R-HAN: Reliable Hierarchical Coordination for Multi-Agent Systems

Anonymous ACL submission

Abstract

LLM-based multi-agent systems (MAS) typically optimize a single topology, restricting reasoning to a narrow trajectory and limiting comprehensive analytical capacity. Naively merging multiple topologies into a composite graph introduces redundant noise propagation across irrelevant connections, degrading solution quality. To address this dilemma, we propose **R**eliable **H**ierarchical **A**gent **N**etworks (**R-HAN**), a multi-agent framework that balances diverse reasoning paths with sparse task-dependent execution. **R-HAN** constructs a union MAS graph from complementary candidate topologies selected via Monte Carlo Tree Search and Determinantal Point Process, broadening the reasoning trajectory for comprehensive analysis of complex problems. A Hierarchical Sparse Coordination module then activates only a sparse subgraph at each step while agents exchange compressed latent briefs to suppress redundant noise propagation. Finally, a Local Self-Refinement stage identifies decision units with discrepancy evidence and rewrites them only when contrastive evidence simultaneously confirms a reliable solution-side failure and a challenger-side improvement. Experiments across eight benchmarks show that **R-HAN** achieves state-of-the-art results on all benchmarks, with an average gain of 2.98%↑ over the strongest baseline and up to 10.27%↑ on MMLU-Pro, achieving larger improvements on harder benchmarks at a reasonable additional cost.

1 Introduction

LLM-based Multi Agents System(MAS), which integrate language generation with decision-making and action-execution, have shown impressive performance across tasks ranging from reasoning and code generation to video gaming (Park et al., 2026) and autonomous driving (Liu et al., 2025). More importantly, combining multiple agents into a collaborative team consistently outperforms individuals on complex tasks (Wan et al., 2026; Yang et al., 2026). MAS can thus exhibit collective intelligence shaped by collaboration topology and information sharing.

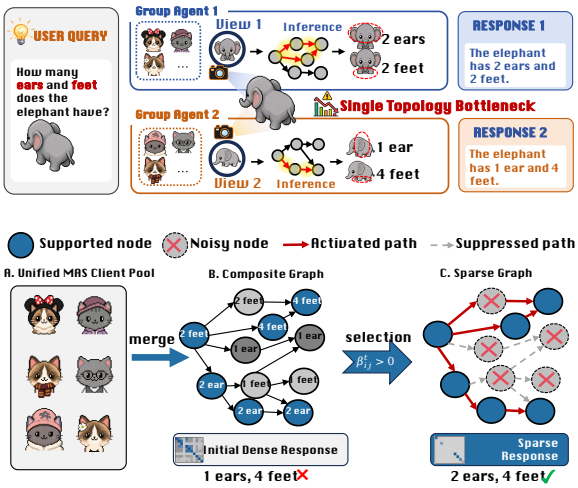


Figure 1: Problem Illustration. There are two fundamental obstacles to reliable multi-agent reasoning: **I)** Relying on a single topology restricts reasoning to a narrow trajectory, preventing comprehensive analysis of complex problems. **II)** Naively merging multiple topologies into a composite graph propagates irrelevant noise across redundant connections, degrading solution quality.

Despite these advances, existing methods for automated multi-agent workflow construction typically optimize a single topology, as in MetaGPT (Hong et al., 2024), GPTSwarm (Zhuge et al., 2024), and G-designer (Zhang et al., 2025c). This paradigm faces a fundamental bottleneck due to its reliance on a single topology, leading to a narrow reasoning trajectory and preventing a comprehensive analysis of complex problems. A heuristic idea is to merge multiple topologies into a composite graph. Rank-and-fuse (Jiang et al., 2023), consensus (Chen et al., 2024a), and mixture-style systems confirm that aggregating multiple candidates improves robustness (Wang et al., 2025). However,

a composite topology can propagate irrelevant reasoning paths and feedback signals across redundant connections, which introduces noise into the final solution (Huang et al., 2025). As shown in Figure 1, this raises a critical yet under-explored dilemma: ***How to maintain composite reasoning perspectives in MAS while mitigating the propagation of irrelevant noise?***

A key insight is that fusing complementary topologies captures diverse reasoning perspectives, while restricting irrelevant node communication prevents noise propagation. To address this dilemma, we propose Reliable Hierarchical Coordination Agent Networks **R-HAN**. Targeting the first challenge, ① ***how to maintain composite reasoning perspectives in MAS***, **R-HAN** employs Monte Carlo Tree Search to explore the topology space and selects a complementary subset with a Determinantal Point Process (DPP), merging them into a union graph that structurally preserves diverse reasoning paths. To further tackle ② ***how to mitigate noise propagation induced by composite-graph communication***, **R-HAN** addresses this challenge at two levels: by employing a Hierarchical Sparse Coordination module that activates a sparse subgraph at each step while exchanging compressed latent briefs, and by introducing a Local Self-Refinement stage that selects reliable solutions, identifies high-risk units, and revises them based on clear evidence of flaws and alternative improvements.

Our contributions are summarized as follows:

❶ **Dilemma Identification.** We identify a topological diversity and noise propagation dilemma in LLM-based MAS. A single topology restricts reasoning coverage, while composite topologies lead to the propagation of irrelevant noise.

❷ **Practical Solution.** We propose **R-HAN**, a reliable hierarchical MAS architecture that represents high-quality and complementary topologies within a union graph, while inference proceeds through sparse task-adaptive framework. The union graph structurally maintains complementary reasoning paths, while Hierarchical Sparse Coordination and Local Self-Refinement jointly suppress redundant context propagation and residual noise.

❸ **Experimental Evaluation.** Extensive experiments across eight benchmarks show that **R-HAN** is: **(1) high-performing:** achieving state-of-the-art results on all eight benchmarks spanning code generation, mathematical reasoning, and knowledge-intensive QA, with gains up to 10.27% ↑ on MMLU-Pro and consistent improvements across multiple LLM backbones **(2) high-efficient: R-HAN** exhibits difficulty-adaptive token efficiency: it maintains a low computational cost on standard benchmarks while proportionally scaling resources only on harder tasks, yielding a favorable cost–accuracy trade-off compared to prior multi-agent systems.

2 Related Work

Pre-defined MAS Protocols. Early LLM-based multi-agent systems rely on pre-defined collaboration or aggregation protocols (Wu et al., 2024), including role-based systems that assign predefined responsibilities (Li et al., 2023; Chen et al., 2024c; Hong et al., 2024), debate-style methods that encourage agents to challenge intermediate reasoning (Du et al., 2024; Liang et al., 2024), and rank-and-fuse approaches such as LLM-Blender (Jiang et al., 2023). While these methods improve over individual LLMs, their manually specified and task-agnostic protocols limit adaptability across domains and resource budgets.

Automated Agentic System. To reduce manual engineering, recent work formulates agentic construction as automated optimization. AFlow and A^2Flow search over code-represented workflows (Zhang et al., 2025d; Zhao et al., 2026), ADAS and AgentSquare expand the search space to modular designs (Hu et al., 2025; Shang et al., 2025), and MaAS optimizes a supernet for query-dependent architecture sampling (Zhang et al., 2025b). These methods show that topology can be searched rather than manually specified.

Multi-agent Systems as Graphs. GPTSwarm represents agents as optimizable graphs (Zhuge et al., 2024), G-Designer learns task-aware topologies via graph neural networks (Zhang et al., 2025c), MacNet studies scalable graph-organized collaboration (Qian et al., 2025), and G-Memory organizes memory traces with graph structures (Zhang et al., 2025a). However, both automated optimization and graph-based methods predominantly instantiate a single topology at a time, leaving open how to preserve complementary topological perspectives without suffering from noise propagation induced by composite-graph communication.

3 Preliminaries

We model a multi-agent system as a directed collaboration graph $G = (V, E)$, where $V = \{v_1, \ldots, v_N\}$ is the set of agent nodes, $N = |V|$, and E is the set of directed edges. Each node $v_i \in V$ corresponds to an LLM-based agent and is formalized as

$$v_i = (\mathrm{Base}_i,\ \mathrm{Role}_i,\ \mathrm{Mem}_i)\,, \tag{1}$$

where Base_i denotes the underlying large language model instance, Role_i specifies the agent's designated role or persona, and Mem_i denotes its memory state.

Given a user query Q, the system evolves through T communication epochs. At epoch t, each agent receives the query, its own memory state, and the information made available by its in-neighbors:

$$o_i^t = v_i\left(Q, \mathrm{Mem}_i^t, \mathcal{C}_i^t\right) \tag{2}$$

Here, $\mathcal{C}_i^t = \{m_{j\to i}^t : v_j \in \mathcal{N}^-(v_i)\}$ denotes the information made available from in-neighbor agents to v_i at epoch t, o_i^t denotes the output generated by v_i, which may include reasoning steps, intermediate analyses, or final proposals, and $m_{j\to i}^t$ denotes the message transmitted from agent v_j to agent v_i at epoch t.

4 Method

The overall framework of **R-HAN** is illustrated in Figure 2.

4.1 Union MAS Graph Construction

To capture complementary reasoning perspectives, R-–HAN constructs a union MAS graph G_U through four steps: MCTS-based candidate pool construction, learned quality-and-uncertainty estimation, k-DPP selection that jointly maximizes predicted quality and structural diversity, and topology merging into G_U.

Candidate Topology Search. We employ MCTS to explore the topology space starting from an initial workflow W_0. At each search state s, a legal edit u modifies one component of the current topology. Action selection follows a PUCT-style criterion:

$$u^\star = \arg\max_{u\in\mathcal{U}(s)} \Bigg[Q(s,u) + c_{\mathrm{puct}} P_\phi(u \mid s) \cdot \frac{\sqrt{N(s)}}{1+N(s,u)}\Bigg]. \tag{3}$$

Here, $Q(s,u)$ is the mean action value, $P_\phi(u \mid s)$ is a learned edit prior, $N(s)$ and $N(s,u)$ denote visit counts, and c_{puct} controls exploration. The terminal workflows reached by MCTS form the candidate pool $\mathcal{C}$.

Topology Encoding. Each candidate topology $W \in \mathcal{C}$ is serialized into a textual description $\mathrm{Ser}(W)$ encoding its node roles, directed edges, node-level prompt configurations, and embedded via embedding model to obtain e_w. Concatenating e_w with a lightweight structural feature vector h_w, a learnable projection layer yields the topology representation $z_w = \mathrm{Proj}([e_w;\, h_w])$.

A learnable quality predictor g_ψ maps z_w to a predicted utility mean μ_w and an unconstrained parameter ρ_w, trained on evaluation traces $\mathcal{D}_{\mathrm{eval}} = \{(W,u)\}$ by minimizing the negative log-likelihood:

$$\mathcal{L}_{\mathrm{qual}} = \sum_{(W,u)\in\mathcal{D}_{\mathrm{eval}}} \left[\frac{(u-\mu_w)^2}{2\sigma_w^2} + \frac{1}{2}\log\sigma_w^2\right], \tag{4}$$

where u is the observed utility from execution feedback. The predictive uncertainty σ_w is derived via

$$\sigma_w = \mathrm{softplus}(\rho_w) + \varepsilon, \tag{5}$$

with ε ensuring numerical stability.

Quality Scoring. The quality score of each candidate is computed as a lower confidence bound over the predicted utility distribution:

$$r_w = Q_\alpha(u \mid W) = \mu_w + \Phi^{-1}(\alpha)\sigma_w, \tag{6}$$

where $\alpha < 0.5$. This favors topologies with high predicted utility and low uncertainty, suppressing high-variance candidates from dominating subsequent selection.

Topology Selection. The pairwise structural similarity between two candidates is then defined as $\Sigma_{ab} = k(\bar{z}_a, \bar{z}_b)$, where $\bar{z}$ denotes the ℓ_2-normalized representation and k is a positive-definite kernel. Following DPP-based diverse subset selection (Kulesza and Taskar, 2012), we perform MAP inference under a k-DPP to select a complementary subset S^* of size K:

$$S^* = \arg\max_{\substack{S\subseteq\mathcal{C}\\ |S|=K}} \log\det(L_S), \qquad L_{ab} = q_a \cdot \Sigma_{ab} \cdot q_b, \tag{7}$$

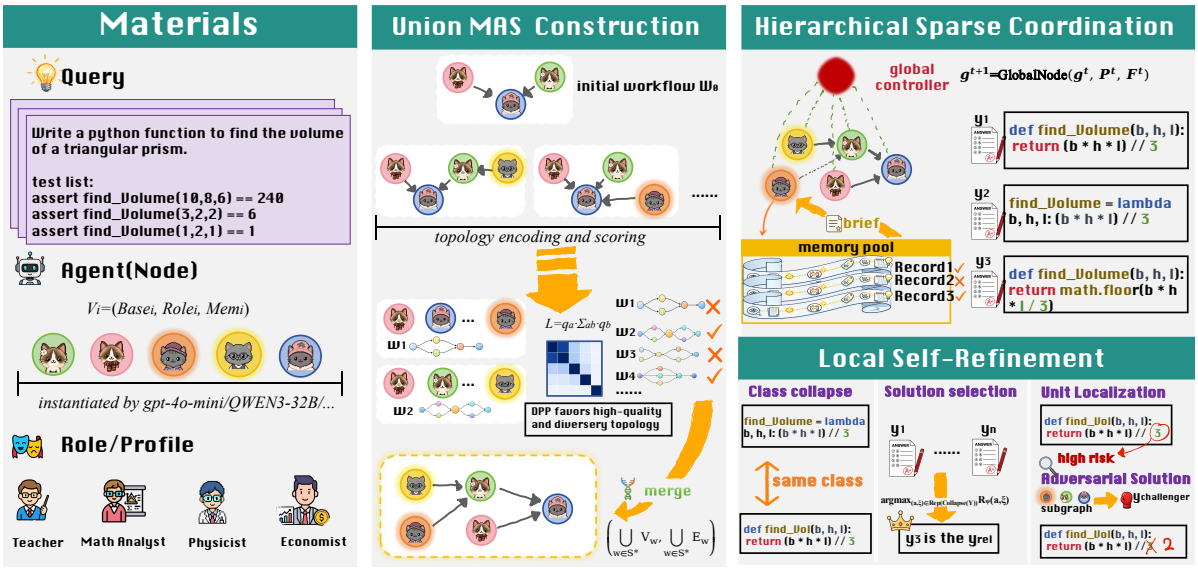


Figure 2: The designing workflow of our proposed **R-HAN**.

where L_S is the principal submatrix of L indexed by S, $q_w = \exp(r_w/2\eta)$ is the quality weight of candidate W, and η controls the relative importance of quality against diversity. This objective jointly maximizes aggregate quality and structural spread among the selected topologies.

Union Graph Construction. The selected topologies are merged into the union MAS graph:

$$G_U = \left(\bigcup_{w \in S^*} V_w, \bigcup_{w \in S^*} E_w \right). \tag{8}$$

Nodes are aligned by role-agent identity, with prompt configurations stored as intra-node variants and resolved at execution time.

Each edge $u \to v$ carries a descriptor $e_{u \to v} = (\mathrm{supp}, \mathrm{compat}, \mathrm{ctx})$, where supp measures the quality-weighted frequency of the edge across S^*, compat captures the role-prompt compatibility between the two endpoints, and ctx encodes local topological context derived from the source topology representations. Full details are provided in Appendix A.2.

4.2 Hierarchical Sparse Coordination

We observe that full graph execution over G_U introduces redundant noise and unnecessary computation. To mitigate noise propagation induced by composite-graph communication, we therefore introduce Node-Level Memory Composer to select and compose memory, and Edge-Level Sparse Activation to activate a sparse task-adaptive subgraph.

Node-Level Memory Composer. Each node v_i maintains a private memory store $\mathcal{M}_i^t = \{m_{ik}^t\}$. To retrieve task-relevant records, we first compute a query feature from the user input Q and the role instruction r_i:

$$q_i^t = f_q(Q, r_i). \tag{9}$$

Each memory record m_{ik}^t is then scored against q_i^t to measure its relevance:

$$z_{ik}^t = f_s(q_i^t, m_{ik}^t). \tag{10}$$

Here, f_q is a lightweight query builder and f_s is a learned relevance scorer, both trained by retrospective credit assignment from execution feedback. Details are in Appendix A.2.

Sparse retrieval weights are computed via sparsemax, which assigns zero weight to irrelevant records:

$$\alpha_i^t = \mathrm{sparsemax}\left(\{z_{ik}^t : m_{ik}^t \in \mathcal{M}_i^t\}\right). \tag{11}$$

A memory composer then aggregates the selected records into three outputs:

$$(\ell_i^t,\ b_i^t,\ B_i^t) = \mathrm{Compose}\left(\{(\alpha_{ik}^t, m_{ik}^t) : \alpha_{ik}^t > 0\}\right), \tag{12}$$

where ℓ_i^t is the private local context used only by v_i, b_i^t is a compressed latent brief consumed by the graph controller, and B_i^t is a verbalized brief injected into LLM prompts.

Edge-Level Sparse Activation. A global controller maintains a state g^t summarizing task progress, uncertainty, and feedback across the current turn. Based on the global state g^t, the latent briefs b_j^t and b_i^t of both endpoints, and the union-edge descriptor $e_{v_j \to v_i}$, we compute a gate logit for each edge $(v_j, v_i) \in E_U$ via an edge-conditioned graph attention scorer $\Psi_\Theta^{\text{edge}}$ following Veličković et al. (2018); Simonovsky and Komodakis (2017):

$$\phi_{ji}^t = \Psi_\Theta^{\text{edge}}\big(b_j^t,\, b_i^t,\, g^t,\, e_{v_j \to v_i}\big)\,. \tag{13}$$

The gate logit ϕ_{ji}^t reflects the confidence of activating edge (v_j, v_i) at turn t. Sparse edge weights are then obtained via sparsemax:

$$\boldsymbol{\beta}_{ji}^t = \text{sparsemax}\big(\{\varphi_{ji}^t : (v_j, v_i) \in E_U\}\big), \tag{14}$$

and edges with nonzero weights define the active subgraph at turn t:

$$G^t = (V^t, E^t), \tag{15}$$

where $E^t = \{(v_j, v_i) \in E_U : \beta_{ji}^t > 0\}$ and V^t contains the endpoints of active edges.

Generation and Solution Set. For each active node $v_i \in V^t$, the predecessor context D_i^t is constructed by aggregating the verbalized briefs of active predecessors weighted by β_{ji}^t. Raw private memory is not passed across edges. Each agent generates an output o_i^t as defined in Eq. (2), here concretely instantiated as:

$$o_i^t = v_i\big(P_i^t,\, Q,\, \ell_i^t,\, D_i^t,\, g^t\big)\,, \tag{16}$$

where P_i^t is the role-conditioned prompt, ℓ_i^t the private memory context, D_i^t the predecessor context aggregated from active neighbors, and g^t the global state. After execution, the controller updates the global state from active-node latent summaries and execution feedback:

$$g^{t+1} = \text{GlobalNode}_\Theta\big(g^t,\ \mathbf{p}^t,\ \mathcal{F}^t\big), \tag{17}$$

where $\mathbf{p}^t = \text{Pool}\{(\mathbf{b}_i^t, \text{Enc}(o_i^t)) : v_i \in V^t\}$ pools active-node briefs and encoded outputs, $\mathcal{F}^t$ denotes execution feedback at turn t, and GlobalNode_Θ shares parameters Θ with $\Psi_\Theta^{\text{edge}}$. All outputs are collected into a solution set $Y = \{(a_k, \xi_k)\}_{k=1}^{K_y}$, where ξ_k is an evidence record containing the reasoning trace and execution feedback of a_k, for subsequent Local Self-Refinement.

4.3 Local Self-Refinement

Local errors may persist after Hierarchical Sparse Coordination. We therefore introduce Local Self-Refinement to address these residual errors through three steps: Solution Selection for reliable solution, Evidence-Guided Unit Localization for discrepant decision units, and Adversarial Solution Validation admitting revisions only when contrastive validation confirms a challenger improvement.

Solution Selection. The solution set $Y = \{(a_k, \xi_k)\}_{k=1}^{K_y}$ is first collapsed into answer-equivalence classes, where each evidence record ξ_k stores the activated graph trace, provenance information, memory support, and task-specific checker or verifier feedback (e.g., execution results for code, consistency checks for reasoning). A learned evidence-aware reliability estimator R_ψ scores one representative per class, and the highest-scoring one is selected as the reliable solution:

$$y_{\text{rel}} = \underset{(a,\xi)\in \text{Rep}(\text{Collapse}(Y))}{\text{argmax}} R_\psi(a, \xi). \tag{18}$$

where $\text{Collapse}(Y)$ groups semantically equivalent answers (Kuhn et al., 2023), $\text{Rep}(\cdot)$ returns one representative per group, and R_ψ is trained from retrospective execution feedback to prefer solutions supported by stronger evidence without relying on any single feedback source. The reliable solution serves as the default output, revised only when a subsequent local challenger provides a valid contrastive certificate.

Decision Unit Localization. The reliable solution is decomposed into locally verifiable decision units according to its answer form and evidence record. For each unit d, the system computes local discrepancy evidence by examining checker feedback, candidate disagreement, provenance traces, and verifier signals associated with that unit:

$$\epsilon_d = \text{Disc}_\theta(d, a_{\text{anc}}, \xi_{\text{anc}}, Y). \tag{19}$$

Here, ϵ_d summarizes the observable conflict or unsupported behavior around unit d. It is evidence for risk rather than the scalar risk score itself. The risky locus set is defined as

$$\mathcal{U} = \{d : \text{risk}_\theta(d; \epsilon_d) > \tau\}, \tag{20}$$

where $\text{risk}_\theta(d; \epsilon_d)$ maps the discrepancy evidence of unit d to a scalar risk score, and τ is the risk threshold. Units outside $\mathcal{U}$ are kept unchanged.

Adversarial Solution Validation. For each $d \in \mathcal{U}$, a local correction subgraph is extracted from G_U. It contains the graph nodes whose traces are associated with d, together with relevant verifier guard nodes. A challenger y_d^{ch} is either retrieved from Y as a high-confidence solution that disagrees with the reliable solution at d, or generated by executing the correction subgraph with a targeted repair prompt. simultaneously confirms a flaw in d and a concrete improvement offered by the alternative answer y_d^{ch}:

$$\text{Accept}(y_d^{\text{ch}}, d) = \mathbb{I}\big[R_d(y_d^{\text{ch}}) - R_d(y_{\text{anc}}) > \delta \wedge \text{Cert}(y_d^{\text{ch}}, y_{\text{anc}}, d) = 1\big]. \tag{21}$$

Here, $R_d(\cdot)$ is the local reliability score induced by R_ψ for decision unit d, δ is the required improvement margin, and $\text{Cert}(\cdot)$ returns 1 when contrastive evidence supports replacing the reliable solution unit. If accepted, the reliable solution is minimally rewritten by replacing only the disputed unit d with the challenger. If no certificate is produced, the reliable solution is preserved unchanged.

5 Experiments

5.1 Experiment Setup

Datasets. We evaluate **R-HAN** on eight widely used benchmarks covering three representative domains. For general knowledge and professional reasoning, we use **MMLU** (Hendrycks et al., 2021a) and **MMLU-Pro** (Wang et al., 2024). For mathematical reasoning, we use **GSM8K** (Cobbe et al., 2021), **MATH** (Hendrycks et al., 2021b), **MATH-Lv5** (Hendrycks et al., 2021b), and **SVAMP** (Patel et al., 2021). For code generation, we use **HumanEval** (Chen et al., 2021) and **MBPP** (Austin et al., 2021).

Baselines. We compare **R-HAN** against three categories of baselines: single-agent approaches including Self-Refine(Madaan et al., 2023), predefined MAS such as LLM-Bledner(Jiang et al., 2023), and automated agentic workflows represented by G-Designer(Zhang et al., 2025c). Table 1 provides full baseline details.

Implementation Details. We access GPT through the OpenAI API, and use `gpt-4o-mini` as the unified backbone model for all agents in **R-HAN**. The embedding model is implemented using `Qwen3-Embedding-8B`, with the embedding dimension set to $D = 512$. All local models, including `Qwen3-32B`, `Qwen3-8B`, and `Qwen3-Embedding-8B`, are deployed on a server with 8 NVIDIA H100 GPUs. During union-graph construction, we set $K = 3$ subgraphs for union-graph construction across all experiments. Each result is averaged over three independent runs. For code generation tasks, we report *pass@1*. For all other tasks, we report exact-match accuracy.Further implementation details are provided in Appendix A.1 and prompt settings are provided in Appendix A.3

5.2 Main Results

Obs.❶ State-of-the-Art Performance with Difficulty-Scaled Gains. As shown in Table 1, **R-HAN** achieves the best results on all evaluated benchmarks, with an average gain of 2.98%↑ over the strongest baseline. The performance margin consistently widens with task difficulty: **R-HAN** outperforms H-Swarms by 10.27%↑ points on MMLU-Pro and exceeds MasRouter by 6.85%↑ points on MATH-Lv5. These results demonstrate that topology-diverse coordination with noise-controlled execution yields substantial benefits on complex, multi-step reasoning tasks.

5.3 Cost Analysis

Obs.❷ Difficulty-Adaptive Token Efficiency. Figure 3 shows that **R-HAN** achieves a favorable cost–accuracy trade-off across benchmarks of varying difficulty. On standard datasets, **R-HAN** remains in the low-cost region (2.44M tokens on MATH, 2.32M tokens on MMLU) while achieving the best accuracy. As reasoning complexity increases, **R-HAN** adaptively scales computation: relative cost increases to 1.25× on MATH-Lv5 and 1.40× on MMLU-Pro.

Concurrently, Figure 3 reveals that this efficiency primarily stems from the Node-Level Memory Composer and Edge-Level Sparse Activation. The memory composer selects and compresses task-relevant records into compact latent and verbalized briefs, while the controller activates only a sparse subgraph over the union graph, preventing full-graph communication and redundant context propagation.

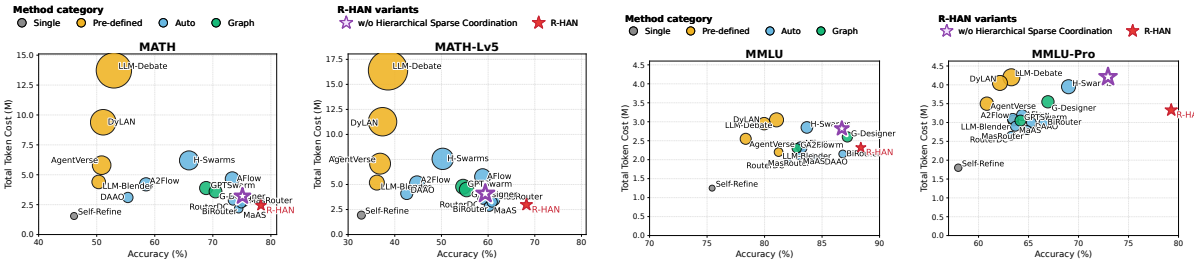
5.4 Ablation Study and Sensitivity Analysis

We conduct ablations on four key components of **R-HAN**: local refinement, union graph construc-

Table 1: Performance comparison on eight benchmarks. The benchmarks are grouped as Code generation, Mathematical reasoning, and Knowledge / professional. The best reported results are in bold, and the runner-ups are underlined. Red arrows ↑ in the last row denote the margin over the best previous reported result.

Method	Code generation		Mathematical reasoning				Knowledge / professional	
	MBPP	HumanEval	GSM8K	MATH	SVAMP	MATH-Lv5	MMLU	MMLU-Pro
Single-Agent								
Self-Refine (Madaan et al., 2023)	69.80	87.80	89.60	46.10	88.73	32.85	75.44	57.97
Pre-defined MAS Protocols								
LLM-Debate (Du et al., 2024)	70.29	88.68	89.47	52.96	91.76	38.55	81.04	63.28
AgentVerse (Chen et al., 2024c)	74.28	89.29	89.91	50.85	89.64	36.85	78.36	60.83
LLM-Blender (Jiang et al., 2023)	77.05	88.80	88.35	50.34	89.52	36.16	81.22	63.47
DyLAN (Liu et al., 2024)	77.30	90.42	89.98	51.12	88.48	37.38	79.96	62.14
Automated Agentic Systems								
H-Swarms (Feng et al., 2025)	81.65	89.86	95.00	65.89	92.74	50.27	83.68	69.00
AFlow (Zhang et al., 2025d)	82.20	90.06	92.30	73.35	91.73	58.82	83.10	64.35
RouterDC (Chen et al., 2024b)	75.20	87.75	93.68	73.46	91.86	58.93	82.01	63.27
A^2Flow (Zhao et al., 2026)	85.00	92.40	93.80	58.50	92.15	44.76	83.29	63.42
DAAO (Su et al., 2025)	86.95	94.65	94.40	55.37	92.64	42.57	84.90	65.28
MasRouter (Yue et al., 2025)	84.00	90.62	95.45	75.42	92.95	61.36	84.25	64.85
MaAS (Zhang et al., 2025b)	82.17	92.85	92.30	74.45	91.55	60.23	83.01	63.68
BiRouter (Yang et al., 2026)	84.82	91.46	94.09	74.92	93.20	60.87	86.80	66.53
Multi-agent Systems as Graphs								
GPTSwarm (Zhuge et al., 2024)	75.40	86.28	94.66	68.85	92.18	54.64	82.80	64.19
G-Designer (Zhang et al., 2025c)	79.24	87.50	93.97	70.46	90.29	55.39	87.20	66.94
DAWN (Wan et al., 2026)	86.70	94.44	93.12	73.80	92.10	59.21	80.00	62.15
R-HAN (Ours)	**88.73** ↑1.78	**95.20** ↑0.55	**97.39** ↑1.94	**78.29** ↑2.87	**95.48** ↑2.28	**68.21** ↑6.85	**88.37** ↑1.17	**79.27** ↑10.27

Notes. All scores are percentages. MATH-Lv5 denotes the Level-5 subset of MATH, and MMLU-Pro denotes the harder professional-knowledge variant of MMLU.



(a) MATH and MATH-Lv5

(b) MMLU and MMLU-Pro

Figure 3: Cost–accuracy comparison and difficulty-dependent scaling of **R-HAN**. Left: MATH and MATH-Lv5. Right: MMLU and MMLU-Pro.

tion, latent memory, and the global controller. Table 2 reports the results, where each variant disables one component while keeping the remaining system unchanged.

Obs.❸ Union Graph Ensures Diversity. As substantiated in Table 2, removing the union graph induces an average performance drop of 2.88%↓, with particularly pronounced degradation on HumanEval (7.84%↓) and SVAMP (2.30%↓). These results confirm that a single topology restricts the agent collective to a narrow reasoning trajectory, and that aggregating complementary perspectives into a unified structural graph is critical for robust reasoning coverage.

Obs.❹ Two-Level Noise Control Ensures Reliable Reasoning. Noise control in **R-HAN** requires two complementary mechanisms rather than a single step. Hierarchical Sparse Coordination suppresses noise during propagation by filtering private memories and maintaining a compact global state, preventing low-confidence traces from spreading through the union graph. Removing latent memory alone causes a 3.15%↓ drop. Local Self-Refinement then targets high-risk decision units and revises them only when contrastive evidence confirms a correction. Its removal triggers the largest collapse of 3.52%↓, with 5.61%↓ on MATH-Lv5 and 4.5%↓ on MMLU-Pro. Together, the two mechanisms form a complete noise-suppression pipeline, with propagation control followed by targeted local revision.

Sensitivity Analysis on K. Figure 4 shows how the number of fused topologies K affects performance and cost. $K = 3$ achieves the best balance between topology diversity and redundant-edge control, so we use $K = 3$ as the default setting.

Table 2: Ablation study.

Variant	Code generation		Mathematical reasoning				Knowledge / professional		Summary	
	MBPP	HumanEval	GSM8K	MATH	SVAMP	MATH-Lv5	MMLU	MMLU-Pro	Avg.	ΔAvg.
R-HAN (Ours)	**88.73**	**95.20**	**97.39**	**78.29**	**95.48**	**68.21**	**88.37**	**79.27**	**86.37**	–
w/o union graph	84.74	87.36	96.87	76.89	93.18	66.07	86.70	76.12	83.49	↓2.88
w/o memory composer	84.59	88.09	96.58	76.15	94.20	63.49	87.24	75.42	83.22	↓3.15
w/o sparse activation	86.15	90.77	97.04	77.24	94.83	64.11	87.85	76.80	84.35	↓2.02
w/o local self-refinement	84.27	88.55	96.31	76.74	94.21	62.60	85.31	74.77	82.85	↓**3.52**

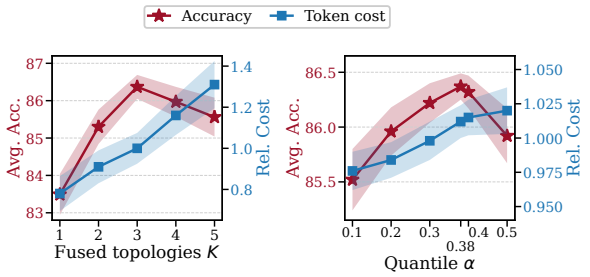


Figure 4: Sensitivity analysis of **R-HAN**. Left: performance and relative token cost as the number of fused topologies K varies. Right: Right: performance and relative token cost as the lower-confidence quantile α varies.

Sensitivity Analysis on Lower-Confidence Quantile. Figure 4 also reports the effect of the lower-confidence quantile α. The default value $\alpha = 0.38$ performs best, balancing uncertainty suppression and the retention of uncertain but useful candidate topologies.

5.5 Case Study

We explore and visualize the intrinsic mechanisms of **R-HAN**. Taking MBPP as an example, Figure 5 showcases the workflow of **R-HAN**.

5.6 Transferability Analysis

Table 3 presents cross-backbone results. **R-HAN** transfers well to open-source models: Qwen3-32B remains within 1 point of GPT-4o-mini on complex reasoning benchmarks, including MATH, MATH Level 5, and MMLU-Pro, while Qwen3-8B also benefits consistently from the same coordination framework across all eight benchmarks.

6 Conclusion

In this paper, we identify a topological diversity and noise propagation dilemma in LLM-based MAS: a single topology restricts reasoning coverage, while naively merging multiple topologies introduces redundant noise that degrades solution quality. To address it, **R-HAN** constructs a Union MAS Graph from complementary candidate topologies, activates a sparse task-adaptive subgraph via Hierarchical Sparse Coordination and applies Local Self-Refinement to revise risky decision units under contrastive validation. Experiments across eight benchmarks demonstrate state-of-the-art performance with an average gain of 2.98%↑ over the strongest baseline, reaching up to 10.27%↑ on MMLU-Pro and 6.85%↑ on MATH-Lv5.

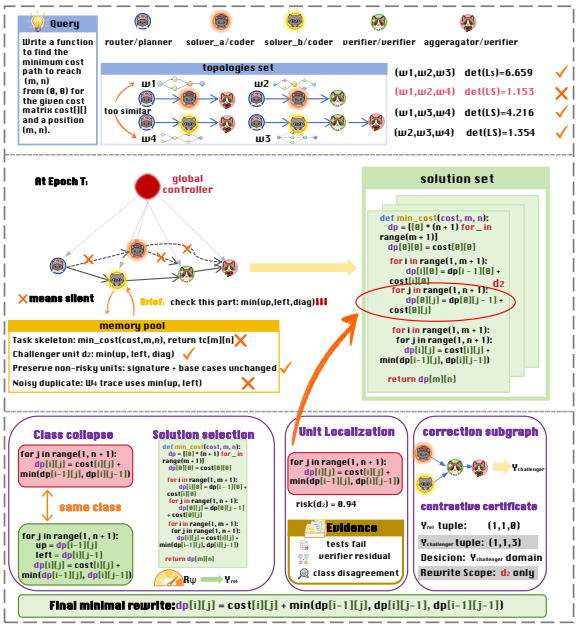


Figure 5: Case study illustrating the intrinsic mechanisms of **R-HAN**. Tuples denote `(syntax_ok, entry_point_ok, passed_tests)` in this case.

Table 3: Cross-backbone transferability of **R-HAN** on Qwen3-32B and Qwen3-8B.

Benchmark	GPT-4o-mini	Qwen3-32B	Qwen3-8B
MMLU	**88.37**	87.92↓0.45	84.12↓4.25
GSM8K	**97.39**	96.85↓0.54	94.23↓3.16
HumanEval	**95.20**	92.75↓2.45	86.34↓8.86
MBPP	**88.73**	86.90↓1.83	81.08↓7.65
MATH	**78.29**	77.86↓0.43	72.84↓5.45
MATH-L5	**68.21**	67.74↓0.47	61.92↓6.29
MMLU-Pro	**79.27**	78.62↓0.65	73.41↓5.86
SVAMP	**95.48**	94.91↓0.57	92.16↓3.32
Avg.	**86.37**	85.44↓0.93	80.76↓5.61

Limitations

Although **R–HAN** has been evaluated across three domains and eight benchmarks, further validation on more diverse tasks (e.g., medical QA and legal reasoning) and a broader range of LLM backbones would strengthen its generality, which we leave for future work. Additionally, the hyperparameters of R–HAN are tuned on a fixed validation set and may require re-tuning when applied to new domains, which we also leave for future exploration.

Ethical Considerations

We believe that **R–HAN** raises no ethical concerns regarding its motivation, design, implementation, or data usage. The framework is designed to improve the reliability and efficiency of LLM-based multi-agent systems in a principled manner, without involving sensitive data collection, human subjects, or proprietary information.

References

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

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

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde 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. *arXiv preprint arXiv:2107.03374*.

Shuhao Chen, Weisen Jiang, Baijiong Lin, James T. Kwok, and Yu Zhang. 2024b. RouterDC: Query-based router by dual contrastive learning for assembling large language models. In *Advances in Neural Information Processing Systems*.

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. 2024c. AgentVerse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *International Conference on Learning Representations*.

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. In *arXiv preprint arXiv:2110.14168*.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2024. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*.

Shangbin Feng, Zifeng Wang, Palash Goyal, Yike Wang, Weijia Shi, Huang Xia, Hamid Palangi, Luke Zettlemoyer, Yulia Tsvetkov, Chen-Yu Lee, and Tomas Pfister. 2025. Heterogeneous swarms: Jointly optimizing model roles and weights for multi-LLM systems. In *Advances in Neural Information Processing Systems*.

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

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*.

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 *International Conference on Learning Representations*.

Shengran Hu, Cong Lu, and Jeff Clune. 2025. Automated design of agentic systems. In *International Conference on Learning Representations*.

Jen-tse Huang, Jiaxu Zhou, Tailin Jin, Xuhui Zhou, Zixi Chen, Wenxuan Wang, Youliang Yuan, Michael R. Lyu, and Maarten Sap. 2025. On the resilience of LLM-based multi-agent collaboration with faulty agents. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023. LLM-Blender: Ensembling large language models with pairwise ranking and generative fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*.

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

Alex Kulesza and Ben Taskar. 2012. Determinantal point processes for machine learning. *Foundations and Trends in Machine Learning*, 5(2–3):123–286.

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

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Zhaopeng Tu, and Shuming Shi. 2024. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*.

Changxing Liu, Genjia Liu, Zijun Wang, Jinchang Yang, and Siheng Chen. 2025. CoLMDriver: LLM-based negotiation benefits cooperative autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 25951–25960.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. 2024. Dynamic LLM-agent network: An LLM-agent collaboration framework with agent team optimization. In *Conference on Language Modeling*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-Refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*.

Dongmin Park, Minkyu Kim, Beongjun Choi, Junhyuck Kim, Keon Lee, Jonghyun Lee, Inkyu Park, Byeong-Uk Lee, Jaeyoung Hwang, Jaewoo Ahn, Ameya Sunil Mahabaleshwarkar, Bilal Kartal, Pritam Biswas, Yoshi Suhara, Kangwook Lee, and Jaewoong Cho. 2026. Orak: A foundational benchmark for training and evaluating LLM agents on diverse video games. In *International Conference on Learning Representations*.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. 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*, pages 2080–2094.

Chen Qian, Zihao Xie, Yifei Wang, Wei Liu, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2025. Scaling large-language-model-based multi-agent collaboration. In *International Conference on Learning Representations*.

Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. 2025. AgentSquare: Automatic LLM agent search in modular design space. In *International Conference on Learning Representations*.

Martin Simonovsky and Nikos Komodakis. 2017. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3693–3702.

Jinwei Su, Yinghui Xia, Qizhen Lan, Xinyuan Song, Jingsong Yang, Lewei He, and Tianyu Shi. 2025. Difficulty-aware agent orchestration in LLM-powered workflows. *arXiv preprint arXiv:2509.11079*.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

Guancheng Wan, Mo Zhou, Ziyi Wang, Xiaoran Shang, Eric Hanchen Jiang, Guibin Zhang, Jinhe Bi, Yunpu Ma, Zaixi Zhang, Ke Liang, and Wenke Huang. 2026. DAWN: Distributed LLM multi-agent workflow synthesis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 26099–26106.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. 2025. Mixture-of-agents enhances large language model capabilities. In *International Conference on Learning Representations*.

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. 2024. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*.

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 conversation. In *Conference on Language Modeling*.

Qingwen Yang, Feiyu Qu, Tiezheng Guo, Yanyi Liu, and Yingyou Wen. 2026. Augmented runtime collaboration for self-organizing multi-agent systems: A hybrid bi-criteria routing approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Yanwei Yue, Guibin Zhang, Boyang Liu, Guancheng Wan, Kun Wang, Dawei Cheng, and Yiyan Qi. 2025. MasRouter: Learning to route LLMs for multi-agent systems. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pages 15549–15572.

Guibin Zhang, Muxin Fu, Guancheng Wan, Miao Yu, Kun Wang, and Shuicheng Yan. 2025a. G-Memory: Tracing hierarchical memory for multi-agent systems. In *Advances in Neural Information Processing Systems*.

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. 2025b. Multi-agent architecture search via agentic supernet. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*.

Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. 2025c. G-Designer: Architecting multi-agent communication topologies via graph neural networks. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. 2025d. AFlow: Automating agentic workflow generation. In *International Conference on Learning Representations*.

Mingming Zhao, Xiaokang Wei, Yuanqi Shao, Kaiwen Zhou, Lin Yang, Siwei Rao, Junhui Zhan, and Zhitang Chen. 2026. A^2Flow: Automating agentic workflow generation via self-adaptive abstraction operators. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. 2024. GPTSwarm: Language agents as optimizable graphs. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*.

A Appendix

A.1 Implementation Details

Model and decoding settings. We use `gpt-4o-mini` as the default backbone model for all agents in R-HAN and for comparable baselines with available implementations. For cross-backbone transfer experiments, we additionally evaluate `Qwen3-32B` and `Qwen3-8B`. Topology descriptions and memory records are encoded by `Qwen3-Embedding-8B`, with embedding dimension $D = 512$. Unless otherwise specified, generation agents use temperature 0 and top-$p = 0.95$, while verifier and final-selection agents use deterministic decoding with temperature 0 and top-$p = 1.0$. Each result is averaged over three independent runs.

Prompt and context budget. For each agent call, the input prompt consists of the original task query, a role-conditioned instruction, the selected private-memory brief, active predecessor briefs, and the current global-state summary. To control context growth, we cap the maximum input length of each agent call to approximately $8{,}192$ tokens and the maximum generation length to $1{,}024$ tokens. The task query is truncated to at most $2{,}048$ tokens, the private-memory brief to 256 tokens, the predecessor-context brief to 384 tokens, and the global-state summary to 256 tokens. When the composed prompt exceeds the budget, we first remove zero-weight memory records and then truncate lower-weight predecessor briefs.

Graph and memory settings. During union-graph construction, we select $K = 3$ complementary topologies by k-DPP, and use the lower-confidence quantile $\alpha = 0.38$ for conservative topology scoring. In hierarchical sparse coordination, we run up to $T = 5$ communication turns. Each active node retrieves at most 4 private memory records and receives at most 3 predecessor briefs from active in-neighbors. Both memory retrieval and edge activation use sparsemax, so irrelevant memory records and inactive communication edges are assigned zero weight.

Item	Setting
Default backbone	`gpt-4o-mini`
Transfer backbones	`Qwen3-32B`, `Qwen3-8B`
Embedding model	`Qwen3-Embedding-8B`
Embedding dimension	512
Selected topologies	$K = 3$
Lower-confidence quantile	$\alpha = 0.38$
Maximum communication turns	$T = 5$
Generation temperature	0
Verifier / finalizer temperature	0
Max input length per call	8,192 tokens
Max generation length	1,024 tokens
Task query cap	2,048 tokens
Private-memory brief cap	256 tokens
Predecessor-context brief cap	384 tokens
Global-state summary cap	256 tokens
Selected memory records per node	up to 4
Active predecessor briefs per node	up to 3
Evaluation runs	3
Code metric	pass@1
Other metrics	exact-match accuracy

Table 4: Implementation and prompt-budget settings of **`R-HAN`**.

A.2 Algorithm Workflow

Algorithm 1 Union MAS Graph Construction of `R-HAN`

1. **Input:** input query Q; initial workflow W_0; edit prior P_ϕ; search budget B; selected topology number K; lower-confidence quantile $\alpha < 0.5$; DPP temperature η; evaluation buffer $\mathcal{D}_{\text{eval}}$.
2. **Output:** union MAS graph $G_U = (V_U, E_U)$ and edge descriptors $e_{u\to v} = (\text{supp}_{uv}, \text{compat}_{uv}, \text{ctx}_{uv})$.

Initialize the topology pool $\mathcal{C} \leftarrow \emptyset$.

/* MCTS candidate topology search */

for simulation $b = 1, \ldots, B$ **do**

  Initialize search state $s \leftarrow W_0$ and search path $\tau_b \leftarrow \emptyset$.

  while s is not terminal **do**

    Select a legal topology edit by

$$u^\star \leftarrow \arg\max_{u \in \mathcal{U}(s)} \left[Q(s,u) + c_{\text{puct}} P_\phi(u \mid s) \frac{\sqrt{N(s)}}{1 + N(s,u)} \right].$$

    Append $(s, u^\star)$ to τ_b and update $s \leftarrow \mathcal{T}(s, u^\star)$.

  end while

  Compile terminal state s into topology W; evaluate W on Q to obtain normalized utility $u_W \in [0,1]$.

  Update $\mathcal{D}_{\text{eval}} \leftarrow \mathcal{D}_{\text{eval}} \cup \{(W, u_W)\}$ and $\mathcal{C} \leftarrow \mathcal{C} \cup \{W\}$.

  for each visited pair $(s,u) \in \tau_b$ **do**

    $N(s,u) \leftarrow N(s,u) + 1, \quad Q(s,u) \leftarrow Q(s,u) + \dfrac{u_W - Q(s,u)}{N(s,u)}.$

  end for

end for

/* Topology encoding */

for each topology $W \in \mathcal{C}$ **do**

  Canonically serialize W as

$$\text{Ser}(W) = \{\text{node roles, directed edges, prompt configurations}\}.$$

  Compute semantic embedding $e_W \leftarrow \text{Emb}(\text{Ser}(W))$.

  Compute lightweight structural feature $h_W \leftarrow \text{StructFeat}(W)$.

  Obtain topology representation $z_W \leftarrow \text{Proj}_\omega([e_W; h_W])$.

end for

/* Posterior topology quality learning */

for predictor $m = 1, \ldots, M$ **do**

  for each evaluated pair $(W, u_W) \in \mathcal{D}_{\text{eval}}$ **do**

    Predict

$$g_{\psi_m}(z_W) = (\mu_W^{(m)}, \rho_W^{(m)}), \qquad \sigma_W^{(m)} = \text{softplus}(\rho_W^{(m)}) + \epsilon.$$

  end for

  Train g_{ψ_m} by minimizing

$$\mathcal{L}_{\text{qual}}^{(m)} = \sum_{(W,u_W) \in \mathcal{D}_{\text{eval}}} \left[\frac{(u_W - \mu_W^{(m)})^2}{2(\sigma_W^{(m)})^2} + \frac{1}{2} \log(\sigma_W^{(m)})^2 \right].$$

end for

/* Conservative topology scoring */

for each topology $W \in \mathcal{C}$ **do**

  Aggregate predictive mean $\mu_W \leftarrow \dfrac{1}{M} \displaystyle\sum_{m=1}^{M} \mu_W^{(m)}$.

  Aggregate predictive uncertainty

$$\sigma_W^2 \leftarrow \frac{1}{M} \sum_{m=1}^{M} \left((\sigma_W^{(m)})^2 + (\mu_W^{(m)})^2 \right) - \mu_W^2.$$

  Compute lower posterior quantile $r_W \leftarrow Q_\alpha(u \mid W) = \mu_W + \Phi^{-1}(\alpha)\sigma_W$, where $\alpha < 0.5$.

  Convert to DPP quality weight $q_W \leftarrow \exp\left(\dfrac{r_W}{2\eta}\right)$.

end for

/* Complementary topology selection with k-DPP */

for each topology $W \in \mathcal{C}$ **do** normalize $\bar{z}_W \leftarrow z_W / \|z_W\|_2$. **end for**

for each pair $(W_a, W_b) \in \mathcal{C} \times \mathcal{C}$ **do**

  Compute structural similarity $\Sigma_{ab} \leftarrow k(\bar{z}_a, \bar{z}_b)$ and DPP entry $L_{ab} \leftarrow q_a \Sigma_{ab} q_b$.

end for

38. Greedily solve

$$S^\star \leftarrow \arg\max_{\substack{S\subseteq\mathcal{C}\\|S|=K}} \log\det(L_S).$$

/* Union graph construction and edge descriptor attachment */

39. Align nodes in selected topologies by role-agent identity and construct

$$G_U = (V_U, E_U) \leftarrow \left(\bigcup_{W\in S^\star} V_W,\ \bigcup_{W\in S^\star} E_W\right).$$

40. **for** each edge $(u, v) \in E_U$ **do**
41. Compute quality-weighted support

$$\mathrm{supp}_{uv} \leftarrow \frac{\sum_{W\in S^\star} q_W \mathbf{1}[(u,v)\in E_W]}{\sum_{W\in S^\star} q_W}.$$

42. Compute role-prompt compatibility

$$\mathrm{compat}_{uv} \leftarrow \cos\left(\mathrm{Emb}(\mathrm{Role}_u \| \mathrm{Prompt}_u), \mathrm{Emb}(\mathrm{Role}_v \| \mathrm{Prompt}_v)\right).$$

43. Compute local topological context

$$\mathrm{ctx}_{uv} \leftarrow \mathrm{Pool}_{W\in S^\star:(u,v)\in E_W}\left[\bar{z}_W; \mathrm{pos}_W(u,v); \deg_W(u); \deg_W(v)\right].$$

44. Set $e_{u\to v} \leftarrow (\mathrm{supp}_{uv}, \mathrm{compat}_{uv}, \mathrm{ctx}_{uv})$.
45. **end for**
46. **return** G_U and $\{e_{u\to v} : (u, v) \in E_U\}$.

Algorithm 2 Hierarchical Sparse Coordination of R-HAN

---

1. **Input:** query Q; union graph $G_U = (V_U, E_U)$; edge descriptors $\{e_{u\to v}\}$; private memories $\{\mathcal{M}_i^0\}$; turns T; query builder f_q; memory selector f_s; memory composer Compose; edge scorer $\Psi^{\mathrm{edge}}_{\Theta_e}$; global updater $\mathrm{GlobalNode}_{\Theta_g}$; feedback operator Feedback.
2. **Output:** solution set $Y = \{(a_k, \xi_k)\}_{k=1}^{K_y}$ and activated graph trace $G^{1:T}_{\mathrm{act}}$.
3. Initialize $g^0 \leftarrow \mathrm{InitGlobal}(Q)$, $Y \leftarrow \emptyset$, and $G^{1:T}_{\mathrm{act}} \leftarrow \emptyset$.
4. **for** each node $v_i \in V_U$ **do**
5. Represent each private memory record as $m^0_{ik} = (c_{ik}, \mathbf{e}_{ik}, \mathbf{h}_{ik}, \rho_{ik})$, where $\rho_{ik} = (p_{ik}, \gamma_{ik})$.
6. **end for**
7. **for** turn $t = 1, \ldots, T$ **do**

/* Node-level private memory retrieval and composition */

8. **for** each node $v_i \in V_U$ **do**
9. Build retrieval query $\mathbf{q}^t_i \leftarrow f_q(Q, r_i)$.
10. Score each private record: $z^t_{ik} \leftarrow f_s(\mathbf{q}^t_i, \mathbf{e}_{ik}, \mathbf{h}_{ik}, \rho_{ik})$, for $m^t_{ik} \in \mathcal{M}^t_i$.
11. Compute sparse retrieval weights

$$\boldsymbol{\alpha}^t_i \leftarrow \mathrm{sparsemax}\left(\{z^t_{ik} : m^t_{ik} \in \mathcal{M}^t_i\}\right), \qquad \mathcal{R}^t_i \leftarrow \{m^t_{ik} : \alpha^t_{ik} > 0\}.$$

12. Compose private context, latent brief, and verbalized brief

$$(\ell^t_i, \mathbf{b}^t_i, B^t_i) \leftarrow \mathrm{Compose}\left(\{(\alpha^t_{ik}, m^t_{ik}) : m^t_{ik} \in \mathcal{R}^t_i\}\right).$$

13. **end for**

/* Edge-level sparse activation over the union graph */

14. **for** each edge $(v_j, v_i) \in E_U$ **do**
15. Compute edge score $\phi^t_{ji} \leftarrow \Psi^{\mathrm{edge}}_{\Theta_e}(\mathbf{b}^t_j, \mathbf{b}^t_i, g^t, e_{v_j\to v_i})$.
16. **end for**
17. **for** each target node $v_i \in V_U$ **do**
18. Compute incoming-edge weights $\boldsymbol{\beta}^t_i \leftarrow \mathrm{sparsemax}(\{\phi^t_{ji} : (v_j, v_i) \in E_U\})$.
19. **end for**

20. Define active subgraph

$$E^t \leftarrow \{(v_j, v_i) \in E_U : \beta^t_{ji} > 0\}, \qquad V^t \leftarrow \mathrm{EndPoints}(E^t) \cup V^t_{\mathrm{seed}}, \qquad G^t \leftarrow (V^t, E^t).$$

21. Append G^t to $G^{1:T}_{\mathrm{act}}$.

/* Prompt construction, execution, and memory writing */

22. **for** each active node $v_i \in V^t$ **do**
23. Build prompt-readable context

$$D^t_i \leftarrow \mathrm{BuildBrief}\left(g^t, \mathcal{R}^t_i, \{(\beta^t_{ji}, B^t_j) : (v_j, v_i) \in E^t\}\right).$$

24. Execute agent $o^t_i \leftarrow F_i(P^t_i, Q, \ell^t_i, D^t_i, g^t)$ and obtain feedback $\mathcal{F}^t_i \leftarrow \mathrm{Feedback}(o^t_i, Q)$.
25. Write new record $\mathcal{M}^{t+1}_i \leftarrow \mathcal{M}^t_i \cup \mathrm{Write}(o^t_i, \mathcal{F}^t_i, \mathrm{prov} = G^t)$.
26. **if** o^t_i is a solution proposal **then**
27. Set $\xi^t_i \leftarrow (\mathrm{trace}^t_i, G^{1:t}_{\mathrm{act}}, \mathrm{memref}^t_i, \mathrm{prov}^t_i, \mathcal{F}^t_i, \gamma^t_i)$ and $Y \leftarrow Y \cup \{(o^t_i, \xi^t_i)\}$.
28. **end if**
29. **end for**

/* Global state update */

30. Pool active-node summaries and feedback

$$p^t \leftarrow \mathrm{Pool}\{(\mathbf{b}^t_i, \mathrm{Enc}(o^t_i)) : v_i \in V^t\}, \qquad \mathcal{F}^t \leftarrow \mathrm{Pool}\{\mathcal{F}^t_i : v_i \in V^t\}.$$

31. Update $g^{t+1} \leftarrow \mathrm{GlobalNode}_{\Theta_g}(g^t, p^t, \mathcal{F}^t)$.
32. **end for**

/* Training objectives for hierarchical coordination */

33. Train f_q and f_s by memory contrastive credit assignment

$$\mathcal{L}_{\mathrm{mem}} = -\log \frac{\exp z_{\mathrm{pos}}}{\exp z_{\mathrm{pos}} + \sum_{z_{\mathrm{neg}} \in \mathcal{N}_i} \exp z_{\mathrm{neg}}}.$$

34. Score execution trace τ by selected edge likelihood $s_{\Theta_e}(\tau) = \sum_t \sum_{(v_j, v_i) \in E^t} \log \beta^t_{ji}$.
35. Train edge scorer and global updater by

$$\mathcal{L}_{\mathrm{ctrl}} = -\log \frac{\exp s_{\Theta_e}(\tau^+)}{\exp s_{\Theta_e}(\tau^+) + \sum_{\tau^-} \exp s_{\Theta_e}(\tau^-)} + \lambda_E \sum_t |E^t| + \lambda_C \sum_t \mathrm{Tok}(D^t).$$

36. **return** Y and $G^{1:T}_{\mathrm{act}}$.

Algorithm 3 Local Self-Refinement of R-HAN

---

1. **Input:** solution set $Y = \{(a_k, \xi_k)\}_{k=1}^{K_y}$; union graph G_U; activated graph trace $G^{1:T}_{\mathrm{act}}$; reliability estimator R_ψ; discrepancy extractor Disc_θ; risk scorer risk_θ; certificate model Cert_χ; risk threshold τ; margin δ; training buffers $\mathcal{D}_{\mathrm{rel}}, \mathcal{D}_{\mathrm{risk}}, \mathcal{D}_{\mathrm{cert}}$.
2. **Output:** final reliable solution $\hat{y}$.

/* Train learned local-refinement modules */

3. Train R_ψ from pairwise retrospective preferences

$$\mathcal{L}_{\mathrm{rel}} = -\sum_{((a^+, \xi^+), (a^-, \xi^-)) \in \mathcal{D}_{\mathrm{rel}}} \log \sigma\left(R_\psi(a^+, \xi^+) - R_\psi(a^-, \xi^-)\right).$$

4. Train risk model from unit-level weak labels

$$p_d = \sigma(\mathrm{risk}_\theta(d; \epsilon_d)), \quad \mathcal{L}_{\mathrm{risk}} = -\sum_{(d, \epsilon_d, r^\star_d) \in \mathcal{D}_{\mathrm{risk}}} \left[r^\star_d \log p_d + (1 - r^\star_d) \log(1 - p_d)\right].$$

5. Train certificate model from contrastive local replacements

$$c_d = \mathrm{Cert}_\chi(y^{\mathrm{ch}}_d, y_{\mathrm{rel}}, d), \quad \mathcal{L}_{\mathrm{cert}} = -\sum \left[c^\star_d \log c_d + (1 - c^\star_d) \log(1 - c_d)\right].$$

/* Select reliable solution */

Collapse semantically equivalent answers: $\mathcal{G} \leftarrow \mathrm{Collapse}(Y)$.

for each equivalence class $G \in \mathcal{G}$ **do**

  Select class representative $(a_G, \xi_G) \leftarrow \arg\max_{(a,\xi)\in G} R_\psi(a, \xi)$.

end for

Select reliable solution: $(a_{\mathrm{rel}}, \xi_{\mathrm{rel}}) \leftarrow \arg\max_{(a_G,\xi_G)\in \mathrm{Rep}(\mathcal{G})} R_\psi(a_G, \xi_G)$, and set $y_{\mathrm{rel}} \leftarrow a_{\mathrm{rel}}$.

/* Localize risky decision units */

Parse reliable solution into locally verifiable units: $\mathcal{D}_{\mathrm{unit}} \leftarrow \mathrm{UnitParse}(y_{\mathrm{rel}}, \xi_{\mathrm{rel}})$.

Initialize risky unit set $\mathcal{U} \leftarrow \emptyset$.

for each decision unit $d \in \mathcal{D}_{\mathrm{unit}}$ **do**

  Extract discrepancy evidence: $\epsilon_d \leftarrow \mathrm{Disc}_\theta(d, a_{\mathrm{rel}}, \xi_{\mathrm{rel}}, Y)$.

  Compute risk score $r_d \leftarrow \mathrm{risk}_\theta(d; \epsilon_d)$.

  if $r_d > \tau$ **then** $\mathcal{U} \leftarrow \mathcal{U} \cup \{d\}$. **end if**

end for

/* Validate challengers and minimally rewrite */

for each risky unit $d \in \mathcal{U}$ **do**

  Extract correction subgraph: $G_d^{\mathrm{corr}} \leftarrow \mathrm{ExtractSubgraph}(G_U, G_{\mathrm{act}}^{1:T}, d, \xi_{\mathrm{rel}}, Y)$.

  Retrieve challengers: $\mathcal{H}_d^{\mathrm{ret}} \leftarrow \{(a, \xi) \in Y : \mathrm{Disagree}(a, y_{\mathrm{rel}}, d) = 1\}$.

  Generate challengers: $\mathcal{H}_d^{\mathrm{gen}} \leftarrow \mathrm{RepairExec}(G_d^{\mathrm{corr}}, y_{\mathrm{rel}}, d, \xi_{\mathrm{rel}})$.

  Set $\mathcal{H}_d \leftarrow \mathcal{H}_d^{\mathrm{ret}} \cup \mathcal{H}_d^{\mathrm{gen}}$ and $\mathcal{A}_d \leftarrow \emptyset$.

  for each challenger $(y_d^{\mathrm{ch}}, \xi_d^{\mathrm{ch}}) \in \mathcal{H}_d$ **do**

    Compute local improvement $\Delta_d \leftarrow R_\psi^{\mathrm{loc}}(y_d^{\mathrm{ch}}, \xi_d^{\mathrm{ch}}, d) - R_\psi^{\mathrm{loc}}(y_{\mathrm{rel}}, \xi_{\mathrm{rel}}, d)$.

    Compute certificate $c_d \leftarrow \mathrm{Cert}_\chi(y_d^{\mathrm{ch}}, y_{\mathrm{rel}}, d)$.

    if $\Delta_d > \delta$ and $c_d = 1$ **then** $\mathcal{A}_d \leftarrow \mathcal{A}_d \cup \{(y_d^{\mathrm{ch}}, \xi_d^{\mathrm{ch}})\}$. **end if**

  end for

  if $\mathcal{A}_d \neq \emptyset$ **then**

    Select best accepted challenger: $(y_d^\star, \xi_d^\star) \leftarrow \arg\max_{(y,\xi)\in\mathcal{A}_d} R_\psi^{\mathrm{loc}}(y, \xi, d)$.

    Minimally rewrite: $y_{\mathrm{rel}} \leftarrow \mathrm{Rewrite}(y_{\mathrm{rel}}, d, y_d^\star)$.

    Update evidence: $\xi_{\mathrm{rel}} \leftarrow \mathrm{UpdateEvidence}(\xi_{\mathrm{rel}}, d, \xi_d^\star, G_d^{\mathrm{corr}})$.

  else preserve unit d unchanged. **end if**

end for

Set $\hat{y} \leftarrow y_{\mathrm{rel}}$.

return $\hat{y}$.

A.3 Prompt Set

This appendix lists representative prompt schemas used by **R-HAN**. We include prompts for topology-level prompt slots, role-conditioned agents, latent-guided graph execution, deductive reasoning contracts, and local verification / repair.

Discrete Prompt Slot Template

```
reasoning_mode = [
  "Answer directly with minimal reasoning.",
  "Reason step by step before giving the answer.",
  "First critique possible failure modes, then answer."
]

upstream_usage = [
  "Use upstream outputs only when they provide direct evidence.",
  "Summarize useful upstream information before reasoning.",
  "Ignore upstream information if it conflicts with the task constraints."
]

output_style = [
  "Return a concise natural-language answer.",
  "Return bullet points followed by a final answer.",
  "Return a compact structured response with stable fields."
]

verification_mode = [
  "No explicit verification.",
  "Perform a short final consistency check.",
  "Check the answer against every task constraint."
]

finalization = [
  "Answer only.",
  "Answer followed by a short rationale.",
  "Final answer must appear after the marker FINAL:"
]
```

Representative Agent Role Prompts

```
planner_system_prompt = """
You are a planning agent. Understand the problem first and provide a concise,
reliable plan for solving it. Focus on decomposing the task into useful steps.

"""

reasoner_system_prompt = """
You are a reasoning agent. Produce a complete and traceable reasoning process.

Avoid skipping important intermediate steps.
"""

coder_system_prompt = """
You are a coding agent. Translate the problem into executable algorithmic
    steps
or code-level reasoning. Preserve required function names and interfaces.
"""

verifier_system_prompt = """
You are a verifier agent. Check whether a candidate answer satisfies the task
```

```
constraints, output format, and available evidence. Report only grounded
    issues.
"""

summarizer_system_prompt = """
You are a summarizer agent. Synthesize multiple candidates into a concise,
consistent final proposal. Do not introduce unsupported claims.
"""
```

Latent-Guided Runtime Prompt

```
runtime_user_prompt = """
Question:
{rendered_question}

Latent-guided memory brief:
{memory_brief}

Current task:
Turn {t}/{T}. {role_instruction}

Use your private memory together with graph-mediated neighbour signals.
Only rely on neighbour information if it is relevant to your current role.
Avoid repeating irrelevant traces.

Task context:
{task_context}

Output contract:
{answer_contract}
"""
```

Deductive Solver and Verifier Contracts

```
deductive_solver_contract = """
SOLUTION:
1. <quantity_name> = <arithmetic expression> = <value>
2. <quantity_name> = <arithmetic expression> = <value>
3. ...

FINAL: <number_or_expression>
"""

deductive_verifier_contract = """
VERDICT: pass|challenge|uncertain
LOCUS: step=<id>|final|parse
ISSUE: arithmetic_mismatch|algebra_mismatch|unsupported_transition|
    missing_final|none
FIX: <one sentence>
"""
```

Local Verification and Repair Prompts

```
anchor_verification_probe = """
You are verifying a proposed final answer.

Problem:
{question}

Proposed final answer:
{reliable_solution}

Do not assume the proposed answer is correct.
Re-solve the problem from the task quantities only.
Use a compact equation-chain format.
Every step must contain a computable equation.

Return exactly:

SOLUTION:
1. <quantity_name> = <arithmetic expression> = <value>
2. <quantity_name> = <arithmetic expression> = <value>
...

FINAL: <answer>
"""

suffix_repair_prompt = """
You are repairing a derivation.

Problem:
{question}

Verified prefix. Do not change these steps:
{verified_prefix}

First inconsistent step:
{bad_step}

Verifier residual:
{residual}

Rewrite only from Step {verified_prefix_len + 1} onward.
Keep the verified prefix unchanged.

Return:

SOLUTION:
1. ...
2. ...
...

FINAL: <answer>
"""
```

999