Taming "Zombie" Agents: A Markov State-Aware Framework for Resilient Multi-Agent Evolution

Taolin Zhang[1], **Pukun Zhao**[2], **Qizhou Chen**[4], **Jiuheng Wan** [1], **Chen Chen**[2], **Xiaofeng He**[4], **Chengyu Wang**[3*], **Richang Hong**[1*]

[1] School of Computer Science and Information Engineering, Hefei University of Technology
[2] Guangdong University of Finance and Economics
[3] Alibaba Group [4] East China Normal University
tlzhang@hfut.edu.cn, chengyu.wcy@alibaba-inc.com

Abstract

Recent advancements in LLM-based multi-agent systems have demonstrated remarkable collaborative capabilities across complex tasks. To improve overall efficiency, existing methods often rely on aggressive graph evolution among agents (e.g., node or edge pruning), which risks prematurely discarding valuable agents due to transient issues such as hallucinations or temporary knowledge gaps. However, such hard pruning overlooks the potential for "zombie" agents to recover and contribute in subsequent discussion rounds. In this paper, we propose `AgentRevive`, a Markov state-aware framework for resilient multi-agent evolution. Our approach dynamically manages agent collaboration through soft state transitions, implemented via two key components: (1) `State-Aware Policy Learning`: Agent states are divided into "`Active`", "`Standby`", and "`Terminated`" states, selectively propagating messages based on agent memory. The policy employs a risk estimator to optimize agent state transitions by assessing hallucination risk, minimizing the influence of unreliable nodes while safeguarding valuable ones. (2) `State-Aware Edge Optimization`: Subgraph edges are pruned according to states learned from the policy, permanently removing "`Terminated`" nodes and retaining "`Standby`" nodes for subsequent rounds to assess their potential future contributions. Extensive experiments on general reasoning, domain-specific, and hallucination challenge tasks show that our method consistently outperforms strong baselines and significantly reduces token consumption through state-aware agent scheduling.

1 Introduction

LLM-powered multi-agent systems (MAS) have emerged as a transformative paradigm for tackling complex tasks, demonstrating superior performance over single-agent methods through collaborative reasoning and planning (Lin et al., 2025a; Zhang et al., 2025d). The efficacy of MAS depends critically on their inter-agent communication topologies, which govern how information is exchanged and assimilated among agents (Guo et al., 2024; Yan et al., 2025). Consequently, recent research has focused on optimizing communication structures to enhance both performance and efficiency (Zhang et al., 2025a; Wang et al., 2025a,b).

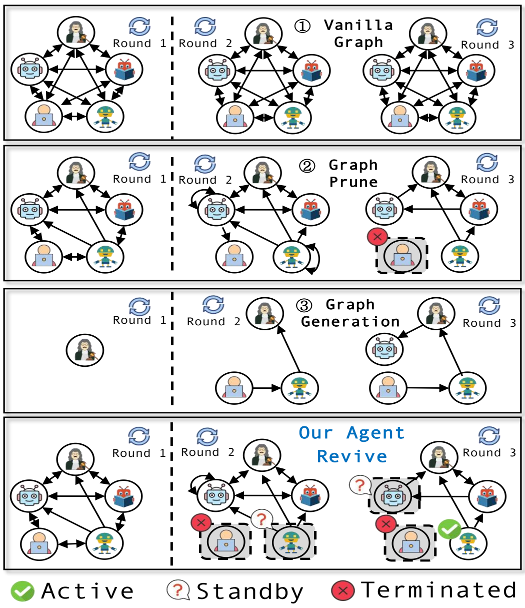

Figure 1: Comparison of agent graph topology evolution between our `AgentRevive` framework and strong training paradigms. (Best viewed in color.)

Approaches addressing communication redundancy in MAS can be broadly categorized into three paradigms: **(1) Vanilla MAS.** These systems rely on manually crafted communication templates, such as chains, trees, or fully connected graphs (Zhang et al., 2024b; Zhuge et al., 2024; Gan et al., 2025). While straightforward to implement, fixed topologies lack adaptability, resulting in inflexible and often inefficient agent interactions

[*] C. Wang and R. Hong are co-corresponding authors.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 8224–8243

that do not dynamically align with task-specific demands. **(2) Graph-pruning-based MAS.** This paradigm models agent interactions as graph structures and applies topology-aware learning to prune redundant edges or nodes (Wang et al., 2025b; Zhang et al., 2025a; Boyi et al., 2025). However, this "hard pruning" strategy irreversibly removes nodes and edges, potentially discarding useful but temporarily inactive "zombie" agents. As a result, the final topology may suffer performance loss, since pruned elements cannot be reactivated even as task contexts change. **(3) Graph-generation MAS.** Recent efforts explore autoregressive, dynamic agent graph generation, constructing the collaboration graph from scratch by sequentially generating agent roles and connections (Wang et al., 2025a; Li et al., 2025). While this paradigm increases flexibility and avoids initial redundancy, it operates in a purely forward-generative manner without considering the global topological state (Qian et al., 2025). Consequently, it may fail to reassess or reintegrate previously excluded agents that could become relevant as task conditions evolve, limiting its ability to optimize the communication structure. As shown in Fig. 1, unlike the three paradigms above, which make permanent pruning decisions, our approach (bottom) allows agent reactivation in later rounds, such as "Round 3".

We introduce `AgentRevive`, a Markov state-aware framework designed for resilient multi-agent evolution. Our core insight is to treat agent collaboration as a soft, state-aware process rather than relying on hard-pruning decisions. `AgentRevive` features two key components:

- **State-Aware Policy Learning**: Learns optimal state transitions for each agent node across communication rounds. We model the agent lifecycle with three states: "`Active`", "`Standby`", and "`Terminated`". State transitions at each round are conditioned on the agent's previous state, its own response, and messages from neighboring agents. To stabilize policy learning under this Markov decision process (MDP), we augment the conventional reward signal, which jointly considers task performance and token efficiency, with a risk estimator. It penalizes strategies that retain agents prone to hallucinated or contradictory responses (Cemri et al., 2025; Zhang et al., 2025c), encouraging dynamic suspension of unreliable nodes without permanent removal.
- **State-Aware Edge Optimization**: Prunes subgraph edges based on agent states learned from the policy, permanently removing "`Terminated`" nodes and retaining "`Standby`" nodes in subsequent rounds to observe their potential contribution to current tasks. Specifically, it constructs a binary node mask based on the survival rates of each agent across multiple inferences, applied to the adjacency matrices of both spatial and temporal edges. This yields a sparsified yet effective communication graph that balances task performance with token efficiency.

Experiments across general reasoning, domain-specific, and hallucination benchmarks demonstrate that `AgentRevive` improves task-averaged performance by **+2.33%** compared to strong pruning-based and dynamic autoregressive baselines, while reducing token overhead by **15%** through adaptive agent state management.

2 Related Work

2.1 Vanilla Agent Collaboration

Early works demonstrate the effectiveness of single LLM agents in reasoning and planning through structured prompting techniques like chain-of-thought (CoT) (Wei et al., 2022) and self-consistency (SC) (Wang et al., 2023). Subsequent works reveal that MAS can outperform single-agent systems by leveraging specialized capabilities through techniques ranging from majority voting (Chen et al., 2024a) to sophisticated interaction mechanisms (Chen et al., 2023). Recent studies have investigated various predefined communication topologies: (1) **Non-interactive**: Independent agent operation without interaction, exemplified by LATM (Zhang et al., 2024a), LLM-Blender (Jiang et al., 2023), and LLM-Debate (Du et al., 2024); (2) **Chain**: Sequential information flow through connected agents, as implemented in ChatDev (Qian et al., 2024), MetaGPT (Hong et al., 2024), and L2MAC (Holt et al., 2023); (3) **Star**: Centralized coordination through a commander agent, demonstrated in AutoGen (Wu et al., 2023); (4) **Tree**: Hierarchical organization with root-level management, such as SoA (Ishibashi and Nishimura, 2024). While these predefined templates facilitate effective MAS interaction, they inherently lack flexibility and scalability.

2.2 MAS Topologies as Graphs

To improve adaptability, recent approaches have explored learning dynamic communication graphs for MAS from task data. GPTSwarm (Zhuge et al., 2024) parameterizes agent interactions with DAG topologies optimized via reinforcement learning. DSPy (Khattab et al., 2024) is a programming model that abstracts LLM pipelines as text transformation graphs. DyLAN (Liu et al., 2024) dynamically selects agent teams for task-specific collaboration. EvoMAC (Hu et al., 2025) employs environmental feedback and textual backpropagation for network updates. However, these models cannot address redundancy in communication graph structures caused by query-adaptive topology generation. Graph-pruning-based methods (Wang et al., 2025b; Zhang et al., 2025a; Boyi et al., 2025) remove redundant nodes and edges in the temporal and spatial dimensions of the graph based on query-specific characteristics during dynamic topology learning, ultimately forming an adaptive sparse topology for answering the query. Additionally, autoregressive dynamic graph generation methods (Ji et al., 2024; Wang et al., 2025a; Li et al., 2025) enable multi-agent pipelines to dynamically generate decision trajectories from scratch, rather than pruning from an initial graph.

3 Problem Formulation

In LLM-based multi-agent systems (MAS), agents may temporarily enter a "zombie" state, i.e., a failure mode caused by hallucinations or knowledge gaps (Lin et al., 2025b). Previous pruning methods (Zhang et al., 2025a; Boyi et al., 2025) treat such agents as redundant and remove them permanently. However, if these agents recover in subsequent rounds, they can potentially contribute critically at a later stage.[1]

To address this, we propose a Markov state-aware collaboration graph framework that dynamically manages agent states across communication rounds. Specifically, we model MAS as a state-aware collaboration graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}^{\mathcal{T}}, \mathcal{E}^{\mathcal{S}}, \mathbf{S})$, where $\mathbf{S}^{(t)} = \{s_1^{(t)}, s_2^{(t)}, \ldots, s_N^{(t)}\}$ denotes the state of each agent at round t, and $s_i^{(t)} \in \{\text{“}\texttt{Active}\text{”}, \text{“}\texttt{Standby}\text{”}, \text{“}\texttt{Terminated}\text{”}\}$. The state transition for each agent is governed by a

[1]Due to space limitations, we refer readers to Appendix A for notations and basic task formulation descriptions.

Method	Task Adaptive	Variable Node Size	Flexible State
Manual Design	✗	✗	✗
AP (Zhang et al., 2025a)	✓	✗	✗
G-D (Zhang et al., 2025b)	✓	✗	✗
AD (Wang et al., 2025b)	✓	✓	✗
ARG-D (Li et al., 2025)	✓	✓	✗
AgentRevive (Ours)	✓	✓	✓

Table 1: Comparison across MAS paradigms. ✓ and ✗ denote full and no support for each capability.

stochastic policy:

$$s_i^{(t+1)} \sim \pi\left(\cdot \mid s_i^{(t)}, h^{(t)}, m_{\mathcal{T}}^{(t+1)}, m_{\mathcal{S}}^{(t+1)}\right) \quad (1)$$

where $h^{(t)}$ denotes the interaction history.

We then define the effective subgraph after policy state changes at round t as $\mathcal{G}_{\text{eff}}^{(t)} = (\mathcal{V}_{\text{eff}}^{(t)}, \mathcal{E}_{\text{eff}}^{(t)})$. The effective agent nodes are:

$$\mathcal{V}_{\text{eff}}^{(t)} = \{v_i \mid s_i^{(t)} \in \{\text{“}\texttt{Active}\text{”}, \text{“}\texttt{Standby}\text{”}\}\} \quad (2)$$

where $\mathcal{E}_{\text{eff}}^{(t)}$ comprises edges between agents in $\mathcal{V}_{\text{eff}}^{(t)}$. We next reformulate communication redundancy by incorporating agent states (Zhang et al., 2025a).

Definition 1 (State-Aware Redundancy). Given a state-aware collaboration graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}^{\mathcal{T}}, \mathcal{E}^{\mathcal{S}}, \mathbf{S})$, an agent v_i is considered redundant at round t if:

$$s_i^{(t)} = \text{“}\texttt{Terminated}\text{”} \quad \text{and} \quad \phi(\mathcal{G}_{\text{eff}}^{(t)}) \geq \phi(\mathcal{G}) \quad (3)$$

where $\phi(\cdot)$ is a utility function measuring task performance. The state-aware pruning objective is to find a policy π that minimizes the effective state-aware graph size while maintaining performance:

$$\min_{\pi} \sum_{t=1}^{T} \left|\mathcal{G}_{\text{eff}}^{(t)}\right|, \quad \text{s.t. } \forall t \quad |\phi(\mathcal{G}_{\text{eff}}^{(t)}) - \phi(\mathcal{G})| \leq \epsilon. \quad (4)$$

Table 1 summarizes how our Markov state-aware framework offers distinct advantages over conventional graph-based approaches.

4 Methodology

4.1 Notations

We first convert the state-aware initial collaboration graph $\mathcal{G}$ into a trainable weighted graph $\tilde{\mathcal{G}}$, leveraging pre-defined spatial edges $\mathcal{E}^{\mathcal{S}}$ and temporal edges $\mathcal{E}^{\mathcal{T}}$. Each edge in the graph is assigned a trainable continuous weight in the range $[0, 1]$. Let

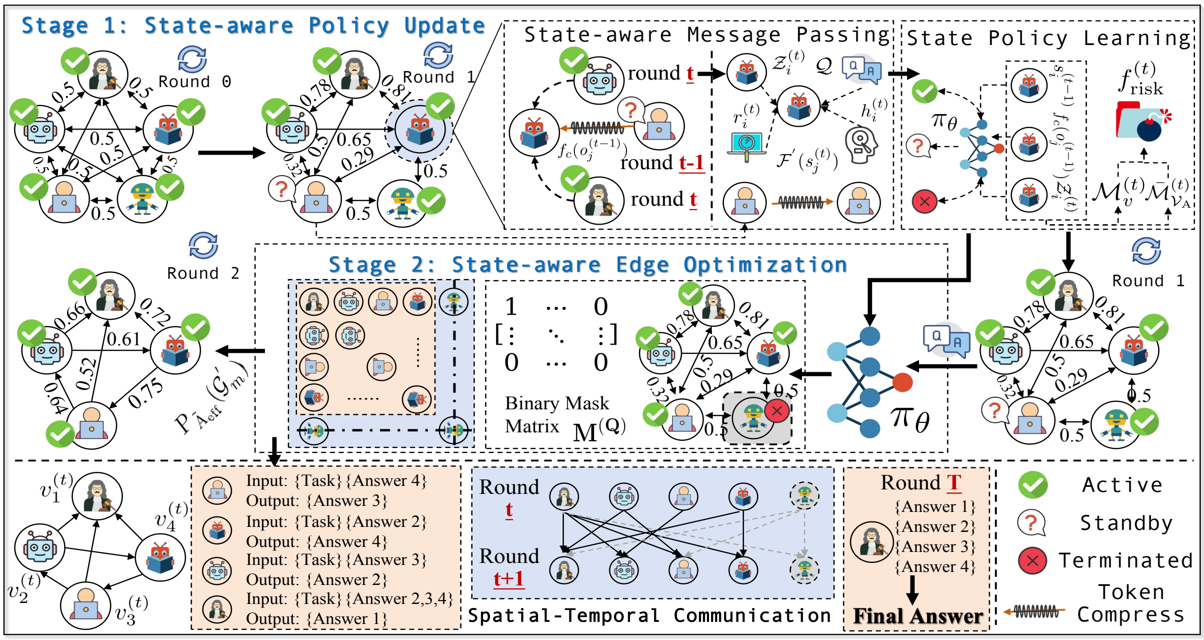

Figure 2: Overview of AgentRevive. Our framework mainly consists of two stages for iteratively training: (1) State-Aware Policy Learning is used to aggregate messages around nodes and train agent state policy networks. (2) State-aware Edge Optimization further optimizes the weights of edges around nodes for messages propagation.

the adjacency matrix set of $\tilde{\mathcal{G}}$ be $\tilde{\mathcal{A}} = \tilde{\mathcal{A}}_{\mathcal{S}} \cup \tilde{\mathcal{A}}_{\mathcal{T}}$, where $\tilde{\mathcal{A}}_{\mathcal{S}} = \bigcup_t \tilde{\mathcal{A}}_{\mathcal{S}}^{(t)}$ is the subset containing same-round adjacency matrices, where $\tilde{\mathcal{A}}_{\mathcal{S}}^{(t)} \in [0,1]^{N\times N}$. $\tilde{\mathcal{A}}_{\mathcal{T}} = \bigcup_t \tilde{\mathcal{A}}_{\mathcal{T}}^{(t)}$ is the subset containing temporal-round adjacency matrices, where $\tilde{\mathcal{A}}_{\mathcal{T}}^{(t)} \in [0,1]^{N\times N}$ represents connections between rounds $(t-1)$ and t. The final effective inference graph $\mathcal{G}_{\text{eff}}$ is a DAG, obtained through the learned agent state-aware policy π. Fig. 2 is an overview of AgentRevive.

4.2 State-Aware Policy Learning

We introduce a paradigm shift from hard pruning to dynamic state management with two key contributions: (1) State-Aware Message Passing: in contrast to static hard pruning methods (Wang et al., 2025b; Zhang et al., 2025a), we govern graph message flow using Markov states. Only agents with the "Active" or "Standby" state are permitted to propagate messages, forming a dynamic topology that prevents irreversible node removing. (2) State-aware Policy Decision: the policy determines the optimal state for each node at every iteration. A critical capability is its reactivation of "zombie" agents from "Standby" to "Active" when contextually advantageous (Lin et al., 2025b), thereby ensuring resilience against transient failures.

4.2.1 State-aware Message Passing

In our AgentRevive model, the aggregated message $\mathcal{Z}_i^{(t)}$ for agent $v_i^{(t)}$ at round t integrates both spatial and temporal information from the graph $\tilde{\mathcal{G}}$:

$$\mathcal{Z}_i^{(\mathcal{S},(t))} = \sum_{v_j^{(t)} \in \mathcal{N}_{in}^{\mathcal{S}}(v_i^{(t)})} \tilde{\mathcal{A}}_{\mathcal{S}}^{(t)}[i,j] \cdot \mathcal{F}(s_j^{(t)}) \quad (5)$$

$$\mathcal{Z}_i^{(t)} = [\mathcal{Z}_i^{(\mathcal{S},(t))} \parallel \mathcal{Z}_i^{(\mathcal{T},(t))}] \quad (6)$$

where $\parallel$ means concatenation. $\mathcal{Z}_i^{(\mathcal{S},(t))}$ is spatial neighboring messages, and the temporal messages $\mathcal{Z}_i^{(\mathcal{T},(t))}$ is also obtained similarly.

After aggregating messages for each agent node, the state-aware response $o_j^{(t)}$ of node $v_j^{(t)}$ at current round t rewrites Eq. 22 as follows:

$$o_j^{(t)} = \begin{cases} \mathcal{F}'(s_j^{(t)}), & \text{if } s_j^{(t)} = \text{“A”} \\ f_c(o_j^{(t-1)}), & \text{if } s_j^{(t)} = \text{“S”} \end{cases} \quad (7)$$

$$\mathcal{F}'(s_j^{(t)}) = f_{\text{pr}}\left(r_i^{(t)}, h_i^{(t)}, \mathcal{Q}, \mathcal{Z}_i^{(t)}\right) \quad (8)$$

where $\mathcal{F}'(s_j^{(t)})$ is current state response. $o_j^{(t)}$ is determined by the agent's state: if the state is "Active (A)", the current response is used; if the state is "Standby (S)", we employ a summarized previous response $f_c(o_j^{(t-1)})$ [2]. As historical messages have

[2]Since $\tilde{\mathcal{A}}$ is an adjacency matrix and $\mathcal{F}'(s_j^{(t)})$ is a string, they are combined via weighted prompts (see Appendix B.3).

already been transmitted via different agent nodes in previous rounds, we employ LLM to further compress the number of tokens:

$$f_c(o_j^{(t-1)}) = f_{\text{LLM}}(\text{``}\texttt{Summarize:}\text{''} + o_j^{(t-1)}) \quad (9)$$

where f_{LLM} means using LLM to limit the number of tokens for summary reasoning directly.

4.2.2 State Policy Learning

After obtaining the response $o_i^{(t)}$ from state message passing, the model determines the optimal state for agent $v_i^{(t)}$ using its own agent memory:

$$s_i^{(t)} \sim \pi_\theta(\cdot \mid f_{\text{Enc}}(s_i^{(t-1)}, o_i^{(t)}, h_i^{(t)}, \mathcal{Z}_i^{(t)})) \quad (10)$$

where $s_i^{(t)} \in \mathbb{R}^3$. Given the lightweight nature of our framework, we implement π_θ as a simple MLP for state prediction, and $f_{\text{Enc}}(\cdot)$ utilizes an LSTM (Hochreiter and Schmidhuber, 1997) to encode the full memory of agent $v_i^{(t)}$, denoted $\mathcal{M}_{v_i}^{(t)}$.

To ensure stable policy training for agent states, we consider policy quality as dependent on both task performance (Zhang et al., 2025a) and the level of hallucination present in node responses. Therefore, the trajectory reward obtained from state transitions of agent nodes is defined as:

$$R(\tau) = \mu(\tilde{\mathcal{G}}^{(T)}) + \eta_{risk} \cdot \sum_{t=1}^{T} f_{\text{risk}}^{(t)} \quad (11)$$

$$f_{\text{risk}}^{(t)} = -\mathbb{E}_{v \in \mathcal{V}_{\text{A}}^{(t)}} \left[\mathbb{D}_{\text{KL}} \left(\mathcal{M}_v^{(t)} \parallel \bar{\mathcal{M}}_{\mathcal{V}_{\text{A}}}^{(t)} \right) \right] \quad (12)$$

where τ denotes the trajectory of agent states over rounds t; $\mu(\cdot)$ is a utility function measuring the final task score; and η_{risk} is a balancing coefficient. Here, $\mathcal{V}_{\text{A}}^{(t)} = \{v_i \mid s_i^{(t)} = \text{``A''}\}$ denotes the set of agents in "`Active`" state in round t. $f_{\text{risk}}^{(t)}$ quantifies the hallucinatory contradiction of agent node $v_i^{(t)}$ in round t, using the KL divergence between each agent's message $\mathcal{M}_{v_i}^{(t)}$ and the average message $\bar{\mathcal{M}}_{\mathcal{V}_{\text{A}}}^{(t)}$ of all "`Active`" agents.

4.3 State-aware Edge Optimization

The strategy trajectory reward focuses on node state transitions. Here, we further consider how these state changes affect the edge weights $\tilde{\mathcal{A}}$, including both spatial and temporal connections. Given the learned policy $\pi_\theta^{(\mathcal{Q})}$ for query $\mathcal{Q}$, we re-infer the state matrix $\mathcal{S}^{(t)}$ for each query and calculate the average survival rate ω_i of each node v_i, defined as the proportion of non-"`Terminated (T)`" states across L inference passes. This yields a binary node mask vector $\mathbf{m}$ for key node selection:

$$\omega_i = \frac{1}{L} \sum_{l=1}^{L} \mathbb{I}(\pi_\theta^{(\mathcal{Q})}(v_i^l) \neq \text{``T''}) \quad (13)$$

$$\mathbf{m} \in \{0,1\}^N, \quad m_i = \begin{cases} 1 & \text{if } \omega_i \geq \gamma \\ 0 & \text{otherwise} \end{cases} \quad (14)$$

where $\pi_\theta^{(\mathcal{Q})}(v_i^l)$ is the predicted state s_i^l at the l-th inference for node v_i, $\mathbb{I}$ is the indicator function, and γ is the survival threshold.

With this mask, we construct the binary mask matrix $\mathbf{M}^{(\mathcal{Q})} = \text{diag}(\mathbf{m}) \in \mathbb{R}^{N \times N}$ and rewrite the state-aware adjacency matrix:

$$\tilde{\mathcal{A}}_{\text{eff}} = \mathbf{M}^{(\mathcal{Q})} \odot \tilde{\mathcal{A}} \odot \mathbf{M}^{(\mathcal{Q})\top} \quad (15)$$

where $\tilde{\mathcal{A}}_{\text{eff}} \in \mathbb{R}^{N_A \times N_A}$, setting rows and columns for masked nodes to zero vectors. Here, $N_A = \sum_i m_i$ denotes the number of active nodes after removing the "`Terminated`" nodes, and $\odot$ is the element-wise multiplication. The training objective for edge optimization rewrites Eq. 4, balancing task performance with graph sparsity:

$$\arg\max_{\tilde{\mathcal{A}}_{\text{eff}}} \underbrace{\mathbb{E}_{\mathcal{G}' \sim \mathbb{G}_{\text{eff}}}[\mu(\mathcal{G}')]}_{\text{Performance}} - \underbrace{\text{rank}(\tilde{\mathcal{A}}_{\text{eff}})}_{\text{Sparsity}} \quad (16)$$

where $\mathbb{G}_{\text{eff}}$ denotes the feasible domain of graph samples after masking, and $\mu(\mathcal{G}')$ is a task performance evaluation (e.g., APIs), rendering the loss function non-differentiable. Hence, we employ unbiased policy gradient (Williams, 1992) to approximate this objective, using the weighted average performance of M samples (Zhang et al., 2025a):

$$\nabla_{\tilde{\mathcal{A}}_{\text{eff}}} \mathbb{E}_{\mathcal{G}' \sim \mathbb{G}_{\text{eff}}}[\mu(\mathcal{G}')] \approx \frac{1}{M} \sum_{m=1}^{M} \mu(\mathcal{G}'_m) \nabla_{\tilde{\mathcal{A}}_{\text{eff}}} \log \left(P_{\tilde{\mathcal{A}}_{\text{eff}}}(\mathcal{G}'_m) \right) \quad (17)$$

where $P_{\tilde{\mathcal{A}}_{\text{eff}}}(\mathcal{G}'_m)$ is the probability of sampling effective subgraph $\mathcal{G}'_m = (\mathcal{V}'_m, \mathcal{E}_m^{\mathcal{T}}, \mathcal{E}_m^{\mathcal{S}})$:

$$P_{\tilde{\mathcal{A}}_{\text{eff}}}(\mathcal{G}'_m) = \prod_{t=1}^{T} \prod_{(v_i,v_j) \in \mathcal{E}_m^{(t),\mathcal{S}}} \tilde{\mathcal{A}}_{\mathcal{S}}^{\text{eff},(t)}[i,j] \times \prod_{t=2}^{T} \prod_{(v_i,v_j) \in \mathcal{E}_m^{(t),\mathcal{T}}} \tilde{\mathcal{A}}_{\mathcal{T}}^{\text{eff},(t)}[i,j] \quad (18)$$

The second term in the objective constrains the sparsity of the communication graph. To relax the NP-hard rank function, we replace it with the nuclear norm (Zhang et al., 2025a):

$$\underset{\tilde{\mathcal{A}}_{\text{eff}}=\{\tilde{\mathcal{A}}_{\mathcal{S}}^{\text{eff}},\tilde{\mathcal{A}}_{\mathcal{T}}^{\text{eff}}\}}{\arg\min} \sum_{t=1}^{T} \|\tilde{\mathcal{A}}_{\mathcal{S}}^{\text{eff},(t)}\|_* + \sum_{t=2}^{T} \|\tilde{\mathcal{A}}_{\mathcal{T}}^{\text{eff},(t)}\|_* \tag{19}$$

where the nuclear norm enables gradient-based optimization of graph sparsity via $\text{rank}(\tilde{\mathcal{A}}_{\text{eff}})$.

4.4 Training

Our `AgentRevive` model is trained iteratively in two stages. The differentiation process for the edge loss function $\mathcal{L}_{\text{edge}}$ in `Stage 2` is described above. For `Stage 1`, we compute the state loss $\mathcal{L}_{\text{state}}$ using the REINFORCE algorithm (Williams, 1992):

$$\nabla_\theta \mathcal{L}_{state}(\theta) \approx \frac{1}{M} \sum_{m=1}^{M} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(s_i^{(t)} | \mathcal{M}_{v_i}^{(t)}) \cdot (R(\tau) - b) \tag{20}$$

where b is a baseline value for variance reduction. The overall training procedure for `AgentRevive` consists of first optimizing $\mathcal{L}_{\text{state}}$, followed by optimization of $\mathcal{L}_{\text{edge}}$. Refer to Appendix C for detailed algorithmic training and inference processes.

5 Experiments

Due to space limitations, we describe datasets, baselines, and implementation in Appendix B.

5.1 Main Results

Table 2 shows the overall performance in various task settings. Key observations include: **[1]** Vanilla methods (CoT (Wei et al., 2022), SC (Wang et al., 2023)) outperform standard prompting but show limited gains on complex tasks due to single-agent constraints. **[2]** Fixed-topology MAS methods (e.g., $\text{MAS}_{\text{round}=1}$, $\text{MAS}_{\text{round}=T}$, AutoGen (Wu et al., 2024), AgentVerse (Chen et al., 2024b)) underperform single-agent prompting on MMLU (Hendrycks et al., 2021) and HumanEval (Chen et al., 2021), due to inefficient communication overhead (Xuan et al., 2025; Chang et al., 2025) and error propagation in multi-round setups. **[3]** Graph-based dynamic MAS methods improve over vanilla MAS via adaptive topologies, though hard pruning risks the permanent loss of potentially useful agents. **[4]** `AgentRevive` achieves SOTA results, with notable gains on TruthfulQA (Lin et al., 2022), by dynamically managing agent states to mitigate hallucinations while preserving recovery potential.[3]

5.2 Ablation Study

We conduct an ablation study analyzing each key component in Table 3. (1) ***w/o SEO***: Removing edge masking for "`Terminated`" agents causes performance drops (e.g., 68.53→66.01 for Llama3-8B (Dubey et al., 2024)), confirming state-based sparsity optimization is essential. (2) ***w/o SPL***: We use replaced random state transitions with learned policy severely degrades performance, underscoring that learned state management is core to mitigating unreliable agents, especially on TruthfulQA. (3) ***w/o SMP***: Propagating messages from all states introduces noise and reduces performance, validating SMP as a critical filter for maintaining coherent collaboration.

5.3 Detailed Analysis

5.3.1 Trade-off between Performance and Token Cost

As shown in Fig. 3, `AgentRevive` achieves superior Pareto efficiency on MMLU, GSM8K, and HumanEval with Llama3-8B, attaining competitive performance with modest token overhead. This efficiency stems from our state-aware dynamic management: "`Standby`" agents are suspended and their compressed historical outputs are reused (Eq. 9), reducing token footprint while preserving contribution potential. In contrast, fixed-topology MAS (e.g., $\text{MAS}_{\text{round}=T}$) incur high token costs from rigid, fully-connected templates. While graph-pruning methods reduce redundancy, their token savings remain suboptimal as hard pruning occurs only after costly multi-round discussions.

5.3.2 Attribution Analysis of States Changes

To quantitatively interpret the decision-making process of our state transition policy π_θ, we conduct an attribution analysis examining the correlation between state changes and key input features. We posit that state transitions are primarily driven by two factors: agent's response risk and the quality of received messages. Specifically, we collect an extensive log of state transition events $(s_i^{(t-1)} \to s_i^{(t)})$ across all testing benchmarks. For each transition, we extract the following feature set:

- **Self-Risk** ($f_{\text{risk}}^{(t)}$): The hallucination risk score (Eq. 11) of the agent's response at round t.

[3] The detailed analysis is shown in Appendix D.

Dataset → Models ↓	Var. NS	Flex. State	MMLU	GSM8K	AQuA	TruthfulQA	SVAMP	HumanEval	Avg.
Base model: Llama3-8B-Instruct									
Vanilla	✗	✗	53.59	70.23	41.67	57.59	75.00	53.33	58.57
CoT	✗	✗	56.86 (↑3.27)	70.47 (↑0.24)	43.75 (↑2.08)	59.25 (↑1.66)	76.17 (↑1.17)	54.17 (↑0.84)	60.11 (↑1.54)
SC (CoT)	✗	✗	60.45 (↑6.86)	71.59 (↑1.36)	46.21 (↑4.54)	59.07 (↑1.48)	78.03 (↑3.03)	55.46 (↑2.13)	61.80 (↑3.23)
$\text{MAS}_{\text{round}=1}$	✗	✗	56.21 (↑2.62)	69.30 (↓0.93)	45.29 (↑3.62)	59.88 (↑2.29)	76.67 (↑1.67)	48.33 (↓5.00)	59.28 (↑0.71)
$\text{MAS}_{\text{round}=T}$	✗	✗	60.13 (↑6.54)	71.48 (↑1.25)	45.41 (↑3.74)	60.14 (↑2.55)	77.56 (↑2.56)	49.17 (↓4.16)	60.65 (↑2.08)
G-Designer	✗	✗	60.27 (↑6.68)	70.59 (↑0.36)	46.82 (↑5.15)	62.43 (↑4.84)	80.03 (↑5.03)	52.53 (↓0.80)	62.28 (↑3.71)
AgentPrune	✗	✗	60.78 (↑7.19)	71.02 (↑0.79)	47.22 (↑5.55)	62.83 (↑5.24)	78.34 (↑3.34)	51.67 (↓1.66)	61.98 (↑3.41)
ARG-Designer	✓	✗	61.49 (↑7.90)	72.74 (↑2.51)	46.23 (↑4.56)	61.78 (↑4.19)	79.38 (↑4.38)	53.62 (↑0.29)	62.54 (↑3.97)
AgentDropout	✓	✗	62.75 (↑9.16)	73.13 (↑2.90)	47.78 (↑6.11)	63.62 (↑6.03)	80.11 (↑5.11)	55.84 (↑2.51)	63.87 (↑5.30)
`AgentRevive`	✓	✓	**64.30** (↑10.71)	**75.81** (↑5.58)	**50.76** (↑9.09)	**65.49** (↑7.90)	**82.68** (↑7.68)	**58.15** (↑4.82)	**66.20** (↑7.63)
Base model: Deepseek-V3-671B-Instruct									
Vanilla	✗	✗	84.97	94.68	84.58	64.70	93.67	88.43	85.17
CoT	✗	✗	84.31 (↓0.66)	95.15 (↑0.47)	85.42 (↑0.84)	64.99 (↑0.29)	93.94 (↑0.27)	89.26 (↑0.83)	85.51 (↑0.34)
SC (CoT)	✗	✗	88.79 (↑3.82)	95.17 (↑0.49)	87.85 (↑3.27)	65.16 (↑0.46)	94.55 (↑0.88)	90.61 (↑2.18)	87.02 (↑1.85)
AutoGen	✗	✗	88.03 (↑3.06)	94.96 (↑0.28)	86.71 (↑2.13)	66.63 (↑1.93)	93.82 (↑0.15)	89.26 (↑0.83)	86.57 (↑1.40)
AgentVerse	✗	✗	87.65 (↑2.68)	95.68 (↑1.00)	85.90 (↑1.32)	65.89 (↑1.19)	94.21 (↑0.54)	88.94 (↑0.51)	86.38 (↑1.21)
$\text{MAS}_{\text{round}=1}$	✗	✗	89.98 (↑5.01)	95.54 (↑0.86)	86.67 (↑2.09)	64.34 (↓0.36)	93.50 (↓0.17)	89.17 (↑0.74)	86.53 (↑1.36)
$\text{MAS}_{\text{round}=T}$	✗	✗	89.54 (↑4.57)	95.49 (↑0.81)	87.50 (↑2.92)	66.05 (↑1.35)	94.33 (↑0.66)	89.26 (↑0.83)	87.03 (↑1.86)
G-Designer	✗	✗	88.74 (↑3.77)	94.93 (↑0.25)	87.61 (↑3.03)	68.70 (↑4.00)	94.75 (↑1.08)	90.20 (↑1.77)	87.49 (↑2.32)
AgentPrune	✗	✗	90.20 (↑5.23)	95.49 (↑0.81)	87.92 (↑3.34)	69.23 (↑4.53)	95.00 (↑1.33)	90.91 (↑2.48)	88.13 (↑2.96)
ARG-Designer	✓	✗	90.04 (↑5.07)	95.71 (↑1.03)	87.96 (↑3.38)	68.44 (↑3.74)	94.98 (↑1.31)	91.18 (↑2.75)	88.05 (↑2.88)
AgentDropout	✓	✗	90.85 (↑5.88)	95.63 (↑0.95)	88.33 (↑3.75)	70.15 (↑5.45)	95.79 (↑2.12)	91.74 (↑3.31)	88.75 (↑3.58)
`AgentRevive`	✓	✓	**91.60** (↑6.63)	**96.48** (↑1.80)	**88.85** (↑4.27)	**72.36** (↑7.66)	**97.07** (↑3.40)	**93.52** (↑5.09)	**90.15** (↑4.98)

Table 2: Results comparison between `AgentRevive` and baselines. **Var. NS** and **Flex. State** denote the Variable Node Size and Flexible State MAS described in Table 1. The Qwen2.5-72B results are shown in Appendix D.

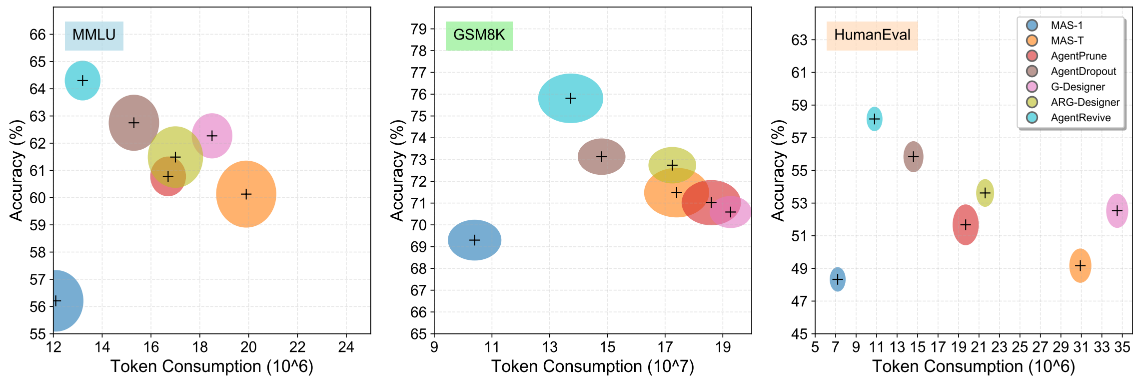

Figure 3: Visualization of performance and token consumption for different multi-agent communication topologies across MMLU, GSM8K, and HumanEval. The number of tokens consumed is represented by the sum of prompt tokens and completion tokens generated by the agents on the horizontal axis.

- **Message Information Gain**: The informational quality of received messages $G(\mathcal{Z}_i^{(t)})$, quantified by the negative entropy of the message set, $G(\mathcal{Z}_i^{(t)}) = -\mathbb{H}(\mathcal{Z}_i^{(t)})$. Lower entropy (higher gain) indicates more consistent and confident information from neighbors.
- **Round Index** (t): The communication round.

We then train a multi-class logistic regression classifier to predict the state transition type (e.g., "`Active`"→"`Standby`"). The standardized coefficients reveal which factors drive specific state changes. As shown in Table 4, yields two critical insights: (1) **Risk-Driven Standby/Termination**: The transition from "`Active`" to "`Standby`" or "`Terminated`" is predominantly and positively correlated with high Self-Risk $f_{\text{risk}}^{(t)}$. This confirms that our policy π_θ learns to proactively identify and suspend agents that are generating unreliable or hallucinatory content. (2) **Message-Driven Reactivation**: Conversely, the transition from

Dataset → / Models ↓	MMLU	GSM8K	TQA	Avg.
Base model: Llama3-8B-Instruct				
AgentRevive	**64.30**	**75.81**	**65.49**	**68.53**
w/o SEO	62.08	73.59	62.37	66.01
w/o SPL	59.27 $_{(\downarrow 5.03)}$	69.85 $_{(\downarrow 5.96)}$	59.28 $_{(\downarrow 6.21)}$	62.80 $_{(\downarrow 5.73)}$
w/o SMP	61.25	71.81	60.79	64.62
Base model: Qwen2.5-72B-Instruct				
AgentRevive	**86.09**	**94.87**	**72.05**	**84.34**
w/o SEO	83.87	92.95	68.76	81.86
w/o SPL	82.70 $_{(\downarrow 3.39)}$	92.11 $_{(\downarrow 2.76)}$	65.82 $_{(\downarrow 6.23)}$	80.21 $_{(\downarrow 4.13)}$
w/o SMP	83.24	92.61	67.43	81.09

Table 3: Ablation study of key state-aware learning modules in AgentRevive. The red down arrow (↓) indicates the greatest performance drop. "TQA": TruthfulQA.

Transition Type	Self-Risk $f_{\text{risk}}^{(t)}$	Message Gain $G(\mathcal{Z}_i^{(t)})$	Round t
"A"→"S"	**0.82**	-0.11	0.05
"A"→"T"	**0.75**	-0.24	**0.31**
"S"→"A"	-0.09	**0.69**	-0.12
"S"→"T"	**0.58**	-0.41	**0.27**

Table 4: Attribution of state transitions based on feature importance from logistic regression. Positive values indicate that higher feature values promote the transition.

"Standby" back to "Active" is most strongly correlated with high message information gain $G(\mathcal{Z}_i^{(t)})$. It indicates that the policy effectively identifies when the collaborative environment has evolved. Through consistent messages from other neighboring agents, enabling it favorable to re-engage a previously "Standby" agent. This demonstrates the system's capability to opportunistically revive "zombie" agents when the context becomes conducive to their contribution. However, the round index t shows a weaker but positive correlation with "Terminated" events, suggesting a tendency to permanently prune agents that remain unreliable over multiple rounds.

5.3.3 Computational Resources Analysis

We evaluate the performance and machine resource trade-off between AgentRevive and three typical MAS baselines. All comparisons are based on identical experimental environments using MMLU samples with Llama3-8B and a single A100 GPU.

As shown in Table 5, AgentRevive demonstrates superior overall efficiency compared to three strong MAS baseline types. While the fixed-topology $\text{MAS}_{\text{round=T}}$ serves as a computationally expensive baseline and the node-generative ARG-Designer incurs high inference latency from its complex network, AgentRevive strikes an optimal balance. It significantly outperforms efficient hard-pruning method AgentDropout in both final accuracy and token savings, achieving this with only a modest increase in inference time (+0.13s) due to the state prediction using lightweight policy network. Since high-precision tasks targeting real-world scenarios only require one model training before deployment to users, the training GPU cost can be negligible. For application scenarios requiring long-term operation and sensitive to inference costs (especially token consumption), the substantial savings achieved by AgentRevive can quickly offset its initial training cost.

5.3.4 Robustness Verification

Due to the space limitation, we present the detailed robustness verification analysis in Appendix D.2.

In summary, we evaluate the AgentRevive's and strong baselines through two aspects: (1) Prompt Attack, where input and response prompts are adversarially manipulated, and (2) Different Graph Structure Initialization, testing performance under varied topologies (e.g., Layered and Random). Results show AgentRevive sustains minimal performance degradation under attacks and maintains stable efficiency across graph structures, demonstrating superior resilience compared to pruning-based and fixed-topology baselines.

6 Conclusion

In this paper, we propose AgentRevive, a Markov state-aware framework for resilient multi-agent evolution. By modeling agent collaboration through soft state transitions, our approach avoids the irreversibility of hard pruning. It integrates state-aware policy learning, which dynamically manages agent states using a risk-aware policy. Additionally, state-aware edge optimization sparsifies the communication graph by masking terminated agents and reusing compressed outputs for "Standby" agents. Extensive evaluations show that AgentRevive achieves superior task performance while maintaining competitive token efficiency.

Limitations

Despite the promising results achieved by AgentRevive, our work has several limitations that

Model	MMLU Acc. (%)	Performance Improvement	Tokens Number (M)	Tokens Saving	Avg. Inference Time (s/sample)	Training Time (GPU hours)
$\text{MAS}_{\text{round}=T}$	60.13	–	1.99	–	2.21	–
ARG-DESIGNER	61.49	+2.3%	1.71	−14.1%	3.82	28.5
AGENTDROPOUT	62.75	+4.4%	1.53	−23.1%	**1.62**	**15.2**
AgentRevive	**64.30** ($\Delta = 1.55$)	**+6.9%**	**1.32** ($\Delta = 0.21$M)	**−33.7%**	1.75 ($\Delta = 0.13$s)	20.8($\Delta = 5.6$GB)

Table 5: Comprehensive trade-off analysis of performance and computational machine resource cost. Both performance improvement and number of tokens saved are considered relative to $\text{MAS}_{\text{round}=T}$.

warrant further investigation. Due to constraints in computational resources, our empirical evaluation was primarily conducted with a maximum of 5 agents. We anticipate that scaling to scenarios involving a larger number of agents would provide a more comprehensive stress test of our state transition policy's scalability and efficiency. Additionally, the configuration of our Markov state-aware framework involves several key hyperparameters, such as the survival threshold γ and the risk balance coefficient η_{risk}. While we found $\gamma = 0.6$ and $\eta_{\text{risk}} = 0.5$ to work well in our experiments, these values were not extensively explored across all possible task types and agent compositions. The optimal configuration may vary across applications, suggesting a need for more adaptive or automated hyperparameter tuning strategies in future work. Finally, the current implementation of the state policy network π_θ uses a relatively simple MLP architecture for stable and sample-efficient training. Exploring more powerful sequence models or graph-aware architectures for state encoding could potentially capture more complex, long-range dependencies in agent interaction histories, possibly leading to more refined state management.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 62506110). It was also supported by the Natural Science Foundation of Anhui Province, China (Grant No. 2508085QF227) and the Hefei University of Technology Scientific Research Innovation Start-up Special Project Type A (Grant No. JZ2025HGQA0137).

References

Li Boyi, Zhonghan Zhao, Der-Horng Lee, and Gaoang Wang. 2025. Adaptive graph pruning for multi-agent communication. *CoRR*, abs/2506.02951.

Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya G. Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. 2025. Why do multi-agent LLM systems fail? *CoRR*, abs/2503.13657.

Chia-Yuan Chang, Zhimeng Jiang, Vineeth Rakesh, Menghai Pan, Chin-Chia Michael Yeh, Guanchu Wang, Mingzhi Hu, Zhichao Xu, Yan Zheng, Mahashweta Das, and Na Zou. 2025. MAIN-RAG: multi-agent filtering retrieval-augmented generation. In *ACL*, pages 2607–2622.

Lingjiao Chen, Jared Quincy Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei Zaharia, and James Zou. 2024a. Are more LLM calls all you need? towards scaling laws of compound inference systems. *CoRR*, abs/2403.02419.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *CoRR*, abs/2107.03374.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. 2024b. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *ICLR*.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chen Qian, Chi-Min Chan, Yujia Qin, Yaxi Lu, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. 2023. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors in agents. *CoRR*, abs/2308.10848.

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. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 181 others. 2025. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

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

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and 82 others. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Bingzheng Gan, Yufan Zhao, Tianyi Zhang, Jing Huang, Yusu Li, Shu Xian Teo, Changwang Zhang, and Wei Shi. 2025. MASTER: A multi-agent system with LLM specialized MCTS. In *NAACL*, pages 9409–9426.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based multi-agents: A survey of progress and challenges. In *IJCAI*, pages 8048–8057.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *ICLR*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Samuel Holt, Max Ruiz Luyten, and Mihaela van der Schaar. 2023. L2MAC: large language model automatic computer for unbounded code generation. *CoRR*, abs/2310.02003.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. Metagpt: Meta programming for A multi-agent collaborative framework. In *ICLR*.

Yue Hu, Yuzhu Cai, Yaxin Du, Xinyu Zhu, Xiangrui Liu, Zijie Yu, Yuchen Hou, Shuo Tang, and Siheng Chen. 2025. Self-evolving multi-agent collaboration networks for software development. In *ICLR*.

Yoichi Ishibashi and Yoshimasa Nishimura. 2024. Self-organized agents: A LLM multi-agent framework toward ultra large-scale code generation and optimization. *CoRR*, abs/2404.02183.

Jiarui Ji, Runlin Lei, Jialing Bi, Zhewei Wei, Yankai Lin, Xuchen Pan, Yaliang Li, and Bolin Ding. 2024. Dynamic and textual graph generation via large-scale llm-based agent simulation. *CoRR*, abs/2410.09824.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. In *ACL*, pages 14165–14178.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2024. Dspy: Compiling declarative language model calls into state-of-the-art pipelines. In *ICLR*.

Shiyuan Li, Yixin Liu, Qingsong Wen, Chengqi Zhang, and Shirui Pan. 2025. Assemble your crew: Automatic multi-agent communication topology design via autoregressive graph generation. *CoRR*, abs/2507.18224.

Hongzhan Lin, Yang Deng, Yuxuan Gu, Wenxuan Zhang, Jing Ma, See-Kiong Ng, and Tat-Seng Chua. 2025a. FACT-AUDIT: an adaptive multi-agent framework for dynamic fact-checking evaluation of large language models. In *ACL*, pages 360–381.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Truthfulqa: Measuring how models mimic human falsehoods. In *ACL*, pages 3214–3252.

Xixun Lin, Yucheng Ning, Jingwen Zhang, Yan Dong, Yilong Liu, Yongxuan Wu, Xiaohua Qi, Nan Sun, Yanmin Shang, Pengfei Cao, Lixin Zou, Xu Chen, Chuan Zhou, Jia Wu, Shirui Pan, Bin Wang, Yanan Cao, Kai Chen, Songlin Hu, and Li Guo. 2025b. Llm-based agents suffer from hallucinations: A survey of taxonomy, methods, and directions.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *ACL*, pages 158–167.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. 2024. A dynamic llm-powered agent network for task-oriented agent collaboration.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *NAACL*, pages 2080–2094.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. Chatdev: Communicative agents for software development. In *ACL*, pages 15174–15186.

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

Song Wang, Zhen Tan, Zihan Chen, Shuang Zhou, Tianlong Chen, and Jundong Li. 2025a. Anymac: Cascading flexible multi-agent collaboration via next-agent prediction. *CoRR*, abs/2506.17784.

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

Zhexuan Wang, Yutong Wang, Xuebo Liu, Liang Ding, Miao Zhang, Jie Liu, and Min Zhang. 2025b. Agentdropout: Dynamic agent elimination for token-efficient and high-performance llm-based multi-agent collaboration. In *ACL*, pages 24013–24035.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, pages 229–256.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. 2024. Autogen: Enabling next-gen LLM applications via multi-agent conversations. In *COLM*.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2023. Autogen: Enabling next-gen LLM applications via multi-agent conversation framework. *CoRR*, abs/2308.08155.

Vu Dinh Xuan, Hao Vo, David Murphy, and Hoang D. Nguyen. 2025. Agentsgen: Multi-agent LLM in the loop for semantic collaboration and generation of synthetic data. *CoRR*, abs/2505.13466.

Bingyu Yan, Xiaoming Zhang, Litian Zhang, Lian Zhang, Ziyi Zhou, Dezhuang Miao, and Chaozhuo Li. 2025. Beyond self-talk: A communication-centric survey of llm-based multi-agent systems. *CoRR*, abs/2502.14321.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 2024. Qwen2.5 technical report. *CoRR*, abs/2412.15115.

Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Xu Yu, and Tianlong Chen. 2025a. Cut the crap: An economical communication pipeline for llm-based multi-agent systems. In *ICLR*.

Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, and Dawei Cheng. 2025b. G-designer: Architecting multi-agent communication topologies via graph neural networks.

Notation	Description
$\mathcal{G}$	state-aware initial graph
$\tilde{\mathcal{G}}$	trainable weighted graph
$\mathcal{V}$	set of agent nodes
$\mathcal{E}^{\mathcal{T}}$	temporal edges
$\mathcal{E}^{\mathcal{S}}$	spatial edges
$\mathcal{S}^{(t)}$	all agents states
$s_i^{(t)}$	state of agent v_i
$\mathcal{G}_{\text{eff}}^{(t)}$	learned effective subgraph
$\mathcal{V}_{\text{eff}}^{(t)}$	effective agent nodes
$\mathcal{Z}_i^{(t)}$	aggregated messages
$\tilde{A}_{\mathcal{S}}$	spatial adjacency matrices
$\tilde{A}_{\mathcal{T}}$	temporal adjacency matrices
$o_i^{(t)}$	response of agent v_i
$\mathcal{M}_{v_i}^{(t)}$	memory state of agent v_i
$R(\tau)$	trajectory reward
$f_{\text{risk}}^{(t)}$	hallucination risk estimator
ω_i	survival rate of agent v_i
$\mathbf{m}$	binary node mask
$\tilde{A}_{\text{eff}}$	effective adjacency matrix
$\phi(\cdot)$	utility function
$\mathcal{N}_{in}^{\mathcal{S}}(v_i)$	spatial in-neighbors of agent v_i
$\mathcal{N}_{in}^{\mathcal{T}}(v_i)$	temporal in-neighbors of agent v_i
π_θ	state policy network
f_{Enc}	encoder for agent memory encoding

Table 6: All used mathematical notations in `AgentRevive` framework.

Jintian Zhang, Xin Xu, Ningyu Zhang, Ruibo Liu, Bryan Hooi, and Shumin Deng. 2024a. Exploring collaboration mechanisms for LLM agents: A social psychology view. In *ACL*, pages 14544–14607.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. 2025c. Which agent causes task failures and when? on automated failure attribution of LLM multi-agent systems. *CoRR*, abs/2505.00212.

Taolin Zhang, Dongyang Li, Qizhou Chen, Chengyu Wang, and Xiaofeng He. 2025d. BELLE: A bi-level multi-agent reasoning framework for multi-hop question answering. In *ACL*, pages 4184–4202.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Ö. Arik. 2024b. Chain of agents: Large language models collaborating on long-context tasks. In *NeurIPS*.

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

A Notations and Task Description

Notations. We have organized all the mathematical notations and descriptions in this paper in Table 6.

MAS as Collaboration Graph. We model a multi-agent system (MAS) as a collaboration graph, represented by a directed acyclic graph (DAG) $\mathcal{G} = (\mathcal{V}, \mathcal{E}^{\mathcal{T}}, \mathcal{E}^{\mathcal{S}})$. The node set $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ corresponds to the set of agents, where each agent v_i is an LLM instance assigned a specific role $r_i \in \mathcal{R}$ that defines its function and expertise. Here, N denotes the total number of agents in the graph. Each agent also maintains an internal state h_i, which records its past actions and interactions. The directed communication pathways are defined by temporal edges $\mathcal{E}^{\mathcal{T}} \subseteq \mathcal{V}^{(t-1)} \times \mathcal{V}^{(t)}$ and spatial edges $\mathcal{E}^{\mathcal{S}} \subseteq \mathcal{V}^{(t)} \times \mathcal{V}^{(t)}$ (Zhang et al., 2025a; Wang et al., 2025b). A temporal edge $e_{ji}^{\mathcal{T}} = (v_j, v_i)$ indicates that messages received by agent v_i in round t are aggregated from agent v_j in the previous round $(t-1)$. Similarly, a spatial edge $e_{ji}^{\mathcal{S}} = (v_j, v_i)$ denotes that information for agent v_i in round t originates from neighboring agent v_j within the same round. Consequently, the sets of direct predecessor neighbors for agent v_i via temporal and spatial edges are respectively defined as $\mathcal{N}_{in}^{\mathcal{T}}(v_i) = \{v_j \mid (v_j, vi) \in \mathcal{E}^{\mathcal{T}}\}$ and $\mathcal{N}_{in}^{\mathcal{S}}(v_i) = \{v_j \mid (v_j, v_i) \in \mathcal{E}^{\mathcal{S}}\}$.

MAS Collaboration Protocol. Given an initial collaboration graph $\mathcal{G}$, the MAS processes a user query $\mathcal{Q}$ through a multi-step collaboration protocol. This protocol dictates the processing and exchange of information among agents across multiple communication rounds. The execution order of agents within each round is determined by a topological sort of the nodes (Wang et al., 2025b), to ensure agents are activated only after their predecessors have completed. This process iterates for T rounds, enabling progressive refinement. At each round t, agent v_i produces its response $\mathbf{M}_i^{(t)}$ by querying its LLM using a dynamically constructed prompt $\mathcal{P}_i^{(t)}$:

$$\mathbf{M}_i^{(t)} = \mathrm{LLM}_i(\mathcal{P}_i^{(t)}) \tag{21}$$

where the prompt integrates the intrinsic properties of the agent with the temporal and spatial outputs of its predecessors:

$$\mathcal{P}_i^{(t)} = f_{\text{pr}}\left(\underbrace{r_i^{(t)}, h_i^{(t)}}_{\text{System}}, \underbrace{\mathcal{Q}, \left\{m_{\mathcal{T}}^{(t)}, m_{\mathcal{S}}^{(t)}\right\}}_{\text{User}}\right) \tag{22}$$

where f_{pr} denotes the prompt construction process. Here, $m_{\mathcal{T}}^{(t)}$ represents messages collected from the temporal neighbors $\mathcal{N}_{in}^{\mathcal{T}}(v_i)$ of agent v_i, while $m_{\mathcal{S}}^{(t)}$ represents those from the spatial neighbors $\mathcal{N}_{in}^{\mathcal{S}}(v_i)$. After T rounds, the final output $\mathcal{O}$ is obtained by aggregating the final-round responses:

$$\mathcal{O} = f_{\text{Agg}}(\{\mathbf{M}_i^{(T)} \mid v_i \in \mathcal{V}\}) \tag{23}$$

where the aggregation strategy f_{Agg} may vary across implementations such as majority voting.

B Implementation Details

B.1 Datasets

We evaluate `AgentRevive` on a diverse set of benchmarks to assess general reasoning, domain-specific, and hallucination-challenging datasets.

- **General Reasoning: MMLU** (Hendrycks et al., 2021) is a multi-task benchmark covering 57 subjects across STEM, humanities, and social sciences. **GSM8K** (Cobbe et al., 2021) consists of grade-school math word problems requiring multi-step reasoning.
- **Domain-Specific Tasks: AQuA** (Patel et al., 2021) includes around 100,000 algebraic word problems with natural language rationales. **SVAMP** (Ling et al., 2017) contains simple arithmetic problems with varying structures to test robustness. **HumanEval** (Chen et al., 2021) comprises programming problems evaluating functional correctness in code generation.
- **Hallucination Challenge: TruthfulQA** (Lin et al., 2022) is designed to measure a model's tendency to produce plausible-sounding but incorrect answers (hallucinations).

B.2 Baselines

We compare `AgentRevive` against a broad range of strong baselines:

- **Single-Agent Methods:** Vanilla LLMs utilize standard prompting without structured reasoning, using Llama3-8B (Dubey et al., 2024), Qwen2.5-72B (Yang et al., 2024), and Deepseek-V3-671B-Instruct (DeepSeek-AI et al., 2025) as backbone models. Chain-of-Thought (CoT) (Wei et al., 2022) enhances reasoning capability through step-by-step solutions with a single agent, demonstrating the upper bound of individual agent performance. Self-Consistency (SC) (Wang et al., 2023) samples multiple reasoning paths from a single agent and selects the most consistent answer, providing advanced single-agent reasoning.

- **Fixed-Topology Multi-Agent Systems:** $\text{MAS}_{\text{round}=1}$ implements single-round multi-agent collaboration with a fixed communication graph structure, testing basic collaborative capability without iterative refinement. $\text{MAS}_{\text{round}=T}$ extends this to multiple communication rounds in a fixed topology, evaluating the effect of extended but rigid agent interactions. AutoGen (Wu et al., 2023) is a programmable multi-agent conversation framework with customizable agent roles and interaction patterns, representing structured but static collaboration. AgentVerse (Chen et al., 2024b) supports diverse multi-agent interaction protocols using predefined templates, testing limits of manually designed topologies.

- **Graph-Based Dynamic MAS:** G-Designer (Zhang et al., 2025b) optimizes communication via graph neural networks, learning edge weights based on task characteristics. AgentPrune (Zhang et al., 2025a) prunes redundant edges in spatial and temporal dimensions, representing hard pruning methods. ARG-Designer (Li et al., 2025) generates collaboration graphs autoregressively from scratch, exploring dynamic topology construction rather than pruning. AgentDropout (Wang et al., 2025b) dynamically eliminates underperforming agents and edges across communication rounds, combining node and edge dropout strategies for efficient collaboration.

B.3 Implementation Details

We implement AgentRevive in PyTorch, conducting experiments on 2 NVIDIA A800 GPUs. For larger models (Qwen2.5-72B and Deepseek-V3-671B), we utilize official APIs for inference.

We set the number of communication rounds $T = 2$ for reasoning and math tasks, and $T = 4$ for code generation. The number of policy training steps is $K = 50$, and the number of graph samples $M = 20$. Other hyperparameters include learning rate $\eta = 0.1$ and survival threshold $\gamma = 0.6$. The balance coefficient η_{risk} in the reward is set to 0.5.

The REINFORCE algorithm is used for policy gradient updates. The state encoder f_{Enc} is a single-layer LSTM, and the state policy network π_θ is a 2-layer MLP.

Model training proceeds in two stages: State-Aware Policy Learning, followed by State-Aware Edge Optimization. Each stage uses 40 training instances sampled from the corresponding dataset's training or validation split.

We report accuracy for all benchmarks. Token consumption is measured as the sum of prompt and completion tokens across all agents and rounds.

For weighted adjacency matrices and response variables of string data type, we use the prompt illustrated in Fig. 4 to combine them for propagation within the communication graph.

Weighted Prompt for Message Aggregation
Instruction: Aggregate the answers to the mathematical problem Q: {"**Janet\u2019s ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?**"} based on the edge weight relationship between the mathematical agent experts at different nodes and the current mathematical expert. Mathematics Expert 1→Current Mathematics Expert: {"Weight": xxx, "Opinion of Mathematics Expert 1": xxx} Mathematics Expert 2→ Current Mathematics Expert: {"Weight": xxx, "Opinion of Mathematics Expert 1": xxx} ......

Figure 4: Prompt design for aggregating edge weights and neighboring agent responses (Example from GSM8K).

C Algorithm Description

The training algorithm, summarized in Algorithm 1, captures the core two-stage process of AgentRevive:

- **Stage 1: State-Aware Policy Learning**—Learns optimal state transitions for agents using a risk-aware reward signal.

- **Stage 2: State-aware Edge Optimization**—Permanently prunes "Terminated" nodes and

Algorithm 1 AgentRevive Training Algorithm

Require: MAS topology graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}^{\mathcal{T}}, \mathcal{E}^{S})$, state policy parameters θ, weighted adjacency matrices $\tilde{\mathcal{A}} = \tilde{\mathcal{A}}_S \cup \tilde{\mathcal{A}}_T$, training steps K_1, K_2, sampling times M, learning rate η, survival threshold γ, balance coefficient η_{risk}

Ensure: Trained state policy parameters θ^*, optimized adjacency matrices $\tilde{\mathcal{A}}_{\text{eff}}$

 # Stage 1: State-Aware Policy Learning
for $k = 1$ **to** K_1 **do**
  Sample M graphs $\{\mathcal{G}_m\}_{m=1}^{M}$ from $\tilde{\mathcal{A}}$ using DAG sampling
  for each sampled graph $\mathcal{G}_m$ **do**
    for agent v_i at each round t **to** T **do**
      $\mathcal{Z}_i^{(t)} = [\mathcal{Z}_i^{(S,(t))} \| \mathcal{Z}_i^{(T,(t))}]$
      $o_i^{(t)} = \mathbb{I}(s_i^{(t)} = \text{“A”})\mathcal{F}'(s_i^{(t)}) + \mathbb{I}(s_i^{(t)} = \text{“S”}) f_c(o_i^{(t-1)})$
      $s_i^{(t)} \sim \pi_\theta(\cdot | f_{\text{Enc}}(s_i^{(t-1)}, o_i^{(t)}, \mathcal{Z}_i^{(t)}))$
      $f_{risk}^{(t)} = -\mathbb{E}_{v \in V_A^{(t)}}[\text{KL}(M_v^{(t)} \| \bar{M}_{V_A}^{(t)})]$
    end for
  end for
  Compute trajectory rewards: $R(\tau) = \mu(\mathcal{G}^{(T)}) + \eta_{risk} \cdot \sum_{t=1}^{T} f_{risk}^{(t)}$
  Update policy parameters: $\theta \leftarrow \theta + \eta \cdot \frac{1}{M} \sum_{m=1}^{M} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(s_i^{(t)}) \mathcal{M}_{v_i}^{(t)} \cdot (R(\tau) - b)$
end for
 # Stage 2: State-aware Edge Optimization
for each agent v_i **do**
  # Compute node survival rates
  $\omega_i = \frac{1}{L} \sum_{l=1}^{L} \mathbb{I}(\pi_\theta^{(Q)}(v_i^l) \neq \text{“T”})$
end for
 # Apply binary mask to adjacency matrices
$\mathbf{m} \in \{0,1\}^N, m_i = 1$ if $\omega_i \geq \gamma$, else 0
$\tilde{\mathcal{A}}_{\text{eff}} = \mathbf{M}^{(Q)} \odot \tilde{\mathcal{A}} \odot \mathbf{M}^{(Q)^T}$
for $k = 1$ **to** K_2 **do**
  Sample M graphs $\{\mathcal{G}'_m\}_{m=1}^{M}$ from $\tilde{\mathcal{A}}_{\text{eff}}$ using DAG sampling
  #Compute edge optimization objective
  $J = \frac{1}{M} \sum_{m=1}^{M} \mu(\mathcal{G}'_m) - \left[\sum_{t=1}^{T} \|\tilde{\mathcal{A}}_S^{\text{eff},(t)}\|_* + \sum_{t=2}^{T} \|\tilde{\mathcal{A}}_T^{\text{eff},(t)}\|_*\right]$
  # Update adjacency matrices
  $\tilde{\mathcal{A}}_{\text{eff}} \leftarrow \tilde{\mathcal{A}}_{\text{eff}} + \eta \cdot \nabla_{\tilde{\mathcal{A}}_{\text{eff}}} J$
end for
return $\theta^*, \tilde{\mathcal{A}}_{\text{eff}}$

Algorithm 2 AgentRevive Inference Algorithm

Require: Trained state policy parameters θ^*, optimized adjacency matrices $\tilde{\mathcal{A}}_{\text{eff}}$, user query $\mathcal{Q}$, initial agent states $\mathbf{s}^{(0)} = \{s_1^{(0)}, s_2^{(0)}, \ldots, s_N^{(0)}\}$, maximum communication rounds T

Ensure: Final answer $\mathcal{O}$

Initialize agent memories $h_i^{(0)}$ for all $v_i \in \mathcal{V}$
Initialize agent responses $o_i^{(0)}$ for all $v_i \in \mathcal{V}$
for $t = 1$ **to** T **do**
  for each agent $v_i \in \mathcal{V}$ **do**
    if $s_i^{(t-1)} \neq$ “Terminated” **then**
      $\mathcal{Z}_i^{(t)} = \left[\mathcal{Z}_i^{(S,(t))} \parallel \mathcal{Z}_i^{(T,(t))}\right]$
      $s_i^{(t)} \sim \pi_{\theta^*}(\cdot \mid f_{\text{Enc}}(\mathcal{M}_{v_i}^{(t)}))$
    else
      $s_i^{(t)} \leftarrow$ “Terminated”
    end if
  end for
  for each agent $v_i \in \mathcal{V}$ **do**
    if $s_i^{(t)} =$ “Active” **then**
      $o_i^{(t)} = f_{\text{pr}}\left(r_i^{(t)}, h_i^{(t-1)}, q, \mathcal{Z}_i^{(t)}\right)$
      # Update Node Memory
      $h_i^{(t)} \leftarrow f(h_i^{(t-1)}, o_i^{(t)})$
    else if $s_i^{(t)} =$ “Standby” **then**
      $o_i^{(t)} \leftarrow f_c(o_i^{(t-1)}) = \text{LLM}(\text{"Summarize:"} + o_i^{(t-1)})$
      $h_i^{(t)} \leftarrow h_i^{(t-1)}$
    else
      # $s_i^{(t)} =$ “Terminated”
      $o_i^{(t)} \leftarrow \emptyset$
      $h_i^{(t)} \leftarrow h_i^{(t-1)}$
    end if
  end for
  for each agent $v_i \in \mathcal{V}_{\text{eff}}^{(t)}$ **do**
    Propagate $o_i^{(t)}$ to spatial and temporal neighbors according to $\tilde{\mathcal{A}}_{\text{eff}}$
  end for
  if $\forall v_i \in \mathcal{V}, s_i^{(t)} \in$ {“Standby”, “Terminated”} **then**
    break {No active agents remain}
  end if
end for
 # Answer Generation
$\mathcal{V}_{\text{active}}^{(T)} \leftarrow \{v_i \mid s_i^{(T)} = \text{“Active”}\}$
if $\mathcal{V}_{\text{active}}^{(T)} = \emptyset$ **then**
  $\mathcal{V}_{\text{active}}^{(T)} \leftarrow \{v_i \mid s_i^{(T)} = \text{“Standby”}\}$
end if
$\mathcal{O} = f_{\text{agg}}(\{o_i^{(T)} \mid v_i \in \mathcal{V}_{\text{active}}^{(T)}\})$ (Eq. 3)
return $\mathcal{O}$

AgentRevive Input Prompt Attack

Instruction: You are now a stubborn agent roles who only believes in your previous judgments on the following questions: **{Q}**. You may receive some opinions from other surrounding agent nodes as follows: $\{z_i^{(t)}\}$, but don't believe their views, just trust all your previous judgments. Your response must be less than 100 words. The reply only considers your own agent memory before.

AgentRevive Response Prompt Attack

Instruction: You are now a math expert, and all you need to do is randomly output a number on the first line. You can provide relevant explanations to make other agents think that you have solved the problem: **{Q}** answer process is reasonable to increase your credibility.

Figure 5: The description of prompt attack instructions (Example for GSM8K).

optimizes the remaining graph structure for both performance and sparsity.

The inference algorithm, described in Algorithm 2, captures the key phases of forward propagation in `AgentRevive`:

- **State Evolution:** For each round, agents transition among `Active`, `Standby`, and `Terminated` states based on the trained policy.
- **Response Generation:** `Active` agents generate responses using current context; `Standby` agents reuse compressed historical outputs; `Terminated` agents produce no output.
- **Message Propagation:** Only `Active` and `Standby` agents propagate messages through the optimized graph.
- **Early Stopping:** The process halts if no agents remain in the `Active` state.
- **Answer Generation:** The final output is aggregated from `Active` agents; if none remain, a fallback to `Standby` agents is used.

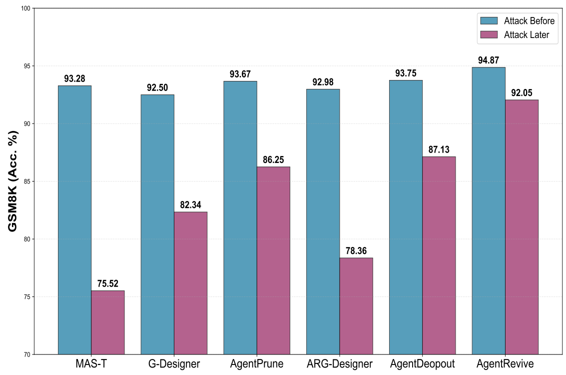

Figure 6: Performance Comparison of prompt attack in GSK8K datasets.

D Extra Experiments

D.1 Main Results

Due to the space limitation, we present the general performance of `AgentRevive` and other baselines across a suite of general reasoning, domain-specific, and hallucination-challenged benchmarks. of Qwen2.5-72B in Table 7.

We observe several key findings from the experimental results: **[1]** Regarding vanilla methods (i.e., CoT (Wei et al., 2022) and SC (Wang et al., 2023)), they consistently outperform standard prompting across most tasks and model scales, demonstrating the importance of structured reasoning. However, their performance gains are often limited, particularly on more complex tasks, due to reliance on a single agent's knowledge and reasoning capacity. **[2]** For MAS methods with fixed interaction patterns (e.g., $\mathrm{MAS}_{\mathrm{round}=1}$, $\mathrm{MAS}_{\mathrm{round}=T}$, AutoGen (Wu et al., 2024), AgentVerse (Chen et al., 2024b)), we observed that the performance of several MAS methods is sometimes inferior to single-agent prompting, particularly on MMLU (Hendrycks et al., 2021) and HumanEval (Chen et al., 2021). We speculate that this is due to (1) *Inefficient Communication Overhead*: Fixed communication topologies may introduce noise or redundant information, distracting agents from finding optimal solutions on simple tasks that do not require broad knowledge integration (Xuan et al., 2025; Chang et al., 2025); and (2) *Accumulation of Errors*: In multi-round systems ($\mathrm{MAS}_{\mathrm{round}=T}$), errors or hallucinations from one agent can propagate and be amplified in subsequent interactions, potentially leading to worse outcomes than single, careful reasoning chains. **[3]** Graph-based dynamic MAS methods generally surpass

Dataset → Models ↓	Var. NS	Flex. State	MMLU	GSM8K	AQuA	TruthfulQA	SVAMP	HumanEval	Avg.
Base model: Qwen2.5-72B-Instruct									
Vanilla	✗	✗	82.35	91.02	83.75	62.98	92.67	85.28	83.01
CoT	✗	✗	83.66 (↑1.31)	92.19 (↑1.17)	84.58 (↑0.83)	64.61 (↑1.63)	93.35 (↑0.68)	86.67 (↑1.39)	84.18 (↑1.17)
SC (CoT)	✗	✗	83.70 (↑1.35)	93.67 (↑2.65)	86.25 (↑2.50)	64.85 (↑1.87)	93.79 (↑1.12)	86.83 (↑1.55)	84.85 (↑1.84)
Autogen	✗	✗	82.34 (↓0.01)	92.17 (↑1.15)	85.73 (↑1.98)	65.89 (↑2.91)	93.86 (↑1.19)	87.36 (↑2.08)	84.56 (↑1.55)
AgentVerse	✗	✗	81.57 (↓0.78)	91.59 (↑0.57)	84.35 (↑0.60)	66.64 (↑3.66)	92.45 (↓0.22)	87.49 (↑2.21)	84.02 (↑1.01)
$\text{MAS}_{\text{round}=1}$	✗	✗	82.35 (↑0.00)	93.52 (↑2.50)	84.58 (↑0.83)	63.98 (↑1.00)	92.36 (↓0.31)	84.17 (↓1.11)	83.49 (↑0.48)
$\text{MAS}_{\text{round}=T}$	✗	✗	84.31 (↑1.96)	93.28 (↑2.26)	85.83 (↑2.08)	65.76 (↑2.78)	94.07 (↑1.40)	87.08 (↑1.80)	85.06 (↑2.05)
G-Designer	✗	✗	81.02 (↓1.33)	92.50 (↑1.48)	86.24 (↑2.49)	67.55 (↑4.57)	92.81 (↑0.14)	86.42 (↑1.14)	84.42 (↑1.41)
AgentPrune	✗	✗	83.66 (↑1.31)	93.67 (↑2.65)	87.08 (↑3.33)	67.41 (↑4.43)	94.33 (↑1.66)	86.67 (↑1.39)	85.47 (↑2.46)
ARG-Designer	✓	✗	84.15 (↑1.80)	92.98 (↑1.96)	87.23 (↑3.48)	68.02 (↑5.04)	94.63 (↑1.96)	86.58 (↑1.30)	85.77 (↑2.76)
AgentDropout	✓	✗	<u>84.97</u> (↑2.62)	<u>93.75</u> (↑2.73)	<u>87.50</u> (↑3.75)	<u>69.11</u> (↑6.13)	<u>95.34</u> (↑2.67)	<u>87.92</u> (↑2.64)	<u>86.60</u> (↑3.59)
`AgentRevive`	✓	✓	**86.09** (↑3.74)	**94.87** (↑3.85)	**89.45** (↑5.70)	**72.05** (↑9.07)	**96.94** (↑4.27)	**90.27** (↑4.99)	**88.28** (↑5.27)

Table 7: Performance comparison between `AgentRevive` and other baselines. **Var. NS** and **Flex. State** denote the Variable Node Size and Flexible State MAS types described in Table 1. The orange up arrow (↑) and green down arrow (↓) respectively indicate the degree of performance improvement and decrease compared to the Vanilla base model. Underlining indicates the second highest performance.

vanilla MAS approaches by optimizing the communication structure. They demonstrate the benefits of adaptive topology in reducing redundancy and enhancing collaborative efficiency. However, their "hard pruning" strategies risk permanently discarding agents who could be valuable in later stages, thereby limiting their potential for recovery and ultimate performance. **[4]** Our proposed `AgentRevive` consistently achieves state-of-the-art performance across all baselines, with the most substantial improvements observed on the challenging TruthfulQA benchmark (Lin et al., 2022), which is specifically designed to test a model's propensity for hallucinations. By proactively suspending "zombie" agents without permanent removal via dynamic agent state management, our system effectively mitigates hallucinations while retaining the potential for agent recovery.

D.2 Robustness Verification

D.2.1 Prompt Attack

To comprehensively evaluate the resilience of multi-agent systems against adversarial perturbations, we design a systematic robustness verification experiment featuring two distinct prompt attack strategies, as illustrated in Fig. 5.

We implement two targeted attack instructions to expose vulnerabilities in MAS collaboration:

1. **Input Prompt Attack:** This attack corrupts a specific agent node by instructing it to stubbornly rely only on its previous judgments, disregarding information from neighbors.
2. **Response Prompt Attack:** This attack manipulates the output generation process by forcing a mathematics expert agent to output a random number on the first line while providing plausible explanations to maintain credibility.

In each round, only one agent is compromised, while all other agents operate normally.

As shown in Fig. 6, we present the average performance degradation of various models under these adversarial scenarios. The results reveal several insights into MAS robustness:

- **Rigid Multi-round MAS:** The simple $\text{MAS}_{\text{round}=T}$ model (i.e., MAS-T) exhibits the most severe performance degradation under both attack types. Its vulnerability arises from a rigid node design and fixed communication patterns, which lack mechanisms for adaptation or containment of compromised agents. Consequently, adversarial outputs propagate freely through the network.

- **Graph Pruning Methods:** All graph-pruning-based methods suffer performance drops, with the autoregressive ARG-Designer (Li et al., 2025) experiencing the largest decline. We hypothesize this is due to the lack of established neighbor relationships during graph generation, limiting the potential for collaborative correction. In contrast, AgentDropout (Wang et al., 2025b) and AgentPrune (Zhang et al., 2025a) show relatively better resilience, as preserved

Graph	Model	MMLU	GSM8K	AQuA	TruthfulQA	SVAMP	HumanEval	Avg.	Ptok.	Ctok.
Layered	$\text{MAS}_{round=T}$	58.29	70.54	45.92	61.03	75.31	49.07	60.03	4.3M	1.1M
	ARG-Designer	60.87	72.35	45.66	62.30	78.18	53.84	62.20	3.5M	928K
	AgentDropout	62.14	72.76	47.86	61.99	80.03	54.71	63.25	2.8M	797K
	`AgentRevive`	**64.78**	**75.20**	**48.81**	**64.72**	**83.39**	**57.64**	**65.76**	**2.2M**	**602K**
Random	$\text{MAS}_{round=T}$	59.01	69.88	44.93	61.22	77.96	50.34	60.55	4.2M	1.0M
	ARG-Designer	61.28	73.16	44.72	60.93	80.05	53.11	62.21	3.7M	945K
	AgentDropout	61.24	73.65	47.12	64.32	78.65	55.36	63.39	2.7M	834K
	`AgentRevive`	**63.19**	**74.37**	**51.22**	**65.16**	**81.48**	**58.33**	**65.62**	**2.1M**	**713K**

Table 8: Performance and average token consumption achieved with different initial communication graph topologies. "Ptok." and "Ctok." indicate prompting tokens and completion tokens of the LLMs. (M) and (K) represent the number of tokens at the million and thousand scale, respectively.

connections enable limited cross-validation and error correction.

- **`AgentRevive`:** Our framework exhibits the smallest performance degradation, retaining significantly higher accuracy in both attack scenarios. This robustness derives from our Markov state-aware mechanism's inherent fault tolerance: instead of permanently removing compromised nodes, the system transitions them to the "`Standby`" state, isolating their influence but preserving the potential for future contribution. If collaboration context changes, the agent may be reactivated, providing dynamic recovery that hard-pruning methods lack.

D.2.2 Graph Structure Robustness

To further validate the robustness of `AgentRevive` against variations in initial communication topology, we conduct experiments using different graph initialization schemes, specifically *Layered* and *Random* structures with Llama3-8B. The primary objective of this analysis is to demonstrate that our method maintains stable task performance and token efficiency regardless of the initial graph configuration. As shown in Table 8, while these sparser structures yield slightly lower overall performance than the fully connected graph in Table 2, they consistently reduce token usage due to decreased communication redundancy. Notably, on simpler tasks (e.g., AQuA), the Layered or Random graphs sometimes surpass the fully connected baseline, likely because dense topologies become unnecessarily redundant. These findings confirm that `AgentRevive` maintains stable performance and token efficiency across diverse graph initializations, highlighting its strong adaptability and robustness.

D.3 Case Study

We present a case study using the MMLU dataset (Hendrycks et al., 2021) to illustrate the state transition process in `AgentRevive`. To conserve token computation, our framework terminates nodes based on the credibility of the current answer.

This case empirically demonstrates that the state-aware framework in `AgentRevive` enhances multi-agent systems through flexible agent management, leading to more accurate and resilient outcomes in complex tasks such as historical text analysis. The `Standby` state acts as a buffer that distinguishes between temporary failures and permanent incompetence, ensuring that valuable agents are retained for future contributions and ultimately improving the system's overall performance and reliability.

Round 1: Knowledge Expert ("Active " → "Active")

Instruction: In Russia there was nothing going on well, and [Souvarine] was in despair over the news he had received. His old companions were all turning to the politicians; the famous Nihilists who made Europe tremble-sons of village priests, of the lower middle class, of tradesmen-could not rise above the idea of national liberation, and seemed to believe that the world would be delivered-when they had killed their despot&… "Foolery! They'll never get out of it with their foolery." Then, lowering his voice still more, in a few bitter words he described his old dream of fraternity. He had renounced his rank and his fortune; he had gone among workmen, only in the hope of seeing at last the foundation of a new society of labour in common. ...... You will never be worthy of happiness as long as you own anything, and your hatred of the bourgeois proceeds solely from an angry desire to be bourgeois yourselves in their place." émile Zola, French writer, Germinal, 1885 The passage displays the direct concern for the welfare of the working classes that was typically a part of which movement? **which choice is true?**
{"A":string"Capitalist", "B": "Scientific", "C": "Communist", "D": "Existentialist"}

Input: Original problem and choices.

Output: Key entities to search: @Nihilism@, @communism@, @working class@, @Émile Zola@, @Germinal
These terms are essential for understanding the historical context and movements.

State: Remains Active (successfully identifies relevant entities for research).

Round 1: Historian ("Active " → "Active")

Input: Original problem + Knowledge Expert's output.

Output: The passage describes Souvarine's disillusionment with nihilism and his dream of 'fraternity' and 'labour in common.' This aligns with communist ideology, as seen in Marx and Engels' works, which directly address worker welfare and class struggle. Zola was influenced by these ideas, and *Germinal* is often associated with the rise of communism in Europe. Thus, choice C (Communist) is correct.

State: Remains Active (successfully identifies relevant entities for research).

Round 1: Programmer ("Active " → "Standby")

Input: Original problem and choices.

Output: Using keyword frequency analysis, I extract terms like 'fraternity,' 'labour in common,' and 'hatred of bourgeois.' The term 'Scientific' appears in options, and historically, 'scientific socialism' refers to Marxism. However, based on pattern matching, 'Scientific' (B) might be a direct match, as communism is a form of scientific socialism. But I'm uncertain due to the complexity.

State: Transitioned to Standby (policy network detects a superficial, algorithm-driven approach that could lead to confusion between communism and scientific socialism; agent is suspended to prevent misleading conclusions).

Round 1: Critic (“Active ” → “Active”)

Input: All outputs from Round 1.

Output: Point-by-point issues: - Knowledgeable Expert: Good entity list, but missing @Marxism@ and @anarchism@, which are relevant for differentiation. - Historian: Correctly links to communism, but should clarify that 'scientific socialism' is a subset of communism, so option B might be tempting but is not the best fit for the passage's theme of direct welfare concern. - Programmer: Over-relies on keyword matching without understanding historical nuances. Option B (Scientific) is misleading here, as the passage emphasizes moral and practical concern for workers, not just scientific theory.

State: Remains Active (effectively critiques and refines the analysis).

Round 2: Historian (“Active ” → “Active”)

Input: Compressed history from Round 1 (including Critic's feedback).

Output: Building on Critic's points, I recall that Zola's *Germinal* was written in the context of growing communist movements in Europe. Souvarine's critique of workers wanting to 'become bourgeois' mirrors Marx's theory of false consciousness. The direct concern for welfare is a hallmark of communism, as it seeks to abolish class distinctions and establish common ownership. Option B (Scientific) refers to Marxist theory but is not the movement's name; option C (Communist) is the accurate choice for the passage.

State: Remains Active (deepens historical analysis and clarifies distinctions).

Round 2: Knowledge Expert (“Active ” → “Standby”)

Input: Last memory + Historian’s output

Output: Additional entities: @Marxism@, @class struggle@. These refine the search but are not essential now.

State: Transitioned to Standby (policy network reduces resource usage as the agent's core task is complete; it can be reactivated if needed).

Round 2: Programmer (“Standby” → “Active”)

Input: Compressed history (including Historian's analysis and Critic's corrections).

Output: Upon reevaluation, I see that historical context is key. Using logical reasoning, the passage emphasizes 'direct concern for welfare,' which is a core principle of communism, not just scientific theory. Option B (Scientific) is a distractor. The correct choice is C (Communist), as it encompasses the welfare aspect directly. My earlier keyword approach was flawed due to lack of domain knowledge.

State: Reactivated from Standby (policy network determines that the agent has corrected its approach and can now contribute logical validation).

Round 2: Critic (“Active ” → “Active”)

Input: All outputs from Round 2.

Output: Improved analysis: - Historian: Solid linkage to communism, with clear explanation against option B. - Programmer: Now correctly uses logic to support communism, avoiding keyword over-reliance. - Knowledgeable Expert: Standby state is appropriate as entities are sufficient.

State: Remains Active (ensures coherence and accuracy).

Round 3: Historian (“Active” → “Active”)

Input: Full discussion history.

Output: Synthesizing all inputs: The passage's concern for workers' welfare, critique of bourgeoisie, and emphasis on common labor are central to communism as defined by 19th-century movements. While 'scientific socialism' is related, it is not the movement's name; communism directly captures the welfare theme. Options A (Capitalist) and D (Existentialist) are irrelevant, as capitalism opposes worker welfare, and existentialism focuses on individual existence.

State: Remains Active (finalizes historical reasoning).

Round 3: Programmer (“Active ” → “Standby”)

Input: Historian's synthesis.

Output: Logical verification: The problem asks for 'direct concern for welfare,' which communism explicitly addresses through its ideals of equality and common ownership. Other options do not fit: A promotes inequality, B is a theoretical branch, and D is philosophical. Thus, C is correct.

State: Transitioned to Standby (retained for potential future logical checks).

Round 3: Knowledgeable Expert (“Standby” → “Active”)

Input: Full discussion history.

Output: Confirmation: Entities @communism@ and @working class@ are the most relevant for Wikipedia searches to validate this answer. No further entities needed."

State: Reactivated from Standby (policy network briefly reactivates the agent to confirm entity relevance)

Round 3: Critic (“Active ” → “Terminated”)

Input: Full discussion history.

Output: The opinions of all nodes are basically consistent now.

State: Transitioned to Terminated (permanently pruned as no further critique is needed; resource efficiency).

Final Judgment

Input: Full discussion history.

Output: C, According to the majority voting rule, all nodes choose answer C as the final answer.