ dialogue rounds, and an agent dropout probability of $p = 0.4$. Task queries are encoded by `all-MiniLM-L6-v2` ($D = 384$), and final answers are determined by majority voting over the last round responses. Implementation details are provided in Appendix F.

4.2 EFFECTIVENESS(RQ1)

To evaluate the effectiveness and robustness of RobMAS, we compare it with fixed topology, reasoning based, and adaptive communication methods on MMLU-Pro, MATH, and Chess under normal, static dropout, and dynamic dropout settings. The results are summarized in Table 1. **Obs.❶ Consistent gains across reasoning domains.** RobMAS achieves the highest accuracy in all nine dataset-setting combinations, with an average accuracy of 65.30%, exceeding the strongest baseline by 3.92%. **Obs.❷ Stronger performance under agent dropout.** Under static and dynamic

Table 1: Performance comparison under normal (Nor.), static dropout (Sta.), and dynamic dropout (Dyn.) settings. Arrows denote accuracy changes from the normal setting, and Improvement shows the gain over the strongest baseline.

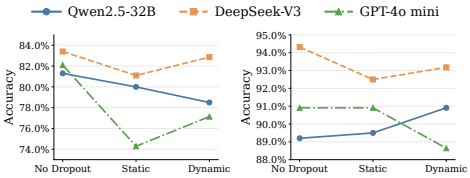
Method	MMLU-Pro			MATH			Chess			
	Nor.	Sta.	Dyn.	Nor.	Sta.	Dyn.	Nor.	Sta.	Dyn.	Avg.
Chain	60.16	57.03$_{\downarrow 3.13}$	56.25$_{\downarrow 3.91}$	64.06	62.50$_{\downarrow 1.56}$	61.72$_{\downarrow 2.34}$	45.45	36.36$_{\downarrow 9.09}$	37.88$_{\downarrow 7.57}$	53.49
Star	60.94	57.81$_{\downarrow 3.13}$	60.16$_{\downarrow 0.78}$	66.41	61.72$_{\downarrow 4.69}$	64.84$_{\downarrow 1.57}$	46.97	51.52$_{\uparrow 4.55}$	43.94$_{\downarrow 3.03}$	57.15
Random	61.72	57.81$_{\downarrow 3.91}$	57.03$_{\downarrow 4.69}$	64.84	60.94$_{\downarrow 3.90}$	61.72$_{\downarrow 3.12}$	53.03	45.45$_{\downarrow 7.58}$	46.97$_{\downarrow 6.06}$	56.61
LLM-Debate	60.16	58.59$_{\downarrow 1.57}$	58.59$_{\downarrow 1.57}$	72.66	69.53$_{\downarrow 3.13}$	69.53$_{\downarrow 3.13}$	43.94	37.88$_{\downarrow 6.06}$	39.39$_{\downarrow 4.55}$	56.70
SC	57.03	57.81$_{\uparrow 0.78}$	52.34$_{\downarrow 4.69}$	64.84	64.06$_{\downarrow 0.78}$	63.28$_{\downarrow 1.56}$	51.52	50.00$_{\downarrow 1.52}$	45.45$_{\downarrow 6.07}$	56.26
ReConcile	59.38	56.25$_{\downarrow 3.13}$	55.47$_{\downarrow 3.91}$	55.47	53.91$_{\downarrow 1.56}$	51.56$_{\downarrow 3.91}$	45.45	39.39$_{\downarrow 6.06}$	42.42$_{\downarrow 3.03}$	51.03
AgentPrune	61.34	57.03$_{\downarrow 4.31}$	57.81$_{\downarrow 3.53}$	69.53	64.84$_{\downarrow 4.69}$	65.62$_{\downarrow 3.91}$	53.03	48.48$_{\downarrow 4.55}$	45.45$_{\downarrow 7.58}$	58.13
EIB-Learner	60.94	57.42$_{\downarrow 3.52}$	57.03$_{\downarrow 3.91}$	60.16	57.03$_{\downarrow 3.13}$	56.25$_{\downarrow 3.91}$	40.91	37.88$_{\downarrow 3.03}$	46.97$_{\uparrow 6.06}$	52.73
G-Designer	62.50	59.38$_{\downarrow 3.12}$	56.25$_{\downarrow 6.25}$	69.53	64.06$_{\downarrow 5.47}$	63.28$_{\downarrow 6.25}$	54.55	50.00$_{\downarrow 4.55}$	51.52$_{\downarrow 3.03}$	59.01
ResMAS	66.41	65.62$_{\downarrow 0.79}$	61.72$_{\downarrow 4.69}$	72.66	71.09$_{\downarrow 1.57}$	68.75$_{\downarrow 3.91}$	53.79	46.97$_{\downarrow 6.82}$	45.45$_{\downarrow 8.34}$	61.38
RobMAS	**68.75**	**67.19**$_{\downarrow 1.56}$	**66.41**$_{\downarrow 2.34}$	**73.44**	**72.66**$_{\downarrow 0.78}$	**71.09**$_{\downarrow 2.35}$	**57.58**	**56.06**$_{\downarrow 1.52}$	**54.55**$_{\downarrow 3.03}$	**65.30**
Improvement	+2.34	+1.57	+4.69	+0.78	+1.57	+1.56	+3.03	+4.54	+3.03	+3.92

dropout, RobMAS consistently maintains higher accuracy than the strongest baselines, with gains of up to 4.69% on MMLU-Pro and 4.54% on Chess. This shows the benefit of adapting the communication structure after agent availability changes. **Obs.❸ Stable performance across dropout patterns.** The degradation from the normal setting remains limited across the three datasets. For example, under static dropout, accuracy decreases by only 0.78% on MATH and 1.52% on Chess, indicating stable performance under varying agent availability.

4.3 Robustness(RQ2)

We evaluate robustness from two perspectives, generalization across backbone LLMs and resilience to different agent dropout rates.

Different Backbone LLMs. We evaluate RobMAS with Qwen2.5-32B-Instruct, DeepSeek-V3, and GPT-4o mini on MMLU and SVAMP with a fixed dropout rate of 40%. As shown in Figure 3, the three backbone models exhibit different sensitivities to communication disruption, yet RobMAS remains effective across all of them. This suggests that the recovery mechanism is not tied to the capability profile of a specific LLM, but can adapt to heterogeneous reasoning behaviors and preserve online agent communication under dropout.



(a) MMLU (b) SVAMP

Figure 3: RobMAS performance across different backbones.

Different Dropout Rates. We further vary the dropout rate from 20% to 80% on MMLU and GSM8K under both static and dynamic dropout, with the dropout sampling constrained to retain at least two online agents in each episode. As shown in Figure 4, RobMAS consistently maintains higher accuracy than Chain and G-Designer, and its advantage becomes more pronounced as the dropout rate increases.

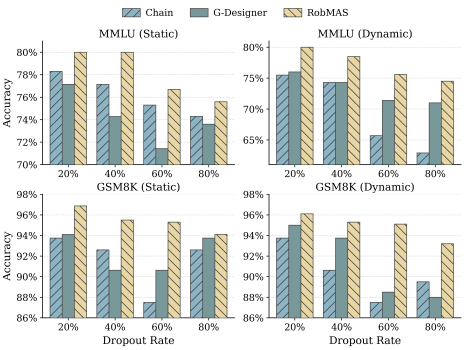


Figure 4: Accuracy under different agent dropout rates in static and dynamic settings.

4.4 Efficiency(RQ3)

For the efficiency evaluation, we use 400 Chess questions with a dynamic dropout rate of $p = 0.4$ and 132 MMLU questions with $p = 0.2$. Table 2 compares accuracy, token consumption, and accuracy per token, where Efficiency Gain denotes the relative improvement in accuracy per token

Table 2: Accuracy and token efficiency comparison between ResMAS and RobMAS.

Dataset	Setting	ResMAS				RobMAS				Efficiency Gain
		Acc.	Tokens ($\times 10^6$)	Tok./Q.	Acc./Token	Acc.	Tokens ($\times 10^6$)	Tok./Q.	Acc./Token	
Chess	No Dropout	53.79	1.785	13,520	0.301	57.58	1.855	14,051	0.310	**3.00%**
	Static	47.73	1.008	7,639	0.473	56.06	1.094	8,291	0.512	**8.23%**
	Dynamic	44.70	0.987	7,481	0.453	54.55	1.093	8,285	0.499	**10.20%**
MMLU	No Dropout	80.75	4.661	11,651	0.173	81.25	4.391	10,979	0.185	**6.81%**
	Static	75.00	3.739	9,347	0.201	77.00	3.515	8,788	0.219	**9.21%**
	Dynamic	78.25	3.715	9,287	0.211	78.50	3.500	8,752	0.224	**6.48%**

over ResMAS. RobMAS achieves higher efficiency in all settings, with gains of 3.00%–10.20% on Chess and 6.48%–9.21% on MMLU. The improvement is more evident under agent dropout, where effective communication becomes more difficult to maintain. These results show that RobMAS improves task performance without relying on proportionally higher communication cost.

4.5 Explainability(RQ4)

Table 3: Ablation results under agent dropout.

Variant	GSM8K		MMLU-Pro	
	Static	Dynamic	Static	Dynamic
w/o Temporal	94.73	95.31	57.03	57.81
w/o Spatial	94.51	93.75	62.50	64.06
w/o Recovery	92.54	92.19	57.03	56.25
Random Recovery	89.06	93.75	64.06	61.72
Greedy Recovery	95.31	94.11	64.06	65.63
RobMAS	**96.88**	**96.88**	**67.19**	**66.41**

To better understand the contribution of different recovery mechanisms in RobMAS, we conduct ablation studies on GSM8K and MMLU-Pro under static and dynamic dropout with $p = 0.4$, and compare the learned policy with alternative recovery strategies. **Obs.❶ Spatial and temporal recovery are complementary.** Removing either stage consistently reduces accuracy, especially on MMLU-Pro. **Obs.❷ Recovery is essential under agent dropout.** Without recovery, RobMAS suffers substantial degradation under both static and dynamic dropout. **Obs.❸ Learned recovery is more effective than heuristic recovery.** Random and greedy recovery consistently underperform RobMAS, indicating that effective recovery requires more than simply adding feasible edges.

4.6 Case Study

Figure 5 illustrates the distinct roles of the two stages. Agent dropout isolates four agents from the original communication structure. Stage 1 restores communication feasibility by constructing the minimum paths for information propagation. Stage 2 then adds two task and temporal state aware edges, enabling A_{10} to aggregate information from A_5, A_7, and A_1. The final topology remains a DAG with limited redundancy, containing only 5 edges for 4 online agents.

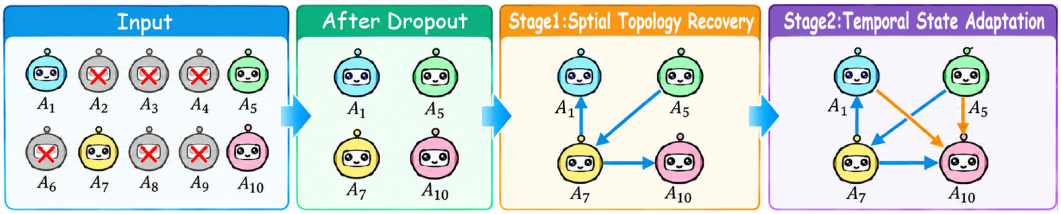


Figure 5: A two stage example of topology recovery with RobMAS under agent dropout.

5 Conclusion

We presented RobMAS, a two stage reinforcement learning framework for robust LLM-based multi agent communication in communication-limited environments. RobMAS combines spatial structural recovery with temporal state adaptation, enabling the communication graph to be reconstructed according to current agent availability while preserving useful recovery history across rounds. Extensive experiments across three reasoning domains show that RobMAS consistently improves robustness under static and dynamic dropout, while maintaining competitive token efficiency and generalizing across different dropout rates and backbone LLMs. Overall, these results demonstrate the importance of adaptive runtime communication recovery for building more resilient LLM-based multi agent systems.

AI USE STATEMENT

No generative AI or large language models were used in any stage of this research or manuscript preparation.

REPRODUCIBILITY STATEMENT

We provide detailed experimental settings and implementation information to facilitate the reproducibility of RobMAS. All comparable experiments use consistent evaluation protocols and fixed random seeds whenever applicable.

- **Method:** The architecture, two-stage recovery procedure, structural objectives, and policy optimization of RobMAS are described in Section 3, with additional implementation details provided in the Appendix.
- **Datasets:** We use publicly available benchmarks, including MMLU-Pro, MATH, Chess, MMLU, GSM8K, and SVAMP. Details provided in the Appendix.
- **Models:** The LLM backbones and corresponding configurations used in our experiments are reported in the experimental setup. The main experiments use Qwen2.5-32B-Instruct, with additional backbone experiments conducted using DeepSeek-V3 and GPT-4o mini.
- **Agent Settings:** We report the number of agents, communication rounds, static and dynamic dropout settings, and dropout probabilities used in all experiments. The same settings are applied to comparable methods for fair evaluation.
- **Hyperparameters:** Training configurations and key hyperparameters for both recovery stages are provided in the Appendix, including the reinforcement learning settings and structural reward coefficients.
- **Baselines:** Baselines are implemented following their original papers and publicly available implementations, under consistent backbone and evaluation settings.

REFERENCES

Mert Cemri, Melissa Z Pan, Shuyi Yang, Lakshya A Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, et al. Why do multi-agent llm systems fail? *Advances in Neural Information Processing Systems*, 38, 2026.

Hongjiang Chen, Xin Zheng, Yixin Liu, Pengfei Jiao, Shiyuan Li, Huan Liu, Zhidong Zhao, Ziqi Xu, Ibrahim Khalil, and Shirui Pan. Goagent: Group-of-agents communication topology generation for llm-based multi-agent systems. *arXiv preprint arXiv:2603.19677*, 2026.

Justin Chen, Swarnadeep Saha, and Mohit Bansal. Reconcile: Round-table conference improves reasoning via consensus among diverse llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 7066–7085. Association for Computational Linguistics, 2024.

Qian Chen, Cong Xin, Yang Cheng, Chen Weize, Yusheng Su, Juyuan Xu, Liu Zhiyuan, and Sun Maosong. Communicative agents for software development. *arXiv preprint arXiv:2307.07924*, 2023.

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

Enjun Du, Xunkai Li, Tian Jin, Zhihan Zhang, Rong-Hua Li, and Guoren Wang. Graphmaster: Automated graph synthesis via llm agents in data-limited environments. *Advances in Neural Information Processing Systems*, 38:103634–103681, 2026.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*, 2023.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021a.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *NeurIPS Datasets and Benchmarks*, 2021b.

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

Jen-tse Huang, Jiaxu Zhou, Tailin Jin, Xuhui Zhou, Zixi Chen, Wenxuan Wang, Youliang Yuan, Michael R Lyu, and Maarten Sap. On the resilience of llm-based multi-agent collaboration with faulty agents. *arXiv preprint arXiv:2408.00989*, 2024.

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

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.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024.

Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. Augmenting large language models with chemistry tools. *Nature machine intelligence*, 6(5):525–535, 2024.

David Henry Mguni, Yaqi Sun, Haojun Chen, Wanrong Yang, Amir Darabi, Larry Olanrewaju Orimoloye, and Yaodong Yang. Learning robust multi-agent policies via selective adversarial fault induction. In *Forty-third International Conference on Machine Learning*, 2025.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pp. 1–22, 2023.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? In *Proceedings of the 2021 conference of the North American chapter of the association for computational linguistics: human language technologies*, pp. 2080–2094, 2021.

Xu Shen, Yixin Liu, Yiwei Dai, Yili Wang, Rui Miao, Yue Tan, Shirui Pan, and Xin Wang. Understanding the information propagation effects of communication topologies in llm-based multi-agent systems. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 12358–12372, 2025.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023.

Kun Wang, Guibin Zhang, ManKit Ye, Xinyu Deng, Dongxia Wang, Xiaobin Hu, Jinyang Guo, Yang Liu, and Yufei Guo. Mas ^{2}: Self-generative, self-configuring, self-rectifying multi-agent systems. In *International Conference on Learning Representations*, volume 2026, pp. 113586–113613, 2026.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems*, 2024.

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

Guang Yang, Tianpei Yang, Jingwen Qiao, Yanqing Wu, Jing Huo, Xingguo Chen, and Yang Gao. Multi-agent reinforcement learning with communication-constrained priors. *Advances in Neural Information Processing Systems*, 38:154040–154060, 2026.

Miao Yu, Shilong Wang, Guibin Zhang, Junyuan Mao, Chenlong Yin, Qijiong Liu, Qingsong Wen, Kun Wang, and Yang Wang. Netsafe: Exploring the topological safety of multi-agent networks. *arXiv preprint arXiv:2410.15686*, 2024.

Miao Yu, Shilong Wang, Guibin Zhang, Junyuan Mao, Chenlong Yin, Qijiong Liu, Kun Wang, Qingsong Wen, and Yang Wang. Netsafe: Exploring the topological safety of multi-agent system. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 2905–2938, 2025.

Sukwon Yun, Jie Peng, Pingzhi Li, Wendong Fan, Jie Chen, James Y Zou, Guohao Li, and Tianlong Chen. Graph-of-agents: A graph-based framework for multi-agent llm collaboration. In *International Conference on Learning Representations*, volume 2026, pp. 19745–19760, 2026.

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. *arXiv preprint arXiv:2410.11782*, 2024a.

Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Yu, and Tianlong Chen. Cut the crap: An economical communication pipeline for llm-based multi-agent systems. In *International Conference on Learning Representations*, volume 2025, pp. 75389–75428, 2025.

Zaibin Zhang, Yongting Zhang, Lijun Li, Jing Shao, Hongzhi Gao, Yu Qiao, Lijun Wang, Huchuan Lu, and Feng Zhao. Psysafe: A comprehensive framework for psychological-based attack, defense, and evaluation of multi-agent system safety. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15202–15231, 2024b.

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. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 35176–35184, 2026.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. Language agents as optimizable graphs. *arXiv preprint arXiv:2402.16823*, 2024.

A NOTATIONS

For clarity, Table 4 summarizes the main mathematical symbols used throughout this paper, covering agent dropout modeling, topology recovery, and temporal state adaptation.

Table 4: Main mathematical symbols used in RobMAS.

Symbol	Meaning
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	Directed communication graph, with agent set $\mathcal{V}$ and edge set $\mathcal{E}$.
N	Total number of agents.
K	Total number of communication rounds.
$m_i^k \in \{0, 1\}$	Availability of agent i at round k: 1 for online and 0 for offline.
p	Agent dropout probability.
$\mathcal{V}_+^k$	Set of online agents at round k.
n	Number of online agents.
P_i^k	Prompt of agent i at round k.
o_i^k	Output of agent i at round k.
$\hat{a}$	Final answer aggregated from the online agents' outputs.
$\mathcal{M}_{\text{rec}}$	MDP for topology recovery, defined by $(\mathcal{S}, \mathcal{U}, P, R, \gamma)$.
s_t^k	Recovery state at step t of round k.
a_t^k	Graph-editing action at step t of round k.
$\pi_\theta(a_t^k \mid s_t^k)$	Parameterized topology recovery policy.
A	Adjacency matrix of the communication graph.
$\widehat{A}$	Normalized adjacency matrix.
X_i	Structural feature of agent i.
z_i	Learned representation of agent i.
$\mathcal{G}_{\text{rec}}$	Recovered communication graph.
S_{spec}	Spectral score measuring connectivity relative to maximum in-degree.
$B_{\max}$	Maximum betweenness centrality, measuring reliance on a single relay agent.
R_t^{spatial}	Spatial reward for structural improvement.
q	Encoded task representation.
u_i	Task-conditioned representation of agent i.
h_i^k	Recurrent TSRA state retaining recovery history for agent i.
R_{task}	Terminal reward determined by task correctness.
R	Joint reward combining spatial and task rewards.

B STRUCTURAL ANALYSIS OF TOPOLOGY RECOVERY

B.1 SPECTRAL CONNECTIVITY AND COMMUNICATION SPARSITY

Proof. Let $\mathcal{G} = \mathcal{G}_{\text{rec}} = (V, \mathcal{E})$, $n = |V|$, and let $\mathcal{G}_{\text{sym}}$ be its symmetrized projection. The normalized Laplacian satisfies the classical spectral graph property

$$\lambda_2(L_{\text{sym}}) > 0 \quad \Longleftrightarrow \quad \mathcal{G}_{\text{sym}} \text{ is connected.} \tag{18}$$

Since

$$S_{\text{spec}}(\mathcal{G}) = \frac{\lambda_2(L_{\text{sym}})}{d_{\max}^{\text{in}}} \geq \eta > 0, \tag{19}$$

we necessarily have

$$\lambda_2(L_{\text{sym}}) > 0. \tag{20}$$

Therefore, the symmetrized projection $\mathcal{G}_{\text{sym}}$ is connected, implying that the recovered directed graph $\mathcal{G}$ is at least weakly connected.

Moreover, for a normalized Laplacian,

$$0 \leq \lambda_i(L_{\text{sym}}) \leq 2. \tag{21}$$

Hence,

$$\eta \leq \frac{\lambda_2(L_{\text{sym}})}{d_{\max}^{\text{in}}} \leq \frac{2}{d_{\max}^{\text{in}}}, \tag{22}$$

which gives

$$d_{\max}^{\text{in}} \leq \frac{2}{\eta}. \tag{23}$$

For any directed graph,

$$|\mathcal{E}| = \sum_{i=1}^{n} d_i^{\text{in}}. \tag{24}$$

Therefore,

$$|\mathcal{E}| \leq n d_{\max}^{\text{in}} \leq \frac{2n}{\eta}. \tag{25}$$

Consequently, when η is treated as a constant,

$$|\mathcal{E}| = O(n). \tag{26}$$

Thus, maintaining a positive lower bound on the spectral ratio not only excludes disconnected symmetrized topologies, but also provides a linear upper bound on the number of communication edges in the recovered graph.

Define the conductance of the symmetrized graph $\mathcal{G}_{\text{sym}}$ as

$$\Phi(\mathcal{G}_{\text{sym}}) = \min_{\substack{S \subset V \\ 0 < \text{vol}(S) \leq \text{vol}(V)/2}} \frac{|\partial S|}{\text{vol}(S)}, \tag{27}$$

where $|\partial S|$ denotes the number of edges crossing between S and its complement.

By Cheeger's inequality for the normalized Laplacian,

$$\frac{\lambda_2(L_{\text{sym}})}{2} \leq \Phi(\mathcal{G}_{\text{sym}}) \leq \sqrt{2\lambda_2(L_{\text{sym}})}. \tag{28}$$

Since

$$S_{\text{spec}}(\mathcal{G}) = \frac{\lambda_2(L_{\text{sym}})}{d_{\max}^{\text{in}}}, \tag{29}$$

we have

$$\lambda_2(L_{\text{sym}}) = S_{\text{spec}}(\mathcal{G}) d_{\max}^{\text{in}}. \tag{30}$$

Therefore, if

$$S_{\text{spec}}(\mathcal{G}) \geq \eta > 0, \tag{31}$$

then

$$\Phi(\mathcal{G}_{\text{sym}}) \geq \frac{\eta d_{\max}^{\text{in}}}{2}. \tag{32}$$

For any nontrivial connected graph, $d_{\max}^{\text{in}} \geq 1$, and hence

$$\Phi(\mathcal{G}_{\text{sym}}) \geq \frac{\eta}{2}. \tag{33}$$

Thus, a positive lower bound on the spectral ratio not only guarantees connectivity of the symmetrized graph but also prevents arbitrarily narrow graph cuts. □

B.2 Single Agent Relay Vulnerability Bound

Proof. Let $\mathcal{G} = \mathcal{G}_{\text{rec}} = (V, \mathcal{E})$ and $n = |V|$. Consider a node $v \in V$ and an ordered node pair (s, t) such that t is reachable from s in $\mathcal{G}$, but becomes unreachable after removing v:

$$s \rightsquigarrow t \quad \text{in } \mathcal{G}, \qquad s \not\rightsquigarrow t \quad \text{in } \mathcal{G} - v. \tag{34}$$

This implies that every directed path from s to t in $\mathcal{G}$ must pass through v. In particular, every shortest path between s and t contains v. Therefore,

$$\sigma_{st}(v) = \sigma_{st}, \tag{35}$$

where σ_{st} is the number of shortest paths from s to t and $\sigma_{st}(v)$ is the number of such paths passing through v. Hence,

$$\frac{\sigma_{st}(v)}{\sigma_{st}} = 1. \tag{36}$$

Let $D(v)$ denote the number of ordered node pairs whose reachability is lost after removing v. Since each such pair contributes one unit to the unnormalized betweenness centrality $B_{\text{raw}}(v)$, we obtain

$$D(v) \leq B_{\text{raw}}(v). \tag{37}$$

Dividing both sides by $(n-1)(n-2)$ gives the normalized quantities

$$\Delta(v) = \frac{D(v)}{(n-1)(n-2)} \leq \frac{B_{\text{raw}}(v)}{(n-1)(n-2)} = B(v). \tag{38}$$

Therefore,

$$\max_{v \in V} \Delta(v) \leq B_{\max}(\mathcal{G}_{\text{rec}}). \tag{39}$$

Thus, minimizing the maximum betweenness centrality limits the worst-case fraction of pairwise reachability that can be disrupted by the removal of a single node. □

B.3 Bounded Residual Perturbation

Proof. For any two feasible actions $a, b \in \mathcal{F}(s)$, their Stage 2 logit difference is

$$\ell\left(a \mid \tilde{s}\right) - \ell\left(b \mid \tilde{s}\right) = \frac{Q_{\text{base}}\left(a \mid s\right) - Q_{\text{base}}\left(b \mid s\right)}{\tau} + \Delta Q_{\text{task}}\left(a \mid \tilde{s}\right) - \Delta Q_{\text{task}}\left(b \mid \tilde{s}\right). \tag{40}$$

By the bounded residual assumption,

$$\left|\Delta Q_{\text{task}}\left(a \mid \tilde{s}\right)\right| \leq \epsilon \tag{41}$$

and

$$\left|\Delta Q_{\text{task}}\left(b \mid \tilde{s}\right)\right| \leq \epsilon. \tag{42}$$

$Therefore$,

$$\Delta Q_{\text{task}}\left(a \mid \tilde{s}\right) - \Delta Q_{\text{task}}\left(b \mid \tilde{s}\right) \geq -2\epsilon. \tag{43}$$

Hence,

$$\ell\left(a \mid \tilde{s}\right) - \ell\left(b \mid \tilde{s}\right) \geq \frac{Q_{\text{base}}\left(a \mid s\right) - Q_{\text{base}}\left(b \mid s\right)}{\tau} - 2\epsilon. \tag{44}$$

If

$$Q_{\text{base}}\left(a \mid s\right) - Q_{\text{base}}\left(b \mid s\right) > 2\tau\epsilon, \tag{45}$$

then

$$\frac{Q_{\text{base}}\left(a \mid s\right) - Q_{\text{base}}\left(b \mid s\right)}{\tau} - 2\epsilon > 0. \tag{46}$$

$Therefore$,

$$\ell\left(a \mid \tilde{s}\right) > \ell\left(b \mid \tilde{s}\right). \tag{47}$$

Hence, the relative preference between a and b is preserved after Stage 2 adaptation. □

C Overall Training and Execution Pipeline

Algorithm 1 summarizes the overall training and execution pipeline of RobMAS. The framework first learns a spatial structural policy to reconstruct a sparse and connected communication graph after agent dropout. It then incorporates task semantics and recovery history through Muse and TSRA, and refines graph-editing decisions using a task-adaptive residual policy. The recovered topology is used for multi-round agent communication, while structural and terminal task rewards jointly guide policy optimization.

Algorithm 1: Two-stage training and execution pipeline of RobMAS.

Input: Task set $\mathcal{D}$; initial graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$; communication rounds K; dropout rate p; edge budgets B_1, B_2; topology score Ψ

Output: Task-adaptive recovery policy π_θ and final answer $\hat{a}$

Initialize the Stage-1 structural policy Q_{base}

/* Stage 1: spatial structural recovery */

while *the Stage-1 policy has not converged* **do**

  Sample agent availability $\mathbf{m}$ and remove offline agents and their incident edges to obtain $\mathcal{G}_0$

  repeat

    $X_i \leftarrow [d_i^{\text{in}}/(N-1), d_i^{\text{out}}/(N-1), m_i]$

    $Z_t \leftarrow \text{GNN}(\widehat{A}_t, X_t) \odot \mathbf{m}$

    $Q_{\text{base}}(a_{ij} \mid s_t) \leftarrow f_{\text{edge}}([z_i, z_j, \bar{z}])$

    $Q_{\text{base}}(\texttt{STOP} \mid s_t) \leftarrow f_{\text{stop}}(\bar{z})$

    Filter edges incident to offline agents, existing edges, and cycle-inducing edges; select the highest-valued valid action a_t

    Apply a_t and compute $R_t^{\text{spatial}} \leftarrow \Psi(\mathcal{G}_{t+1}) - \Psi(\mathcal{G}_t) - \lambda_{\text{edge}}\mathbb{I}[a_t \neq \texttt{STOP}]$

    Update Q_{base} using $(s_t, a_t, R_t^{\text{spatial}}, s_{t+1})$

  until $a_t = \texttt{STOP}$ **or** B_1 *edges have been added*

end

Retain Q_{base} as the Stage-1 structural prior and initialize the task-adaptive residual policy

/* Stage 2: temporal state adaptation */

for *each task query* $q \in \mathcal{D}$ **do**

  Initialize $h_i^0 = 0$ and dialogue history $\mathcal{H} = \varnothing$

  for $k = 1$ **to** K **do**

    Sample $\mathbf{m}^k$, remove offline agents, and apply the Stage-1 policy to obtain $\mathcal{G}_0^k$

    repeat

      $u_{i,t}^k \leftarrow \text{Muse}(z_{i,t}^k, q)$

      $h_{i,t}^k \leftarrow \text{TSRA}(u_{i,t}^k, h_{i,t-1}^k)$ for online agents; otherwise retain $h_{i,t-1}^k$

      $\Delta Q_{\text{task}}(a_{ij} \mid \tilde{s}_t^k) \leftarrow f_{\text{taskedge}}([h_i, h_j, \bar{h}])$

      $\Delta Q_{\text{task}}(\texttt{STOP} \mid \tilde{s}_t^k) \leftarrow f_{\text{taskstop}}(\bar{h})$

      $\ell(a) \leftarrow \dfrac{Q_{\text{base}}(a \mid s_t^k)}{\tau} + \Delta Q_{\text{task}}(a \mid \tilde{s}_t^k)$

      Filter invalid actions, select $a_t^k \sim \pi_\theta = \text{softmax}(\ell)$, and update $\mathcal{G}_t^k$

    until $a_t^k = \texttt{STOP}$ **or** B_2 *edges have been added*

    Execute online agents in a topological order of $\mathcal{G}_{\text{rec}}^k$ and append their responses to $\mathcal{H}$

  end

  $\hat{a} \leftarrow \text{Aggregate}(\text{final-round responses})$

  Set $R_{\text{task}} \in \{+1, -1\}$ and compute

$$R_t \leftarrow w_{\text{spatial}} R_t^{\text{spatial}} + w_{\text{task}} \mathbb{I}[t = T] R_{\text{task}}.$$

  Update θ to maximize $J(\theta) = \mathbb{E}_{\pi_\theta}[\sum_{\tau=0}^{T-1} \gamma^\tau r_\tau]$

end

return $\pi_\theta, \hat{a}$

D BENCHMARK DETAILS.

We evaluate RobMAS on six benchmarks spanning general knowledge, mathematical reasoning, and game reasoning.

MMLU-Pro (Wang et al., 2024) is a challenging extension of MMLU that contains multiple-choice questions from a broad range of academic disciplines. Compared with conventional four-choice benchmarks, each question can contain up to ten candidate answers, increasing the difficulty of both reasoning and answer selection.

MATH (Hendrycks et al., 2021b) evaluates mathematical problem solving across multiple mathematical domains and difficulty levels. We use only Level-5 problems, the most challenging subset, which typically require multi-step reasoning and more complex solution procedures.

Chess (Srivastava et al., 2023), derived from BIG-Bench, evaluates sequential game reasoning. Given the preceding move history of a chess game, agents are required to infer a valid subsequent move, testing their ability to reason over structured and temporally dependent game states.

These three benchmarks are used during training and for in-domain evaluation of the recovery policy.

To evaluate whether the learned recovery policy transfers beyond the training benchmarks, we additionally use three datasets without any further policy training. **MMLU** (Hendrycks et al., 2021a) evaluates broad domain knowledge and reasoning through multiple choice questions covering a diverse set of academic subjects. **GSM8K** (Cobbe et al., 2021) consists of grade school mathematical word problems that require multi step arithmetic reasoning. **SVAMP** (Patel et al., 2021) is a mathematical word problem benchmark designed to evaluate robustness to variations in problem structure and linguistic formulation, requiring models to identify the relevant quantities and arithmetic relations rather than relying on superficial textual patterns. Together, MMLU, GSM8K, and SVAMP provide complementary evaluations of the transferability of RobMAS across general knowledge and mathematical reasoning tasks.

E BASELINE DETAILS.

We compare RobMAS with three categories of baselines covering different multi-agent communication strategies.

Fixed Topologies. We consider **Chain**, **Star**, and **Random**. These baselines use predefined communication structures without adapting their topology to task inputs or agent dropout. They represent different levels of connectivity and communication centralization.

Communication Paradigms. We include **LLM-Debate** (Du et al., 2024), **Self-Consistency (SC)** (Wang et al., 2023), and **ReConcile** (Chen et al., 2024). LLM-Debate enables multiple agents to iteratively exchange and refine their responses. SC obtains multiple independent reasoning trajectories and determines the final answer through consistency among sampled outputs. ReConcile iteratively aggregates and reconciles different agent responses to reach a consensus.

Adaptive Topologies. We compare RobMAS with **G-Designer** (Zhang et al., 2024a), **ResMAS** (Zhou et al., 2026), **AgentPrune**, and **EIB-Learner**. G-Designer optimizes the communication graph according to task performance, while ResMAS explicitly improves the resilience of multi-agent systems to agent dropout through topology and prompt optimization. AgentPrune learns to remove redundant spatial and temporal communication connections to obtain more efficient multi-agent structures. EIB-Learner learns effective communication structures by balancing task-relevant information exchange and communication efficiency. Together, these methods provide representative comparisons with adaptive communication structure optimization.

F IMPLEMENTATION DETAILS

We provide the implementation and training details of the proposed two-stage recovery framework. The framework consists of Stage 1 for topology-oriented spatial recovery and Stage 2 for task-driven temporal adaptation. Unless otherwise specified, all experiments use 10 agents and three communication rounds.

Stage 1: Spatial Topology Recovery. In Stage 1, we train the topology recovery policy to reconstruct the surviving communication graph after agent dropout. The graph state is encoded by a two-layer GNN, and candidate edge additions and the STOP action are evaluated by the edge/stop prediction head. Training is performed with experience replay and a target network. We use an ϵ-greedy exploration strategy, where ϵ decreases from 1.0 to 0.05 during training. At each recovery round, the policy is allowed to add at most four communication edges.

Table 5: Hyperparameter settings for Stage 1 spatial topology recovery.

Hyperparameter	**Value**
GNN layers	2
Edge/stop head hidden dimension	64
Learning rate	1×10^{-3}
Discount factor (γ)	0.95
Replay buffer capacity	20,000
Batch size	4
Gradient clip norm	5
Target network update interval	200 steps
ϵ start / end	1.0 / 0.05
ϵ decay	Exponential
Number of agents	10
Communication rounds	3
Edge budget	4 per round
TD loss	Smooth L1 loss

Stage 2: Temporal Task Adaptation. Stage 2 further optimizes the recovery policy with task feedback using PPO. The query is encoded using `all-MiniLM-L6-v2`, and the recurrent state is maintained across topology editing decisions and communication rounds. Newly introduced modules are optimized with Adam using a learning rate of 2×10^{-4}. Stage 1 first provides the initial recovery with a maximum budget of four added edges per round, after which the PPO policy can introduce at most two additional edges per round. The terminal task reward is $+1$ for a correct answer and -1 otherwise, while each added edge incurs a cost of 0.05.

Table 6: Hyperparameter settings for Stage 2 temporal task adaptation.

Hyperparameter	**Value**
GNN layers	2
Query encoder	`all-MiniLM-L6-v2`
Actor/Critic hidden dimension	64
Optimizer	Adam
Learning rate	2×10^{-4}
Gradient clip norm	0.5
Number of agents	10
Communication rounds	3
Edge budget	2 per round
Value loss coefficient	0.5
Task reward	$+1/-1$
Edge cost	0.05
Evaluation	Greedy

Evaluation Protocol. During evaluation, topology recovery actions are selected greedily without exploration. We set the agent dropout probability to 0.4 unless otherwise specified. For dynamic dropout, agent availability is resampled at each communication round, whereas for static dropout, the sampled availability pattern remains unchanged throughout the episode. The same number of agents, communication rounds, LLM generation settings, and task evaluation protocol are used across methods to ensure a controlled comparison.

G VISUALIZATION OF SPATIAL CONNECTIVITY EVOLUTION

To provide an intuitive illustration of topology adaptation under agent dropout, we select a representative task from the MMLU-Pro dataset and visualize the evolution of its spatial connectivity over three communication rounds. As shown in Figure 6, each entry (i, j) represents the cumulative occurrence frequency of the directed connection from agent i to agent j up to the corresponding round. Warmer colors indicate more persistent connections. As the set of available agents changes, RobMAS dynamically reorganizes the communication topology while preserving several stable information-transmission paths across rounds.

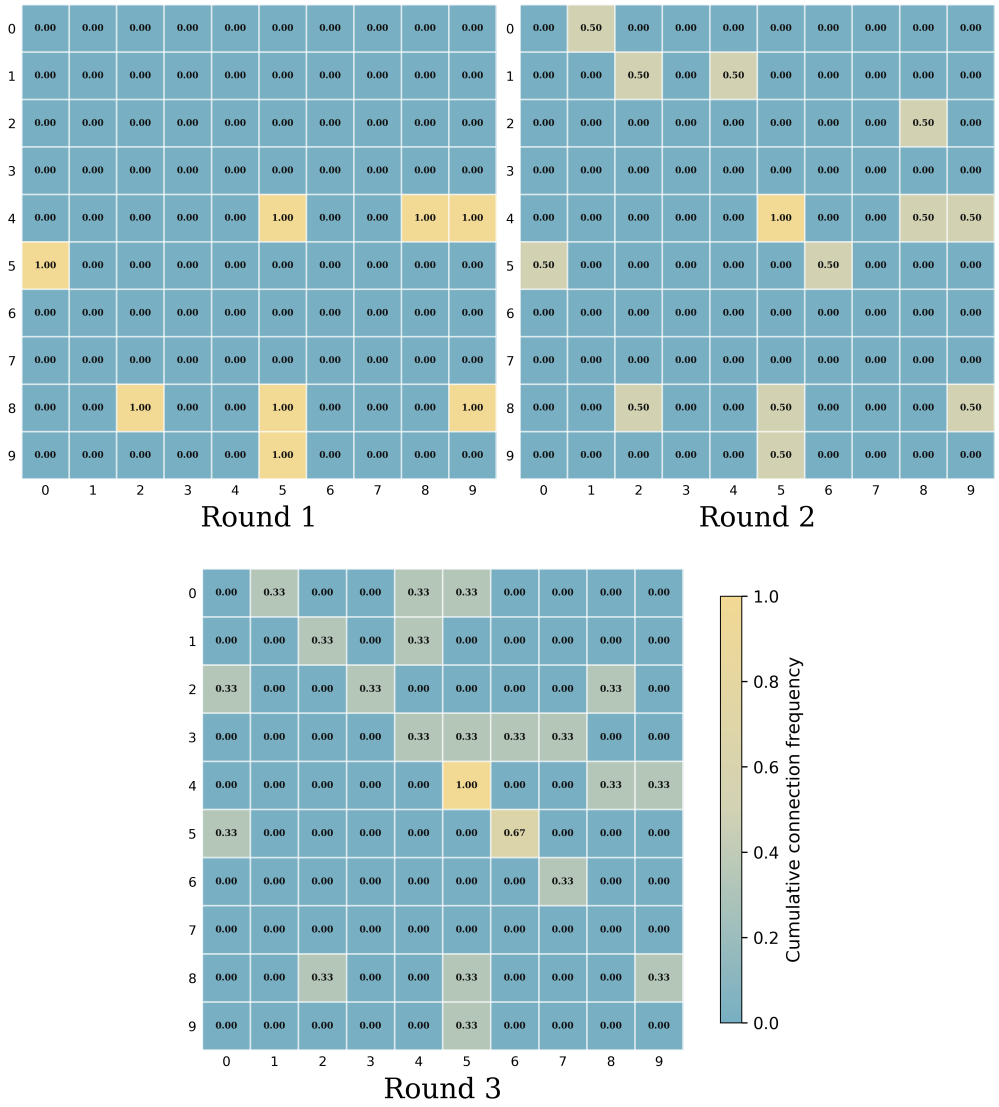


Figure 6: Evolution of spatial connectivity for a representative task from the MMLU-Pro dataset under agent dropout. The upper panel shows the connectivity patterns in Rounds 1 and 2, while the lower panel shows the pattern in Round 3. Cell (i, j) denotes the cumulative occurrence frequency of the directed connection from agent i to agent j.

H PROMPT STRUCTURE

Prompt Structure for the MMLU-Pro Task

Description:
This prompt activates an MMLU-Pro agent within the executing MAS. The agent solves a multiple-choice question by independently analyzing the question and the available answer choices. When responses from other agents are available, the agent uses them only as references and verifies their reasoning independently.

```
You are an expert problem solver for multiple-choice questions.
Analyze the given question independently and select the single best
answer from the provided choices.  When responses from other agents
are available, use them only as references and verify their reasoning
before reaching your own conclusion.

## Goal
Determine the correct answer to the provided MMLU-Pro question.
The choices are labeled with capital letters in their given order,
beginning with A.

## Task Requirements
1.  Analyze the Question:  Identify the knowledge and reasoning
required to answer the question accurately.
2.  Interpret the Choices:  Assign the labels A, B, C, and so on to
the choices in their provided order.
3.  Select an Existing Choice:  Choose exactly one answer from the
available choices.  Do not output a letter that does not correspond
to a provided choice.
4.  Reason Independently:  Derive the answer using your own knowledge
and logical analysis.
5.  Evaluate References:  Previous-round answers and answers from
predecessor agents may be used as references, but they may be
incomplete or incorrect.  Verify every proposed answer and its
reasoning independently.
6.  Preserve Valid Answers:  Keep a previously supported answer
unless specific evidence or reasoning shows that it is incorrect.
Do not change a valid answer merely because another agent disagrees.
7.  Explain Briefly:  Give a concise and well-reasoned explanation
for the selected choice.
8.  Output Format:  The final line must be exactly the answer is X,
where X is one valid capital-letter choice.  Do not place punctuation
or any additional text on the final line.

### MMLU-Pro Task
Question:
{question}

Choices:
{choices}

### Reference Answers
The following section may contain the available previous-round
answer and the current-round answers from predecessor agents.  If
no reference answers are available, solve the task independently.
{reference_answers}
```

Prompt Structure for the MATH Task

Description:
This prompt activates a mathematical problem-solving agent within the executing MAS. Each agent receives an agent-specific role that emphasizes a different reasoning, calculation, or

verification strategy. The agent solves the complete problem independently and uses responses from other agents only as references whose correctness must be verified.

```
## Agent Specific Role
{agent_role}
Apply the assigned role while solving the complete mathematical
problem.  The role determines which aspects of the solution require
particular attention, but it does not restrict the agent to only one
part of the problem.  Independently verify the final result and do
not change a valid answer merely because another agent disagrees.

## Goal
Solve the provided mathematical problem and determine the requested
value or expression.  Present a concise derivation and a clearly
identified final answer.

## Task Requirements
1.  Follow the Assigned Role:  Apply the reasoning or verification
strategy specified by the agent-specific role while still solving the
entire problem.
2.  Understand the Problem:  Identify the given information, the
quantity to be determined, and all relevant mathematical constraints.
3.  Solve Independently:  Derive the result using valid mathematical
principles rather than relying on agreement among other agents.
4.  Verify Correctness:  Check the decisive reasoning steps,
calculations, signs, domains, boundary cases, exceptional values,
and whether the result satisfies the original conditions.
5.  Evaluate References:  Previous-round answers and current-round
answers from predecessor agents may be used as references, but they
may be incorrect.  Verify every proposed result and its supporting
reasoning independently.
6.  Preserve Valid Answers:  Keep a previously supported answer
unless a specific mathematical error makes it invalid.  Do not change
an answer only because another agent disagrees.
7.  Explain Briefly:  Give a concise and coherent derivation
containing the essential steps needed to justify the result.
8.  Output Format:  The final line must contain exactly one final
answer in the form \boxed{answer}.  Do not place any additional text
on the final line.

### Round Specific Instruction
The instruction below specifies whether the current round requires an
independent derivation, a critical review, or a final synthesis.
{round_instruction}

### MATH Task
Problem:
{problem}

### Reference Answers
The following section may contain the agent's own answer from the
previous round and the current-round answers from predecessor agents.
These responses may contain incorrect calculations, unsupported
conclusions, or irrelevant information.  Verify them independently
before using any part of their reasoning.  If no reference answers
are available, solve the problem independently.
{reference_answers}

### Shared Demonstrations
The following examples demonstrate the required concise solution
style and boxed-answer format.  The same demonstrations are provided
to every agent.
{few_shot_examples}
```

Prompt Structure for the Chess Task

Description:
This prompt activates a Chess agent within the executing MAS. The agent reconstructs a chess position from a sequence of moves and determines one legal destination square for the queried piece. When responses from other agents are available, the agent uses them only as references and verifies their legality independently.

```
You are a chess rules expert. Independently reconstruct the board
from the given move sequence and select one legal destination for the
queried piece. Do not change a valid answer merely because another
agent disagrees.

## Goal
Starting from the standard chess position, apply every move in the
provided sequence in order. Then identify one legal destination
square for the piece located on the queried square.

## Task Requirements
1. Reconstruct the Board: Apply every move in the move sequence
chronologically from the standard chess starting position.
2. Identify the Piece: Determine which piece currently occupies the
queried square.
3. Verify Legality: Select a destination that is legal in the
reconstructed position, considering movement rules, occupied squares,
checks, and all other applicable chess rules.
4. Evaluate References: Previous-round answers and answers from
predecessor agents may be used as references, but they may be
incorrect. Verify every proposed move independently.
5. Preserve Valid Answers: Keep a previously supported destination
unless a specific chess rule makes it illegal. Do not change an
answer only because another agent disagrees.
6. Explain Briefly: Give a concise explanation of why the selected
destination is legal.
7. Output Format: The final line must contain exactly one
destination square matching the regular expression [a-h][1-8]. Do
not place any additional text on the final line.

### Chess Task
Move sequence:
{game}

Queried square:
{piece}

### Reference Answers
The following section may contain the available previous-round
answer and the current-round answers from predecessor agents. If
no reference answers are available, solve the task independently.
{reference_answers}
```