

ST-EVO: Towards Generative Spatio-Temporal Evolution of Multi-Agent Communication Topologies

Xingjian Wu*, Xvyuan Liu*, Junkai Lu*, Siyuan Wang, Xiangfei Qiu,
Yang Shu, Jilin Hu, Chenjuan Guo, Bin Yang✉

East China Normal University

{xjwu, xvyuanliu, jkl, sywang, xfqiu}@stu.ecnu.edu.cn,

{yshu, jlhu, cjguo, byang}@dase.ecnu.edu.cn

Abstract

LLM-powered Multi-Agent Systems (MAS) have emerged as an effective approach towards collaborative intelligence, and have attracted wide research interests. Among them, “self-evolving” MAS, treated as a more flexible and powerful technical route, can construct task-adaptive workflows or communication topologies, instead of relying on a predefined static structure template. Current self-evolving MAS mainly focus on Spatial Evolving or Temporal Evolving paradigm, which only considers the single dimension of evolution and does not fully incentivize LLMs’ collaborative capability. In this work, we start from a novel Spatio-Temporal perspective by proposing **ST-EVO**, which supports dialogue-wise communication scheduling with a compact yet powerful flow-matching based Scheduler. To make precise Spatio-Temporal scheduling, the Scheduler can also perceive the uncertainty of MAS, and possesses self-feedback ability to learn from accumulated experience. Extensive experiments on nine benchmarks demonstrate the state-of-the-art performance of **ST-EVO**, achieving about 5%–25% accuracy improvement.

1 Introduction

LLM-powered agentic systems (Significant Gravititas, 2023; Inc., 2024; Wu et al., 2023), which integrate LLMs’ reasoning capabilities with external functionalities, have exhibited impressive performance and facilitated a surge of practical applications like information search (Li et al., 2025b), data analysis (Wu et al., 2026b), code generation (Zheng et al., 2023), deep research (Zheng et al., 2025) and autonomous driving (Jin et al., 2023). To further improve the performance and robustness in complex scenarios, multi-agent systems (MAS) are studied to achieve collaborative intelligence. Generally, multi-agent systems (Ding et al., 2024; Han et al., 2024) are customized for a certain type of task, with a meticulously-designed workflow or

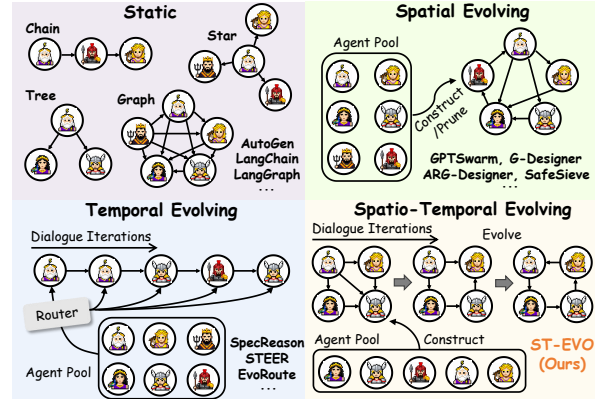


Figure 1: Existing multi-agent systems. Our proposed **ST-EVO** pioneers the exploration of Spatio-Temporal Evolving multi-agent systems.

communication topology, focusing on how agents effectively utilize, exchange, and induce information reciprocally.

Classic multi-agent systems often adopt static topologies—see Figure 1 (top left), which organizes agents as sequential workflows (chain) (Wei et al., 2022; Hong et al., 2024), task splitting mode (tree) (Yao et al., 2023, 2022), or task assignment mode (star) (Wu et al., 2023). Since they are all special cases of graphs, some recent multi-agent systems (Inc., 2024; Wu et al., 2023) adopt graphs as their template structures and demonstrate remarkable performance. However, these task-specific, static designs rely on expert intervention, and lack automation and adaptation when scenario changes. Therefore, current research interests gradually shift towards dynamic workflow orchestration (Niu et al., 2025; Zhang et al., 2025a; Lee et al., 2025) and communication topology design (Zhang et al.; Li et al., 2025a; Zhang et al., 2025c), aiming at task-specific or query-specific automated fine-grained customization, known as “self-evolving” MAS (Fang et al., 2025; Wang et al., 2025b).

As shown in Figure 1, we categorize existing

self-evolving multi-agent systems into “Spatial Evolving” and “Temporal Evolving”. Considering the Spatial Evolving MAS, the mainstream approach is to first formulate the communication topology as a graph, then learn from some cases to train a “designer” (often compact neural networks), which can design topologies for specific tasks (Liu et al., 2023; Zhuge et al., 2024) or queries (Zhang et al.; Li et al., 2025a; Zhang et al., 2025c). However, the designed topologies are fixed throughout the entire conversation process for a query. That means, if a complex query needs multiple dialogue iterations to solve, the communication topology cannot be dynamically tuned during iterations to allocate computing resources and functions among different agent nodes, *which leads to low flexibility and robustness, and may cause high token consumption*. On the other hand, though the Temporal Evolving MAS makes temporal split on workflows, most of them focus on routing single agent in the proper position (Niu et al., 2025; Zhang et al., 2025a; Lee et al., 2025), *which neglects the potential of multi-agent collaboration in key dialogue iterations*.

The questions in single-dimension evolution posed above can be explicitly summarized as: ❶ **Accuracy**. How to fully unleash the collaborative capabilities of MAS in different dialogue iterations to fulfill the changing demand? ❷ **Stability**. How to ensure the scheduling is reasonable and robust? ❸ **Efficiency**. How to ensure efficiency considering the token consumption of MAS and the cost of scheduling? To tackle these problems, we propose ST-EVO, the first MAS with the capability of Spatio-Temporal Evolution, achieving dialogue-wise temporal scheduling for spatial communication topologies.

The motivation of ST-EVO is to precisely schedule multi-agent collaborations from both temporal and spatial dimensions, preserving accuracy, stability, and efficiency. To achieve this, we believe the core elements lie in: ♣ **Appropriate data structures**. The structure should be able to well maintain the Spatio-Temporal trajectories, providing the basis for scheduling. ♠ **A compact but effective scheduler**. The scheduling is frequent, and the demand for accuracy is high, thus calling for a lightweight and powerful scheduler. ♦ **System state perception**. The scheduler should consider the system state, such as whether it is reliable or not, thus leading to correct scheduling scheme. ♥ **Self-feedback ability**. The system needs to accu-

mulate experience and learn underlying scheduling principles from it.

Based on above-mentioned design principles, ST-EVO adopts *Spatio-Temporal Graph* to maintain the scheduling trajectories, which is an important structure in Spatio-Temporal data mining (Wang et al., 2020; Yang et al., 2026), able to characterize the spatial correlations (agent collaboration) at each temporal index (dialogue iteration). Considering the scheduler, we meticulously devise a compact neural network, driven by an MLP-based *Flow-Matching* (Lipman et al., 2023) module, which possesses strong generative mechanism while keeping lightweight, and can work as a stochastic interpolant (Wu et al., 2026a) to directly evolve the communication topology temporally.

Additionally, to perceive the system state and provide self-feedback ability, we devise mechanisms driven by both entropy and experience, then internalize them through preference optimization. First, we supervise the system state with the help of *Entropy Distribution* to evaluate the uncertainty of the MAS, and we internalize it into the scheduler via reinforcement learning. Second, considering the self-feedback ability, we draw inspiration from *Experience Learning* (Zhang et al., 2025b; Tang et al., 2025), and construct a retrieve-augment module to accumulate successful scheduling trajectories as experience. During training, we also internalize the experience, which retrieves corresponding historical scheduling trajectories as reference, and regularizes the current scheduling scheme.

Our Contributions can be summarized as:

- ❶ We introduce ST-EVO, pioneering the exploration of Spatio-Temporal Evolving MAS. It can support dialogue-wise real-time scheduling through generative Spatio-Temporal evolution of communication topologies.
- ❷ We devise a compact generative scheduler driven by Flow-Matching. It can learn underlying scheduling principles from entropy distribution and historical experience, thus possessing strong query-aware and dialogue-wise scheduling capability.
- ❸ Comprehensive experiments on 9 benchmarks from multiple real-world tasks demonstrate the state-of-the-art performance of our proposed ST-EVO. All the code and datasets are available at the anonymous repository <https://github.com/decisionintelligence/ST-EVO>.

2 Related Works

2.1 LLM-powered Multi-Agent Systems

Recent studies introduce a surge of multi-agent systems to incentivize LLMs’ swarm intelligence in complex scenarios. The communication topology occupies a central position in them and includes multiple structures. Among them, non-interactive systems organize agents parallelly without explicit communication, such as Mixture-of-Agents (Wang et al., 2025a) and LLM-Debate (Estornell and Liu, 2024); Chain MAS arranges agents sequentially as a workflow, such as MetaGPT (Hong et al., 2024) and ChatDev (Qian et al., 2024); Tree MAS works in a task-splitting mode with a chief node and hierarchical child nodes (Yao et al., 2023); Star MAS works in a task-assignment mode with a central node and multiple edge nodes, seen in AutoGen (Wu et al., 2023) and MiniGrid (Zhou et al., 2024); Graph MAS, as the most general form, can represent all of above-posed topologies and are widely applied (Inc., 2024; Zhuge et al., 2024; Wu et al., 2023).

2.2 Self-evolving Multi-Agent Systems

To improve efficiency, recent research interests shift towards “self-evolving” MAS (Wang et al., 2025b; Fang et al., 2025), which can contribute to MAS customization for changing tasks. These studies characterize multi-agent communication topologies as graphs (Liu et al., 2023; Zhuge et al., 2024; Zhang et al.) to organize and represent relationships between agents. Compared with conventional static communication topologies, they support task-aware (Liu et al., 2023; Zhuge et al., 2024) or even query-aware (Zhang et al.; Li et al., 2025a; Zhang et al., 2025c) dynamic communication topology customization. We categorize them into Spatial Evolving and Temporal Evolving MAS. For Spatial Evolving MAS, they mainly focus on the performance, robustness, efficiency, and redundancy (Zhang et al., 2025c) when customizing the topology. However, they pay little attention to temporal scheduling which is crucial in MAS with dialogue-wise varying resource demands. For Temporal Evolving MAS (Zhang et al., 2025a; Niu et al., 2025; Lee et al., 2025), though they consider the temporal scheduling in different iterations, most of them neglect utilizing the collaborative capability of multiple agents inside each iteration. In this work, we propose **ST-EVO**, the first Spatio-Temporal Evolving MAS, which supports temporal

scheduling for spatial communication topologies.

3 Formalization

In this section, we formalize key concepts in **ST-EVO** from a spatio-temporal perspective, and introduce the notations for clear clarification.

3.1 Communication Topology

In **ST-EVO**, a multi-agent system is modeled as a Spatio-Temporal Graph $\mathcal{G} = \{\mathcal{G}_t\}_{t=1}^T = \{(\mathcal{V}, \mathcal{E}_t, \mathbf{A}_t)\}_{t=1}^T$, where t records the dialogue iteration. Note that the total iterations T is flexible and not pre-defined. $\mathcal{V} = \{v_1, v_2, \dots, v_N\}$ denotes N agent nodes in MAS:

$$v_i = \{\text{LLM}_i, \text{Profile}_i, \text{Tool}_i, \text{State}_i\}, \quad (1)$$

where each agent v_i is instantiated through a LLM with Profile (system prompts) and Tool, and its historical memory is maintained in State. During operation, each agent v_t treats Profile _{i} and State _{i} as a prefix, and receives user prompts or responses from other agents in MAS.

In **ST-EVO**, the communication topology $\mathcal{G}_t$ changes over time, contributing to the varying communication link $\mathcal{E}_t$, which is described via the adjacent matrix $\mathbf{A}_t \in \{0, 1\}^{N \times N}$, where $\mathbf{A}_t[i, j] = 1$ denotes the edge $e_t[i, j] \in \mathcal{E}_t$ is connected and the information flows from v_i to v_j , vice versa.

3.2 Scheduling Pipeline

Given a query $\mathcal{Q}$, **ST-EVO** first constructs the communication topology $\mathcal{G}_1$ from an initial anchor topology $\mathcal{G}_0$ for the first dialogue iteration. For simple questions, they can be easily tackled at once. For harder ones, multi-iteration dialogues are executed and fine-grained scheduling is planned through a Scheduler.

In the t -th dialogue, the Scheduler considers the problem $\mathcal{Q}$, the iteration t , and the topology from the last iteration $\mathcal{G}_{t-1}$ to determine the topology of this iteration $\mathcal{G}_t$:

$$\mathcal{G}_t = \text{Scheduler}(\mathcal{Q}, t, \mathcal{G}_{t-1}) \quad (2)$$

Then the communication topology $\mathcal{G}_t$ is compiled into a *topological* execution order $\mathcal{S}_t$ of the N agents in MAS:

$$\mathcal{S}_t = [v_{s_1}^t, v_{s_2}^t, \dots, v_{s_N}^t], \quad (3)$$

$$\text{s.t. } \forall j > i, v_{s_j}^t \notin \mathcal{N}_{\text{in}}(v_{s_i}^t) \quad (4)$$

After **ST-EVO** executes T iterations, the final solution is formed or partially formed inside the agent

states, an aggregation operation like majority voting is applied to summarize for the output.

4 ST-EVO

As illustrated in Figure 2, our proposed **ST-EVO** builds multi-agent systems from a spatio-temporal perspective. (1) **ST-EVO** supports query-aware MAS construction, which builds agents with provided LLMs, profiles and Tools, and generates the initial communication topology based on the query. (2) **ST-EVO** supports spatio-temporal scheduling, where a generative Scheduler is devised to evolve the dialogue-wise spatial communication topologies. (3) The Scheduler is driven by entropy and experience to perceive the system states and learn from accumulated experience. (4) The Scheduler is optimized on dozens of samples to internalize the state-aware capability and experience, thus supporting precise spatio-temporal scheduling.

4.1 Constructing Multi-agent Systems

Given a query $\mathcal{Q}$, **ST-EVO** can adaptively build an effective MAS to handle it. Specifically, it first initializes an anchor MAS with a pre-defined communication topology $\mathcal{G}_0 = (\mathcal{V}, \mathcal{E}_0, \mathcal{A}_0)$. It begins by instantiating agents in $\mathcal{V}$. For each agent v_i , the base model, profile, and tool are assigned to it, as defined in Equation (1). To further characterize the MAS, we utilize the agents’ descriptions to construct the representative features of the nodes:

$$\mathcal{T}_i = \text{Text}(\text{LLM}_i + \text{Profile}_i + \text{Tool}_i), \quad (5)$$

$$x_i^{\text{node}} = \text{NodeEncoder}(\mathcal{T}_i), \quad (6)$$

where $\mathcal{T}_i$ is the textual description of agent v_i , and x_i^{node} is the node feature generated by a sentence transformer (Reimers and Gurevych, 2019), which can transform agents’ descriptions into the fix-length embeddings. We denote the agent node features as $X^{\text{node}} = [x_1^{\text{node}}, x_2^{\text{node}}, \dots, x_N^{\text{node}}]$.

We then provide an empirically favorable communication topology $\mathcal{A}_0 \in \{0, 1\}^{N \times N}$ as the anchor topology. Take an instance in math reasoning tasks given 4 agents: Manager, Analyzer, Mathematician, and Coder. The anchor topology can be organized as a Chain, i.e., “Manager $\rightarrow$ Mathematician $\rightarrow$ Analyzer $\rightarrow$ Coder”, where the Manager assigns the task to Mathematician and Analyzer to identify the essential mathematical problem and conduct a step-by-step analysis, finally the Coder solve numerical values through programming.

Taking the anchor communication topology $\mathcal{G}_0 = (\mathcal{V}, \mathcal{E}_0, \mathcal{A}_0)$ as a good starting point, we then utilize the generative Scheduler to construct the query-adaptive communication topology:

$$\mathcal{G}_1 = \text{Scheduler}(\mathcal{Q}, t = 1, \mathcal{G}_0), \quad (7)$$

where $\mathcal{G}_1$ is the generated communication topology for the first iteration of dialogue. We introduce the detailed design of the Scheduler as follows.

4.2 Generative Scheduler

As introduced in Section 3, given an initial anchor topology $\mathcal{G}_0$, the Spatio-Temporal Scheduling is to autoregressively plan the future topologies $[\mathcal{G}_1, \mathcal{G}_2, \dots, \mathcal{G}_T]$ for dialogue iterations. And it needs to simultaneously ensure ❶ accuracy, ❷ stability, and ❸ efficiency.

Intuitively, adopting generative mechanisms can ensure the diversity and quality of the scheduling trajectories, and avoid overfitting during training. However, most generative mechanisms like DDPM, GAN, and VAE cannot balance these three aspects. To tackle this, since the scheduling task in **ST-EVO** is Spatio-Temporal Graph Generation, we draw inspiration from spatio-temporal data mining, for which lightweight flow-matching networks (Liu et al., 2025; Wu et al., 2026a) are competent.

Specifically, we devise a compact flow-matching based Scheduler to ensure the ❷ stability and ❸ efficiency. For ❶ accuracy, we utilize an important property of flow-matching to serve as a stochastic interpolant, which can start from a given point, instead of a random Gaussian noise, bridging the transport path between two arbitrary distributions (Wu et al., 2026a). Its core architecture consists of a GCN (Zhang et al., 2019) and an MLP-based flow-matching network (FM-Net). The FM-Net utilizes the AdaLN (Peebles and Xie, 2023) to integrate the condition $\mathbf{H}_t$. In t -th dialogue, given the query $\mathcal{Q}$ and topology $\mathcal{G}_{t-1} = (\mathcal{V}, \mathcal{E}_{t-1}, \mathcal{A}_{t-1})$, the condition $\mathbf{H}_t$ is:

$$X^{\text{query}} = \text{Embedding}(\mathcal{Q}), \quad (8)$$

$$X^{\text{pos}} = \text{Sinusoidal}(t), \quad (9)$$

$$\mathbf{H}_t = X^{\text{query}} + X^{\text{pos}} \in \mathbb{R}^d \quad (10)$$

It enables the Scheduler to perceive the task information and the iteration information. Then the flow-matching process occurs in the latent space of

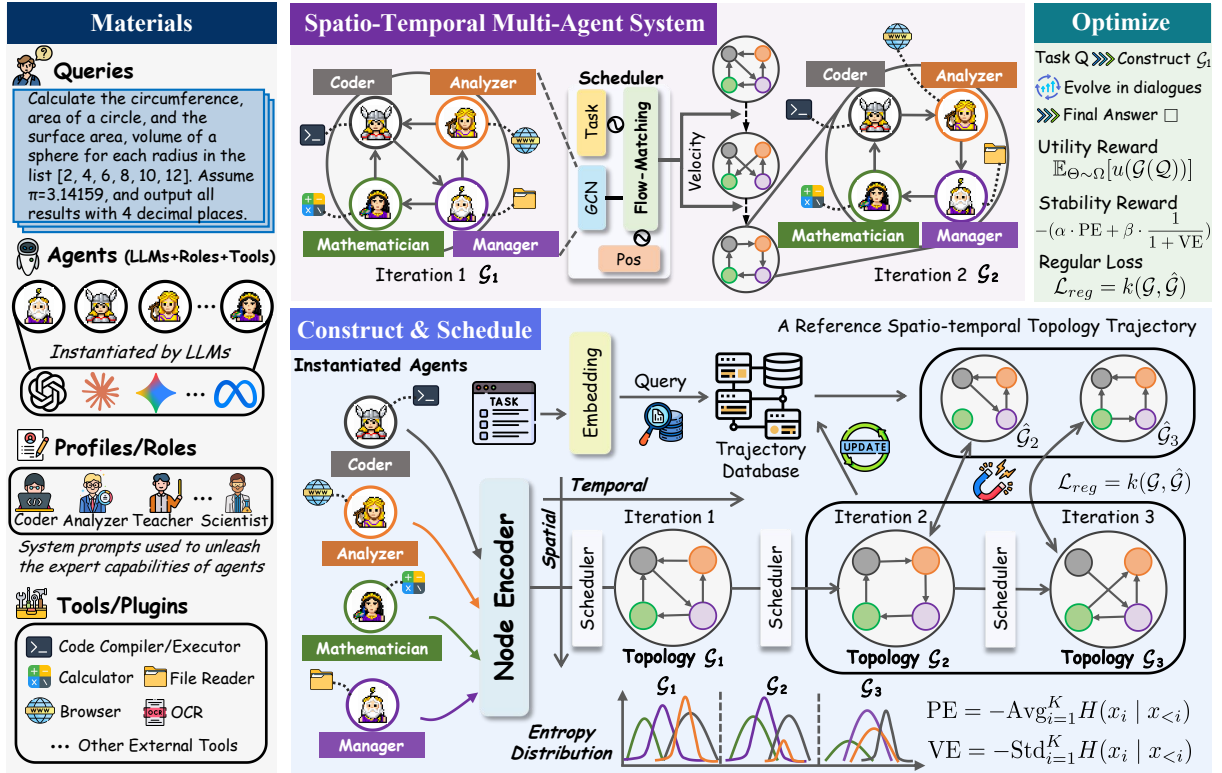


Figure 2: The overview of ST-EVO. The framework follows a macro-to-micro logic: Construct & Schedule outlines the core macro-workflow where the Scheduler iteratively refines the communication topology $\mathcal{G}_i$. Supporting this, Materials details the foundational components used to instantiate agents, while the Spatio-Temporal Multi-Agent System illustrates the step-wise topology evolution via graph flow matching. Finally, Optimize formulates the trajectory-retrieval regularized loss and the stability reward that align the Scheduler’s preferences.

graph embeddings, with $\mathcal{G}_{t-1}$ as the starting point:

$$L_{t-1} = \text{GCN}(X^{\text{node}}, A_{t-1}) + \epsilon, \quad (11)$$

$$L_{t-(1-p)} = p \cdot L_t + (1-p) \cdot L_{t-1}, \quad (12)$$

$$\hat{V}_p = \text{FM-Net}(L_{t-(1-p)}, p, \mathbf{H}_t), \quad (13)$$

where $L_{t-1}, L_t \in \mathbb{R}^{N \times d}$ are the graph embeddings. L_{t-1} denotes the embedding of the starting point $\mathcal{G}_{t-1}$, and L_t is the expected embedding of the end point $\mathcal{G}_t$ (we introduce this in Section 4.3). $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is a Gaussian noise to increase the diversity during training. The goal of Flow-Matching (FM-Net) is: given current position $L_{t-(1-p)}$ and condition $\mathbf{H}_t$, fit the velocity field between the start and end points. Note that $p \in [0, 1]$, we adopt the conditional optimal transport path to model a uniform velocity field, thus the optimization objective $\mathcal{L}_{\text{reg}}$ is:

$$\mathbb{E}_p ||\hat{V}_p \cdot W - (L_t \cdot W - L_{t-1} \cdot W)||^2, \quad (14)$$

where $W \in \mathbb{R}^{d \times N}$ is the projection matrix to transform graph embeddings into adjacent matrices. We also use normalization, and differentiable Bernoulli

resampling (Jang et al., 2016) to preserve adjacent matrices $\in \{0, 1\}^{N \times N}$. In the inference phase, the sampling process is a discretized integration process—see Algorithm 1.

Algorithm 1 Graph Flow Matching

- 1: Given condition $\mathbf{H}_t$, total steps P , and Topology $\mathcal{G}_{t-1} = (X^{\text{node}}, A_{t-1})$.
- 2: Sample a Gaussian noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
- 3: $\Delta p = 1/P$, $\hat{L} = \text{GCN}(X, A_{t-1}) + \epsilon$
- 4: **for** j **in** $\{0, 1, \dots, P-1\}$ **do**
- 5: $\hat{V}_j \leftarrow \text{FM-Net}(L_{t-(1-j\Delta p)}, j\Delta p, \mathbf{H}_t)$
- 6: $\hat{L} \leftarrow \hat{L} + \hat{V}_j \Delta p$
- 7: **end for**
- 8: **Return:** $\hat{A}_t = \text{Discretize}(\hat{L}W) \in \{0, 1\}^{N \times N}$

4.3 Scheduling Multi-agent Systems

After obtaining the compact generative Scheduler, ST-EVO possesses the fundamental infrastructure for spatio-temporal scheduling, while ensuring performance, flexibility, and efficiency. The next step is to support state-aware scheduling, and enhance system’s self-feedback ability.

To precisely supervise the MAS state, we draw inspiration from uncertainty estimation (MacKay, 1992), measuring the predictive entropy PE given input x by calculating the conditional entropy $H_i = H(x_i|x_{<i})$:

$$H_i = \sum_{x_i} P_{\Theta}(x_i|x_{<i}) \log P_{\Theta}(x_i|x_{<i}), \quad (15)$$

$$\text{PE} = -\frac{1}{K} \sum_{i=1}^K H(x_i|x_{<i}), \quad (16)$$

where high PE signals high uncertainty. We also consider the VarEntropy (VE), which reflects the internal self-consistency in MAS:

$$\text{VE} = -\text{Std}_{i=1}^K(H(x_i|x_{<i})) \quad (17)$$

With the help of PE and VE, we can classify system states into fine-grained categories—see Table 1.

PE	VE	System State Description
Low (↓)	Low (↓)	High Confidence (Deterministic)
High (↑)	High (↑)	Conflicting Uncertainty (Branching)
High (↑)	Low (↓)	Uniform Ignorance (Cluelessness)
Low (↓)	High (↑)	Overconfident Anomaly (Rare state)

Table 1: Taxonomy of System States based on Predictive Entropy (PE) and VarEntropy (VE).

Based on the summary in Table 1, we devise a quantitative indicator, named Stability Reward ($\mathcal{R}_{\text{sta}}$), to evaluate the stability of MAS:

$$\mathcal{R}_{\text{sta}} = -(\alpha \cdot \text{PE} + \beta \cdot \frac{1}{1 + \text{VE}}), \quad (18)$$

where higher $\mathcal{R}_{\text{sta}}$ value indicates higher stability. Note that for MAS stability measurement, $\mathcal{R}_{\text{sta}}$ is calculated over the top 10%–20% high-entropy tokens to prevent the stability signal from being diluted by trivial, high-confidence function words. We use $\mathcal{R}_{\text{sta}}$ as a reward model in training ST-EVO to align the preference into the Scheduler.

On the other hand, to enhance the self-feedback ability, we devise an experience-driven method. Specifically, it is designed as a retrieve-augment system to store the historical spatio-temporal scheduling trajectories, where the indices are query embeddings: X^{query} . The Update strategy relies on $\mathcal{R}_{\text{sta}}$ and token consumption, considering the stability and efficiency, to store the successfully executed scheduling trajectories during training.

After several cold-start iterations, multiple trajectories are stored, and can be used in experience

training. In ST-EVO, we retrieve the most similar scheduling trajectory $\hat{\mathcal{G}}$ with current X^{query} , which can reflect the historical scheduling preference of the similar task. Empirically, we store the latent graph embeddings $[\hat{L}_1, \hat{L}_2, \dots, \hat{L}_T]$, which are convenient to training the flow-matching based Scheduler. Recalling the Equation (14), we surrogate the L_t with $\hat{L}_t$ as successful experience for reference, regularizing the Scheduler. The full details can be found in Appendix A.2.

4.4 Optimizing ST-EVO

The ultimate objective is to achieve an effective few-shot paradigm, which trains ST-EVO on several samples, and efficiently work on numerous downstream queries. Considering the Spatio-Temporal Scheduling Communication Topologies $\mathcal{G}$ for query $\mathcal{Q}$, the utility objective is:

$$\arg \max_{\Theta} \mathbb{E}_{\Theta \sim \Omega} [u(\mathcal{G}(\mathcal{Q}))], \quad (19)$$

where Θ are the parameters of Scheduler, Ω are the parameter space, and $u(\cdot)$ denote the utility. Since Equation (19) is inherently intractable and non-differentiable due to the external API calls for LLMs, we adopt standard approaches (Zhang et al.; Li et al., 2025a; Zhang et al., 2025c) to apply the policy gradient for optimization:

$$\frac{1}{M} \sum_{m=1}^M u(\mathcal{G}^m(\mathcal{Q})) \nabla_{\Theta} (P(\mathcal{G}^m)), \quad (20)$$

where $\mathcal{G}^m$ are sampled spatio-temporal scheduling trajectories, and $\{\mathcal{G}^m(\mathcal{Q})\}_{m=1}^M$ are the outputs. $P(\mathcal{G}^m)$ is the probability of $\mathcal{G}^m$ being sampled and can be calculated through $\prod_{j=1}^T P(\mathcal{G}_j^m)$. The overall training objective of ST-EVO consists of three parts, where Equation (18), (20) are reward models, and Equation (14) is a regularization loss function. Full details can be found in Appendix A.1.

5 Experiments

5.1 Experimental Settings

Benchmarks We evaluate ST-EVO on 9 benchmarks: MMLU (Hendrycks et al.) for general reasoning, GSM8K (Cobbe et al., 2021), MultiArith (Roy and Roth, 2015), SVAMP (Patel et al., 2021), and AQuA (Ling et al., 2017) for math reasoning, HumanEval (Chen, 2021) and DS-1000 (Lai et al., 2023) for code generation, HotpotQA (Yang et al., 2018) for web search, and DDXPlus (Fansi Tchango et al., 2022) for medical reasoning.

Method	MMLU	GSM8K	MultiArith	SVAMP	AQuA	HumanEval	DS-1000	HotpotQA	DDXPlus	Avg.
Vanilla	80.47	87.15	93.40	87.25	69.27	73.28	38.40	82.35	56.40	74.22
<i>Single-Agent Systems</i>										
CoT	81.85 ^{↑1.38}	87.35 ^{↑0.20}	93.60 ^{↑0.20}	87.90 ^{↑0.65}	72.40 ^{↑3.13}	74.05 ^{↑0.77}	42.35 ^{↑3.95}	81.04 ^{↑1.31}	64.22 ^{↑7.82}	76.08
SC	82.95 ^{↑2.48}	87.66 ^{↑0.51}	94.40 ^{↑1.00}	87.40 ^{↑0.15}	71.85 ^{↑2.58}	75.82 ^{↑2.54}	37.66 ^{↓0.74}	82.24 ^{↓0.11}	66.43 ^{↑10.03}	76.27
<i>Multi-Agent Systems</i>										
Chain	82.40 ^{↑1.93}	87.23 ^{↑0.08}	92.88 ^{↓0.52}	87.16 ^{↓0.09}	69.85 ^{↑0.58}	73.40 ^{↑0.12}	38.25 ^{↓0.15}	80.77 ^{↓1.58}	65.28 ^{↑8.88}	75.25
Tree	81.55 ^{↑1.08}	86.35 ^{↓0.80}	92.95 ^{↓0.45}	88.25 ^{↑1.00}	71.44 ^{↑2.17}	75.33 ^{↑2.05}	43.50 ^{↑5.10}	81.50 ^{↓0.85}	69.25 ^{↑12.85}	76.68
Star	79.64 ^{↓0.83}	85.25 ^{↓1.90}	93.25 ^{↓0.15}	87.60 ^{↑0.35}	68.50 ^{↓0.77}	76.28 ^{↑3.00}	41.82 ^{↑3.42}	83.52 ^{↑1.17}	68.25 ^{↑11.85}	76.01
Complete Graph	82.75 ^{↑2.28}	88.05 ^{↑0.90}	94.73 ^{↑1.33}	88.20 ^{↑0.95}	70.44 ^{↑1.17}	84.25 ^{↑10.97}	45.05 ^{↑6.65}	83.35 ^{↑1.00}	70.35 ^{↑13.95}	78.57
Random Graph	83.20 ^{↑2.73}	88.25 ^{↑1.10}	94.80 ^{↑1.40}	87.10 ^{↓0.15}	70.16 ^{↑0.89}	84.14 ^{↑10.86}	47.48 ^{↑9.08}	84.28 ^{↑1.93}	70.44 ^{↑14.04}	78.87
AutoGen	83.25 ^{↑2.78}	89.40 ^{↑2.25}	95.05 ^{↑1.65}	88.15 ^{↑0.90}	71.50 ^{↑2.23}	83.50 ^{↑10.22}	48.55 ^{↑10.15}	83.25 ^{↑0.90}	71.83 ^{↑15.43}	79.39
LangGraph	84.10 ^{↑3.63}	88.75 ^{↑1.60}	95.10 ^{↑1.70}	88.03 ^{↑0.78}	71.27 ^{↑2.00}	84.66 ^{↑11.38}	47.95 ^{↑9.55}	84.92 ^{↑2.57}	72.50 ^{↑16.10}	79.70
LLM-Debate	83.28 ^{↑2.81}	89.95 ^{↑2.80}	95.45 ^{↑2.05}	88.30 ^{↑1.05}	73.52 ^{↑4.25}	85.24 ^{↑11.96}	49.26 ^{↑10.86}	84.55 ^{↑2.20}	71.33 ^{↑14.93}	80.10
<i>Spatial Evolving Multi-Agent Systems</i>										
GPTSwarm	83.80 ^{↑3.33}	90.20 ^{↑3.05}	96.28 ^{↑2.88}	87.28 ^{↑0.03}	75.44 ^{↑6.17}	87.14 ^{↑13.86}	52.35 ^{↑13.95}	85.40 ^{↑3.05}	75.57 ^{↑19.17}	81.50
G-Designer	84.63 ^{↑4.16}	93.35 ^{↑6.20}	97.60 ^{↑4.20}	89.52 ^{↑2.27}	76.30 ^{↑7.03}	89.40 ^{↑16.12}	54.82 ^{↑16.42}	87.66 ^{↑5.31}	76.42 ^{↑20.02}	83.30
ARG-Designer	85.04 ^{↑4.57}	94.40 ^{↑7.25}	97.35 ^{↑3.95}	88.95 ^{↑1.70}	79.48 ^{↑10.21}	88.25 ^{↑14.97}	55.60 ^{↑17.20}	86.45 ^{↑4.10}	76.03 ^{↑19.63}	83.51
SafeSieve	84.65 ^{↑4.18}	93.20 ^{↑6.05}	97.80 ^{↑4.40}	90.10 ^{↑2.85}	80.40 ^{↑11.13}	90.15 ^{↑16.87}	53.73 ^{↑15.33}	87.22 ^{↑4.87}	77.94 ^{↑21.54}	83.91
<i>Temporal Evolving Multi-Agent Systems</i>										
AFlow	83.22 ^{↑2.75}	89.50 ^{↑2.35}	94.28 ^{↑0.88}	86.48 ^{↓0.77}	78.20 ^{↑8.93}	89.25 ^{↑15.97}	46.82 ^{↑8.42}	90.04 ^{↑7.69}	74.25 ^{↑17.85}	81.34
SpecReason	84.20 ^{↑3.73}	91.40 ^{↑4.25}	95.40 ^{↑2.00}	87.62 ^{↑0.37}	81.36 ^{↑12.09}	91.22 ^{↑17.94}	50.08 ^{↑11.68}	88.29 ^{↑5.94}	76.58 ^{↑20.18}	82.91
STEER	84.40 ^{↑3.93}	93.50 ^{↑6.35}	95.82 ^{↑2.42}	87.40 ^{↑0.15}	82.66 ^{↑13.39}	89.02 ^{↑15.74}	51.34 ^{↑12.94}	89.35 ^{↑7.00}	75.20 ^{↑18.80}	83.19
<i>Spatio-Temporal Evolving Multi-Agent Systems</i>										
ST-EVO (Ours)	89.85 ^{↑9.38}	97.60 ^{↑10.45}	99.05 ^{↑5.65}	94.72 ^{↑7.47}	86.56 ^{↑17.29}	94.36 ^{↑21.08}	58.65 ^{↑20.25}	92.52 ^{↑10.17}	82.50 ^{↑26.10}	88.42

Table 2: Accuracy (%) comparison with four types baselines, including Single-Agent Systems, Multi-Agent Systems, Spatial Evolving Multi-Agent Systems, and Temporal Evolving Multi-Agent Systems. All baselines and ST-EVO are driven by gpt-oss-120b (Vanilla). **Red**: the best, **Blue**: the runner-up.

Baselines We compare ST-EVO with static approaches like COT (Wei et al., 2022), Self-Consistency (Wang et al., 2023), Chain, Tree, Star, Complete Graph, Random Graph, AutoGen (Wu et al., 2023), LangGraph (Inc., 2024), LLM-Debate (Du et al., 2023), Spatial Evolving approaches like GPTSwarm (Zhuge et al., 2024), G-Designer (Zhang et al.), ARG-Designer (Li et al., 2025a), and SafeSieve (Zhang et al., 2025c), Temporal Evolving approaches like AFlow (Zhang et al., 2025a), SpecReason (Damani et al., 2024) and STEER (Lee et al., 2025).

Implementation Details We run ST-EVO and all baselines with gpt-oss-120b. To synthesize dialogue history and generate the final solution, we implement a summarizer agent, maintaining $T = 3$ across all baseline comparisons. For the Node Encoder and textual embedding, we adopt all-MiniLM-L6-v2 with an embedding dimension of $D = 384$. For all datasets, we use 40–80 queries to train the Scheduler. And the initial anchor topology $\mathcal{G}_0$ is customized for each dataset.

5.2 Main Results

Results in Table 2 demonstrate that ST-EVO achieves consistent state-of-the-art performance across nine benchmarks, with average accuracy

improvements (against Vanilla) from 5.65% to 26.10%. Compared with Temporal or Spatial Evolving Multi-Agent Systems, ST-EVO possesses more stable performance improvement when faced with heterogeneous tasks, from simple tasks on MultiArith (5.65% ↑) and SVAMP (7.47% ↑) to hard tasks on DS-1000 (20.25% ↑) and DDXPlus (26.10% ↑). The larger improvements on harder tasks demonstrate the effectiveness of Spatio-Temporal Scheduling.

5.3 Method Analysis

Efficiency Analysis We analyze the efficiency of ST-EVO in Figure 3. When adopting the same number of agents in MAS, ST-EVO outperforms all baselines a lot with less token consumption, showing the capability to fully schedule and utilize the computing resources and specific functions in appropriate moments. Compared with previous advanced baseline ARG-Designer and SafeSieve, ST-EVO only needs nearly 50% of token consumption on MMLU and HumanEval, but with higher accuracy (about 5% ↑). We also consider the scalability in Table 8 and 9. Results show that the complexities of baselines like GPTSwarm and G-Designer grow linearly as the number of agents or dialogue iterations increases, while ST-EVO has a lower com-

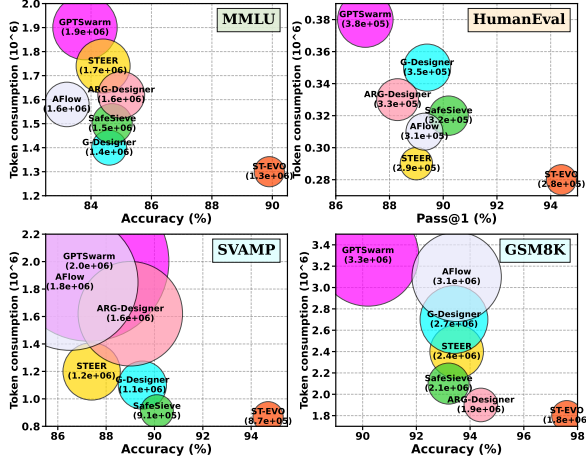


Figure 3: Visualization of performance metrics and token consumption of different MAS across MMLU, HumanEval, GSM8K, SVAMP. The diameters of the circles represent the scales of token consumption.

plexity growth rate, since spatio-temporal scheduling can effectively avoid unnecessary communication during execution.

Robustness Analysis We validate ST-EVO’s robustness in Figure 4. To demonstrate whether the state-aware scheduling is useful, we first conduct experiments on all 9 benchmarks and report the average results of accuracy and stability $e^{\mathcal{R}_{sta}}$. For approaches, we select G-Designer, ARG-Designer, SafeSieve and ST-EVO since they can be trained using the Stability Reward $\mathcal{R}_{sta}$.

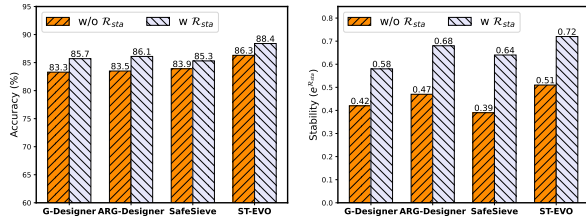


Figure 4: We utilize the accuracy (%) and $e^{\mathcal{R}_{sta}} \in [0, 1]$ to visualize the stability of ST-EVO. Higher values indicate better stabilities.

Observations show that all approaches can be improved on both the accuracy and stability using the reward $\mathcal{R}_{sta}$. This demonstrates that $\mathcal{R}_{sta}$ can effectively reflect the uncertainty of the MAS, and aligning this preference in Scheduler can lead to generalization on unseen downstream queries. Note that ST-EVO has the most improvement.

We then conduct experiments to validate whether ST-EVO can fight against external disturbances. Following the experimental settings of prompt injection attack in G-Designer (Zhang et al.), we compare ST-EVO with other baselines in Figure 5.

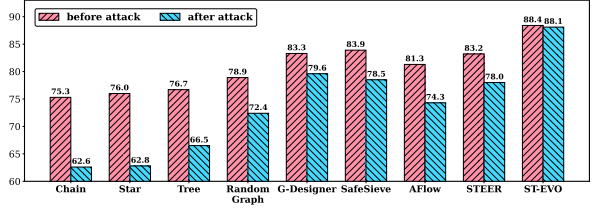


Figure 5: We compare the accuracy (%) of multiple multi-agent systems before and after prompt attacks on all benchmarks, and report the average results.

Results show that static multi-agent systems suffer significant accuracy degradation under partial system attacks, with drops about 6.5%–12.7%. Among self-evolving multi-agents, G-Designer and SafeSieve benefit from task-adaptive topology design, only suffer drops under 5%. Thanks to such adaptive topology adjustment, ST-EVO gains exceptional robustness against adversarial attacks, and can preserve nearly consistent performance before and after the attacks.

Ablation Studies We conduct ablation studies on ST-EVO—see Table 3: (1) w/o Scheduling, which limits the Scheduler to output a communication topology for the entire query, (2) w/o $\mathcal{L}_{reg}$, which does not rely on the experience, and (3) w/o Node Encoder, which removes the task feature X^{query} in scheduling, and using adaptive representations to replace the node features.

Variant	MMLU	HumanEval	SVAMP	GSM8K
ST-EVO	89.9	94.4	94.7	97.6
w/o Scheduling	84.7	89.6	88.6	92.5
w/o $\mathcal{L}_{reg}$	84.9	88.2	90.3	93.7
w/o Node Encoder	82.0	84.8	87.5	94.8

Table 3: Ablation studies tested on MMLU, HumanEval, SVAMP, and GSM8K benchmarks.

We observe that w/o Scheduling and w/o Node Encoder matters a lot, because the lack of them would break the core query-adaptive Spatio-Temporal Scheduling mechanism of ST-EVO. w/o $\mathcal{L}_{reg}$ may also make the MAS vulnerable to interference, thus degrading the performance.

6 Conclusion

In this work, we introduce ST-EVO, the first Spatio-Temporal Evolving MAS, which supports query-aware and dialogue-wise communication topology scheduling. Comprehensive experiments on 9 benchmarks demonstrate the state-of-the-art performance of ST-EVO, and it can ensure both efficiency and robustness on complex scenarios.

Limitations

While **ST-EVO** demonstrates advancement on multiple tasks, it has limitations. Since it follows the few-shot learning paradigm, the Scheduler needs to be trained on a few corresponding samples to recognize the task characteristics. However, in some real-world scenarios with scarce data, this mechanism may lose its advantages. In the future, we aim at building a fully zero-shot and effective mechanism to support construction and scheduling for multi-agent systems.

Ethical considerations

As for ethical considerations, our study only employs standard public datasets and commonly used open-source frameworks, and we find no obvious ethical issues in this work.

Acknowledgments

This work was partially supported by National Natural Science Foundation of China (62372179). Xingjian Wu, Xvyuan Liu, and Junkai Lu contributed equally to this work, and Bin Yang is the corresponding author. We use LLMs to conduct grammar checking and polishing on our manually-written contents.

References

- Mark Chen. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.
- 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.
- Mehul Damani, Idan Shenfeld, Andi Peng, Andreea Bobu, and Jacob Andreas. 2024. Learning how hard to think: Input-adaptive allocation of lm computation. *arXiv preprint arXiv:2410.04707*.
- Han Ding, Yinheng Li, Junhao Wang, and Hang Chen. 2024. Large language model agent in financial trading: A survey. *arXiv preprint arXiv:2408.06361*.
- Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2023. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first International Conference on Machine Learning*.
- Andrew Estornell and Yang Liu. 2024. Multi-llm debate: Framework, principals, and interventions. *Advances in Neural Information Processing Systems*, 37:28938–28964.
- Jinyuan Fang, Yanwen Peng, Xi Zhang, Yingxu Wang, Xinhao Yi, Guibin Zhang, Yi Xu, Bin Wu, Siwei Liu, Zihao Li, Zhaochun Ren, Nikos Aletras, Xi Wang, Han Zhou, and Zaiqiao Meng. 2025. [A comprehensive survey of self-evolving AI agents: A new paradigm bridging foundation models and lifelong agentic systems](#). *CoRR*, abs/2508.07407.
- Arsene Fansi Tchango, Rishab Goel, Zhi Wen, Julien Martel, and Joumana Ghosn. 2022. Ddxplus: A new dataset for automatic medical diagnosis. *Advances in neural information processing systems*, 35:31306–31318.
- Shanshan Han, Qifan Zhang, Yuhang Yao, Weizhao Jin, and Zhaozhuo Xu. 2024. Llm multi-agent systems: Challenges and open problems. *arXiv preprint arXiv:2402.03578*.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.
- Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiaowu 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 *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- LangChain Inc. 2024. Langgraph. <https://github.com/langchain-ai/langgraph>.
- Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.
- Ye Jin, Xiaoxi Shen, Huiling Peng, Xiaolan Liu, Jingli Qin, Jiayang Li, Jintao Xie, Peizhong Gao, Guyue Zhou, and Jiangtao Gong. 2023. Surrealdriver: Designing generative driver agent simulation framework in urban contexts based on large language model. *arXiv preprint arXiv:2309.13193*, 5(7):8.
- Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Wen-tau Yih, Daniel Fried, Sida Wang, and Tao Yu. 2023. Ds-1000: A natural and reliable benchmark for data science code generation. In *International Conference on Machine Learning*, pages 18319–18345. PMLR.
- Sangmook Lee, Dohyung Kim, Hyukhun Koh, Nakyeong Yang, and Kyomin Jung. 2025. Confidence-guided stepwise model routing for cost-efficient reasoning. *arXiv preprint arXiv:2511.06190*.
- Shiyuan Li, Yixin Liu, Qingsong Wen, Chengqi Zhang, and Shirui Pan. 2025a. Assemble your crew: Automatic multi-agent communication topology design via autoregressive graph generation. *arXiv preprint arXiv:2507.18224*.

- Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025b. Search-o1: Agentic search-enhanced large reasoning models. *arXiv preprint arXiv:2501.05366*.
- Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167.
- Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. 2023. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.
- Yong Liu, Guo Qin, Zhiyuan Shi, Zhi Chen, Caiyin Yang, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. 2025. Sundial: A family of highly capable time series foundation models. In *Forty-second International Conference on Machine Learning*.
- Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. 2023. Dynamic llm-agent network: An llm-agent collaboration framework with agent team optimization. *arXiv preprint arXiv:2310.02170*.
- David JC MacKay. 1992. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472.
- Boye Niu, Yiliao Song, Kai Lian, Yifan Shen, Yu Yao, Kun Zhang, and Tongliang Liu. 2025. Flow: Modularized agentic workflow automation. *arXiv preprint arXiv:2501.07834*.
- Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*.
- William Peebles and Saining Xie. 2023. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205.
- 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 *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 15174–15186. Association for Computational Linguistics.
- Nils Reimers and Iryna Gurevych. 2019. **Sentence-bert: Sentence embeddings using siamese bert-networks**. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3980–3990. Association for Computational Linguistics.
- Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1743–1752.
- Significant Gravitas. 2023. Autogpt. <https://github.com/Significant-Gravitas/AutoGPT>.
- Xiangru Tang, Tianyu Hu, Muyang Ye, Yanjun Shao, Xunjian Yin, Siru Ouyang, Wangchunshu Zhou, Pan Lu, Zhuosheng Zhang, Yilun Zhao, Arman Cohan, and Mark Gerstein. 2025. **Chemagent: Self-updating library in large language models improves chemical reasoning**. *CoRR*, abs/2501.06590.
- Junlin Wang, WANG Jue, Ben Athiwaratkun, Ce Zhang, and James Zou. 2025a. Mixture-of-agents enhances large language model capabilities. In *The Thirteenth International Conference on Learning Representations*.
- Senzhang Wang, Jiannong Cao, and S Yu Philip. 2020. Deep learning for spatio-temporal data mining: A survey. *IEEE transactions on knowledge and data engineering*, 34(8):3681–3700.
- 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 *The Eleventh International Conference on Learning Representations*.
- Yingxu Wang, Siwei Liu, Jinyuan Fang, and Zaiqiao Meng. 2025b. Evoagentx: An automated framework for evolving agentic workflows. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 643–655.
- 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 *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.
- 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.
- Xingjian Wu, Jianxin Jin, Wanghui Qiu, Peng Chen, Yang Shu, Bin Yang, and Chenjuan Guo. 2026a. Aurora: Towards universal generative multimodal time series forecasting. In *ICLR*.

- Xingjian Wu, Junkai Lu, Zhengyu Li, Xiangfei Qiu, Jilin Hu, Chenjuan Guo, Christian S. Jensen, and Bin Yang. 2026b. Timeart: Towards agentic time series reasoning via tool-augmentation.
- Yiyuan Yang, Ming Jin, Haomin Wen, Chaoli Zhang, Yuxuan Liang, Lintao Ma, Yi Wang, Chenghao Liu, Bin Yang, Zenglin Xu, Shirui Pan, and Qingsong Wen. 2026. [A survey on diffusion models for time series and spatio-temporal data](#). *ACM Comput. Surv.*, 58(8):196:1–196:39.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models, 2023. URL <https://arxiv.org/abs/2305.10601>, 3:1.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.
- Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. G-designer: Architecting multi-agent communication topologies via graph neural networks. In *Forty-second International Conference on Machine Learning*.
- 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. 2025a. [Aflow: Automating agentic workflow generation](#). In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.
- Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, Jian Xie, Yuxuan Sun, Boyu Gou, Qi Qi, Zihang Meng, Jianwei Yang, Ning Zhang, Xian Li, Ashish Shah, and 11 others. 2025b. [Agent learning via early experience](#). *CoRR*, abs/2510.08558.
- Ruijia Zhang, Xinyan Zhao, Ruixiang Wang, Sigen Chen, Guibin Zhang, An Zhang, Kun Wang, and Qingsong Wen. 2025c. Safesieve: From heuristics to experience in progressive pruning for llm-based multi-agent communication. *arXiv preprint arXiv:2508.11733*.
- Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. 2019. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1):1–23.
- Qinkai Zheng, Xiao Xia, Xu Zou, Yuxiao Dong, Shan Wang, Yufei Xue, Lei Shen, Zihan Wang, Andi Wang, Yang Li, Teng Su, Zhilin Yang, and Jie Tang. 2023. [Codegeex: A pre-trained model for code generation with multilingual benchmarking on humaneval-x](#). In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, pages 5673–5684. ACM.
- Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. 2025. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. *arXiv preprint arXiv:2504.03160*.
- Zihao Zhou, Bin Hu, Chenyang Zhao, Pu Zhang, and Bin Liu. 2024. Large language model as a policy teacher for training reinforcement learning agents. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 5671–5679.
- Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. 2024. Gptswarm: Language agents as optimizable graphs. In *Forty-first International Conference on Machine Learning*.

A Details of Methodology

We introduce some additional technical details in this section, related to the specific implementations.

A.1 Details of Optimization Objectives

Equation (18), (20), and (14) introduce three optimization objectives of ST-EVO. Among them, Equation (18), (20) are optimized as reward models. Intuitively, we merge them as $\mathcal{L}_{\text{utility}}$:

$$-\frac{1}{M} \sum_{m=1}^M e^{\mathcal{R}_{\text{sta}}} \cdot u(\mathcal{G}^m(\mathcal{Q})) \cdot \nabla_{\Theta}(P(\mathcal{G}^m)), \quad (21)$$

where $\mathcal{R}_{\text{sta}} \in \mathbb{R}^-$, so that $e^{\mathcal{R}_{\text{sta}}} \in [0, 1]$ can be treated as a confidence level, which reflects the stability of the scheduling trajectory. And the total loss function $\mathcal{L}$ is:

$$\mathcal{L} = \mathcal{L}_{\text{utility}} + \gamma \cdot \mathcal{L}_{\text{reg}} \quad (22)$$

A.2 Details of Retrieve-Augment

To efficiently manage temporal node states under memory constraints, we introduce our retrieval mechanism. Unlike traditional graph stores, our memory $\mathcal{M}$ with capacity L stores latent tensor representations. Each item $m_i \in \mathcal{M}$ is a tuple $(\mathbf{k}_i, \mathcal{H}_i, c_i, u_i, \alpha_i)$, where $\mathbf{k}_i$ is the query embedding key, $\mathcal{H}_i$ is the sequence of node feature tensors (values), c_i is the computational cost, u_i is the Stability Reward (i.e., $\mathcal{R}_{\text{sta}}$), and α_i tracks access frequency.

Retrieval and Update. Given a query $\mathbf{q}$, we retrieve the top- k items maximizing the dot-product similarity $\mathbf{k}_i^\top \mathbf{q}$. To maintain the most valuable information within the fixed capacity L , we employ a value-based eviction policy. When the buffer is full, a new candidate m_{new} replaces the item with the minimum retention score $S(m)$ if $S(m_{\text{new}}) > \min_j S(m_j)$. The score $S(m)$ balances reliability, efficiency, and popularity:

$$S(m_i) = \frac{1 + \log(\alpha_i + 1)}{c_i \cdot |u_i| + \epsilon} \quad (23)$$

where ϵ is a smoothing term. This policy prioritizes items with high stability ($|u_i| \rightarrow 0$) and low computational cost (c_i), while logarithmically scaling with access frequency to preserve useful knowledge without saturation.

A.3 Entropy Estimation without Token Log-Probabilities

The original stability reward uses token-level log-probabilities to compute Predictive Entropy (PE) and VarEntropy (VE). When an inference backend does not expose token probabilities, we use two alternative estimators. The first is a sample-based estimator that draws K responses for the same prompt. Let $p(a)$ denote the empirical frequency of answer a among these responses. We estimate answer-level uncertainty as

$$\widehat{\text{PE}} = - \sum_a p(a) \log p(a). \quad (24)$$

We estimate response-level variation from the mean pairwise cosine distance between response embeddings. Given embeddings $\{\mathbf{e}_i\}_{i=1}^K$, this proxy is

$$\widehat{\text{VE}} = 1 - \frac{2}{K(K-1)} \sum_{i < j} \cos(\mathbf{e}_i, \mathbf{e}_j). \quad (25)$$

The second alternative is a lightweight confidence probe: a small MLP predicts PE and VE directly from output-text features and therefore requires neither token probabilities nor repeated API calls.

Entropy Estimator	Avg. Acc.	PE Corr.	Extra Calls
Logprob-based	94.15	1.00	0
Sample-based ($K = 5$)	92.95	0.85	5×
Sample-based ($K = 3$)	92.23	—	3×
Confidence probe	92.40	0.78	0
w/o PE/VE	91.80	—	0

Table 4: Comparison of entropy estimators. Average accuracy is computed over MMLU, HumanEval, SVAMP, and GSM8K. PE Corr. is the Pearson correlation with logprob-based PE, and extra calls are measured per agent and iteration.

Table 4 shows that sample-based estimation with $K = 5$ is only 1.20 points below the logprob-based implementation and remains 1.15 points above removing entropy altogether. The confidence probe also preserves most of the benefit without additional model calls. Thus, token probabilities provide the most accurate and efficient signal when available, but they are not a strict deployment requirement for ST-EVO.

B Details of Experiments

B.1 Dataset Statistics

To comprehensively evaluate the model’s capabilities, we selected a diverse set of benchmarks across

Category	Dataset	Answer Type	Metric	#Test	License
General reasoning	MMLU	Multi-choice	Acc.	153	MIT License
	HotpotQA	Span	F1	7,405	CC BY-SA 4.0
Math reasoning	GSM8K	Number	Acc.	1,319	MIT License
	MultiArith	Number	Acc.	600	Unspecified
	SVAMP	Number	Acc.	1,000	MIT License
	AQuA	Multi-choice	Acc.	254	Apache-2.0
Code generation	HumanEval	Code	Pass@1	164	MIT License
	DS-1000	Code	Acc.	1,000	CC BY-SA 4.0
Medical reasoning	DDXPlus	Multi-choice	Acc.	282,906	CC BY-SA 4.0

Table 5: Dataset descriptions and statistics.

four primary categories: general reasoning, mathematical reasoning, code generation, and medical reasoning. Table 5 provides a detailed summary of these datasets, including their respective evaluation metrics, test set sizes, and licenses.

B.2 Evaluation on Challenging Agentic Benchmarks

The nine standard benchmarks provide controlled coverage of reasoning, code generation, retrieval, and medical tasks. To test whether dynamic coordination remains useful in more realistic long-horizon settings, we additionally evaluate on SWE-Bench Verified, Terminal-Bench, MLE-Bench, and WebArena. These benchmarks require codebase-level debugging, terminal interaction, end-to-end machine-learning engineering, and interactive web navigation, respectively. All methods use gpt-oss-120b under the same communication-based multi-agent protocol; consequently, the absolute numbers are not directly comparable with single-agent tool-use pipelines that use different scaffolds.

As shown in Table 6, ST-EVO improves over the strongest baseline by 5.5, 6.1, 5.5, and 6.4 points on the four benchmarks, respectively. These gains are at least as large as those on the standard benchmark suite, indicating that dialogue-wise topology adaptation becomes more valuable as tasks require longer planning horizons and more heterogeneous forms of collaboration.

B.3 Generalization across Backbone Models

We evaluate whether the learned scheduler depends on a particular backbone by replacing gpt-oss-120b with five additional self-deployable models. The models cover dense, mixture-of-experts (MoE), and code-specialized architectures, with active parameter counts ranging from 3B to 120B. Table 7 reports averages over the original nine benchmarks.

ST-EVO consistently improves over the strongest baseline across all model sizes and architectures. The gains are larger for models with only 3B active parameters (5.79–5.96 points) than for the 120B dense model (4.51 points), suggesting that adaptive collaboration can compensate for weaker individual agents. Across the 3B-active models, PE/VE remains correlated with output quality (Pearson $r > 0.65$), supporting entropy-aware scheduling at smaller scales.

B.4 Scalability

C Ablation Study on the Scheduler

To further examine the contribution of the scheduler in ST-EVO, we evaluate different scheduling strategies and generative mechanisms, conduct combined module ablations, analyze the sensitivity to core hyperparameters, and quantify topology changes between adjacent iterations.

C.1 Ablation on scheduling strategies.

We first design several simple scheduler variants for comparison. First, *w/ LLM Scheduler* replaces our learned scheduler with a language model, where gpt-oss-120b directly reasons over the current dialogue state and determines the communication topology at each reasoning step. Second, *w/o Scheduling* only uses our generated scheduler to produce the initial communication topology, and then keeps this topology fixed throughout the entire multi-turn reasoning process, which resembles the static-topology setting in G-Designer. Third, *w/ Random Scheduler* randomly samples a communication topology at each reasoning step, without considering the evolving intermediate reasoning states.

As shown in Table 10, ST-EVO consistently outperforms all scheduler variants across the four benchmarks. Replacing the learned scheduler with an LLM-based scheduler leads to an average drop of 3.05%, suggesting that directly prompting a language model to design the topology is insufficient for stable and fine-grained multi-step coordination. The *w/o Scheduling* variant further degrades the average performance by 5.30%, indicating that using a fixed topology cannot effectively adapt to the evolving reasoning states during multi-agent collaboration. The random scheduler performs the worst, with an average drop of 6.15%, which confirms that dynamically changing the topology alone is not enough; the topology must be optimized ac-

Method	SWE-Bench	Terminal-Bench	MLE-Bench	WebArena
Vanilla (single)	37.4	32.6	9.2	15.7
Chain (4 agents)	39.1	34.8	10.6	17.9
Complete Graph	40.8	36.3	12.1	20.3
G-Designer	44.2	39.7	15.8	24.6
SafeSieve	45.6	40.9	16.4	25.8
STEER	<u>46.3</u>	<u>41.5</u>	<u>17.2</u>	<u>26.7</u>
ST-EVO	51.8	47.6	22.7	33.1

Table 6: Results (%) on four challenging agentic benchmarks. All methods use gpt-oss-120b and the same communication-based multi-agent protocol.

Backbone	Architecture	Vanilla	Best Baseline	ST-EVO	Gain
gpt-oss-120b	120B Dense	74.22	83.91	88.42	+4.51
Qwen3.5-122B-A10B	122B MoE (10B active)	71.99	81.75	86.48	+4.73
Qwen3.6-27B	27B Dense	70.05	79.88	84.83	+4.95
Qwen3.6-35B-A3B	35B MoE (3B active)	64.16	75.18	81.14	+5.96
Qwen3-30B-A3B	30B MoE (3B active)	62.71	73.74	79.59	+5.85
Qwen3-Coder-30B-A3B	30B MoE (3B active)	61.67	72.50	78.29	+5.79

Table 7: Average accuracy (%) across the nine benchmarks with different backbone models. Gain is measured against the strongest baseline for each backbone.

#Agents	5	10	20
Random Graph			
Accuracy (%)	82.7	83.3	83.6
Time (min)	14.2	31.4	59.7
#Tokens	431,064	1,317,148	4,245,925
GPTSwarm			
Accuracy (%)	83.6	84.2	84.8
Time (min)	67.8	201.8	422.5
#Tokens	1,672,201	5,103,328	14,054,825
G-Designer			
Accuracy (%)	84.1	85.2	87.3
Time (min)	22.4	41.8	75.23
#Tokens	1,422,309	2,887,532	6,130,221
AFlow			
Accuracy (%)	83.0	84.5	85.1
Time (min)	36.7	77.2	135.4
#Tokens	1,851,925	3,485,925	7,292,296
STEER			
Accuracy (%)	84.0	86.2	87.8
Time (min)	48.2	105.6	233.2
#Tokens	2,025,125	4,256,252	9,026,290
ST-EVO			
Accuracy (%)	89.2	91.6	93.3
Time (min)	15.2	28.4	51.6
#Tokens	1,162,296	1,825,926	3,326,037

Table 8: Comparison of accuracy, time, and token consumption across different agent configurations. We use the MMLU benchmark and gpt-oss-120b as the base LLM.

#Iterations	3	6	12
Random Graph			
Accuracy (%)	82.7	83.1	83.3
Time (min)	14.2	27.4	52.5
#Tokens	431,064	816,266	1,425,295
GPTSwarm			
Accuracy (%)	83.6	84.0	84.2
Time (min)	67.8	112.5	199.4
#Tokens	1,672,201	3,025,162	5,925,102
G-Designer			
Accuracy (%)	84.1	83.8	84.3
Time (min)	22.4	42.5	79.2
#Tokens	1,422,309	3,015,285	5,723,293
AFlow			
Accuracy (%)	83.0	83.4	83.6
Time (min)	36.7	67.2	122.4
#Tokens	1,851,925	3,425,291	6,902,946
STEER			
Accuracy (%)	84.0	84.3	84.5
Time (min)	48.2	87.2	144.5
#Tokens	2,025,125	3,821,229	7,402,409
ST-EVO			
Accuracy (%)	89.2	89.4	89.6
Time (min)	15.2	26.4	44.2
#Tokens	1,162,296	1,823,394	2,901,034

Table 9: Comparison of accuracy, time, and token consumption across different dialogue iterations. We use the MMLU benchmark and gpt-oss-120b as the base LLM.

Variant	MMLU	HumanEval	SVAMP	GSM8K	Avg.
ST-EVO	89.9	94.4	94.7	97.6	94.15
w/ LLM Scheduler	86.8	91.2	91.5	94.9	91.10
w/o Scheduling	84.7	89.6	88.6	92.5	88.85
w/ Random Scheduler	83.8	88.7	87.9	91.6	88.00

Table 10: Ablation study on different scheduling strategies. We report the accuracy (%) on MMLU, HumanEval, SVAMP, and GSM8K, as well as the average performance.

cording to the current reasoning context. These results demonstrate that the proposed scheduler is essential for enabling effective step-wise communication and coordination among agents.

C.2 Ablation on generative mechanisms

We further study why **ST-EVO** adopts Flow Matching as the generative mechanism for topology scheduling. The key motivation is that Flow Matching can naturally serve as an interpolator: it starts from an arbitrary source distribution and learns a velocity field that transports it toward the target distribution. This property allows us to directly take the communication topology from the previous reasoning step as the starting point, and then learn the transition toward the target topology of the next step. In this way, the starting topology and the conditional information are explicitly decoupled, making the scheduler better aligned with the dynamic nature of multi-step topology evolution.

By contrast, VAE- and Diffusion-based generative mechanisms need to inject the previous communication topology into the conditional network to achieve a similar effect. However, the conditional network also needs to encode task features and step-dependent features. Coupling the previous topology with these heterogeneous conditions may make the conditioning signal entangled and hinder the extraction of key scheduling information. To verify this point, we construct two additional variants. For *w/ Diffusion Scheduler*, we adopt DDIM as the sampling mechanism, keep the same MLP-based scheduler architecture as **ST-EVO**, and use AdaLN for condition injection. The training objective is changed to a denoising objective with the negative ELBO loss. For *w/ VAE Scheduler*, we adopt a conditional VAE decoder with the same condition-injection strategy, and train it with the negative ELBO loss.

Table 11 shows that the Flow-Matching-based scheduler achieves the best performance on all benchmarks. Compared with **ST-EVO**, the

Variant	MMLU	HumanEval	SVAMP	GSM8K	Avg.
ST-EVO	89.9	94.4	94.7	97.6	94.15
w/ Diffusion Scheduler	87.6	92.0	92.4	95.5	91.88
w/ VAE Scheduler	87.1	91.6	91.8	95.0	91.38

Table 11: Ablation study on different generative mechanisms for topology scheduling. We report the accuracy (%) on MMLU, HumanEval, SVAMP, and GSM8K, as well as the average performance.

Diffusion-based scheduler suffers an average drop of 2.27%, while the VAE-based scheduler leads to a larger average drop of 2.77%. These results support our design choice: modeling topology evolution as a flow from the previous communication topology to the next target topology provides a more direct and effective formulation than coupling the previous topology into the conditional network of VAE- or Diffusion-based generators. Therefore, Flow Matching is better suited for learning communication transitions in **ST-EVO**.

C.3 Combined Ablation of Feedback Modules

The stability reward provides real-time state feedback, whereas retrieval provides historical scheduling experience. We jointly remove these components to determine whether one can substitute for the other.

Variant	MMLU	HumanEval	SVAMP	GSM8K	Avg.
ST-EVO (full)	89.90	94.40	94.70	97.60	94.15
w/o Entropy Reward	87.23	91.76	92.41	95.82	91.81
w/o Retrieval	84.86	88.24	90.31	93.72	89.28
w/o Both	83.17	86.43	87.94	91.76	87.33

Table 12: Combined ablation of the entropy-based stability reward and retrieve-augment module. All values are accuracy (%).

Removing the entropy reward and retrieval separately decreases the average by 2.34 and 4.87 points, respectively. Removing both produces a 6.82-point drop, close to the 7.21-point drop predicted by adding the two individual effects. This result indicates that the modules provide largely complementary information: entropy characterizes the current system state, while retrieval supplies relevant historical trajectories.

C.4 Sensitivity to Core Hyperparameters

We vary the stability-reward weights (α, β) , trajectory-cache capacity L , and number of Flow-Matching integration steps P . Unless varied, the other hyperparameters use their default values.

Balanced weights achieve the best average performance. Setting either coefficient to zero causes

α	β	MMLU	HumanEval	SVAMP	GSM8K / Avg.
0.3	0.7	88.17	92.68	93.14	96.33 / 92.58
0.5	0.5	89.85	94.36	94.72	97.60 / 94.13
0.7	0.3	89.06	93.51	93.85	96.92 / 93.34
1.0	0.0	87.43	91.74	91.96	95.48 / 91.65
0.0	1.0	86.72	91.15	91.37	94.93 / 91.04

Table 13: Sensitivity to the stability-reward weights α and β . Results are accuracy (%).

a drop of at least 2.48 points, confirming that PE and VE capture complementary aspects of system uncertainty.

Capacity L	MMLU	HumanEval	SVAMP	GSM8K	Avg.
0	84.86	88.24	90.31	93.72	89.28
25	87.74	91.93	92.76	95.81	92.06
50	88.93	93.47	93.85	96.74	93.25
100	89.90	94.40	94.70	97.60	94.15
200	89.82	94.31	94.63	97.54	94.08
500	89.79	94.28	94.58	97.51	94.04

Table 14: Sensitivity to trajectory-cache capacity. Results are accuracy (%).

Performance improves steadily up to $L = 100$ and then saturates. A moderate cache therefore captures sufficient reusable experience, while larger caches provide no material gain and may introduce less relevant trajectories.

P	MMLU	HumanEval	SVAMP	GSM8K	Avg.	Time (ms)
10	86.53	90.74	91.18	94.46	90.73	12
25	88.19	92.83	93.41	96.23	92.67	24
50	89.24	93.79	94.05	96.98	93.52	38
100	89.85	94.36	94.72	97.60	94.13	65
200	89.91	94.38	94.74	97.62	94.16	125
500	89.94	94.42	94.79	97.65	94.20	305

Table 15: Accuracy–latency trade-off for the number of Flow-Matching integration steps P .

Accuracy largely saturates at $P = 100$: increasing P from 100 to 500 improves the average by only 0.07 points while increasing topology-generation time from 65 to 305 ms. We therefore use $P = 100$ as the default accuracy–efficiency trade-off.

C.5 Topological Oscillation and Smoothing

We measure the similarity of adjacent communication topologies using the Jaccard similarity between their edge sets,

$$J(\mathcal{E}_{t-1}, \mathcal{E}_t) = \frac{|\mathcal{E}_{t-1} \cap \mathcal{E}_t|}{|\mathcal{E}_{t-1} \cup \mathcal{E}_t|}. \quad (26)$$

A higher value denotes a smoother transition. To explicitly discourage abrupt changes, we optionally add an adjacency regularizer

$$\mathcal{L}_{\text{smooth}} = \lambda \|A_t - A_{t-1}\|_1. \quad (27)$$

Method	Jaccard	Std.	Oscillation Rate
Random Scheduler	0.31	0.22	78.4%
G-Designer (fixed)	1.00	0.00	0.0%
ST-EVO w/o smoothing	0.72	0.15	22.3%
ST-EVO w/ smoothing	0.81	0.09	12.8%

Table 16: Similarity and oscillation of communication topologies between adjacent dialogue iterations.

Variant	MMLU	HumanEval	SVAMP	GSM8K	Jaccard
w/o smoothing	89.85	94.36	94.72	97.60	0.72
$\lambda = 0.01$	89.78	94.29	94.64	97.53	0.78
$\lambda = 0.05$	89.71	94.18	94.51	97.42	0.81
$\lambda = 0.10$	89.32	93.81	94.03	97.06	0.88
$\lambda = 0.50$	87.46	91.53	92.18	95.27	0.95

Table 17: Effect of topology-smoothing strength on accuracy (%) and adjacent-topology similarity.

Even without an explicit penalty, Flow Matching yields relatively gradual transitions ($J = 0.72$). Small smoothing weights further reduce oscillation with less than 0.2 points of accuracy loss, whereas excessive smoothing prevents necessary topology adaptation. We use $\lambda = 0.01$ when smoother deployment behavior is preferred; otherwise, no explicit smoothing is required.

C.6 Generalization Study via Cross-Dataset Transfer

We further conduct a transfer study to examine the generalization capability of ST-EVO. In this experiment, $A \rightarrow B$ denotes that the scheduler is few-shot trained on the source dataset A and then directly evaluated on the target dataset B . We hypothesize that both the scheduling trajectory memory and the learned scheduler parameters are transferable across tasks with similar reasoning structures. Therefore, we first consider two in-domain transfer settings: GSM8K $\rightarrow$ SVAMP for mathematical reasoning, and HumanEval $\rightarrow$ DS-1000 for code reasoning. We also evaluate two cross-domain transfer settings, including GSM8K $\rightarrow$ HumanEval and HumanEval $\rightarrow$ GSM8K, to study whether the learned scheduling ability can generalize across substantially different task domains.

As shown in Table 18, ST-EVO exhibits strong transferability when the source and target datasets belong to the same task domain. For GSM8K $\rightarrow$ SVAMP, the transferred scheduler achieves 93.85% accuracy, only 0.87% lower than ST-EVO trained on the target dataset. Similarly, for HumanEval $\rightarrow$ DS-1000, the transferred scheduler obtains 57.42% accuracy, with a moderate drop of 1.23%. These results suggest that the scheduling trajectory mem-

Transfer Setting	Target Domain	Original on Target	Transfer Acc.	Drop
GSM8K → SVAMP	Math → Math	94.72	93.85	0.87
HumanEval → DS-1000	Code → Code	58.65	57.42	1.23
GSM8K → HumanEval	Math → Code	94.36	88.54	5.82
HumanEval → GSM8K	Code → Math	97.60	90.71	6.89

Table 18: Generalization study via cross-dataset transfer. $A \rightarrow B$ denotes that the scheduler is few-shot trained on dataset A and directly evaluated on dataset B . “Original on Target” denotes the performance of **ST-EVO** trained directly on the target dataset, while “Transfer Acc.” denotes the performance under the transfer setting.

ory and learned scheduler parameters can capture reusable coordination patterns within the same type of reasoning task.

Cross-domain transfer is more challenging. When transferring from GSM8K to HumanEval, the accuracy decreases from 94.36% to 88.54%, and transferring from HumanEval to GSM8K decreases the accuracy from 97.60% to 90.71%. This indicates that mathematical reasoning and code reasoning require different communication patterns among agents. Nevertheless, the transferred models still achieve competitive performance and remain stronger than static multi-agent systems, suggesting that **ST-EVO** learns non-trivial scheduling behaviors that can partially generalize beyond the source domain.

C.7 Zero-Shot Cross-Task Transfer

The preceding experiment allows limited target-task adaptation. We further evaluate a stricter zero-shot setting in which a scheduler trained on source task A is directly applied to target task B without any target-task training query.

Table 19 quantifies a clear domain gap. In-domain zero-shot transfer incurs a 2.32–5.58 point drop, whereas cross-domain transfer incurs a 9.59–13.43 point drop. Together with Table 18, these results show that the scheduler learns reusable behaviors within a task family, while a small amount of target-domain adaptation substantially closes the gap for structurally different tasks.

D Details of Baselines

For fair comparison, all baselines use gpt-oss-120b, the same prompt template as **ST-EVO**, and decoding temperature 0.1. For method-specific hyperparameters, we follow the original papers whenever available and tune validation-dependent thresholds under the same training budget of 40–80 queries.

D.1 Positioning Relative to Agent Harnesses

Agent harnesses address a broader engineering problem than communication-topology scheduling: they may provide tool integration, memory management, sandboxing, sub-agent creation, and deployment infrastructure. Representative commercial and research systems—including Claude Code, Codex, OpenHands, and MASAI—primarily expose execution or workflow abstractions. By contrast, **ST-EVO** is a learned scheduling module whose output is the communication graph used at each dialogue iteration. It does not replace a harness and is agnostic to its tool and sandbox implementations; instead, it can be integrated into a harness to replace a fixed or manually specified inter-agent communication pattern. This distinction also explains why harness-level systems are not treated as directly comparable topology-learning baselines in the main experiments.

E Case Study

We showcase some scheduling trajectories in this section—see Figure 6–10.

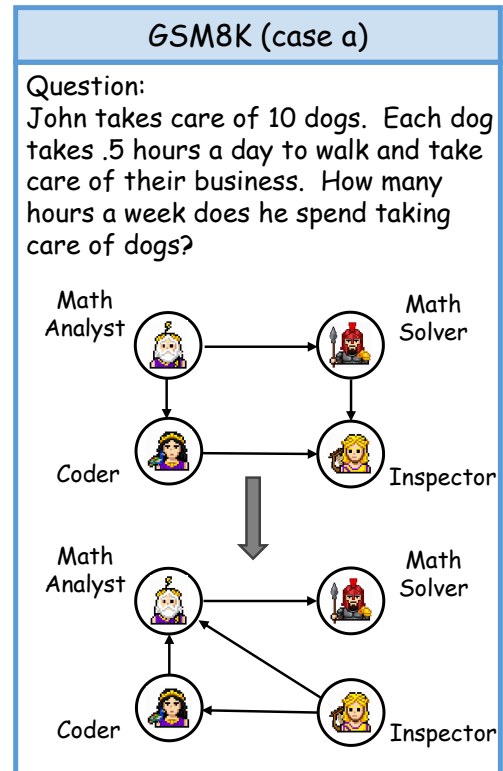


Figure 6: Case study on GSM8K.

Transfer Setting	Domain	Target-Trained	Zero-Shot	Drop
GSM8K → SVAMP	Math → Math	94.72	91.38	3.34
GSM8K → MultiArith	Math → Math	99.05	96.73	2.32
HumanEval → DS-1000	Code → Code	58.65	53.07	5.58
GSM8K → HumanEval	Math → Code	94.36	81.92	12.44
HumanEval → GSM8K	Code → Math	97.60	84.17	13.43
GSM8K → MMLU	Math → General	89.85	80.26	9.59
HumanEval → HotpotQA	Code → Web	92.52	79.63	12.89

Table 19: Zero-shot cross-task transfer. Target-Trained denotes a scheduler trained on the target dataset; Zero-Shot directly transfers the source scheduler without target-task queries. All values are accuracy (%).

Hyperparameter	Value
Backbone LLM	gpt-oss-120b
Prompt template	Same as ST-EVO
Decoding temperature	0.1
Optimization target	Graph edges and prompts
Graph optimizer	REINFORCE
Initial edge probability	0.10
Learning rate	0.40
Sampled graphs per iteration	20
Training budget	40–80 queries
Stopping criterion	Budget exhausted

Table 20: Key hyperparameter settings of GPTSwarm.

Hyperparameter	Value
Backbone LLM	gpt-oss-120b
Prompt template	Same as ST-EVO
Decoding temperature	0.1
Routing granularity	Reasoning step
Draft model	gpt-oss-120b
Verifier model	gpt-oss-120b
Verification strategy	LLM-as-judge
Utility score range	0 to 9
Acceptance threshold candidates	3, 5, 7, 9
Acceptance threshold selection	Validation-selected
Fallback strategy	Regenerate rejected step
Training budget	40–80 queries

Table 23: Key hyperparameter settings of SpecReason.

Hyperparameter	Value
Backbone LLM	gpt-oss-120b
Prompt template	Same as ST-EVO
Decoding temperature	0.1
Communication rounds	3
Node encoder	all-MiniLM-L6-v2
Node embedding dimension	384
Topology sampling times	10
Anchor regularization weight	0.01
Sparsity regularization weight	0.10
Training budget	40–80 queries

Table 21: Key hyperparameter settings of G-Designer.

Hyperparameter	Value
Backbone LLM	gpt-oss-120b
Prompt template	Same as ST-EVO
Decoding temperature	0.1
Routing granularity	Reasoning step
Routing signal	Logit-based confidence
Confidence estimator	Maximum logit confidence
Calibration model	Two-component Gaussian mixture model
External router	No
Routing threshold search interval	0.10
Routing threshold selection	Validation-selected
Fallback strategy	Route to high-confidence branch
Training budget	40–80 queries

Table 24: Key hyperparameter settings of STEER.

Hyperparameter	Value
Backbone LLM	gpt-oss-120b
Prompt template	Same as ST-EVO
Decoding temperature	0.1
Search algorithm	Monte Carlo Tree Search
Workflow representation	Code-represented workflow
Optimized components	Workflow edges and node prompts
Workflow operator set	Original AFlow operators
Maximum optimization rounds	20
Training budget	40–80 queries
Early stopping criterion	Validation score not improved

Table 22: Key hyperparameter settings of AFlow.

Hyperparameter	Value
Backbone LLM	gpt-oss-120b
Prompt template	Same as ST-EVO
Decoding temperature	0.1
Initial topology	Fully connected graph
Pruning granularity	Communication edge
Initial edge scoring	Semantic compatibility
Adaptive edge scoring	Historical contribution
Clustering strategy	0-extension clustering
Pruning schedule	Progressive pruning
Greedy top-k pruning	No
Training budget	40–80 queries
Final topology selection	Validation-selected

Table 25: Key hyperparameter settings of SafeSieve.

Hyperparameter	Value
Backbone LLM	gpt-oss-120b
Prompt template	Same as ST-EVO
Decoding temperature	0.1
Topology generation paradigm	Autoregressive graph generation
Graph initialization	Empty graph
Node generation strategy	Role selection from agent pool
Edge generation strategy	Sequential edge prediction
Node encoder	a11-MiniLM-L6-v2
Node embedding dimension	384
Communication rounds	3
Training strategy	Two-stage curriculum learning
Training objective	Negative log-likelihood
Node-edge loss weight	0.20
Training budget	40–80 queries
Final aggregation	Summarizer agent

Table 26: Key hyperparameter settings of ARG-Designer.

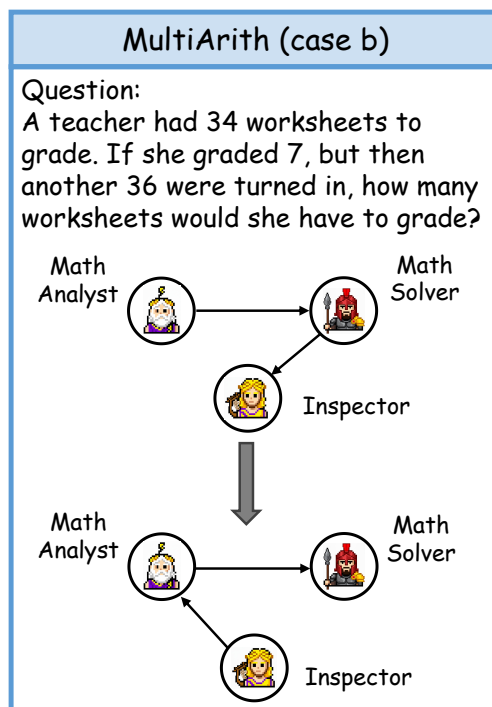


Figure 7: Case study on MultiArith.

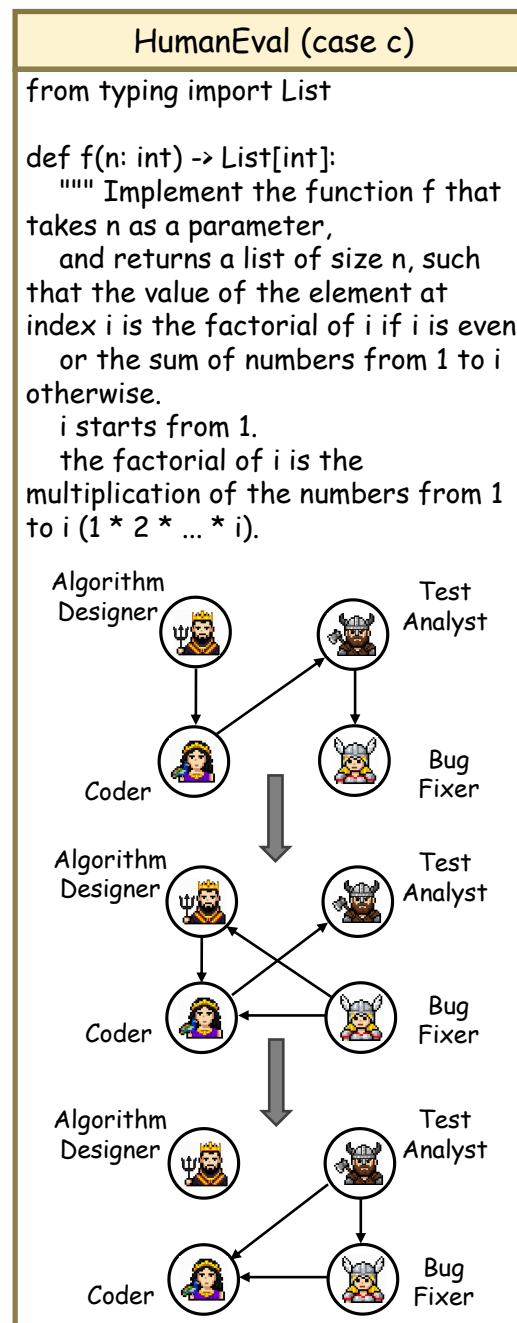


Figure 8: Case study on HumanEval.

DS-1000 (case d)

```
I have following pandas
dataframe :\n\n\nimport pandas as
pd\nfrom pandas import Series,
DataFrame\nndata =
DataFrame({'Qu1': ['apple', 'potato',
'cheese', 'banana', 'cheese', 'banana',
'cheese', 'potato', 'egg'],\n
'Qu2': ['sausage', 'banana', 'apple',
'apple', 'apple', 'sausage', 'banana',
'banana', 'banana'],\n
'Qu3':
['apple', 'potato', 'sausage', 'cheese',
'cheese', 'potato', 'cheese', 'potato',
'egg']})\n\n\nI'd like to change
values in columns Qu1 according to
value_counts() when value count great
or equal 3 and change values in
columns Qu2 and Qu3 according to
value_counts() when value count great
or equal 2.
```

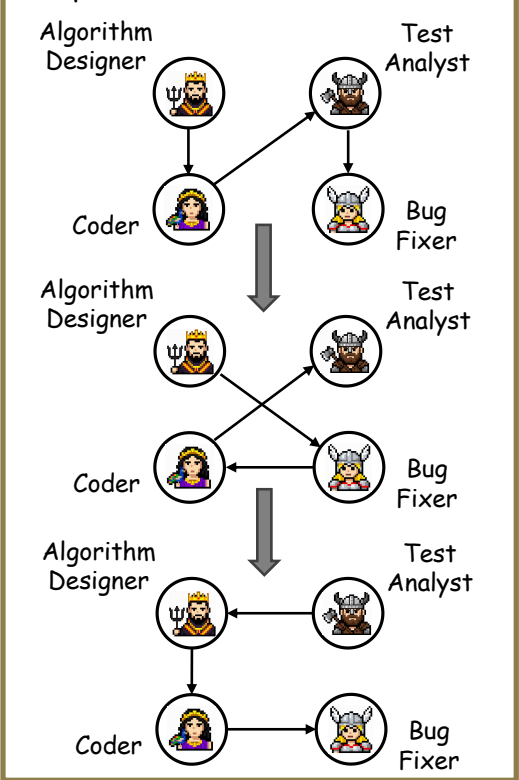


Figure 9: Case study on DS-1000.

MMLU (case e)

Which of the following is not one of nor follows directly from Kepler's laws?

A. As a planet moves around its orbit it sweeps out equal areas in equal times.

B. The orbit of each planet about the Sun is an ellipse with the Sun at one focus.

C. The force of attraction between any two objects decreases with the square of the distance between their centers.

D. A planet travels faster when it is nearer to the Sun and slower when it is farther from the Sun.

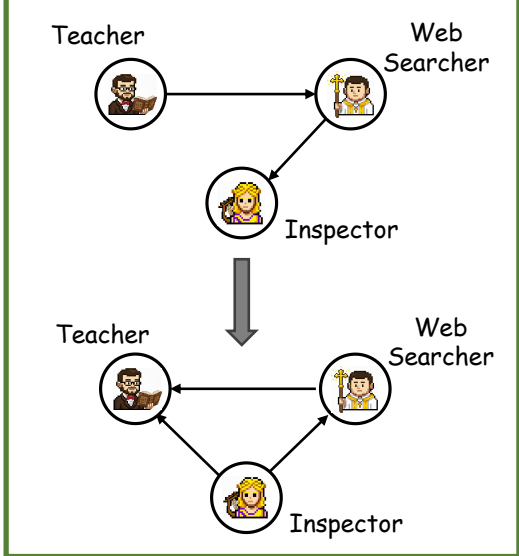


Figure 10: Case study on MMLU.

F More Robustness Analyses

We further evaluate the robustness of **ST-EVO** under different fractions of compromised agents. We consistently use five agents for all multi-agent systems and conduct experiments on MMLU, SVAMP, and HumanEval, which respectively cover question answering, mathematical reasoning, and code generation. For each system, Attack ($k/5$) denotes that k out of five agents are injected with an adversarial system prompt that instructs them to produce plausible but incorrect answers. We compare **ST-EVO** with Random Graph, a static multi-agent system with randomly constructed communication topology, and SafeSieve, a stronger scheduling-based baseline.

As shown in Table 27, **ST-EVO** consistently suffers the smallest performance degradation under all attack ratios. When only one agent is compromised, **ST-EVO** only drops by 0.60% on average, while Random Graph and SafeSieve drop by 6.83% and 4.84%, respectively. As the attack ratio increases to Attack (2/5), the average drop of **ST-EVO** remains limited to 2.12%, which is substantially smaller than the drops of Random Graph (10.82%) and SafeSieve (7.79%). Even under the strongest Attack (3/5) setting, where more than half of the agents are compromised, **ST-EVO** still preserves an average accuracy of 87.34%, with a drop of only 4.69%. In contrast, Random Graph and SafeSieve suffer much larger average drops of 15.75% and 11.63%, respectively.

These results demonstrate that **ST-EVO** is more robust to adversarial agents than both static and scheduling-based baselines. The static topology in Random Graph makes the system vulnerable to corrupted information propagation, since compromised agents can continuously influence the same communication paths. SafeSieve improves robustness by introducing scheduling mechanisms, but still suffers notable degradation as the number of compromised agents increases. By contrast, **ST-EVO** dynamically adjusts the communication topology across reasoning steps, which helps reduce the influence of unreliable agents and preserve useful information flow among normal agents. This confirms that the learned spatio-temporal scheduling mechanism is effective not only for improving accuracy under clean settings, but also for maintaining stable reasoning performance under adversarial perturbations.

G More Scalability Analyses

We provide more scalability analyses from three complementary perspectives: the number of agents, the number of reasoning iterations, and the number of training queries. The corresponding results are reported in Tables 28–30, Tables 31–33, and Table 34, respectively.

Scalability with respect to the number of agents.

Tables 28–30 compare different methods under increasing numbers of agents. Across MMLU, HumanEval, and SVAMP, **ST-EVO** consistently achieves the best accuracy under all agent scales. More importantly, **ST-EVO** exhibits a favorable accuracy–efficiency trade-off when the number of agents increases. For example, on MMLU, increasing the agent number from 5 to 40 improves the accuracy of **ST-EVO** from 89.2% to 94.1%, while the time cost remains lower than most optimization-based baselines at large scales. Similarly, on SVAMP, **ST-EVO** achieves the highest accuracy with the lowest time and token consumption across all agent numbers. These results indicate that the scheduler can effectively exploit additional agents while avoiding excessive communication overhead.

Scalability with respect to the number of reasoning iterations.

Tables 31–33 further evaluate the scalability of different methods under varying numbers of reasoning iterations. Compared with baselines, **ST-EVO** maintains consistently higher accuracy as the number of iterations increases. Meanwhile, the additional cost introduced by more iterations grows moderately, showing that **ST-EVO** does not rely on exhaustive multi-round communication. This is particularly important for multi-agent reasoning, since naively increasing the number of iterations can quickly amplify both latency and token consumption. The results suggest that **ST-EVO** can obtain stable gains from additional reasoning steps while preserving computational efficiency.

Scalability with respect to the number of training queries.

Table 34 studies how the performance of **ST-EVO** changes with different numbers of training queries. The results show a clear and consistent improvement as the number of training queries increases from 10 to 80. With only 10 training queries, **ST-EVO** already outperforms strong baselines such as LLM-Debate and G-Designer on average. As more training queries are used, the performance further improves across diverse

Method	Attack Ratio	MMLU	SVAMP	HumanEval	Avg.
Random Graph	Clean (0/5)	83.20	87.10	84.14	84.81
	Attack (1/5)	76.48 (↓6.72)	79.72 (↓7.38)	77.73 (↓6.41)	77.98 (↓6.83)
	Attack (2/5)	72.36 (↓10.84)	75.64 (↓11.46)	73.97 (↓10.17)	73.99 (↓10.82)
	Attack (3/5)	67.83 (↓15.37)	70.82 (↓16.28)	68.52 (↓15.62)	69.06 (↓15.75)
SafeSieve	Clean (0/5)	84.65	90.10	90.15	88.30
	Attack (1/5)	80.93 (↓3.72)	85.15 (↓4.95)	84.31 (↓5.84)	83.46 (↓4.84)
	Attack (2/5)	78.47 (↓6.18)	82.03 (↓8.07)	81.03 (↓9.12)	80.51 (↓7.79)
	Attack (3/5)	75.01 (↓9.64)	78.58 (↓11.52)	76.42 (↓13.73)	76.67 (↓11.63)
ST-EVO	Clean (0/5)	89.20	93.70	93.20	92.03
	Attack (1/5)	88.57 (↓0.63)	93.49 (↓0.21)	92.23 (↓0.97)	91.43 (↓0.60)
	Attack (2/5)	87.34 (↓1.86)	92.07 (↓1.63)	90.31 (↓2.89)	89.91 (↓2.12)
	Attack (3/5)	85.08 (↓4.12)	89.96 (↓3.74)	86.99 (↓6.21)	87.34 (↓4.69)

Table 27: Robustness analyses under different fractions of compromised agents. We consistently use five agents for all multi-agent systems. We report the accuracy (%) on MMLU, SVAMP, and HumanEval, as well as the average performance. Attack ($k/5$) denotes that k out of five agents are injected with an adversarial system prompt. The values in parentheses indicate the absolute performance drop compared with the corresponding clean setting.

benchmarks, including general question answering, mathematical reasoning, coding, web reasoning, and medical reasoning. This indicates that the scheduler can effectively benefit from additional few-shot training trajectories and gradually learn more generalizable communication patterns.

Overall, these scalability analyses demonstrate that ST-EVO scales well along three important axes: agent size, reasoning depth, and training data size. It consistently improves accuracy while maintaining relatively low time and token costs, suggesting that the proposed scheduler provides an effective and efficient mechanism for coordinating multi-agent reasoning systems at larger scales.

H Deployment Cost Analysis

We report the standalone cost of the scheduler to complement the end-to-end time and token measurements in the main experiments. The scheduler contains 2.8M parameters, comprising a 1.2M-parameter graph encoder and a 1.6M-parameter Flow-Matching network.

Table 35 shows that entropy computation and Flow-Matching scheduling account for only 0.5% and 0.7% of an iteration, respectively, compared with 93.9% for LLM inference. Entropy computation takes 45 ms in total: 5 ms to extract log-probabilities, 15 ms to compute token entropy, 12 ms to aggregate PE, 8 ms to compute VE, and 5 ms for state classification. Its cost scales approximately linearly with the number of agents.

Although modeling the temporal dimension

makes ST-EVO moderately more expensive to train than static topology optimizers, the scheduler remains lightweight relative to LLM inference: its cold start is below one second, steady-state topology generation takes 65 ms, and inference requires 0.3 GB of GPU memory.

I Reproducibility Analysis.

We set a relatively low decoding temperature of 0.1 in our experiments, which leads to high overall reproducibility. To further verify this, we conduct experiments on all nine benchmarks, running each trial three times and reporting the mean and standard deviation in Table 38. It can be seen that ST-EVO maintains high performance stability across different types of tasks, with standard deviations consistently below 0.5% on all benchmarks.

Method	Metric	5 Agents	10 Agents	20 Agents	40 Agents
Random Graph	Accuracy (%)	82.7	83.3	83.6	83.8
	Time (minutes)	14.2	31.4	59.7	118.5
	#Tokens	431,064	1,317,148	4,245,925	13,184,762
GPTSwarm	Accuracy (%)	83.6	84.2	84.8	85.1
	Time (minutes)	67.8	201.8	422.5	825.0
	#Tokens	1,672,201	5,103,328	14,054,825	39,487,316
G-Designer	Accuracy (%)	84.1	85.2	87.3	88.1
	Time (minutes)	22.4	41.8	75.23	138.6
	#Tokens	1,422,309	2,887,532	6,130,221	12,873,904
AFlow	Accuracy (%)	83.0	84.5	85.1	85.5
	Time (minutes)	36.7	77.2	135.4	248.0
	#Tokens	1,851,925	3,485,925	7,292,296	15,176,842
STEER	Accuracy (%)	84.0	86.2	87.8	88.7
	Time (minutes)	48.2	105.6	233.2	492.0
	#Tokens	2,025,125	4,256,252	9,026,290	19,367,814
ST-EVO	Accuracy (%)	89.2	91.6	93.3	94.1
	Time (minutes)	15.2	28.4	51.6	94.8
	#Tokens	1,162,296	1,825,926	3,326,037	5,942,183

Table 28: Scalability comparison on MMLU under different numbers of agents.

Method	Metric	5 Agents	10 Agents	20 Agents	40 Agents
Random Graph	Accuracy (%)	83.6	84.1	84.6	84.9
	Time (minutes)	3.8	7.5	15.2	31.4
	#Tokens	158,426	316,917	641,382	1,296,745
GPTSwarm	Accuracy (%)	86.5	87.1	87.8	88.2
	Time (minutes)	8.9	17.6	35.9	73.8
	#Tokens	189,742	381,653	776,438	1,592,817
G-Designer	Accuracy (%)	88.3	89.4	90.2	90.7
	Time (minutes)	5.1	10.4	21.5	44.7
	#Tokens	176,385	352,914	719,682	1,463,259
AFlow	Accuracy (%)	88.4	89.3	89.9	90.3
	Time (minutes)	6.4	13.1	27.2	56.9
	#Tokens	154,628	309,847	631,295	1,286,374
STEER	Accuracy (%)	88.1	89.0	89.8	90.4
	Time (minutes)	6.8	13.9	29.6	62.7
	#Tokens	145,736	291,284	598,417	1,239,506
ST-EVO	Accuracy (%)	93.2	94.4	95.0	95.4
	Time (minutes)	4.2	8.3	16.9	34.8
	#Tokens	138,945	281,376	574,829	1,176,942

Table 29: Scalability comparison on HumanEval under different numbers of agents.

Method	Metric	5 Agents	10 Agents	20 Agents	40 Agents
Random Graph	Accuracy (%)	86.6	87.1	87.5	87.8
	Time (minutes)	10.6	21.4	43.8	89.7
	#Tokens	648,372	1,294,816	2,631,457	5,382,904
GPTSwarm	Accuracy (%)	86.7	87.3	87.8	88.1
	Time (minutes)	25.8	52.6	108.4	223.7
	#Tokens	1,006,481	2,018,736	4,126,592	8,482,317
G-Designer	Accuracy (%)	88.5	89.5	90.2	90.7
	Time (minutes)	12.2	24.9	51.3	106.8
	#Tokens	552,946	1,108,375	2,264,831	4,682,519
AFlow	Accuracy (%)	85.9	86.5	87.0	87.4
	Time (minutes)	21.7	44.3	91.5	189.6
	#Tokens	904,218	1,815,604	3,714,925	7,658,341
STEER	Accuracy (%)	86.8	87.4	88.1	88.6
	Time (minutes)	16.3	33.5	70.1	146.8
	#Tokens	601,739	1,207,482	2,493,816	5,182,647
ST-EVO	Accuracy (%)	93.7	94.7	95.3	95.8
	Time (minutes)	9.3	18.8	38.7	80.4
	#Tokens	436,215	872,946	1,785,324	3,692,781

Table 30: Scalability comparison on SVAMP under different numbers of agents.

Method	Metric	3 Iterations	6 Iterations	12 Iterations
Random Graph	Accuracy (%)	82.7	83.1	83.3
	Time (minutes)	14.2	27.4	52.5
	#Tokens	431,064	816,266	1,425,295
GPTSwarm	Accuracy (%)	83.6	84.0	84.2
	Time (minutes)	67.8	112.5	199.4
	#Tokens	1,672,201	3,025,162	5,925,102
G-Designer	Accuracy (%)	84.1	83.8	84.3
	Time (minutes)	22.4	42.5	79.2
	#Tokens	1,422,309	3,015,285	5,723,293
AFlow	Accuracy (%)	83.0	83.4	83.6
	Time (minutes)	36.7	67.2	122.4
	#Tokens	1,851,925	3,425,291	6,902,946
STEER	Accuracy (%)	84.0	84.3	84.5
	Time (minutes)	48.2	87.2	144.5
	#Tokens	2,025,125	3,821,229	7,402,409
ST-EVO	Accuracy (%)	89.2	89.4	89.6
	Time (minutes)	15.2	26.4	44.2
	#Tokens	1,162,296	1,823,394	2,901,034

Table 31: Scalability comparison on MMLU under different numbers of iterations.

Method	Metric	3 Iterations	6 Iterations	12 Iterations
Random Graph	Accuracy (%)	84.1	84.5	84.8
	Time (minutes)	5.6	10.8	20.9
	#Tokens	246,382	467,915	834,276
GPTSwarm	Accuracy (%)	87.1	87.5	87.8
	Time (minutes)	18.4	32.7	60.8
	#Tokens	381,624	704,892	1,326,517
G-Designer	Accuracy (%)	89.4	89.7	90.1
	Time (minutes)	10.6	19.8	37.4
	#Tokens	352,768	671,435	1,238,692
AFlow	Accuracy (%)	89.3	89.6	89.9
	Time (minutes)	13.2	24.5	45.7
	#Tokens	309,846	587,293	1,087,641
STEER	Accuracy (%)	89.0	89.4	89.8
	Time (minutes)	14.7	26.9	49.6
	#Tokens	291,573	548,926	1,015,384
ST-EVO	Accuracy (%)	94.4	94.7	95.0
	Time (minutes)	8.9	15.6	28.3
	#Tokens	281,394	492,618	857,936

Table 32: Scalability comparison on HumanEval under different numbers of iterations.

Method	Metric	3 Iterations	6 Iterations	12 Iterations
Random Graph	Accuracy (%)	86.6	87.0	87.2
	Time (minutes)	10.6	20.5	39.2
	#Tokens	648,372	1,227,762	2,143,815
GPTSwarm	Accuracy (%)	86.7	87.1	87.3
	Time (minutes)	25.8	42.8	75.9
	#Tokens	1,006,481	1,820,815	3,566,259
G-Designer	Accuracy (%)	88.5	88.2	88.7
	Time (minutes)	12.2	23.1	43.1
	#Tokens	552,946	1,172,242	2,225,024
AFlow	Accuracy (%)	85.9	86.3	86.5
	Time (minutes)	21.7	39.7	72.4
	#Tokens	904,218	1,672,427	3,370,422
STEER	Accuracy (%)	86.8	87.1	87.3
	Time (minutes)	16.3	29.5	48.9
	#Tokens	601,739	1,135,427	2,199,528
ST-EVO	Accuracy (%)	93.7	93.9	94.1
	Time (minutes)	9.3	16.2	27.0
	#Tokens	436,215	684,328	1,088,771

Table 33: Scalability comparison on SVAMP under different numbers of iterations.

Methods	MMLU	GSM8K	MultiArith	SVAMP	AQuA	HumanEval	DS-1000	HotpotQA	DDXPlus	Avg.
LLM-Debate	83.28	89.95	95.45	88.30	73.52	85.24	49.26	84.55	71.33	80.10
G-Designer	84.63	93.35	97.60	89.52	76.30	89.40	54.82	87.66	76.42	83.30
ST-EVO (10 Queries)	85.35	93.85	97.45	89.85	77.60	89.70	55.05	88.10	76.95	83.77
ST-EVO (20 Queries)	87.10	95.20	98.20	91.35	81.10	91.55	56.35	89.75	79.15	85.53
ST-EVO (40 Queries)	89.85	96.85	99.05	93.70	84.90	93.30	57.65	91.55	81.40	87.58
ST-EVO (60 Queries)	90.10	97.60	99.12	94.72	86.10	94.10	58.30	92.20	82.25	88.28
ST-EVO (80 Queries)	90.25	97.75	99.18	94.90	86.56	94.36	58.65	92.52	82.50	88.52

Table 34: Performance comparison under different numbers of training queries.

Component	Time (ms)	Percentage
LLM inference (4 agents)	8,500	93.9%
Entropy computation	45	0.5%
Flow-Matching scheduling	65	0.7%
Cache retrieval	12	0.1%
Topology parsing and routing	8	0.1%
Inter-agent message passing	420	4.6%
Total	9,050	100%

Table 35: Inference-latency breakdown per dialogue iteration with four agents.

Metric	Value
Scheduler parameters	2.8M
Training hardware	1 × NVIDIA 3090 (24 GB)
Training GPU memory	4.2 GB
Training time (40 queries)	~35 min
Training time (80 queries)	~68 min
LLM calls for training (80 queries)	~2,400
Model loading time	0.8 s
First-query scheduling latency	72 ms
Steady-state scheduling latency	65 ms
Inference GPU memory	0.3 GB

Table 36: Training and inference cost of the **ST-EVO** scheduler. Scheduling latency is measured with $P = 100$ integration steps.

Method	Training Time	GPU Memory	Parameters
G-Designer	~45 min	3.8 GB	2.1M
ARG-Designer	~55 min	4.5 GB	3.2M
SafeSieve	~50 min	3.5 GB	1.8M
ST-EVO	~68 min	4.2 GB	2.8M

Table 37: Scheduler training-cost comparison under an 80-query budget.

	MMLU	GSM8K	MultiArith	SVAMP	AQuA
ST-EVO	89.82±0.18	97.55±0.21	99.01±0.12	94.68±0.24	86.48±0.36
	HumanEval	DS-1000	HotpotQA	DDXPlus	
ST-EVO	94.29±0.31	58.72±0.42	92.49±0.27	82.43±0.39	

Table 38: Mean and standard deviation of **ST-EVO** across three independent runs.