ExComm: Exploration-Stage Communication for Error-Resilient Agentic Test-Time Scaling

Woomin Song[1] **Beomjun Kim**[1] **Daewon Choi**[1] **Sai Muralidhar Jayanthi**[2]
Saket Dingliwal[3†] **Jinwoo Shin**[1] **Aram Galstyan**[2]
[1]KAIST [2]Amazon AGI [3]Together AI

Abstract

A common failure mode in long-horizon agentic test-time scaling is error propagation, where factual errors or invalid deductions introduced at intermediate steps persist in the agent's belief state and contaminate later reasoning. Existing test-time scaling methods provide limited control over this process, as they often rely on agents to detect their own mistakes, select among flawed trajectories, or refine solutions only after errors have already shaped the reasoning path. We propose ExComm, a communication protocol for exploration-stage agentic test-time scaling. ExComm is motivated by the empirical observation that the majority of intermediate errors in parallel agentic reasoning produce cross-agent factual conflicts. Leveraging the iterative structure of agentic workflows, ExComm periodically audits agent belief states to detect such conflicts, resolves them through a dedicated tool-based verification loop, and returns concise, targeted feedback to the involved agents. Corrections are incorporated through soft belief updates, which append verified feedback rather than overwriting existing beliefs. Furthermore, to prevent collapsing trajectory diversity due to communication, ExComm further introduces a trajectory diversification module that redirects redundant trajectories toward orthogonal strategies. Experiments on AIME 2024, AIME 2025, and GAIA with Gemini-2.5-Flash-Lite and Qwen3.5-4B show that ExComm consistently outperforms strong test-time scaling baselines, achieving average performance gains of 5.7% and 5.2% over the best-performing baselines, respectively. Further analyses demonstrate improved error recovery, favorable scaling behavior, stronger diversity than adapted communication baselines, and the best performance-cost trade-off among the evaluated methods.

1 Introduction

Test-time scaling has emerged as a powerful approach for improving LLM performance without additional training [1]. By allocating additional computation at inference time, methods such as sequential revision [2, 3], parallel sampling strategies including Best-of-N and Self-Consistency [4, 5], and tree search techniques [6, 7] have achieved substantial gains across reasoning benchmarks. Recently, this paradigm has been extended to agentic workflows [8, 9], where models solve problems through multiple rounds of generation, tool use, and interaction with an environment.

A common failure mode in long-horizon problem solving is error propagation, where factual errors or invalid deductions introduced at intermediate steps persist in the agent's belief and contaminate later decisions. Existing test-time scaling methods provide limited control over this process. Parallel sampling relies heavily on agents detecting their own mistakes, which is often unreliable [10, 11]. Tree search mitigates errors through branch selection rather than explicit correction. Therefore, errors

[†]Work done at Amazon.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

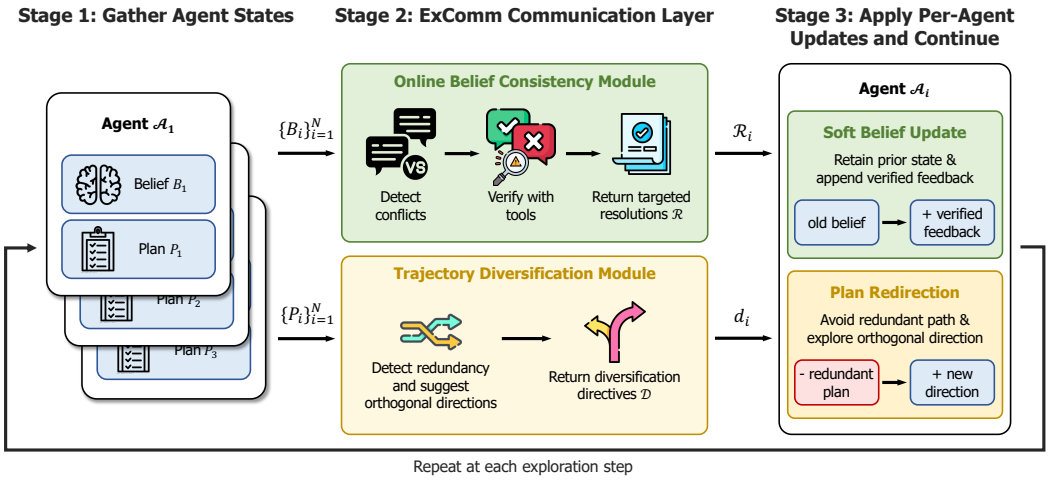


Figure 1: **Overview of ExComm.** ExComm augments a standard agentic test-time scaling loop with an exploration-stage communication step. After each execution step, the Online Belief Consistency Module gathers agent belief states $\{B_i\}_{i=1}^N$, detects factual conflicts, resolves them through tool-augmented verification, and produces a set of targeted resolutions $\mathcal{R}$. Each agent $\mathcal{A}_i$ receives only the relevant subset $\mathcal{R}_i \subseteq \mathcal{R}$, which is applied as a soft belief update by appending verified feedback without overwriting the prior belief state. In parallel, the Trajectory Diversification Module analyzes agent plans $\{P_i\}_{i=1}^N$, identifies redundant strategies, and produces a set of diversification directives $\mathcal{D}$. When a directive is assigned to agent $\mathcal{A}_i$, it receives $d_i \in \mathcal{D}$ and updates its plan to encourage orthogonal exploration. The updated agents then continue the base execution loop at the next exploration step.

that are not pruned can continue to shape downstream reasoning. Post-hoc aggregation or refinement methods intervene only after errors have already shaped the reasoning trajectory.

At the same time, agentic workflows provide a natural point of intervention for error control. Unlike static reasoning, they involve dynamic interaction with an environment, where agents alternate between reasoning and actions such as tool calls while updating intermediate beliefs over multiple rounds. This iterative structure creates a natural opportunity for early error detection and correction during the exploration stage of agentic test-time scaling, where agents are still gathering evidence and forming their reasoning trajectories, before candidate solutions are created. We hypothesize that structured communication among parallel reasoning trajectories at this stage can help detect and resolve belief-level errors before they propagate, ultimately improving final answer quality.

To this end, we propose ExComm, an exploration-stage communication protocol for agentic test-time scaling. ExComm is motivated by the empirical observation that the majority of intermediate errors in parallel agentic reasoning introduce cross-agent factual conflicts (67–71% across benchmarks, see Section 3.2), making them potentially detectable without relying on self-correction. Figure 1 illustrates the core components of ExComm. When a conflict is detected, the Online Belief Consistency Module resolves the discrepancy using its tool-based verification loop. The resulting correction is then returned to the involved agents as targeted feedback. Each agent applies a soft belief update, appending the correction to its belief state rather than overwriting the belief. This allows agents to benefit from verified information while retaining the ability to recover from potential verifier errors.

A key challenge in exploration-stage communication is the risk of diversity collapse. Excessive information sharing can prematurely synchronize reasoning trajectories, undermining the benefits of parallel scaling. To avoid this, the consistency module provides targeted feedback that addresses only the specific inconsistency, rather than a broad summary of other agents' beliefs. ExComm further incorporates a trajectory diversification module, which monitors high-level strategies across agents and encourages orthogonal directions when trajectories become overly similar. By combining targeted belief correction with explicit diversity preservation, ExComm mitigates error propagation while maintaining the benefits of parallel exploration.

We evaluate ExComm on challenging tool-based reasoning setups, including AIME 2024 and AIME 2025 [12] with code interpreter access, and GAIA [13] with code interpreter, file management tools, and web search. Experiments on Gemini-2.5-Flash-Lite [14] and Qwen3.5-4B [15] show that ExComm consistently outperforms strong test-time scaling baselines, including sequential revision, parallel scaling, and test-time tree search. ExComm achieves average performance gains of 5.7% and 5.2% over the best-performing baselines with the two models, respectively. Further analysis shows that ExComm substantially improves error recovery during reasoning, supporting our claim that exploration-stage communication helps agents correct erroneous beliefs before they propagate.

Additional analyses show that ExComm scales with both model size and the number of agents, consistently outperforming baselines across settings. We also compare against adapted communication protocols, including Multi-Agent Debate [16] and Mixture-of-Agents [17], and show that ExComm achieves stronger accuracy while preserving higher trajectory diversity. Ablation studies validate the contributions of both the consistency module and the trajectory diversification module to accuracy and error recovery. Finally, efficiency analysis shows that ExComm achieves the best performance-cost trade-off among the evaluated methods, making it a practical approach for agentic test-time scaling.

2 Related Works

LLM Agents. Large Language Models (LLMs) have evolved from static text generators into agents capable of solving complex, real-world tasks [18–20]. By integrating tools such as calculators, search engines, and APIs, agentic systems can extend standalone models with capabilities such as precise computation and real-time information retrieval [21, 22]. Unlike static reasoning, agentic workflows are often implemented through a ReAct loop [18], alternating between reasoning and tool use. This iterative structure provides a natural intervention point during exploration, before a final answer is produced, which ExComm uses for error recovery.

Test-Time Scaling. Test-time scaling (TTS) improves model performance by allocating additional computation during inference without modifying model parameters [1]. Common approaches include parallel sampling [4, 5], sequential revision [2, 3], and search-based methods [6, 23, 7]. While most work studies TTS in static, text-based settings, its application to agentic workflows remains relatively underexplored [8]. We study an agentic TTS framework focused on preventing error propagation.

One line of TTS research improves reasoning through communication among multiple LLM agents [16, 17]. Representative methods such as Multi-Agent Debate and Mixture-of-Agents iteratively exchange, aggregate, or refine intermediate outputs to improve candidate solutions. ExComm is designed for a different problem: exploration-stage error control. Rather than refining candidate solutions, ExComm detects factual conflicts during execution, resolves them through tool-based verification, and returns concise, targeted feedback before errors propagate.

3 ExComm

In this section, we present the architecture of ExComm. We begin in Section 3.1 by formalizing a base agent loop that serves as the minimal substrate on top of which our framework is built. We then introduce the two core modules of ExComm. In Section 3.2, we describe an online belief consistency module that detects and resolves factual inconsistencies across parallel agents, preventing erroneous beliefs from propagating through subsequent reasoning steps. In Section 3.3, we present a trajectory diversification module that jointly analyzes agents' plans and selectively redirects redundant trajectories toward orthogonal reasoning strategies. Together, these modules allow the agents to detect and recover from errors early-on, preventing the error from propagating and contaminating the later reasoning stages. An overview of the resulting control flow is provided in Figure 1.

3.1 Base Agent Loop

Before introducing our approach, we first formalize the problem setup. Motivated by recent progress in plan-based execution and memory-based agents, we consider an agent that maintains explicit plans and a structured belief. Formally, a stateful problem-solving agent is defined as

$$\mathcal{A} := (B, P, \mathcal{T}),$$

where B, P, and $\mathcal{T}$ denote the agent's belief, plan, and available toolkit, respectively.

The belief B is a compact document containing factual information available to the agent about the problem, including the problem specification itself as well as information acquired during the problem-solving process. The plan P is a to-do list that records both executed steps and remaining planned actions. The toolkit $\mathcal{T}$ represents the set of tools available to the agent during execution.

Algorithm 1 illustrates the agent loop used throughout this work. Given a problem instance p, the agent first initializes its belief state B using the information provided in the problem, and then constructs an initial execution plan based on p and B. The agent subsequently proceeds through a sequence of execution steps, iteratively processing tasks corresponding to individual items in the plan.

At each execution step, the agent selects a task t from the current plan and executes it using a ReAct-style loop with access to the toolkit $\mathcal{T}$. Upon completion of the task, the agent updates its belief B using the resulting execution log ξ_t. The plan P is then optionally revised based on the updated belief and the execution log. We refer to the resulting sequence of execution steps as a *reasoning trajectory*.

Algorithm 1 Base Agent Loop

procedure INITIALIZEAGENT($p, \mathcal{T}$)
  $B \leftarrow$ INITBELIEFSTATE(p)
  $P \leftarrow$ PLANINIT(p, B)
  return $B, P, \mathcal{T}$
end procedure
procedure AGENTSTEP($B, P, \mathcal{T}$)
  /* Deliberation */
  $t \leftarrow$ SELECTTASK(P, B)
  /* Execution Loop (ReAct) */
  $\xi_t \leftarrow$ EXECUTETASK($t, B, \mathcal{T}$)
  /* Belief update */
  $B' \leftarrow$ UPDATEBELIEF(B, ξ_t)
  /* Optional replanning */
  $P' \leftarrow$ REPLAN(P, B', ξ_t)
  return B', P'
end procedure
procedure AGENTLOOP($p, \mathcal{T}$)
  $B, P, \mathcal{T} \leftarrow$ INITIALIZEAGENT($p, \mathcal{T}$)
  while not TERMINATE(P, B) **do**
    $B, P \leftarrow$ AGENTSTEP($B, P, \mathcal{T}$)
  end while
  return B
end procedure

In the following sections, we describe the core components of ExComm in detail. The proposed modules are applied on the parallel problem-solving agents after they finish each execution step.

3.2 Online Belief Consistency Module

Long-horizon agentic reasoning is vulnerable to error propagation, where an incorrect intermediate claim, once internalized, impacts all subsequent reasoning steps. Motivated by the intuition that detecting conflicts across multiple reasoning trajectories is substantially easier than identifying errors within a single trajectory, we introduce the Online Belief Consistency Module.

This module identifies mutually exclusive factual conflicts across the beliefs of parallel agents, employs a tool-based reasoning loop to resolve these conflicts, and applies concise, targeted updates to the agents' beliefs to improve factual correctness. In contrast to post-hoc verification methods that operate on preliminary solutions, our module is integrated directly into the agent loop immediately after each execution step. By acting on the beliefs that govern future planning and execution, the module stabilizes agent reasoning before downstream decisions are made.

Observation. In Table 1, we categorize errors in parallel agentic reasoning trajectories into three types. Specifically, we define errors as conflicting, common, or neutral based on whether other agents hold opposing beliefs, matching incorrect beliefs, or no relevant beliefs at the same reasoning step. Notably, 67.0% to 71.5% of errors are categorized as conflicting, suggesting that most errors are potentially detectable by inspecting factual inconsistencies across multiple agents' beliefs. This observation naturally motivates our design of the consistency module.

Table 1: **Error Type Analysis.** Error types in generation trajectories from four parallel Gemini-2.5-Flash-Lite agents, averaged over three critic models. Details in Section C.

Error Type	AIME	GAIA
Common	5.6%	13.6%
Neutral	22.9%	19.4%
Conflicting (Detectable)	**71.5%**	**67.0%**

Conflict extraction. After parallel problem-solving agents $\{\mathcal{A}_i\}_{i=1}^N$ complete an execution step, the consistency module collects their beliefs $\{B_i\}_{i=1}^N$ and extracts conflicts, defined as sets of mutually exclusive factual claims. Each conflict c is represented as a structured object containing the information described alongside. The module may also flag factual errors from individual agents, similar in spirit to self-reflection. A conflict instance is created for each set of mutually exclusive factual claims, and all conflicts are gathered into a conflict report $\mathcal{C}$.

Conflict resolution via agentic reasoning. Given the conflict report $\mathcal{C}$, the consistency module initiates a dedicated ReAct loop to resolve the identified conflicts. The resolver uses the same model and tools as the solver agents, but focuses solely on resolving factual conflicts. This loop is decoupled from the parallel agents, preventing direct access to one another's belief states and preserving diversity. After resolution, the module produces a set of resolutions $\mathcal{R}$. Each resolution $r \in \mathcal{R}$ is transmitted only to the agents involved in the conflict, as specified in the `agents` field. This targeted design provides corrections only to relevant agents, preventing others from being influenced by unrelated information.

Structure of a Conflict

```
conflict:
  agents: [0, 1, 2]
  description: |
    Brief description of the conflict
  claims:
    agent_0: Claim from agent 0
    agent_1: Claim from agent 1
    agent_2: Claim from agent 2
```

Structure of a Resolution

```
resolution:
  agents: [0, 1, 2]
  description: |
    Brief description of the conflict
  claims:
    agent_0: Claim from agent 0
    agent_1: Claim from agent 1
    agent_2: Claim from agent 2
  correct_claim: |
    Claim verified by module
  justification: |
    How the claim was verified
```

Soft belief update. Once an agent $\mathcal{A}_i$ receives the resolutions associated with its index, it updates its belief B_i accordingly. A straightforward approach is to directly replace conflicting belief entries with the resolved values, which we refer to as a hard update. However, conflict resolution is not guaranteed to be perfect, and incorrect resolutions may occasionally be produced. Blindly overwriting belief entries therefore risks introducing new factual errors that may propagate through subsequent reasoning steps. Such errors are particularly difficult to detect later, since multiple agents may share the same incorrect belief, reducing the likelihood of future conflicts. Moreover, even correct resolutions can reduce trajectory diversity if they repeatedly overwrite agent-specific beliefs with shared information.

To mitigate these risks, we adopt a soft update strategy. Each resolution is appended to B_i rather than replacing the original conflicting information. The resolution is treated as an external suggestion that contradicts the existing belief and may itself be correct or incorrect. This preserves the agent's ability to recover from erroneous resolutions while encouraging it to re-examine the conflicting belief in subsequent reasoning. If the resolution is incorrect, the remaining inconsistency can be more easily detected by the agent later in the trajectory.

3.3 Trajectory Diversification Module

Maintaining diverse trajectories is critical for effective exploration in parallel agentic reasoning. It also supports the Consistency Module, since factual conflicts are easier to detect when agents pursue sufficiently distinct reasoning paths. However, agents initialized with identical prompts may suffer from trajectory collapse, especially when sharing information. To mitigate this, we introduce a Trajectory Diversification Module, which updates agent plans after each execution step to keep parallel trajectories sufficiently diverse.

Batched analysis of plan-level redundancy. Similar to the Consistency Module, the diversification module adopts a centralized design. Given the set of plans $\{P_i\}_{i=1}^{N}$ from parallel agents $\{\mathcal{A}_i\}_{i=1}^{N}$, the module jointly analyzes all plans to identify redundancy and underexplored reasoning directions. Operating over this global plan configuration, the module produces a structured set of diversification directives $\mathcal{D}$. Specifically, it checks whether multiple trajectories are overly similar or whether plausible orthogonal reasoning paths remain unexplored. Each directive specifies how a particular agent's plan should be shifted to encourage exploration, and at most one directive is assigned to each agent at a given step.

Structure of a Diversification Directive

```
directive:
  target_agent_index: |
    Index of the agent to modify plan
  modification_instruction: |
    Instruction specifying how to
        orthogonally shift the plan
```

This batched formulation ensures globally consistent diversification decisions and avoids oscillatory or conflicting local interventions. Once generated, directives are transmitted only to their corresponding

Table 2: **Benchmark Evaluations.** Comparison of test-time scaling methods on AIME 2024, AIME 2025, and GAIA. For AIME, agents have access to a Python code interpreter (C). For GAIA, agents have access to a Python code interpreter (C), file management tools (F), and web search tools (S). Parallel methods use four problem-solving agents. Due to cost constraints, GAIA L1/L2 is evaluated on a 50-sample subset. Scores are averaged over three runs, with the best values highlighted in **bold**. We report the standard errors for the average scores.

	N > 1?	AIME 24	AIME 2025	GAIA L1	GAIA L2	GAIA L3	Average
Available Tools		C	C	C / F / S	C / F / S	C / F / S	
Gemini-2.5-Flash-Lite							
Base Agent	✗	61.4	46.1	35.8	21.5	5.9	34.1$\pm$0.61
Sequential Revision (SR)	✗	60.3	46.9	35.2	21.8	4.9	33.8$\pm$0.62
Independent Scaling	✓	65.3	52.2	44.0	28.5	9.0	39.8$\pm$0.84
Independent Scaling + SR	✓	65.3	54.2	42.7	29.2	5.6	39.4$\pm$0.87
Tree Search	✓	67.2	53.9	41.0	30.0	10.2	40.5$\pm$2.32
ExComm (Ours)	✓	**72.5**	**60.6**	**52.2**	**30.8**	**14.8**	**46.2$\pm$0.78**
Qwen3.5-4B							
Base Agent	✗	76.9	60.0	53.7	38.8	16.4	49.2$\pm$0.26
Sequential Revision (SR)	✗	78.6	60.6	54.0	36.7	21.0	50.2$\pm$0.58
Independent Scaling	✓	84.2	70.6	61.7	46.8	20.1	56.7$\pm$1.34
Independent Scaling + SR	✓	83.3	72.2	60.8	43.7	26.5	57.3$\pm$0.40
Tree Search	✓	78.3	62.8	58.7	45.3	16.7	52.4$\pm$0.71
ExComm (Ours)	✓	**87.8**	**75.8**	**69.5**	**50.7**	**28.7**	**62.5$\pm$1.19**

agents. As with the consistency module, this targeted design prevents agents from inspecting other agents' plans, since they receive only the resulting directives rather than the full plans.

Plan shifting via explicit instructions. Diversification directives are applied independently to the targeted agents. Unlike the soft belief update used in the consistency module, plan shifting is enforced once a directive is issued. Because different directives are assigned to different agents, this intervention does not risk inducing mode collapse. Rather, it is designed to explicitly promote diversity across trajectories. By confining intervention to the plan level while leaving task execution fully autonomous, the diversification module expands coverage of the solution space while preserving the coherence and interpretability of each reasoning trajectory.

4 Experiments

In this section, we evaluate ExComm across diverse reasoning benchmarks and analysis settings. We first describe the common experimental setup and baselines. In Section 4.1, we present main benchmark results comparing our method against standard test-time scaling strategies across multiple backbone models. We then analyze the sources of performance gains through error recovery analysis in Section 4.2. Next, we demonstrate the scalability of our framework in Section 4.3 by scaling model size and the number of agents. In Section 4.4, we compare ExComm with other communication protocols. Section 4.5 provides an ablation study examining the contributions of individual components and their effects on error recovery. Finally, in Section 4.6, we discuss the efficiency of our approach.

Common setup and baselines. Unless otherwise specified, experiments are conducted within the CoSight framework, which we treat as an instantiation of the base agent loop described in Algorithm 1. We compare against the following baselines: Base Agent ($N = 1$), which performs a single pass without test-time scaling; Sequential Revision ($N = 1$), which introduces periodic self-revision; Independent Scaling, where multiple agents run in parallel without communication; Independent Scaling + Sequential Revision, which is a parallel version of Sequential Revision; Tree Search, implemented as beam search over reasoning trajectories; and ExComm (ours), which augments the base loop with the Online Belief Consistency Module and the Trajectory Diversification Module.

Unless otherwise stated, we use $N = 4$ solver agents for all parallel scaling methods. We evaluate all methods on AIME 2024, AIME 2025, and GAIA, reporting majority-vote accuracy.

4.1 Main Results

This experiment evaluates whether ExComm improves performance over commonly used test-time scaling strategies. We compare single solver, sequential self-revision, independent parallel scaling with and without self-revision, and tree search baselines against our method across diverse benchmarks spanning mathematical reasoning (AIME 2024, AIME 2025 [12]) and multi-level agentic reasoning (GAIA Levels 1-3 [13]). For reliability, all scores are averaged over three repeated runs.

As shown in Table 2, our method consistently outperforms the baselines across models and benchmarks, achieving average-score gains of 5.7% and 5.2% over the best-performing baseline. Independent scaling and tree search provide moderate improvements over a single solver. Sequential revision sometimes yields additional gains but remains limited by its reliance on error detection within a single reasoning trajectory. In contrast, our communicative approach achieves the strongest performance by combining the Consistency Module with trajectory diversification, mitigating error propagation and improving search-space diversity.

4.2 Error Recovery Analysis

To better understand why ExComm outperforms independent and sequential baselines, we investigate the mechanisms of error detection and correction during execution by comparing the error recovery rates across different methods.

Experimental Setup. Adopting the step-wise evaluation methodology from ProcessBench [24], we compare agent execution results against ground-truth reference solutions to identify factual or logical errors and determine whether they are later resolved. Our primary metric is the Error Recovery Rate, defined as the fraction of unique errors that are subsequently corrected within the same trajectory (see Section B.2 for the formal definition). To ensure a robust analysis, we employ a critic model to identify factual errors and their recovery steps, adapting the critic model design from ProcessBench. Furthermore, we repeat the evaluation with three different critic models, namely Gemini-3-Flash [25], GPT-5-nano [26], and Gemini-2.5-Flash-Lite [14], and take the average of their results.

Table 3: **Error Recovery Rates.** Comparison of error recovery rates (%) across benchmarks, defined as the percentage of factual or logical errors that are corrected during execution. Scores are averaged over three critic models. Detailed experiment setup is outlined in Section B.1.

	AIME		GAIA			
Method	2024	2025	L1	L2	L3	Avg.
Gemini-2.5-Flash-Lite						
Independent	32.6	20.5	16.6	12.9	9.7	18.5
Independent + SR	28.7	25.2	18.6	12.8	14.4	19.9
Tree Search	22.5	10.8	12.3	10.6	8.4	12.9
ExComm (Ours)	**50.6**	**43.9**	**42.7**	**30.7**	**25.6**	**38.7**
Qwen3.5-4B						
Independent	67.1	59.6	43.3	38.5	30.7	47.8
Independent + SR	69.8	71.2	55.5	46.2	45.7	57.7
Tree Search	66.6	59.3	44.6	48.9	37.5	51.4
ExComm (Ours)	**70.7**	**76.3**	**61.7**	**53.7**	**46.5**	**61.8**

Results. As shown in Table 3, ExComm shows superior error detection and correction compared to baselines. ExComm significantly outperforms all baselines, achieving average recovery rates of 38.7% and 61.8% for the Gemini and Qwen models, respectively, being consistently higher than the best-performing baselines. Our framework achieves this resilience through its sophisticated communicative design, that allows agents to collaboratively identify and correct invalid reasoning steps that would otherwise propagate unchecked in isolated settings.

4.3 Performance with Further Scaling

To evaluate the scalability of ExComm, we extend the evaluation in two dimensions. First, we scale the parameter size of the base LLM to test whether the gains of our framework extend to larger models. To this end, we evaluate our approach on Qwen3-235B-A22B-Instruct [27]. Second, we scale the number of solver agents to test whether the benefits of the communication protocol extend to more agents. Specifically, we double the agent count and evaluate Gemini-2.5-Flash-Lite with 8 parallel solver agents instead of 4. As shown in Table 4, ExComm consistently outperforms the baselines in both scaling regimes, demonstrating the scalability of our approach. This suggests that the effectiveness of our approach is complementary to underlying model capacity rather than specific to a particular model size or number of agents.

Table 4: **Scaling experiments.** We evaluate ExComm under two scaling regimes: (a) model scaling with Qwen3-235B-A22B-Instruct and (b) agent scaling with $N = 8$ parallel Gemini-2.5-Flash-Lite agents. We report majority-voting accuracy across AIME and GAIA, averaged over three runs. The best values are highlighted in **bold**, and runner-up values are underlined.

(a) Model Scaling: Qwen3-235B-A22B-Instruct

	AIME		GAIA			
Method	2024	2025	L1	L2	L3	Avg.
Base Agent	80.6	69.7	53.7	39.7	21.0	52.9
Seq. Revision	80.8	69.2	53.2	39.3	23.8	53.3
Independent	89.2	76.4	60.8	46.5	22.8	59.1
Independent + SR	88.9	74.2	59.3	45.0	**29.9**	59.5
Tree Search	86.1	75.0	57.7	45.7	25.3	58.0
ExComm (Ours)	**89.7**	**83.1**	**62.7**	**50.2**	29.3	**63.0**

(b) Agent Scaling: Gemini-2.5-Flash-Lite ($N = 8$)

	AIME		GAIA			
Method	2024	2025	L1	L2	L3	Avg.
Independent	71.5	60.4	46.7	32.7	12.5	44.7
Independent + SR	69.4	61.5	48.1	32.2	9.6	44.2
Tree Search	67.8	53.9	48.6	30.9	9.3	42.1
ExComm (Ours)	**74.6**	**64.0**	**59.3**	**40.8**	**14.0**	**50.5**

4.4 Comparison Against Other Communication Protocols

Many representative communication protocols, including Multi-Agent Debate [16] and Mixture-of-Agents [17], are primarily designed for *solution-stage refinement*. In this setup, agents exchange, critique, or aggregate preliminary answers and the accompanying reasoning trajectory to improve candidate solutions. In contrast, ExComm targets a different problem: *exploration-stage error control*. Rather than intervening after preliminary answers have already formed, ExComm intervenes during the agent execution loop to detect factual conflicts, resolve them before they propagate, and preserve trajectory diversity for subsequent reasoning.

Table 5: **Communication Baselines.** Comparison of ExComm against Multi-Agent Debate and Mixture-of-Agents, adapted to the exploration-stage agent loop using Gemini-2.5-Flash-Lite. We report majority-voting accuracy and diversity scores, defined as $100 -$ Self-BLEU.

	AIME		GAIA			
Method	2024	2025	L1	L2	L3	Avg.
Majority Voting Accuracy						
Multi-Agent Debate	64.4	52.8	49.3	30.0	8.3	41.0
Mixture-of-Agents	63.9	52.5	51.3	**34.0**	11.1	42.6
ExComm (Ours)	**72.5**	**60.6**	**52.2**	30.8	**14.8**	**46.2**
Diversity Score (Trajectory-Level)						
Multi-Agent Debate	53.3	54.7	51.1	51.5	54.2	53.0
Mixture-of-Agents	2.8	2.5	5.4	5.7	4.5	4.2
ExComm (Ours)	**63.1**	**62.1**	**59.7**	**58.7**	**59.1**	**60.5**
Diversity Score (Step-Level)						
Multi-Agent Debate	46.2	45.4	42.3	42.3	44.8	44.2
Mixture-of-Agents	10.9	11.3	13.3	13.3	11.8	12.1
ExComm (Ours)	**55.4**	**52.6**	**51.0**	**48.2**	**49.1**	**51.3**

In this section, we compare ExComm with two representative communication protocols, namely Multi-Agent Debate and Mixture-of-Agents. In Multi-Agent Debate, agents repeatedly share their progress with one another and revise their reasoning based on peer responses. In Mixture-of-Agents, agents' outputs are centrally aggregated at each synchronization step, and subsequent reasoning proceeds from the aggregated state. These methods were originally designed for iterative refinement of candidate solutions, rather than exploration-stage communication within an agent execution loop. We therefore adapt them to this setting as strong communication baselines, allowing us to test whether generic information exchange is sufficient or whether the targeted design of ExComm is necessary for reducing error propagation.

As shown in Table 5, ExComm consistently outperforms both protocols across benchmarks. This gap reflects the different objective of our communication design. Multi-Agent Debate and Mixture-of-Agents exchange broad intermediate information, which can improve consensus but may also cause premature synchronization of reasoning trajectories. In contrast, ExComm uses targeted communication, sharing only conflict resolutions with the agents involved and applies explicit trajectory diversification to maintain independent exploration. This allows ExComm to reduce error propagation without collapsing agents into a shared reasoning path.

This distinction is further supported by the diversity analysis. Following prior work [28, 29], we adopt $100 -$ Self-BLEU to measure diversity among agent trajectories. In addition to trajectory-level diversity, we measure step-level diversity by comparing the reasoning traces of parallel agents at the same step. For both metrics, ExComm consistently shows higher trajectory diversity across benchmarks, indicating that exploration-stage communication can improve reliability while preserving diverse reasoning paths.

4.5 Ablation and Analysis

In this section, we analyze ExComm to isolate the contributions of the Consistency Module, soft belief updates, and trajectory diversification. Unless otherwise specified, we use $N = 4$ solver agents across all methods in this section.

As shown in Table 6, adding the Consistency Module without soft belief updates slightly degrades GAIA Level 3 accuracy, suggesting that conflict resolution alone may introduce harmful synchronization in more open-ended search spaces. Soft belief updates mitigate this issue by appending resolutions as external suggestions rather than overwriting agent beliefs. This improves performance on both benchmarks, indicating that agents benefit from peer-derived corrections while retaining their independent reasoning trajectories.

Table 6: **Component Ablation.** Ablation of ExComm on AIME 2024 and GAIA Level 3 using Gemini-2.5-Flash-Lite. We report majority-voting accuracy and error recovery rates. Soft belief updates improve belief consistency by avoiding rigid synchronization, while trajectory diversification yields the largest gains, especially on GAIA.

	AIME 2024	GAIA Level 3
Majority Voting Accuracy		
Independent	65.3	9.0
+ Consistency Module	68.3	8.0
+ Soft Belief Updates	71.4	9.3
+ Diversification Module	**72.5**	**14.8**
Error Recovery Rates		
Independent	32.6	9.7
+ Consistency Module	33.5	12.0
+ Soft Belief Updates	33.6	13.3
+ Diversification Module	**50.6**	**25.6**

Finally, adding the Diversification Module yields the strongest performance, with particularly large gains on GAIA Level 3 and error recovery rates. This suggests that diversification is essential in large search spaces, where agents must explore complementary reasoning paths rather than prematurely converge on a single trajectory. Overall, both components are necessary. Belief consistency improves error detection and correction, while diversification preserves broad search-space coverage and substantially improves recovery from earlier errors.

4.6 Cost Analysis

To assess the efficiency of ExComm, we analyze the per-sample API cost of each method and summarize the performance-cost trade-off in Figure 2. ExComm achieves the best trade-off, matching or outperforming the $N = 8$ baselines with only $N = 4$ solver agents while requiring lower API cost. This suggests that targeted exploration-stage communication can be more cost-effective than simply increasing the number of independent trajectories. We also report latency in Table 15. For example, ExComm has a similar per-sample latency to the $N = 8$ Independent + SR baseline (10.45 min. vs. 9.57 min.), while requiring significantly less API cost and improving accuracy on both AIME and GAIA. Overall, ExComm achieves a favorable performance-cost trade-off while maintaining latency in a comparable range to parallel baselines.

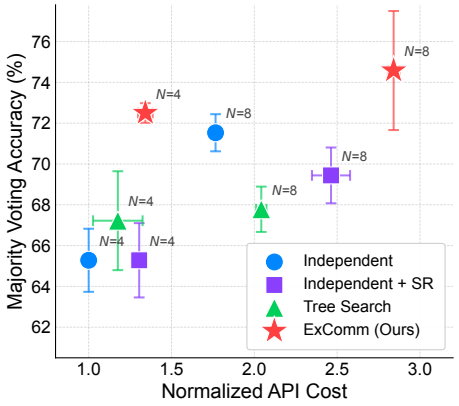


Figure 2: **Performance-Cost Trade-Off.** We plot majority-voting accuracy versus normalized API cost on AIME 2024 using Gemini-2.5-Flash-Lite. Error bars are standard errors.

5 Conclusion

We introduce ExComm, a communication protocol for the exploration stage of agentic test-time scaling. By auditing agents' beliefs for factual inconsistencies and coordinating plan-level diversification, ExComm directly addresses a key failure mode of agentic systems, specifically the propagation of incorrect intermediate beliefs that corrupt downstream reasoning. Our results suggest that the design space of exploration-stage communication (what to share, when to intervene, and how to balance correction with diversity) is a promising and largely unexplored axis for scaling agentic reasoning. As demonstrated in the results, targeted interventions in this space yield substantial gains over both independent scaling and adapted solution-stage communication protocols.

Acknowledgements

For WS, BK, DC and JS: This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.RS-2019-II190075 Artificial Intelligence Graduate School Program (KAIST); No.RS-2022-II220184, 2022-0-00184, Development and Study of AI Technologies to Inexpensively Conform to Evolving Policy on Ethics)

References

[1] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

[2] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594, 2023.

[3] Kuang-Huei Lee, Ian Fischer, Yueh-Hua Wu, Dave Marwood, Shumeet Baluja, Dale Schuurmans, and Xinyun Chen. Evolving deeper llm thinking. *arXiv preprint arXiv:2501.09891*, 2025.

[4] Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. Universal self-consistency for large language model generation. *arXiv preprint arXiv:2311.17311*, 2023.

[5] Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

[6] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

[7] Edward Beeching, Lewis Tunstall, and Sasha Rush. Scaling test time compute for open models. `https://huggingface.co/spaces/HuggingFaceH4/blogpost-scaling-test-time-compute`, 2024. Hugging Face Blog, Accessed: 2025-09-20.

[8] King Zhu, Hanhao Li, Siwei Wu, Tianshun Xing, Dehua Ma, Xiangru Tang, Minghao Liu, Jian Yang, Jiaheng Liu, Yuchen Eleanor Jiang, et al. Scaling test-time compute for llm agents. *arXiv preprint arXiv:2506.12928*, 2025.

[9] Hongwei Zhang, Ji Lu, Shiqing Jiang, Chenxiang Zhu, Li Xie, Chen Zhong, Haoran Chen, Yurui Zhu, Yongsheng Du, Yanqin Gao, et al. Co-sight: Enhancing llm-based agents via conflict-aware meta-verification and trustworthy reasoning with structured facts. *arXiv preprint arXiv:2510.21557*, 2025.

[10] Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*, 2023.

[11] Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. When can llms actually correct their own mistakes? a critical survey of self-correction of llms. *Transactions of the Association for Computational Linguistics*, 12:1417–1440, 2024.

[12] Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions. *arXiv preprint arXiv:2505.23281*, 2025.

[13] Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants. In *The Twelfth International Conference on Learning Representations*, 2024.

[14] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

[15] Qwen Team. Qwen3.5: Towards native multimodal agents. *Qwen. URL: https://qwen.ai/blog?id=qwen3.5 (Access Date: 07. 05. 2026)*, 2026.

[16] Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first international conference on machine learning*, 2024.

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

[18] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2023.

[19] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36:38154–38180, 2023.

[20] Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, et al. Openhands: An open platform for ai software developers as generalist agents, 2024b. *URL https://arxiv. org/abs/2407.16741*, 2(4):9, 2024.

[21] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems 36 (2023): 68539-68551.*, 2023.

[22] Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Yongliang Shen, Kan Ren, Dongsheng Li, and Deqing Yang. Easytool: Enhancing llm-based agents with concise tool instruction. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 951–972, 2025.

[23] Ye Tian, Baolin Peng, Linfeng Song, Lifeng Jin, Dian Yu, Lei Han, Haitao Mi, and Dong Yu. Toward self-improvement of llms via imagination, searching, and criticizing. *Advances in Neural Information Processing Systems*, 37:52723–52748, 2024.

[24] Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Processbench: Identifying process errors in mathematical reasoning, 2025. URL `https://arxiv.org/abs/2412.06559`.

[25] Sundar Pichai, Demis Hassabis, and Koray Kavukcuoglu. A new era of intelligence with gemini 3. *Google. URL: https://blog.google/products-and-platforms/products/gemini/gemini-3/(Access Date: 07. 05. 2026)*, 2025.

[26] Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

[27] Qwen Team. Qwen3 technical report, 2025. URL `https://arxiv.org/abs/2505.09388`.

[28] Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. Encouraging divergent thinking in large language models through multi-agent debate, 2024. URL `https://arxiv.org/abs/2305.19118`.

[29] Tianhui Zhang, Bei Peng, and Danushka Bollegala. Improving diversity of commonsense generation by large language models via in-context learning, 2024. URL https://arxiv.org/abs/2404.16807.

[30] LangChain Team. LangGraph, 2024. URL https://www.langchain.com/blog/langgraph.

[31] Aymeric Roucher, Albert Villanova del Moral, Thomas Wolf, Leandro von Werra, and Erik Kaunismäki. 'smolagents': a smol library to build great agentic systems. https://github.com/huggingface/smolagents, 2025.

A Further Discussions

A.1 Broader Impact Statement

This work studies methods for improving the reliability of agentic test-time scaling through exploration-stage communication and error control. By detecting belief-level inconsistencies among parallel solver agents and resolving them before they propagate, ExComm may contribute to more robust language-model deployment in complex reasoning settings. The proposed framework also aims to improve the performance-cost trade-off by using targeted coordination rather than simply increasing the number of independent trajectories.

At the same time, more capable automated reasoning systems may also be misused or over-trusted, especially in high-stakes domains where incorrect outputs can have significant consequences. As with other advances in language-model reasoning, these techniques should be applied responsibly, with appropriate human oversight, domain-specific validation, and consideration of their broader societal impacts.

A.2 Limitations and Future Work

Our work focuses on error propagation during the exploration stage of agentic test-time scaling, where agents are still forming reasoning trajectories through iterative reasoning and tool use. This scope is intentional, as early belief-level errors can influence many downstream decisions. However, our method is not designed to replace aggregation or refinement after candidate answers are generated.

Another consideration is the quality of the consistency module. ExComm uses a dedicated agentic verifier to resolve factual conflicts, and its effectiveness depends on the verifier's ability to produce reliable corrections. Since incorrect resolutions may introduce new errors, ExComm uses soft belief updates that present corrections as external suggestions rather than replacing the original beliefs. Empirically, ExComm improves performance across all tested models, but in extreme cases, repeated incorrect verification results could still mislead agents and degrade performance. Improving verifier reliability and uncertainty estimation is therefore a promising direction for making exploration-stage communication more robust.

B Implementation Details

B.1 Design of Critic Model

The Critic Model framework, as introduced in ProcessBench [24], was originally employed in the mathematical domain to evaluate candidate solutions for a given problem and identify the first occurrence of an error. They reported that critic models utilizing proprietary language models (e.g., o1-mini) with simple prompting achieved high agreement with human annotations. Notably, these models were able to not only correctly localize the erroneous step but also provide a detailed thinking process and explanation.

Building upon this foundation, we adapt the ProcessBench prompt template to suit our specific objectives. We introduce two primary modifications:

1. **Additional Use of Reference Solutions.** Unlike ProcessBench, which aimed to benchmark the performance of the critic itself, our objective is to maximize the accuracy of the critique. To this end, we provide the critic with a ground-truth reference solution. This is particularly beneficial for tool-augmented benchmarks like GAIA; without a reference solution containing necessary intermediate results (e.g., outputs from web searches), the critic is prone to misjudgment due to missing context.
2. **Identification of Error Recovery.** While ProcessBench focused solely on detecting the existence and location of the first error, we extend this scope to determine if the error is also recovered in subsequent steps. Although detecting error recovery is a more complex task, the provision of a reference solution significantly mitigates the difficulty. By treating the reference solution (which contains key intermediate results) as a roadmap, the model can effectively compare execution traces and verify if the reasoning trajectory realigns with the correct path.

Critic Model Prompt Template

```
You are an expert Critic. Your goal is to evaluate the solving process
of a specific agent ({agent_id}) regarding the given problem.

# Problem: {problem}

# Reference Solution: {reference_solution}
# Reference Answer: {reference_answer}

# Model Execution of {agent_id}
The execution log below belongs to {agent_id}. Focus on {agent_id}'s workflow only.
<model_execution>
  <step_0>
    Action: {Execution results of step 0}
    Memory Diff: {Updated Intermediate Results of step 0}
  </step_0>
  ...
  <step_K>
    Action: {Execution results of step K}
    Memory Diff: {Updated Intermediate Results of step K}
  </step_K>
</model_execution>

# Your Task
1. Analyze the "Action" and "Memory Diff" of {agent_id}.
2. Identify "Factual Errors" committed by {agent_id}.
   - Definition: A Factual Error is where {agent_id} incorrectly derives
     an intermediate result or maintains an incorrect intermediate result
     that impacts decision-making.
   - Exclusions: Minor tool errors, or errors made by other agents.

3. For each Factual Error:
   - Error Type: Description.
   - Occurrence Step: Step number where {agent_id} introduced the error.
   - Recovered Step: Step number where the error was corrected (or "N/A").

4. Final Check:
   - If the agent's final answer is INCORRECT (differs from Reference Answer),
     there MUST be at least one Factual Error that is "N/A" (Unrecovered).
   - Do NOT mark an error as "Recovered" if the agent proceeded to a wrong
     conclusion based on a related misconception.

# Output Format
Provide your analysis in the following XML format:
<error_recovery_analysis>
  <error>
    <description>...</description>
    <occurrence_step>...</occurrence_step>
    <recovered_step>...</recovered_step>
  </error>
</error_recovery_analysis>
```

Figure 3: The prompt template used for the critic model.

Our critic model utilizes a structured prompt designed to detect factual errors and their subsequent recovery. The inputs to the prompt (see Figure 3) include the problem statement, a ground-truth reference solution, and the target agent's step-by-step execution log. The critic is instructed to identify "Factual Errors", defined as incorrect derivations impacting decision-making, while ignoring minor tool slips or external noise. Crucially, the prompt enforces a temporal analysis: for each error, the critic must pinpoint the occurrence step and, if applicable, the recovery step. To ensure consistency, the instructions mandate that any trajectory resulting in an incorrect final answer must be associated with at least one unrecovered error.

B.2 Definition of Error Recovery Rate

We define Error Recovery Rate as the fraction of unique critic-identified errors that are subsequently corrected within the same trajectory. For each task q and agent i, let $\mathcal{E}_{q,i}$ denote the set of unique error records. Each error e has a recovery step r_e, where $r_e = \text{N/A}$ if the error is never recovered. We compute

$$\frac{\sum_{q,i} \sum_{e \in \mathcal{E}_{q,i}} \mathbf{1}[r_e \neq \text{N/A}]}{\sum_{q,i} |\mathcal{E}_{q,i}|}. \tag{1}$$

B.3 Tree Search Baseline

Our tree-search baseline performs beam-style search over N agent trajectories. After each execution step, an LLM-based process reward model (PRM) assigns a fitness score to each trajectory using the evaluation prompt adapted from OAgents [8]. The scorer uses a 0–10 scale and evaluates progress toward task completion (with the highest priority), the severity and stability of factual or logical errors, execution efficiency, and consistency with the current plan.

The scorer receives the original task, current belief state, current plan, and most recently completed execution step, and uses the same backbone model as the solver agents. At each round, the top-scoring $N/2$ trajectories are retained, while lower-scoring trajectories are replaced by copies of the retained trajectories and replanned with a diversity instruction. Final answers are aggregated through majority voting over the N completed trajectories.

C Error Type Analysis

This section describes the experimental setup for the error type classification reported in Table 1. The goal is to determine, for each factual error in independent-baseline trajectories, whether the error would be detectable through cross-agent conflict analysis.

Error identification. We use an LLM-as-a-Judge approach to identify factual errors in each agent's execution trajectory. The critic model setup follows the same design described in Section B.1. Given a problem, a reference solution, and an agent's step-by-step execution log, the critic identifies factual errors and the steps at which they occur. For each identified error, we then classify whether the error would be detectable by examining the beliefs of other parallel agents at the corresponding step.

Classification procedure. For each identified error, we examine the reasoning logs of the other three parallel agents at the same execution step. A critic model classifies the error into one of the following categories based on the other agents' beliefs at the time of the error:

- **Conflicting:** At least one other agent has established a different factual claim on the same topic, whether correct or incorrect. The presence of a conflicting claim makes the error detectable via cross-agent comparison.
- **Common:** All other agents make the same or an equivalent error. No conflicting claim exists, so the error is not detectable through cross-agent conflict analysis.
- **Neutral:** No other agent explicitly discusses or contradicts this specific fact in their completed reasoning at the given step. The error goes unmentioned.

To ensure temporal fairness, each other agent's reasoning log is truncated to include only steps up to and including the step at which the error occurs. This prevents the classification from relying on information that would not yet be available at the time the error is committed.

Evaluation. We run the classification using three critic models (Gemini-3-Flash, GPT-5-nano, and Gemini-2.5-Flash-Lite) and report the average across all three. The analysis is conducted on errors identified from four parallel Gemini-2.5-Flash-Lite agents running Independent scaling on AIME 2024, AIME 2025, and GAIA. The prompt template used for the classification is shown in Figure 4. This analysis provides an upper bound on the proportion of errors detectable through cross-agent conflicts.

Error Type Classification Prompt Template

```
You are an expert analyst evaluating multi-agent reasoning systems.
Your task is to classify whether a specific factual error made by
one agent is detectable via cross-agent conflict analysis.

# Task
{question}

# Error Made by Agent {expert_i}
The following error was introduced at step {occurrence_step}:
"{error_description}"

# Reasoning Logs of Other Agents
Below are the reasoning logs of the other {n_others} agents working on the same task.
{other_agent_logs}

# Your Task
Classify the error above into exactly ONE of these categories,
based on the other agents' logs:

CATEGORY 1 (Conflicting -- correct): At least one other agent has
ALREADY ESTABLISHED the CORRECT value/fact that directly contradicts
the error.

CATEGORY 2 (Conflicting -- incorrect): At least one other agent has
already established a DIFFERENT value/fact on the same topic, but
that value is also incorrect.

CATEGORY 3 (Shared): All other agents make the same error.

CATEGORY 4 (Silent): No other agent explicitly discusses, confirms,
or contradicts this specific fact.

CATEGORY 5 (Not an error): The described "error" does not appear to
be a genuine factual mistake. Category 5 entries are excluded from
the final computation.

Think step by step. Then output your classification in the following
XML format:
<conflict_detectability_analysis>
<reasoning>Your step-by-step analysis here.</reasoning>
<category>1</category>
</conflict_detectability_analysis>
```

Figure 4: Prompt template for error type classification. Categories 1 and 2 are merged as "Conflicting" in Table 1. Category 3 maps to "Common". Category 4 maps to "Neutral"

D Additional Results

D.1 Verifier Reliability and Failure Modes

Since ExComm relies on a dedicated verifier to resolve factual conflicts, incorrect resolutions may introduce new errors into the agents' belief states. To assess this risk, we investigate the reliability of the verifier and how ExComm's safeguards mitigate the downstream impact of incorrect resolutions. We use Gemini-3-Flash as the judge model to evaluate verifier outcomes and their downstream effects.

Verifier reliability. As shown in Table 7, the verifier produces correct resolutions in 60.2% of effective verification events, while 39.8% are incorrect. However, incorrect resolutions do not necessarily propagate into persistent errors. ExComm employs soft belief updates, selective communication, and subsequent consistency checks to mitigate their impact.

Table 7: **Verifier Resolution Outcomes.** Outcomes of verifier resolutions and error recovery. We report proportions (%) over effective verification events, with standard errors over three runs.

Verifier Outcome	Proportion (%)
Correct	$60.2_{\pm 3.4}$
Incorrect	$39.8_{\pm 3.4}$
Re-detected following soft updates	$10.8_{\pm 1.5}$
Re-detected through other safeguards	$12.1_{\pm 1.5}$
Never re-detected	$16.4_{\pm 1.7}$
Other outcomes	$0.5_{\pm 0.2}$

Soft belief updates. Rather than forcing agents to adopt the verifier output, soft updates preserve alternative beliefs. This prevents uniform convergence to an incorrect resolution in 14.7% of all verification events (36.9% of incorrect resolutions). The preserved disagreement subsequently enables 10.8% of all resolutions (27.1% of incorrect resolutions) to be re-detected through cross-agent conflicts.

Additional safeguards. Selective communication propagates resolutions only to agents involved in the original conflict, allowing uninvolved agents to independently generate conflicting claims. The consistency module can also flag factual errors in individual agent outputs without explicit inter-agent disagreement. Together, these mechanisms enable the later re-detection of an additional 12.1% of all verifier resolutions (30.4% of incorrect resolutions).

Lasting degradation. The raw verifier error rate does not directly measure its downstream impact. We define lasting degradation as cases where an incorrect resolution causes a receiving agent to replace a correct belief with an incorrect one and fail to recover before finalization. As shown in Table 8, lasting degradation occurs in only 4.5% of all resolution deliveries and 11.4% of incorrect-resolution deliveries. These results suggest that ExComm's safeguards substantially limit the downstream impact of imperfect verification.

Table 8: **Lasting Degradation.** Lasting degradation caused by incorrect verifier resolutions at the receiving-agent level. We report proportions (%) with standard errors over three runs.

Delivery Population	Lasting Degradation (%)
All resolution deliveries	$4.5_{\pm 0.6}$
Incorrect-resolution deliveries	$11.4_{\pm 1.3}$

D.2 Budget-Matched Evaluations

To test whether additional inference compute alone explains the improvements, we extend each $N = 4$ baseline with additional self-revision steps until its measured API cost exceeds that of ExComm. We also compare against $N = 8$ baselines. Table 9 shows that ExComm achieves higher accuracy on both AIME and GAIA than all extended or larger-scale baselines.

Table 9: **Budget-Matched Evaluations.** Comparison of test-time scaling methods under controlled API budgets using Gemini-2.5-Flash-Lite. Cost denotes mean per-sample API cost (USD) on AIME 2024. AIME and GAIA scores are averaged over AIME 2024/2025 and GAIA L1/L2/L3, respectively, with standard errors over three runs.

Method	N	Cost (USD)	AIME	GAIA
Independent (Extended)	4	0.325	$61.8_{\pm 1.60}$	$27.1_{\pm 1.71}$
Independent + SR (Extended)	4	0.327	$59.2_{\pm 0.96}$	$25.5_{\pm 1.02}$
Tree Search (Extended)	4	0.331	$60.7_{\pm 2.05}$	$28.2_{\pm 1.31}$
Independent	8	0.415	$66.0_{\pm 1.01}$	$30.6_{\pm 1.24}$
Independent + SR	8	0.578	$65.5_{\pm 0.25}$	$30.0_{\pm 1.95}$
Tree Search	8	0.479	$60.8_{\pm 2.10}$	$29.6_{\pm 1.45}$
ExComm (Ours)	**4**	**0.318**	$\mathbf{66.5_{\pm 1.19}}$	$\mathbf{32.6_{\pm 1.03}}$

D.3 Verifier-Free Ablations

To isolate the contribution of centralized verification, we evaluate a verifier-free variant of ExComm that retains exploration-stage communication and trajectory diversification. We further evaluate a cost-matched variant that reallocates the saved verifier budget to one additional round of self-revision per agent. As shown in Table 10, both verifier-free variants retain substantial gains over the baselines, while the full method achieves the highest accuracy on both AIME and GAIA. The full method also outperforms the cost-matched variant at comparable cost and latency, demonstrating that the verifier's contribution cannot be explained solely by additional inference compute.

Importantly, the verifier uses the same backbone model and benchmark-specific tools as the solver agents, differing only in its specialized conflict-resolution role and access to conflicting beliefs. Thus, the additional gains of the full method reflect the benefit of targeted verification rather than a stronger external model.

Table 10: **Verifier-Free Ablations.** Comparison of ExComm with and without centralized verification using Gemini-2.5-Flash-Lite and $N = 4$ solver agents. AIME and GAIA scores are averaged over AIME 2024/2025 and GAIA L1/L2/L3, respectively, with standard errors over three runs. Cost and latency are normalized to the full method.

Method	AIME	GAIA	Relative Cost	Relative Latency
ExComm (Full)	**66.5**$\pm$**1.19**	**32.6**$\pm$**1.03**	100.0%	100.0%
ExComm (Verifier-free)	62.9$\pm$1.25	31.6$\pm$0.47	94.1%	85.3%
ExComm (Verifier-free, cost-matched)	62.1$\pm$0.42	30.1$\pm$1.35	100.1%	101.5%

D.4 Generalization Beyond Co-Sight

To evaluate whether the improvements of ExComm generalize beyond Co-Sight, we integrate the same exploration-stage communication protocol into two additional agent frameworks, LangGraph [30] and smolagents [31]. We evaluate both frameworks on AIME and GAIA, incorporating the benchmark-specific tools required for each setting. As shown in Table 11, ExComm consistently outperforms all baselines across both frameworks and benchmarks, demonstrating the portability of our communication protocol across different agent implementations.

Table 11: **Generalization Across Agent Frameworks.** Comparison of test-time scaling methods on LangGraph and smolagents using Gemini-2.5-Flash-Lite. AIME and GAIA scores are averaged over AIME 2024/2025 and GAIA L1/L2/L3, respectively. We report majority-voting accuracy with standard errors over three runs.

Method	N	AIME	GAIA
LangGraph			
Base Agent	1	56.9$\pm$0.74	24.8$\pm$0.86
Sequential Revision (SR)	1	56.7$\pm$2.44	23.8$\pm$0.68
Independent Scaling	4	63.1$\pm$0.50	30.4$\pm$2.03
Independent Scaling + SR	4	63.3$\pm$2.73	28.3$\pm$1.25
Tree Search	4	62.9$\pm$1.97	30.4$\pm$0.40
ExComm (Ours)	4	**68.2**$\pm$**1.00**	**33.1**$\pm$**1.31**
smolagents			
Base Agent	1	55.3$\pm$1.00	24.3$\pm$1.23
Sequential Revision (SR)	1	56.8$\pm$1.81	22.0$\pm$0.42
Independent Scaling	4	62.4$\pm$0.61	29.4$\pm$0.84
Independent Scaling + SR	4	65.6$\pm$1.23	26.2$\pm$1.29
Tree Search	4	61.3$\pm$1.50	30.9$\pm$1.77
ExComm (Ours)	4	**68.6**$\pm$**2.68**	**33.6**$\pm$**1.24**

D.5 Extended Diversity Analysis

To examine whether the diversity gains extend beyond lexical variation, we evaluate semantic and category-based diversity using the same agent plan documents as in the Self-BLEU analysis. Each document records the agent's evolving subgoals and execution notes, including reasoning, actions, and tool interactions. We use Gemini-3-Flash for the LLM-as-a-judge evaluations.

Semantic diversity. We measure the mean pairwise cosine distance between embeddings of agent plans, scaled by 100. As shown in Table 12, ExComm achieves the highest semantic diversity across both embedding models, benchmarks, and evaluation granularities. These results suggest that the diversity gains extend beyond surface-level lexical differences.

Table 12: **Semantic Diversity.** Mean pairwise cosine distance between agent plan embeddings, scaled by 100. We report standard errors over three runs, with higher values indicating greater diversity.

Granularity	Method	AIME	GAIA
	all-mpnet-base-v2		
Step	MAD	16.2±0.32	14.3±0.33
	MoA	4.7±0.11	4.7±0.08
	ExComm (Ours)	**17.8**±**0.23**	**16.5**±**0.19**
Trajectory	MAD	20.3±0.36	17.6±0.62
	MoA	1.2±0.08	2.1±0.13
	ExComm (Ours)	**22.9**±**0.20**	**20.4**±**0.30**
	bge-large-en-v1.5		
Step	MAD	8.5±0.20	7.6±0.11
	MoA	2.5±0.05	2.6±0.05
	ExComm (Ours)	**9.5**±**0.09**	**9.0**±**0.12**
Trajectory	MAD	10.2±0.18	8.8±0.22
	MoA	0.6±0.02	1.1±0.04
	ExComm (Ours)	**11.5**±**0.14**	**10.7**±**0.22**

Category-based diversity. We further evaluate diversity in solution strategies and explored subproblems using an LLM judge. Given the plan documents of four agents from MAD, MoA, and ExComm, the judge identifies distinct strategies and subproblems and assigns them to each agent's plan. We measure strategy diversity as the number of distinct strategy categories normalized by agent count, and subproblem diversity as the mean pairwise Jaccard distance between explored-subproblem sets. As shown in Tables 13a and 13b, ExComm consistently achieves the highest diversity across both metrics, providing more direct evidence of diverse solution strategies and exploration directions.

Table 13: **Category-Based Diversity.** Comparison of solution-strategy and explored-subproblem diversity across communication methods. Strategy diversity measures the number of distinct strategy categories normalized by agent count (%), while subproblem diversity measures the mean pairwise Jaccard distance between explored-subproblem sets, scaled by 100. We report standard errors over three runs, with higher values indicating greater diversity.

(a) Solution-Strategy Diversity

Granularity	Method	AIME	GAIA
Step	MAD	47.3±0.65	46.4±0.63
	MoA	28.6±0.34	30.0±0.25
	ExComm (Ours)	**61.0**±**0.64**	**59.1**±**0.29**
Trajectory	MAD	45.4±1.44	48.9±1.36
	MoA	25.4±0.24	26.0±0.34
	ExComm (Ours)	**62.1**±**0.48**	**63.9**±**0.46**

(b) Explored-Subproblem Diversity

Granularity	Method	AIME	GAIA
Step	MAD	19.3±0.72	13.4±0.37
	MoA	3.4±0.47	3.2±0.17
	ExComm (Ours)	**27.1**±**0.75**	**19.3**±**0.12**
Trajectory	MAD	15.1±0.88	14.5±0.88
	MoA	0.1±0.11	1.2±0.40
	ExComm (Ours)	**24.9**±**0.49**	**24.3**±**1.29**

D.6 Additional Evaluation on Humanity's Last Exam

To evaluate generalization beyond competition mathematics and open-ended agent tasks, we test ExComm on a 100-example multiple-choice subset of Humanity's Last Exam (HLE) using Gemini-2.5-Flash-Lite. As shown in Table 14, ExComm achieves the highest accuracy among the evaluated methods, extending the observed benefits to multidisciplinary expert-level reasoning.

Table 14: **Humanity's Last Exam Evaluation.** Comparison of test-time scaling methods on a 100-example multiple-choice subset of HLE using Gemini-2.5-Flash-Lite. We report majority-voting accuracy with standard errors over three runs.

Method	N	Accuracy (%)
Base Agent	1	$7.2_{\pm 0.55}$
Sequential Revision (SR)	1	$7.0_{\pm 0.52}$
Independent Scaling	4	$7.8_{\pm 0.62}$
Independent Scaling + SR	4	$7.4_{\pm 0.96}$
Tree Search	4	$8.3_{\pm 0.67}$
ExComm (Ours)	4	$\mathbf{10.4}_{\pm \mathbf{1.79}}$

D.7 Latency Analysis

Table 15 reports the per-sample latency of each method. The additional latency of ExComm relative to Independent scaling mainly comes from the Consistency Module, which runs a separate tool-augmented reasoning loop, along with the plan analysis performed by the Trajectory Diversification Module. Despite this overhead, ExComm at $N = 4$ outperforms all $N = 8$ baselines on both AIME and GAIA, suggesting that targeted coordination uses each trajectory more effectively.

Table 15: **Latency Analysis.** We report median per-sample latency (minutes) with Gemini-2.5-Flash-Lite. Accuracy columns show the average over AIME 2024/2025 and GAIA L1/L2/L3, respectively. ExComm at $N = 4$ outperforms scaled-up baselines at $N = 8$ while maintaining modest additional latency.

	N	Latency (min.)	Accuracy (AIME)	Accuracy (GAIA)
Independent	8	7.94	66.0	30.6
Independent + SR	8	9.57	65.5	30.0
Tree Search	8	7.38	60.8	29.6
ExComm (Ours)	4	10.45	**66.5**	**32.6**

E Example Illustrations

This section provides concrete examples of the structured objects used by the Consistency Module and the Trajectory Diversification Module, illustrating the format of conflicts, resolutions, and diversification directives introduced in Section 3.

E.1 Conflict and Resolution Objects

A *conflict* is a structured object that identifies mutually exclusive factual claims held by different agents at the same reasoning step. The `claims` field records each involved agent's specific claim on the contested topic. A *resolution* is the output of the tool-augmented verification loop that adjudicates the conflict. It contains the verified correct claim, a justification, and the list of agents that should receive the correction.

As a concrete illustration, consider a scenario where agents need to identify the largest two-digit number divisible by both 6 and 8 as an intermediate step. Agent 1 correctly computes 96 (noting $\text{LCM}(6, 8) = 24$ and $24 \times 4 = 96$). Agent 2 stops at 72 ($24 \times 3 = 72$, having not searched far enough). The Consistency Module flags this disagreement and extracts the following conflict:

Example Conflict

```
conflict:
  agents: [1, 2]
  description: |
    Agents disagree on the largest two-digit number
    divisible by both 6 and 8. Agent 1 identifies 96;
    Agent 2 identifies 72.
  claims:
    agent_1: "96"
    agent_2: "72"
```

The Consistency Module then invokes a tool-augmented ReAct loop that tests both candidates via the Python interpreter, without assuming either agent is correct. It verifies divisibility of both 96 and 72 by 6 and 8, confirms both are valid multiples, and selects the larger one:

Example Resolution

```
resolution:
  agents: [1, 2]
  description: |
    Agents disagree on the largest two-digit number
    divisible by both 6 and 8.
  claims:
    agent_1: "96"
    agent_2: "72"
  correct_claim: "96"
  reason: |
    Both 96 and 72 are divisible by 6 and 8 (verified
    by Python). Since 96 > 72, 96 is the largest.
    Agent 2 stopped its search prematurely.
```

This resolution is then incorporated into the involved agents' belief states via soft updates, appending the verified information as an external suggestion rather than overwriting existing beliefs.

E.2 Diversification Directive

Suppose Agents 0 and 1 both plan to solve the same sub-problem by iterating downward from 100 and checking whether each number is divisible by both 6 and 8. The Trajectory Diversification Module detects this redundancy and issues a directive to redirect one agent toward an orthogonal approach:

Example Diversification Directive

```
directive:
  target_agent_index: 1
  modification_instruction: |
    Instead of iterating downward from 100, compute the
    set of all positive integers less than 100 that are
    divisible by 6, then filter for those also divisible
    by 8, and return the maximum of the resulting set.
```

This shifts Agent 1 from a sequential brute-force search to a set-based filtering approach, ensuring that the two agents explore complementary strategies for the same sub-problem.

F Case Studies

We present representative case studies from evaluation trajectories to illustrate how the Consistency Module and the Trajectory Diversification Module operate in practice.

Consistency Module (AIME 2024 #19). An agent incorrectly modeled a geometric locus as a circle rather than an astroid. The Consistency Module flagged the geometric conflict across agents, verified the astroid equation via tool-augmented reasoning, and issued a soft update. The failing agent recovered and adopted the correct geometric model without losing its prior reasoning context.

Diversification Module (AIME 2025-II #6). Agent plans initially converged on coordinate geometry, creating a shared structural bottleneck. The Trajectory Diversification Module detected this redundancy and directed one agent to pivot to synthetic geometry. This produced an orthogonal trajectory that uniquely solved the task.

Soft update robustness (GAIA e1fc63a2). The Consistency Module erroneously corrected a valid calculation due to unit confusion. Because corrections are applied as soft updates rather than hard overwrites, agents retained their valid derivations alongside the flawed suggestion. They recognized the verifier's error in subsequent reasoning steps and answered correctly. This case illustrates that the soft update design is necessary for robustness against imperfect conflict resolution.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: They clearly reflect the paper's contributions and scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Section A.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: No theoretical components.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We outlined the details in the experiment section and appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: No code release at the current stage.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: See experiments section.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: 3 rounds of repetitive experiments are done, and we report the standard errors.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See efficiency section.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Code of ethics are followed.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See Section A.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: No such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Appropriate citations and acknowledgements are made.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: No new asset.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No human experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No human experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: The use of LLMs in our proposed method, including conflict resolution and trajectory diversification, is described in Section 3. Additional details on LLM-based evaluation are provided in the appendix.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.