WHAT DO AGENTS MISS WHEN THEY DECIDE? EXPLORING EPISTEMIC GAPS IN LLM-BASED MULTI-AGENT SYSTEMS THROUGH SIBLING PROBING

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM-based multi-agent systems distribute decision making across agents, such as planning and execution. Recent research has shown that failures can arise when a decision lacks necessary information support despite correct execution, a phenomenon known as an *epistemic gap*. Existing approaches mainly use retrospective feedback from past interactions to calibrate subsequent decisions. However, such feedback offers limited insight into which decision contains an *epistemic gap*, as decisions are made top-down with obtained information, while later information emerges bottom-up and outcomes entangle multiple decisions. We formulate *decision-level epistemic exploration* in MAS to acquire information that resolves such gaps. Two challenges arise: *epistemic gaps can accumulate across hierarchical decision levels*, making it unclear which decision level to explore; *explored trajectories entangle information across decision levels*, making decision-specific information requirements difficult to extract. We design HIVE, which starts from an observed failure and probes alternatives in a bottom-up order to mitigate the effects of epistemic gap accumulation, while constructing explored trajectories to disentangle role-specific information requirements through trajectory comparisons. Based on these comparisons, HIVE identifies why current knowledge fails to support the decision and updates role-specific knowledge online within the current task. Across six benchmarks, HIVE improves task success by 4.46% over the compared baselines while reducing average trajectory length by 23.67%.

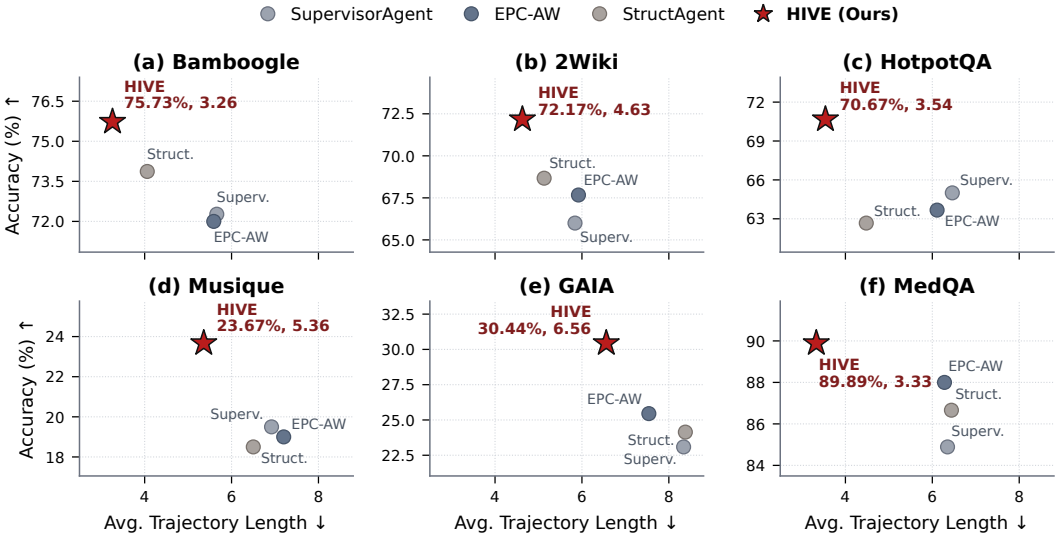

Figure 1: HIVE achieves higher accuracy with shorter trajectory length across all six benchmarks. Each panel plots accuracy (%) against the average length of trajectory for HIVE and three baselines (SupervisorAgent, EPC-AW, and StructAgent). The *upper-left region* is preferable (higher accuracy, shorter trajectory length). HIVE consistently occupies this favorable region across six benchmarks.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

Large language model (LLM)-based multi-agent systems (MAS) have emerged as a promising paradigm for solving complex tasks by organizing agents with specialized roles (Qi et al., 2026; Yan et al., 2025). By distributing decision making across agents, such as planning and execution, MAS combines complementary capabilities through modular workflows (Li et al., 2026b; Yang et al., 2026). However, interactions among agents can also introduce additional failure modes, making reliable task completion challenging (Cemri et al., 2026; Hammond et al., 2025). LLM-based MAS may fail even when the planned action is correctly executed, as a decision may lack the information needed to support it, leaving the intended goal unsatisfied (Qiao et al., 2026; Wang et al., 2026). Thus, system reliability depends not only on correct execution, but also on whether decisions are adequately supported by the information available to them.

This perspective has motivated growing interest in the *epistemic gap* in the LLM-based MAS (Wang et al., 2026; Yadkori et al., 2024), which concerns whether decisions are adequately supported by the information available to agents. Existing studies address this problem through different forms of adaptation. One line of work calibrates planning decisions by evaluating whether they remain supported under different information conditions and refining subsequent decisions based on past discrepancies (Wang et al., 2026), while another assesses peer reliability from agents' interaction histories to guide subsequent decisions (Zhou et al., 2026b). Despite their different mechanisms, these approaches primarily use information from *past interactions* to improve subsequent decisions.

However, such retrospective information does not reveal which decision was unsupported or what information was missing to support it. In LLM-based MAS, decisions are made *top-down*, with higher-level choices of agents constraining lower-level decisions, while decision-relevant information is revealed *bottom-up* through subsequent execution. This creates a temporal mismatch, as the information needed to support a decision may only emerge after it is made, while the resulting information is entangled across decisions. For example, in a Planner-Executor workflow, the Planner generates a subgoal and selects a tool to solve it, constraining the Executor's parameter decision (Li et al., 2026b; Wang et al., 2026). The Executor carries out the action, and whether the tool and its parameters can fulfill the subgoal may only become clear after execution. An unsatisfied subgoal can indicate that the joint decision lacked the information needed to support it, even when the Executor correctly executed the action. This constitutes an epistemic gap in the preceding decision. History-based reflection can improve subsequent decisions, but provides limited insight into which decision lacked information support and what was missing (Li et al., 2026b; Wang et al., 2026). This limitation motivates exploring information underlying individual decisions beyond retrospective reflection.

We therefore formulate this problem as *decision-level epistemic exploration in LLM-based MAS*, where agents actively acquire the information needed to resolve epistemic gaps at individual decision levels. This problem presents two fundamental challenges. ***First, epistemic gaps can accumulate across hierarchical decision levels.*** Decisions are interdependent, and thus an epistemic gap at a downstream level can undermine the information support for upstream decisions. For example, without the information needed to configure a tool's parameters, the Executor cannot provide sufficient support for the Planner to assess whether the tool can satisfy the subgoal. Thus, failure does not reveal what information is missing, making it unclear which decision level to explore. While some work (Lee et al., 2026; Kim et al., 2026b) explores information acquisition, it mainly targets single-agent settings with unconstrained exploration, rather than epistemic gaps in hierarchical MAS decisions. ***Second, explored trajectories entangle information across decision levels.*** An explored trajectory reflects not only the target decision, but also other decisions made at different levels and the encountered environmental states. The outcome of this trajectory, therefore, does not directly reveal what information the target decision requires, making decision-specific information requirements difficult to extract. However, within LLM-based MAS, existing approaches primarily use information from past interactions to calibrate subsequent planning decisions, without explicitly identifying the information requirements of individual decisions (Wang et al., 2026; Zhou et al., 2026b).

To address these challenges, we propose **HIVE** (**H**ierarchical **I**ntervention for **V**erifying **E**pistemic gaps), a framework for decision-level epistemic exploration in LLM-based MAS. Drawing on Pearl's causal hierarchy (Pearl, 2010), HIVE progresses from observing the original trajectory to intervening on individual decisions and comparing outcomes. To handle the accumulation of epistemic gaps across hierarchical decision levels, HIVE probes alternatives in a bottom-up order. It first probes

alternative parameters while preserving the tool and subgoal, expanding to tools and subgoals only when needed. Each probe intervenes on one decision and is executed and verified in the environment, constructing a *sibling trajectory* that shares the pre-intervention context but differs in one decision, enabling matched comparisons across decision roles. Contrasting successful and unsuccessful siblings identifies distinguishing information supporting successful decisions, while repeated failures indicate missing information requirements. HIVE then reasons about why the current knowledge fails to support the decision and updates role-specific knowledge online within the current task.

We evaluate HIVE across six benchmarks, as highlighted in Figure 1, where it consistently outperforms existing MAS baselines with higher accuracy and shorter trajectory lengths (§5.2). Ablation studies show that hierarchical sibling probing and online knowledge updating provide complementary gains (§5.3). Comparisons with direct LLM-based attribution demonstrate the importance of hierarchical sibling probing for effective decision refinement (§5.4). These results show consistent gains across the evaluated LLM backbones (§5.5). Finally, HIVE substantially shortens task trajectories by suppressing redundant MAS iterations, improving overall efficiency (§5.6).

Our contributions are summarized as follows:

- We formulate *decision-level epistemic exploration* in LLM-based MAS as the active acquisition of decision-relevant information to resolve epistemic gaps across decision levels.
- We introduce HIVE with hierarchical sibling probing to handle epistemic gaps across hierarchical decisions by probing alternatives in a bottom-up order and expanding the intervention scope.
- To disentangle information across decision roles, HIVE derives decision-specific information requirements from controlled sibling trajectories and uses their outcome differences and commonalities to refine role-specific knowledge, turning exploratory trajectories into reusable knowledge.
- Experiments show that HIVE improves task success by 4.46% over the compared baselines while reducing average trajectory length by 23.67%.

2 Related Work

Failure and Repair in LLM-based Multi-agent Systems. LLM-based multi-agent systems (MAS) remain vulnerable to reasoning, coordination, and cascading failures (Cemri et al., 2026; Zhang et al., 2026; Li et al., 2026a). Existing approaches mainly perform *post-hoc diagnosis* of responsible agents or steps (Zhang et al., 2025; Epperson et al., 2025; Ma et al., 2026), or *runtime mechanism* through rollback, reflection, supervision, or local correction (Liang et al., 2025; Shen et al., 2026; Lin et al., 2026). Other methods ground execution in structured and verified information (Wu et al., 2026). However, existing approaches mainly diagnose or repair such failures from observed trajectories or feedback. HIVE focuses on *epistemic gaps*, but probes individual decisions by executing alternative choices and comparing the resulting sibling trajectories to identify missing information.

Epistemic Reasoning and Calibration in Multi-agent Systems. Prior work studies epistemic gaps through calibration, uncertainty estimation, and knowledge-awareness (Woodruff & Hewitt, 2026; Shao et al., 2026), with recent MAS work assessing plan support, collaborator reliability, and information gained through communication (Wang et al., 2026; Zhou et al., 2026b; Qiao et al., 2026). These approaches primarily evaluate or estimate epistemic support from information already available. They therefore provide ways to assess whether an agent has sufficient support, but do not directly specify how the system should acquire missing support after a decision failure. HIVE instead treats verifier-confirmed failure as an actionable epistemic signal, probing the decision hierarchy to acquire missing information and convert it into role-specific knowledge.

Active Information Seeking and Exploration in LLM Agents. Active information acquisition has been studied through environment observations (Lee et al., 2026; Kim et al., 2026a), exploration strategies (Jiang et al., 2026; Szot et al., 2026), skill discovery (Yang et al., 2025b), and peer-agent exploration (Choi et al., 2026); related work also considers when external tool use is epistemically necessary (Hongru et al.). However, these approaches do not address *epistemic gap exploration in LLM-based MAS*, which faces two MAS-specific challenges: epistemic gaps can accumulate across hierarchical decisions, and explored trajectories entangle information across decisions. HIVE probes alternative decisions upward to locate missing information, then compares sibling trajectories to derive decision-specific requirements and distill them into role-specific online knowledge.

3 PROBLEM FORMULATION

LLM-based MAS operation. Given an input query q, an LLM-based MAS consists of three specialized roles (Wang et al., 2026; Li et al., 2026b): a Planner A^P that determines the next subgoal and selects the tool, an Executor A^E that determines how to instantiate the selected tool, and a Verifier A^V that evaluates the resulting state. The MAS iteratively invokes these roles to acquire the information required for the final answer.

At iteration t, the system state is $S_t = (q, K_t, M_t)$, where $K_t = (K_t^P, K_t^E)$ denotes the task-local knowledge that supports the Planner and Executor in making their respective decisions, and M_t denotes task-local shared system memory accumulated from prior interactions. Given S_t, the Planner selects a subgoal and tool, $(g_t, \tau_t) = A^P(S_t)$, and the Executor determines the corresponding tool parameters, $\theta_t = A^E(S_t, g_t, \tau_t)$. The resulting joint decision is denoted by $d_t = (g_t, \tau_t, \theta_t)$ and executed in the environment to produce an execution trajectory $\xi_t = \mathrm{Exec}(S_t, d_t)$. The resulting experience is stored in the memory M_{t+1}.

After execution, the Verifier evaluates both subgoal satisfaction and termination:

$$(Y_t, C_t) = A^V(q, S_t, d_t, \xi_t), \qquad Y_t, C_t \in \{0, 1\}, \tag{1}$$

where $Y_t = 0$ indicates an *iteration failure*, meaning that the execution fails to satisfy the current plan's subgoal, while $C_t = 1$ indicates that the current state contains sufficient information to produce the final answer and terminate the MAS.

Epistemic gaps An *epistemic gap* in LLM-based MAS refers to the lack of information necessary for making a correct decision at a given level of hierarchical decision-making. At iteration t, let $\mathcal{I}_t^\ell$ denote the information available at decision level $\ell \in \{g, \tau, \theta\}$. Given the hierarchical decision order $g \to \tau \to \theta$, the available information at each level is represented as

$$\mathcal{I}_t^\ell = \mathrm{Extract}\left(S_t, d_t^{<\ell}\right), \qquad \ell \in \{g, \tau, \theta\}, \tag{2}$$

where $\mathrm{Extract}(\cdot)$ constructs the decision-specific information from the system state and the decisions made at preceding levels, and $d_t^{<\ell}$ denotes those preceding decisions. Let $\mathcal{I}_t^{\ell,*}$ denote the information necessary to support a decision satisfying the corresponding objective at level ℓ. This information is latent and cannot be directly evaluated during execution. The epistemic gap at level ℓ is defined as

$$\mathcal{G}_t^\ell = \mathcal{I}_t^{\ell,*} \setminus \mathcal{I}_t^\ell, \qquad \ell \in \{g, \tau, \theta\}, \tag{3}$$

representing the information necessary for the decision at level ℓ but unavailable when that decision is made. Although decisions are made upstream to downstream, $g \to \tau \to \theta$, information needed to identify epistemic gaps may emerge in the reverse direction, $\theta \to \tau \to g$, as downstream decisions and outcomes can reveal information relevant to upstream decisions.

Problem statement Decision-level epistemic exploration aims to identify missing information behind hierarchical decisions. When an iteration fails, an exploration procedure obtains m trajectories $\{\xi_{t+1}, \ldots, \xi_{t+m}\}$ by varying decisions from the failed trajectory. These trajectories are analyzed to extract decision-relevant knowledge, which is incorporated into the subsequent state S_{t+m+1}. The available information at each decision level is then updated as

$$\mathcal{I}_{t+m+1}^\ell = \mathrm{Extract}\left(S_{t+m+1}, d_{t+m+1}^{<\ell}\right), \qquad \ell \in \{g, \tau, \theta\}. \tag{4}$$

The objective is to acquire decision-relevant information that reduces epistemic gaps, supports subsequent decisions, and improves task success.

4 HIVE: DECISION-LEVEL EPISTEMIC EXPLORATION

Inspired by Pearl's causal hierarchy (Pearl, 2010), HIVE compares outcomes under observed and intervened decisions to probe epistemic gaps. The overall workflow is shown in Fig. 2 and detailed in §4.1. When the Verifier detects an iteration failure, HIVE is triggered to diagnose the underlying epistemic gap. Because epistemic gaps can accumulate across hierarchical decision levels, HIVE probes decision alternatives in a bottom-up order, expanding the intervention scope only when lower-level alternatives fail (§4.2). To disentangle information in the explored trajectories across decision roles, HIVE constructs sibling trajectories and derives the information requirements for each decision level (§4.3). HIVE then reasons about why current knowledge fails to support the decision and updates role-specific knowledge online within the current task (§4.4).

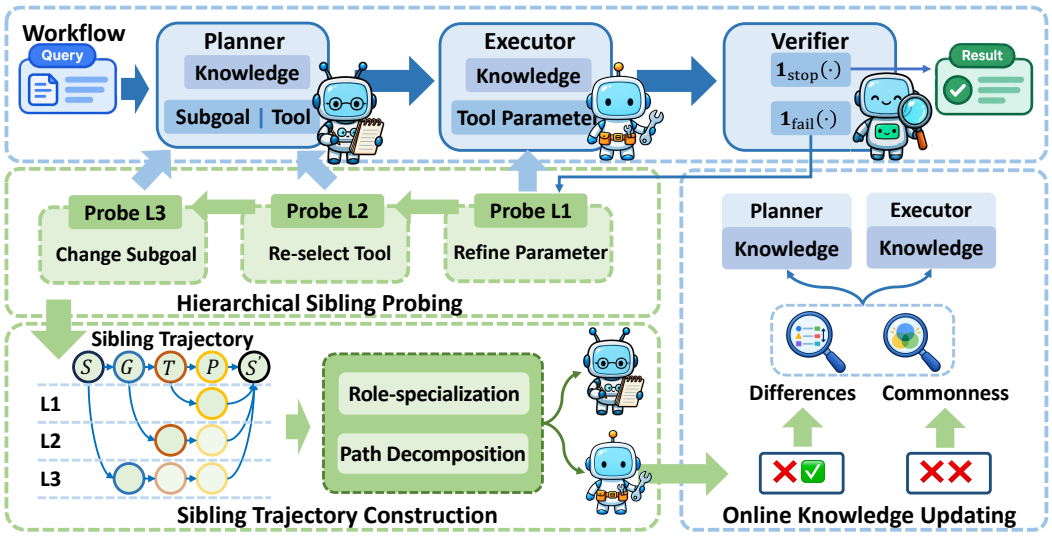

Figure 2: Overview. HIVE explores hierarchical decision alternatives and updates role-specific knowledge from their executed trajectories, while retaining explored siblings as reusable demonstrations.

4.1 System Workflow

Pearl's causal hierarchy (Pearl, 2010) distinguishes three reasoning levels: *observation* of what happened, *intervention* on altered conditions, and *counterfactual reasoning* through outcome comparison. This progression guides HIVE for epistemic gap exploration: observation identifies failure, intervention isolates decision roles, and counterfactual-inspired comparison reasons about why existing knowledge fails to support correct decisions, guiding knowledge updates.

HIVE is triggered when the Verifier identifies an iteration failure, i.e., $Y_t = 0$, where t anchors the exploration episode. A sibling trajectory is an alternative execution from the same pre-intervention state, obtained by intervening at one decision level while keeping upstream decisions fixed and regenerating downstream decisions as needed. HIVE probes alternatives from parameters θ to tools τ and subgoals g, advancing only when the current alternative fails, and terminates upon success or after all three levels are explored. Parameter probing reuses the selected tool and bypasses a Planner call, whereas tool- and subgoal-level probing require new Planner decisions. Each probing constitutes one MAS iteration and is counted in the trajectory length. Thus, if m_t siblings are explored, normal execution resumes at step $t+m_t+1$ with the updated knowledge available to the Planner and Executor.

4.2 Hierarchical Sibling Probing

HIVE fixes the failed decision context S_t and probes decision alternatives in the order $\theta \to \tau \to g$. At each level, HIVE intervenes on the selected decision level, keeps all upstream decisions fixed, and regenerates downstream decisions conditioned on the intervention. In particular, *parameter probing bypasses Planner invocation.* Each sibling constitutes an independent MAS iteration.

Consider a failed trajectory ξ_t at iteration t, where $Y_t = 0$, with failed decision $d_t = (g_t, \tau_t, \theta_t)$. Let $\ell_1 = \theta$, $\ell_2 = \tau$, and $\ell_3 = g$ denote the probing order. The corresponding sibling decisions are

$$d_t^{[\theta]} = (g_t, \tau_t, \tilde{\theta}^{[\theta]}), \quad d_t^{[\tau]} = (g_t, \tilde{\tau}^{[\tau]}, \tilde{\theta}^{[\tau]}), \quad d_t^{[g]} = (\tilde{g}^{[g]}, \tilde{\tau}^{[g]}, \tilde{\theta}^{[g]}). \tag{5}$$

where parameter probing replaces only θ, tool-level probing replaces τ with $\tilde{\tau}^{[\tau]}$ and regenerates $\tilde{\theta}^{[\tau]}$, and subgoal-level probing replaces g with $\tilde{g}^{[g]}$ and regenerates both $\tilde{\tau}^{[g]}$ and $\tilde{\theta}^{[g]}$.

HIVE may terminate early when an alternative satisfies the intended subgoal. Let $\mathcal{T}_t \subseteq \{\theta, \tau, g\}$ denote the explored levels, with $m_t = |\mathcal{T}_t|$. For $k = 1, \ldots, m_t$, let ℓ_k denote the k-th explored level in the order $\theta \to \tau \to g$. The corresponding sibling trajectory is $\xi_t^{[\ell_k]} = \text{Exec}(S_t, d_t^{[\ell_k]})$. The shared subscript t identifies the failed decision that anchors the probing episode, while the superscript $[\ell_k]$

identifies the decision level being probed. Thus, sibling trajectories are independent alternatives from the same state S_t. Each sibling trajectory is evaluated by the Verifier:

$$\left(Y_t^{[\ell_k]}, C_t^{[\ell_k]}\right) = A^V\left(q, S_t, d_t^{[\ell_k]}, \xi_t^{[\ell_k]}\right), \qquad Y_t^{[\ell_k]}, C_t^{[\ell_k]} \in \{0,1\}. \tag{6}$$

HIVE stops when an alternative satisfies the intended subgoal $Y_t^{[\ell_k]} = 1$ or when all three levels have been explored. The trajectories collected during the exploration episode anchored at t are

$$\Xi_t = \{\xi_t\} \cup \left\{\xi_t^{[\ell_k]} : k = 1, \ldots, m_t\right\}. \tag{7}$$

4.3 Sibling Trajectory Construction

HIVE projects sibling trajectories in §4.2 into role-specific views before extracting knowledge. For each sibling trajectory $\xi \in \Xi_t$, HIVE represents its decision transition as $\delta(\xi) = (S^\xi, g^\xi, \tau^\xi, \theta^\xi, \hat{S}^\xi)$, where S^ξ and $\hat{S}^\xi$ denote the initial and resulting states of the corresponding execution. HIVE then projects each transition into role-specific views based on the knowledge required by each role:

$$\rho^P(\delta(\xi)) = \left(S^\xi, g^\xi, \tau^\xi, \hat{S}^\xi\right), \qquad \rho^E(\delta(\xi)) = \left(S^\xi, g^\xi, \tau^\xi, \theta^\xi, \hat{S}^\xi\right). \tag{8}$$

The Planner view compares subgoal and tool selections across sibling trajectories, while the Executor view compares tool parameterizations under the corresponding subgoal and tool. In particular, parameter-level siblings share the same g and τ, making θ the varying decision in their comparison. These role-specific views isolate the information relevant to updating the corresponding decision knowledge. Applying these projections to the trajectories collected around the failed decision gives

$$\Xi_t^P = \left\{\rho^P(\delta(\xi)) : \xi \in \Xi_t\right\}, \qquad \Xi_t^E = \left\{\rho^E(\delta(\xi)) : \xi \in \Xi_t\right\}. \tag{9}$$

4.4 Online Knowledge Updating

HIVE compares the role-specific views of sibling trajectories to identify conditions that distinguish successful and failed decisions under the same system state. These comparisons provide analysis for reasoning about why current knowledge fails to support the decision and what information is missing.

For each role $i \in \{P, E\}$, HIVE analyzes the corresponding sibling views. When successful and failed trajectories coexist, it extracts conditions that distinguish their outcomes. When all sibling trajectories fail, it instead extracts conditions consistently shared by the failed trajectories, providing evidence about conditions under which the current information remains insufficient:

$$\mathcal{E}_t^i = \begin{cases} \mathrm{Diff}\left(\Xi_{t,+}^i, \Xi_{t,-}^i\right), & \Xi_{t,+}^i \neq \varnothing, \\ \mathrm{Common}\left(\Xi_{t,-}^i\right), & \Xi_{t,+}^i = \varnothing, \end{cases} \qquad i \in \{P, E\}. \tag{10}$$

where $\Xi_{t,+}^i$ and $\Xi_{t,-}^i$ denote the successful and failed role-specific views collected during the hierarchical sibling probing episode anchored at the failed decision at step t. The resulting analysis $\mathcal{E}_t^i$ captures decision-relevant conditions revealed by these comparisons.

HIVE then reconciles the extracted analysis with role-specific knowledge and retains the explored sibling trajectories as reusable demonstrations:

$$\left(K_{t+m_t+1}^P, K_{t+m_t+1}^E\right) = \mathcal{U}\left(K_t^P, K_t^E, \{\mathcal{E}_t^i\}_{i \in \{P,E\}}\right). \tag{11}$$

The updating operation $\mathcal{U}$ adds missing knowledge, strengthens existing knowledge supported by consistent evidence, and refines conflicting knowledge to better align with the extracted evidence.

4.5 Integration with the MAS Iteration Loop

After the online knowledge update, the updated $K_{t+m_t+1}^P$ and $K_{t+m_t+1}^E$ guide the next Planner and Executor decisions. Thus, an HIVE episode anchors sibling probing at the failed decision at step t, executes each explored alternative as an independent MAS iteration from the same state S_t at steps $t{+}1, \ldots, t{+}m_t$, updates the role-specific knowledge, and then resumes the normal MAS loop at step $t + m_t + 1$. This closes the adaptation loop by converting the failed decision and sibling trajectories into role-specific knowledge online within the current task.

5 EXPERIMENT

5.1 EXPERIMENTAL SETUP

Datasets. We evaluate HIVE on six benchmarks covering multi-hop question answering (**Bamboogle** (Press et al., 2023), **2Wiki** (Ho et al., 2020), **HotpotQA** (Yang et al., 2018), and **Musique** (Trivedi et al., 2022)), agentic tasks (**GAIA** (Mialon et al., 2023)), and medical reasoning (**MedQA** (Yang et al., 2025a)). They cover multi-step reasoning, information retrieval, tool use, and domain knowledge, providing diverse settings for evaluating LLM-based MAS performance. Detailed dataset descriptions and statistics are provided in Appendix B.1.

Baselines. We compare HIVE with four representative approaches. **Clear MAS**(Li et al., 2026b) provides workflow execution through AgentFlow's MAS framework, without its RL algorithm. **EPC-AW**(Wang et al., 2026) calibrates epistemic support for decision selection. **SupervisorAgent**(Lin et al., 2026) uses external supervision for intervention. **StructAgent**(Wu et al., 2026) grounds execution in structured states and verified information. HIVE instead acquires decision-relevant knowledge through hierarchical sibling probing. All methods use the same task instances, model, environment, and evaluation protocol. Implementation details are in Appendix B.2.

Implementation Details. All agents in HIVE, including the Planner, Executor, and Verifier, use Qwen3.5-27B (Qwen Team, 2026a) and are served with `vLLM` on two NVIDIA A800 GPUs. Unless otherwise specified, the generation temperature is set to 0. For EPC-AW, the Planner samples $n = 9$ candidate next-step plans with temperature 0.9, following its original configuration. For SupervisorAgent, we add its supervisory agent to the shared Planner-Executor-Verifier execution environment while preserving its original intervention mechanism. All methods are limited to a maximum trajectory length of 10 steps, interact with the same five tools, and use GPT-4o as the automatic evaluator, following prior work (Wang et al., 2026; Li et al., 2026b). We report the mean accuracy over three independent runs. Further implementation details are provided in Appendix B.3.

Table 1: Accuracy (%) comparison of different approaches on six benchmarks with Qwen3.5-27B. Results are averaged over three independent runs. The best result is highlighted in **bold**. Δ denotes the absolute improvement of HIVE over the best-performing baseline on each benchmark.

Method	Bamboogle	2Wiki	HotpotQA	Musique	GAIA	MedQA	Avg.
Clear MAS	64.27	55.83	54.67	15.00	18.11	74.88	47.13
SupervisorAgent	72.27	66.00	<u>65.00</u>	<u>19.50</u>	23.09	84.89	55.12
EPC-AW	72.00	67.67	63.67	19.00	<u>25.45</u>	<u>88.00</u>	<u>55.97</u>
StructAgent	<u>73.87</u>	<u>68.67</u>	62.67	18.50	24.14	86.66	55.75
HIVE	**75.73**	**72.17**	**70.67**	**23.67**	**30.44**	**89.89**	**60.43**
Δ	+1.86	+3.50	+5.67	+4.17	+4.99	+1.89	+4.46

5.2 MAIN RESULTS

Table 1 compares HIVE with three complementary approaches: external supervision in SupervisorAgent, plan calibration in EPC-AW, and state- and information-aware execution in StructAgent. HIVE achieves the best accuracy on all six benchmarks, improving the task accuracy from 55.97% to 60.43% on average, with gains of 1.86-5.67 points.

The baselines show distinct dataset-specific strengths. StructAgent performs relatively well on Bamboogle and 2Wiki, consistent with its emphasis on retaining verified information, while EPC-AW is strongest on GAIA and MedQA, where conservative plan calibration is particularly useful. SupervisorAgent leads the baselines on HotpotQA and Musique. HIVE, however, consistently outperforms all three, with especially large gains on HotpotQA ($+5.67$), GAIA ($+4.99$), and Musique ($+4.17$). This pattern indicates that HIVE complements information retention and plan calibration by directly probing decision-specific gaps and refining the responsible parameter, tool, or subgoal.

Table 2: Ablation study of hierarchical sibling probing and online knowledge updating. R"w/" denotes adding the corresponding component to the Clear MAS.

Variant	Bamboogle	2Wiki	HotpotQA	Musique	GAIA	MedQA
Clear MAS	64.27	55.83	54.67	15.00	18.11	74.88
w/ Hierarchical Sibling Probing	68.80	67.83	66.67	18.00	24.41	88.00
w/ Update Planner Knowledge	72.27	69.17	69.33	19.17	28.09	88.33
w/ Update Executor Knowledge	73.33	71.00	68.00	21.33	28.35	89.55
HIVE	**75.73**	**72.17**	**70.67**	**23.67**	**30.44**	**89.89**

5.3 Ablation Study

The ablation results demonstrate that hierarchical sibling probing and online knowledge updating provide clear and complementary benefits. Hierarchical sibling probing alone improves Clear MAS by 3.00-13.12 points across all six benchmarks, with particularly large gains on 2Wiki, HotpotQA, and MedQA. By actively exploring alternative decisions, probing obtains additional decision-relevant information that is unavailable from the original failed trajectory.

Online knowledge updating further improves the probing-only variant on every benchmark, with executor-side knowledge generally providing larger gains than planner-side knowledge. The additional gain from executor-side knowledge reaches 3.33 points on Musique and 3.94 points on GAIA, while probing yields gains of up to 12.00 points on Bamboogle, 2Wiki, and HotpotQA and 13.12 points on MedQA. These results show that sibling exploration can acquire useful information beyond retrospective feedback, while role-specialized knowledge helps subsequent agents avoid redundant errors and make better-informed decisions along the task trajectory.

5.4 Replacing Hierarchical Sibling Probing with LLM-based Attribution

We compare HIVE with a controlled variant that replaces hierarchical sibling probing with LLM-based attribution, while retaining the same sibling-trajectory construction and knowledge update. Given an iteration failure, an LLM directly attributes the failure to one decision level: tool parameters, the selected tool, or the subgoal. This attributed level determines the repair level. HIVE achieves higher accuracy than this variant across all benchmarks with available results.

The gap stems from how the repair level is determined. LLM-based attribution relies on a single LLM judgment to identify the level, so an incorrect attribution may direct repair to an inappropriate level. In contrast, HIVE replaces this judgment with hierarchical sibling probing, progressively testing alternative decisions and using their execution outcomes to determine the repair level.

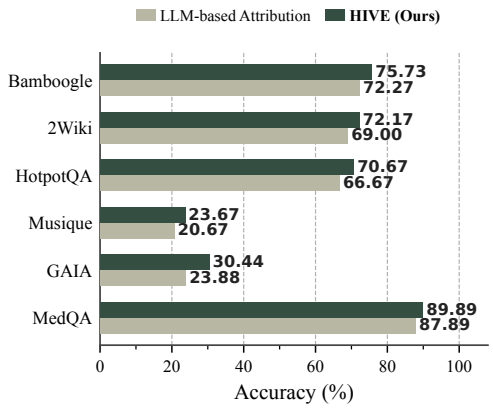

Figure 3: Comparison of LLM-based attribution and hierarchical sibling probing for decision repair.

5.5 Generalization across LLMs

To examine whether the effectiveness of HIVE depends on a particular backbone, we evaluate all methods with two additional LLMs, DeepSeek-v4-flash (Xu et al., 2026) and Qwen3.7-flash (Qwen Team, 2026b). As shown in Fig. 4, HIVE consistently achieves the best performance across both backbones and all six benchmarks. The improvements are particularly pronounced on reasoning-intensive tasks such as HotpotQA and GAIA, indicating that the benefit of hierarchical probing is not tied to a specific backbone's reasoning capability. Instead, HIVE provides a model-independent mechanism for acquiring and organizing decision-relevant evidence, making it complementary to the underlying LLM. Detailed numerical results are reported in Appendix B.4.

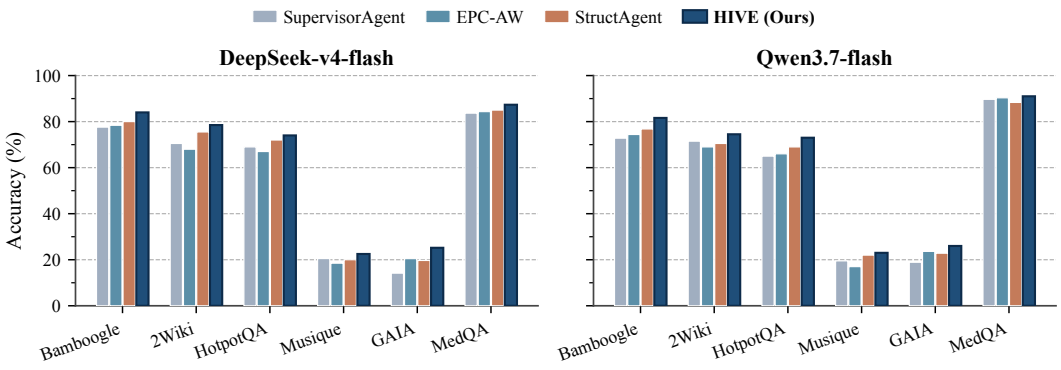

Figure 4: Generalization across LLM backbones. Accuracy (%) of HIVE and baselines on six benchmarks under DeepSeek-v4-flash (left) and Qwen3.7-flash (right).

5.6 Efficiency Analysis

HIVE maintains the same LLM-call budget as Clear MAS while substantially shortening execution trajectories. Parameter-level probing bypasses the Planner and directly reuses the current tool, while tool- and subgoal-level probing replace the corresponding Planner decisions. After probing terminates, one Verifier call assesses the probed outcomes and updates the online knowledge. Thus, HIVE redistributes inference calls across decision exploration and verification rather than increasing the total LLM-call budget.

Moreover, HIVE requires only 4.45 iterations per task on average, compared with 5.83-6.60 for the baselines, corresponding to a 23.7%-32.6% reduction. Two mechanisms account for this reduction. First, sibling probing explores alternative parameter, tool, or subgoal decisions when the current decision is insufficiently supported, allowing HIVE to move into a different decision space rather than repeatedly following an unproductive path. Second, HIVE suppresses redundant execution loops (Table 7), defined as two consecutive steps with the same subgoal and tool, highly similar parameters, and no target information obtained from the latter step, and measured as the proportion of trajectories containing at least one such loop. HIVE's redundant-loop rate is only 2.7%, compared with 26.3%-29.1% for the baselines, corresponding to a 90.3% reduction relative to their mean. Together, targeted decision-space exploration and redundant-loop suppression reduce recovery steps while maintaining higher task success. Detailed results and analysis are provided in Appendix C.3.

Table 3: Average trajectory length of different approaches on six benchmarks with Qwen3.5-27B. Lower values indicate fewer execution steps.

Method	Bamboogle	2Wiki	HotpotQA	Musique	GAIA	MedQA	Avg.
SupervisorAgent	5.66	5.84	6.46	6.92	8.34	6.35	6.60
EPC-AW	5.59	5.92	6.11	7.20	7.54	6.28	6.44
StructAgent	4.06	5.13	4.48	6.50	8.38	6.44	5.83
HIVE	**3.26**	**4.63**	**3.54**	**5.36**	**6.56**	**3.33**	**4.45**

6 Conclusion

We introduced HIVE, a framework for decision-level epistemic exploration in LLM-based multi-agent systems. HIVE addresses iteration failure ambiguity by hierarchically probing sibling parameter, tool, and subgoal decisions, verifying their outcomes in the environment, and retaining the resulting knowledge for subsequent decisions. This structured exploration enables agents to identify and acquire knowledge at the decision level where support is missing. Across six benchmarks, HIVE improves task success by 4.46 percentage points over the compared baselines while reducing average trajectory length by 23.67%. These results suggest that MAS adaptation is not only about repairing failed decisions, but also about acquiring decision knowledge for better subsequent decisions.

AI USE STATEMENT

In this work, we used generative AI tools to assist with manuscript language polishing and editing, as well as to assist with code modification and debugging. We did not use generative AI tools to generate or determine the core research ideas, methodology, experimental design, experimental results, or scientific claims of this work. We have reviewed and verified all AI-assisted content. In particular, AI-assisted code modifications were manually inspected and tested by the authors for correctness, and all AI-assisted textual revisions were reviewed by the authors against the intended technical content. We take full responsibility for the final content of this work, including all text, claims, code, and other artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not involve human subjects, personally identifiable information, or the collection of new sensitive data. The experiments use existing benchmark datasets and publicly available language models. We are not aware of any additional ethical concerns specific to this work.

REPRODUCIBILITY STATEMENT

We provide the implementation and experimental code as supplementary material to facilitate the reproduction of our results. The main paper describes the problem formulation and the HIVE framework, while the appendix provides additional algorithmic details, experimental settings, and ablation configurations. We also report the datasets, baselines, evaluation protocol, and implementation details used in our experiments. The released code contains the necessary components for reproducing the reported experiments, subject to the availability of the underlying language models and external services used by the benchmark environments.

REFERENCES

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

Hyeong Kyu Choi, Jiatong Li, Wendi Li, Xin Eric Wang, and Sharon Li. Multi-agent llms fail to explore each other. *arXiv preprint arXiv:2607.11250*, 2026.

Will Epperson, Gagan Bansal, Victor C Dibia, Adam Fourney, Jack Gerrits, Erkang Zhu, and Saleema Amershi. Interactive debugging and steering of multi-agent ai systems. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pp. 1–15, 2025.

Lewis Hammond, Alan Chan, Jesse Clifton, Jason Hoelscher-Obermaier, Akbir Khan, Euan McLean, Chandler Smith, Wolfram Barfuss, Jakob Foerster, Tomáš Gavenčiak, et al. Multi-agent risks from advanced ai. *arXiv preprint arXiv:2502.14143*, 2025.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, 2020.

WANG Hongru, Cheng Qian, Manling Li, Jiahao Qiu, XUE Boyang, Mengdi Wang, Heng Ji, Amos Storkey, and Kam-Fai Wong. Position: Agents should invoke external tools only when epistemically necessary. In *Forty-third International Conference on Machine Learning Position Paper Track*.

Yulun Jiang, Liangze Jiang, Damien Teney, Michael Moor, and Maria Brbic. Meta-rl induces exploration in language agents. In *International Conference on Learning Representations*, volume 2026, pp. 3279–3303, 2026.

Remy Kim, Minseung Lee, Shuo Li, and Osbert Bastani. Seekergym: A benchmark for reliable information seeking. *arXiv preprint arXiv:2604.17143*, 2026a.

Takyoung Kim, Jinseok Nam, Chandrayee Basu, Xing Fan, Chengyuan Ma, Heng Ji, Gokhan Tur, and Dilek Hakkani-Tür. Rein: Conversational error recovery with reasoning inception. In *The Fourteenth International Conference on Learning Representations*, 2026b.

Ka Yiu Lee, Yuxuan Huang, Zhiyuan He, Huichi Zhou, Weilin Luo, Kun Shao, Meng Fang, and Jun Wang. Infoseeker: A scalable hierarchical parallel agent framework for web information seeking. *arXiv preprint arXiv:2604.02971*, 2026.

Yuyang Li, Yiran Dou, Jie-Jing Shao, Yueming Lyu, Ivor Tsang, and Haiyan Yin. Skilltracer: Structural failure attribution and refinement of agentic skills in long-horizon web tasks. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 2897–2908, 2026a.

Zhuofeng Li, Haoxiang Zhang, Seungju Han, Sheng Liu, Jianwen Xie, Yu Zhang, Yejin Choi, James Y Zou, and Pan Lu. In-the-flow agentic system optimization for effective planning and tool use. In *International Conference on Learning Representations*, volume 2026, pp. 50524–50570, 2026b.

Churong Liang, Jinling Gan, Kairan Hong, Qiushi Tian, Zongze Wu, and Runnan Li. Coco: Cognitive operating system with continuous oversight for multi-agent workflow reliability. *arXiv preprint arXiv:2508.13815*, 2025.

Fulin Lin, Ruishan Fang, Hongwei Wang, Tao Lin, et al. Stop wasting your tokens: Towards efficient runtime multi-agent systems. In *International Conference on Learning Representations*, volume 2026, pp. 113497–113528, 2026.

Yi-Cheng Lin, Kang-Chieh Chen, Zhe-Yan Li, Tzu-Heng Wu, Tzu-Hsuan Wu, Kuan-Yu Chen, Hung-yi Lee, and Yun-Nung Chen. Creativity in llm-based multi-agent systems: A survey. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 27572–27595, 2025.

Ming Ma, Jue Zhang, Fangkai Yang, Yu Kang, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. Dover: Intervention-driven auto debugging for llm multi-agent systems. In *International Conference on Learning Representations*, volume 2026, pp. 28059–28087, 2026.

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*, 2023.

Judea Pearl. An introduction to causal inference. *The international journal of biostatistics*, 6(2):7, 2010.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, 2023.

Shihao Qi, Jie Ma, Rui Xing, Wei Guo, Xiao Huang, Zhitao Gao, Jianhao Deng, Jun Liu, Lingling Zhang, Bifan Wei, et al. Beyond individual intelligence: Surveying collaboration, failure attribution, and self-evolution in llm-based multi-agent systems. *arXiv preprint arXiv:2605.14892*, 2026.

Dan Qiao, Binbin Chen, Fengyu Cai, Jianlong Chen, Wenhao Li, Fuxin Jiang, Zuzhi Chen, Hongyuan Zha, Tieying Zhang, and Baoxiang Wang. Epistemic gain, aleatoric cost: Uncertainty decomposition in multi-agent debate for math reasoning. In *Forty-third International Conference on Machine Learning*, 2026.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026a. URL `https://qwen.ai/blog?id=qwen3.5`.

Qwen Team. Qwen3.7: The agent frontier, May 2026b. URL `https://qwen.ai/blog?id=qwen3.7`.

Stephan Rabanser, Sayash Kapoor, Peter Kirgis, Kangheng Liu, Saiteja Utpala, and Arvind Narayanan. Towards a science of ai agent reliability. *International Conference on Machine Learning*, 2026.

Valentin Rodionov and Shamil Assylbekov. Traces: A benchmark for epistemic reliability in scientific reasoning by llms. *arXiv preprint arXiv:2608.11415*, 2026.

Krishna Ronanki. Facilitating trustworthy human-agent collaboration in llm-based multi-agent system oriented software engineering. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering*, pp. 1333–1337, 2025.

Jiaqi Shao, Yuxiang Lin, Munish P Lohani, Yufeng Miao, and Bing Luo. Do llm agents know how to ground, recover, and assess? evaluating epistemic competence in information-seeking agents. In *International Conference on Learning Representations*, volume 2026, pp. 152647–152679, 2026.

Xu Shen, Qi Zhang, Song Wang, Zhen Tan, Xinyu Zhao, Laura Yao, Vaishnav Tadiparthi, Hossein Nourkhiz Mahjoub, Ehsan Moradi Pari, Kwonjoon Lee, et al. Metacognitive self-correction for multi-agent system via prototype-guided next-execution reconstruction. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 23320–23337, 2026.

Andrew Szot, Michael Kirchhof, Omar Attia, and Alexander Toshev. Expanding llm agent boundaries with strategy-guided exploration. *arXiv preprint arXiv:2603.02045*, 2026.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Zehao Wang, Shilong Jin, Zhao Cao, and Lanjun Wang. When planning fails despite correct execution: On epistemic calibration for llm-based multi-agent systems. In *Forty-third International Conference on Machine Learning*, 2026.

Earl Woodruff and Jim Hewitt. Epistemic agency in the age of large language models: Design principles for knowledge-building ai. *AI*, 7(3):99, 2026.

Wenyi Wu, Sibo Zhu, Kun Zhou, Aayush Salvi, Zixuan Song, and Biwei Huang. Structagent: Harness long-horizon digital agents with unified causal structure. *arXiv preprint arXiv:2607.11388*, 2026.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

Yasin A Yadkori, Ilja Kuzborskij, András György, and Csaba Szepesvári. To believe or not to believe your llm: Iterative prompting for estimating epistemic uncertainty. *Advances in Neural Information Processing Systems*, 37:58077–58117, 2024.

Bingyu Yan, Zhibo Zhou, Litian Zhang, Lian Zhang, Ziyi Zhou, Dezhuang Miao, Zhoujun Li, Chaozhuo Li, and Xiaoming Zhang. Beyond self-talk: A communication-centric survey of llm-based multi-agent systems. *Front. Comput. Sci.*, 2025.

Hang Yang, Hao Chen, Hui Guo, Yineng Chen, Ching-Sheng Lin, Shu Hu, Jinrong Hu, Xi Wu, and Xin Wang. Llm-medqa: Enhancing medical question answering through case studies in large language models. In *2025 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2025a.

Yingxuan Yang, Huacan Chai, Shuai Shao, Yuanyi Song, Siyuan Qi, Renting Rui, and Weinan Zhang. Agentnet: Decentralized evolutionary coordination for llm-based multi-agent systems. *Advances in Neural Information Processing Systems*, 38:107309–107336, 2026.

Yongjin Yang, Sinjae Kang, Juyong Lee, Dongjun Lee, Se-Young Yun, and Kimin Lee. Automated skill discovery for language agents through exploration and iterative feedback. *arXiv preprint arXiv:2506.04287*, 2025b.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2369–2380, 2018.

Guibin Zhang, Junhao Wang, Junjie Chen, Wangchunshu Zhou, Kun Wang, and Shuicheng Yan. Agentracer: Who is inducing failure in the llm agentic systems? In *International Conference on Learning Representations*, volume 2026, pp. 11377–11399, 2026.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. Which agent causes task failures and when? On automated failure attribution of LLM multi-agent systems. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 76583–76599. PMLR, 2025.

Caishen Zhou, Yihong Tang, Kehai Chen, Xuefeng Bai, Shuhan Qi, Li Shen, and Min Zhang. From fragmentation to systematic design: Architecting llm-based multi-agent systems. 2026a.

Ruiwen Zhou, Maojia Song, Xiaobao Wu, Sitao Cheng, Xunjian Yin, Yuxi Xie, Zhuoqun Hao, Wenyue Hua, Liangming Pan, Soujanya Poria, et al. Epistemic context learning: Building trust the right way in llm-based multi-agent systems. *arXiv preprint arXiv:2601.21742*, 2026b.

A Detailed HIVE Algorithm

This section provides the operational details of HIVE. Algorithm 1 summarizes the complete procedure, including hierarchical sibling probing, trajectory construction, and knowledge updating.

Algorithm 1 HIVE: Hierarchical Sibling Probing and Online Knowledge Updating

Require: Query q, failed state S_t, failed decision $d_t = (g_t, \tau_t, \theta_t)$, failed trajectory ξ_t
Require: Role-specific knowledge (K_t^P, K_t^E) and memory M_t
Ensure: Updated knowledge $(K_{t+m_t+1}^P, K_{t+m_t+1}^E)$

Initialize $\Xi_t \leftarrow \{\xi_t\}$ and $m_t \leftarrow 0$
for $\ell \in \{\theta, \tau, g\}$ **do**
  $m_t \leftarrow m_t + 1$
  if $\ell = \theta$ **then**
    Generate an alternative parameter $\tilde{\theta}^{[\theta]}$ with (g_t, τ_t) fixed
    $d_t^{[\theta]} \leftarrow (g_t, \tau_t, \tilde{\theta}^{[\theta]})$
  else if $\ell = \tau$ **then**
    Generate an alternative tool $\tilde{\tau}^{[\tau]}$ for g_t
    $\tilde{\theta}^{[\tau]} \leftarrow A^E(S_t, g_t, \tilde{\tau}^{[\tau]})$
    $d_t^{[\tau]} \leftarrow (g_t, \tilde{\tau}^{[\tau]}, \tilde{\theta}^{[\tau]})$
  else
    Generate an alternative subgoal $\tilde{g}^{[g]}$
    $(\tilde{\tau}^{[g]}, \tilde{\theta}^{[g]}) \leftarrow \mathrm{RegenerateDownstream}(S_t, \tilde{g}^{[g]})$
    $d_t^{[g]} \leftarrow (\tilde{g}^{[g]}, \tilde{\tau}^{[g]}, \tilde{\theta}^{[g]})$
  end if
  $\xi_t^{[\ell]} \leftarrow \mathrm{Exec}(S_t, d_t^{[\ell]})$
  $(Y_t^{[\ell]}, C_t^{[\ell]}) \leftarrow A^V(q, S_t, d_t^{[\ell]}, \xi_t^{[\ell]})$
  $\Xi_t \leftarrow \Xi_t \cup \{\xi_t^{[\ell]}\}$
  if $Y_t^{[\ell]} = 1$ **then**
    break
  end if
end for
Construct role-specific views from Ξ_t
Reflect on the resulting views to obtain $\mathcal{E}_t^P$ and $\mathcal{E}_t^E$
$(K_{t+m_t+1}^P, K_{t+m_t+1}^E) \leftarrow \mathcal{U}(K_t^P, K_t^E, \{\mathcal{E}_t^i\}_{i\in\{P,E\}})$
$M_{t+m_t+1} \leftarrow \mathrm{MemUpdate}(M_t, \Xi_t)$
return $(K_{t+m_t+1}^P, K_{t+m_t+1}^E)$

HIVE is triggered when the Verifier identifies an iteration failure, $Y_t = 0$. Starting from the state S_t and decision $d_t = (g_t, \tau_t, \theta_t)$, HIVE probes alternative decisions in the order $\theta \to \tau \to g$. Each

sibling probing is initialized from S_t, executed independently, and evaluated by the Verifier. Probing stops when a sibling trajectory satisfies its subgoal or when all three levels have been explored.

Hierarchical Sibling probing. At the parameter level, HIVE keeps (g_t, τ_t) fixed and generates an alternative parameter, bypassing the Planner. At the tool level, it replaces τ_t and regenerates θ with the Executor. At the subgoal level, it replaces g_t and regenerates both downstream decisions. All sibling probings are independently initialized from the common failed state S_t.

Reflection and update. After probing, HIVE projects the sibling trajectories into the role-specific views defined in the main text. Reflection compares role-specific views from successful and failed trajectories when both are available and otherwise extracts conditions shared by the failed views. The resulting evidence is reconciled with the existing role-specific knowledge through $\mathcal{U}$, while the explored trajectories are retained as reusable demonstrations.

Execution accounting. If m_t sibling trajectories are explored, they occupy m_t MAS iterations for accounting purposes. These iterations are independently initialized from S_t rather than forming a sequential execution chain. The updated knowledge is therefore used at the next normal execution step $t + m_t + 1$. The dedicated knowledge-update operation is not counted as an additional MAS execution iteration.

B Experimental Details

B.1 Dataset Details

We evaluate HIVE on six benchmarks covering different combinations of multi-step reasoning, external information retrieval, tool interaction, and domain-specific knowledge. Table 4 summarizes the evaluation subsets used in our experiments. We retain the evaluation subsets adopted by Agent-Flow (Li et al., 2026b) and EPC-AW (Wang et al., 2026) so that all methods are evaluated on exactly the same task instances.

Bamboogle (Press et al., 2023) is a manually constructed multi-step reasoning benchmark designed to evaluate compositional inference over interconnected facts. Its questions typically require several sequential inference steps, making intermediate decisions important for obtaining the final answer.

2Wiki (Ho et al., 2020) combines structured knowledge from Wikidata with textual evidence from Wikipedia. Solving an instance requires retrieving and integrating information across heterogeneous sources, with intermediate retrieval results often affecting subsequent reasoning steps.

HotpotQA (Yang et al., 2018) is a multi-hop question answering benchmark constructed from Wikipedia articles. Questions require the system to identify relevant evidence from multiple documents and combine the retrieved information to reach the final answer.

Musique (Trivedi et al., 2022) focuses on multi-step reasoning with tightly connected inference chains. Later reasoning steps depend on information obtained or inferred at earlier stages, making the benchmark suitable for evaluating decision reliability under sequential dependencies.

GAIA (Mialon et al., 2023) evaluates general-purpose agents on open-world tasks that involve multi-step reasoning, web search, navigation, and tool use. Compared with conventional question answering benchmarks, GAIA requires agents to acquire and process information from external environments as part of task solving.

MedQA (Yang et al., 2025a) contains multiple-choice questions derived from professional medical licensing examinations. The benchmark primarily evaluates domain-specific reasoning grounded in medical knowledge and therefore provides a setting with substantially less dependence on external web retrieval.

Collectively, these benchmarks provide diverse environments in which agents may rely on internal reasoning, external evidence, tool interaction, or domain knowledge. This diversity allows us to examine whether HIVE can improve decision making when the information available to the role-specialized agents is insufficient for a successful execution.

Table 4: Evaluation subsets used in our experiments. The same task instances are fixed across all compared methods.

Dataset	Bamboogle	2Wiki	HotpotQA	Musique	GAIA	MedQA
# Samples	125	200	100	200	127	300

B.2 Baseline Details

We compare HIVE with four representative approaches that address multi-agent planning, runtime supervision, epistemic calibration, or structured execution. Each baseline is characterized below by its original architectural mechanism and the primary source of its decision-making or intervention capability.

Clear MAS[1] (Li et al., 2026b) follows the role-specialized agentic workflow introduced by AgentFlow, which decomposes execution into Planner, Executor, Verifier, and Generator modules connected through evolving system memory. The Planner determines the next step, the Executor carries out the selected operation, and the Verifier evaluates the resulting outcome, while the Generator supports information aggregation and subsequent decision making. The original framework further introduces Flow-GRPO for in-the-flow optimization of the Planner; this learning component is excluded from the Clear MAS configuration considered here.

EPC-AW[2] (Wang et al., 2026) addresses epistemic miscalibration in planning through heterogeneous information states across role-specialized agents. Its Information-consistency-based Plan Selection generates multiple candidate plans and evaluates their support under different agent information conditions, favoring plans whose evaluations remain stable across these conditions. Its Consistency-guided Epistemic State Refinement further accumulates discrepancies between locally preferred and information-consistent plans into persistent epistemic constraints that guide subsequent planning.

SupervisorAgent[3] (Lin et al., 2026) introduces a lightweight runtime supervisor that operates alongside an existing MAS without modifying its core agent architecture. An LLM-free adaptive filter first identifies high-risk interactions, after which the supervisor constructs a context from task goals, agent traces, and interaction summaries and selects an intervention from a graded action space, including approval, guidance, observation correction, and explicit verification. The design therefore focuses on selective runtime intervention at critical interactions rather than continuously supervising every execution step.

StructAgent[4] (Wu et al., 2026) organizes long-horizon execution around a unified, evidence-backed task state shared by a Planner, Actor, and Verifier. The Planner selects the next subgoal from structured task progress, the Actor performs the corresponding interaction, and the Verifier determines whether sufficient evidence supports a state transition. This verifier-centered state representation enables explicit progress checkpointing, evidence-grounded completion, failure attribution, and targeted recovery when execution does not satisfy the current task state.

B.3 Implementation Details

Model and Inference. All experiments use Qwen3.5-27B (Qwen Team, 2026a) as the underlying language model. The model is served with `vLLM` on a server equipped with two NVIDIA A800 GPUs. The same model backbone is used for HIVE and all baselines. HIVE uses three role-specialized agents: a Planner, an Executor, and a Verifier. The Planner selects subgoals and tools, the Executor determines the parameters for the selected tool, and the Verifier evaluates whether the resulting state satisfies the intended subgoal. This role assignment is fixed across all experiments.

Generation Settings. All methods are limited to a maximum trajectory length of 10 steps. EPC-AW follows its original sampling configuration, in which the Planner samples $n = 9$ candidate next-step

[1] https://github.com/lupantech/AgentFlow
[2] https://github.com/wzhSteve/EPC-AW
[3] https://github.com/LINs-lab/SupervisorAgent
[4] https://github.com/WenyiWU0111/StructAgent

plans with temperature 0.9 and selects the candidate with the strongest estimated epistemic support. For HIVE and other components using single-sample generation, temperature 0 is used for decision generation, execution, verification, and knowledge updates. Decoding settings are otherwise kept consistent across corresponding components when applicable.

Repeated Runs. Each experiment is run independently three times (Li et al., 2026b; Wang et al., 2026). The same evaluation instances are used across the three runs and across all methods. We report the mean performance over the three independent runs.

Evaluation. We use GPT-4o as an automatic evaluator to determine whether the final agent prediction matches the ground-truth answer (Li et al., 2026b; Wang et al., 2026). The same evaluator and evaluation procedure are applied to all methods. For each benchmark, we report the mean task-level success rate over the three independent runs.

Tool Configuration. All methods use the same execution interface and have access to the same tools (Li et al., 2026b; Wang et al., 2026). Tool outputs are returned to agents as environment observations and can be used by subsequent decisions according to each method's architecture. HIVE's probing operations invoke the same underlying tools as the standard execution process and do not introduce additional information sources. All methods have access to the same five tools:

- **Base Generator**: provides general-purpose text generation and reasoning for tasks that do not require a dedicated external tool.
- **Python Coder**: generates and executes Python code and returns the execution results to the agent. It is used for computational tasks and programmatic information processing.
- **Google Search**: retrieves relevant web pages in response to a search query.
- **Wikipedia Search**: retrieves information from Wikipedia for tasks involving encyclopedic knowledge.
- **Web Search**: processes retrieved web pages and extracts or summarizes information relevant to the current task.

Google Search Pipeline. To reduce potential interference from language models embedded in commercial search APIs, we implement Google Search through a local browser-based retrieval pipeline. For each search query, a local Chrome-based interface retrieves the top-5 returned web pages. The retrieved pages are then passed to the Web Search component for page-level processing.

Because the evaluated datasets and their solutions may be publicly available online, we apply keyword-based filtering during retrieval to reduce direct exposure to dataset-specific resources. In particular, pages corresponding to HuggingFace repositories and dataset-specific pages are excluded from the retrieved results. After filtering, retained web pages are processed by the Web Search tool and summarized using the underlying language model. The same retrieval and filtering procedure is applied to all methods throughout evaluation.

B.4 Detailed Results for Generalization across LLMs

Table 5 reports the exact numerical results corresponding to Fig. 4. HIVE consistently achieves the best accuracy across both DeepSeek-v4-flash and Qwen3.7-flash on all six benchmarks. On DeepSeek-v4-flash, HIVE improves over the compared baselines by 4.00, 3.00, 2.00, 2.00, 4.72, and 2.33 percentage points on Bamboogle, 2Wiki, HotpotQA, Musique, GAIA, and MedQA, respectively. The average improvement is 3.01 percentage points across the six benchmarks. On Qwen3.7-flash, the corresponding improvements are 4.80, 3.00, 4.00, 1.00, 2.36, and 0.67 percentage points, yielding an average improvement of 2.64 percentage points. Thus, HIVE provides consistent gains across different LLM backbones rather than relying on a specific model's behavior.

The improvements are also distributed across the benchmarks with different task characteristics. For DeepSeek-v4-flash, HIVE reaches 84.00% on Bamboogle and 78.50% on 2Wiki, exceeding the compared baselines by 4.00 and 3.00 points, respectively. On HotpotQA and Musique, HIVE achieves 74.00% and 22.50%, improving by 2.00 points on both benchmarks. The largest gain is observed on GAIA, where HIVE improves from 20.47% to 25.19%, a 4.72-point increase. Similarly, with Qwen3.7-flash, HIVE improves over the best-performing baseline on every benchmark, with gains ranging from 0.67 points on MedQA to 4.80 points on Bamboogle. Notably, HIVE maintains

Table 5: Detailed accuracy (%) results for generalization across different LLM backbones. The best result for each benchmark under each backbone is shown in **bold**, while the best-performing baseline is underlined.

Backbone	Method	Bamboogle	2Wiki	HotpotQA	Musique	GAIA	MedQA
DeepSeek-v4-flash	SupervisorAgent	77.60	70.50	69.00	20.50	14.17	83.67
	EPC-AW	78.40	68.00	67.00	18.50	20.47	84.33
	StructAgent	80.00	75.50	72.00	20.00	19.68	85.00
	HIVE	**84.00**	**78.50**	**74.00**	**22.50**	**25.19**	**87.33**
Qwen3.7-flash	SupervisorAgent	72.80	71.50	65.00	19.50	18.90	89.67
	EPC-AW	74.40	69.00	66.00	17.00	23.62	90.33
	StructAgent	76.80	70.50	69.00	22.00	22.83	88.33
	HIVE	**81.60**	**74.50**	**73.00**	**23.00**	**25.98**	**91.00**

its advantage on both reasoning-oriented benchmarks such as Bamboogle and HotpotQA and more challenging settings such as GAIA, indicating that its decision-level intervention is not tied to a particular LLM or benchmark type.

C Additional Efficiency Analysis

This appendix provides additional analysis of the computational and execution efficiency of HIVE from three complementary perspectives: LLM-call overhead, average trajectory length, and redundant execution loops. All methods use the same underlying LLM, tool configurations, and execution environment. The analysis therefore separates inference overhead from the number and quality of environment-facing execution steps.

C.1 LLM Call Accounting

HIVE does not increase the number of LLM calls compared with Clear MAS. Its hierarchical sibling probing changes how inference calls are allocated across the Planner, Executor, and Verifier, but does not introduce an additional Planner-Executor inference round.

At the parameter level, HIVE reuses the tool selected by the Planner and directly probes an alternative parameterization with the Executor. Therefore, parameter probing does not require another Planner call. At the tool and subgoal levels, HIVE replaces the current decision with an alternative decision and executes the resulting alternative within the same execution process. After probing, the Verifier assesses the probed outcome and updates the stored knowledge. Thus, probing changes the allocation of inference calls across roles rather than adding an additional LLM-call round.

The corresponding accounting is summarized in Table 6. HIVE therefore reallocates one LLM call from the Planner to the Verifier rather than increasing the overall LLM-call budget.

Table 6: LLM-call accounting for HIVE after an iteration failure. For parameter-level probing, bypassing the Planner saves one LLM call, which is reallocated to Verifier-based knowledge updating. Thus, the net change in LLM calls is zero.

Role	Change in LLM calls under HIVE
Planner	-1 for parameter probing
Executor	No change
Verifier	$+1$ for knowledge updating
Net change	**0**

C.2 Average Trajectory Length

We measure trajectory length as the number of MAS iterations required to process a task. One iteration consists of a Planner-Executor interaction with the environment followed by verification. The same counting convention is applied to all methods, including HIVE's sibling probing iterations.

For each benchmark, trajectory length is averaged over all tasks and three independent runs. All methods use the same underlying LLM, tool configurations, and execution environment. The resulting average trajectory lengths are reported in Table 3. HIVE achieves an average trajectory length of 4.45 iterations across the six benchmarks, compared with 5.83, 6.44, and 6.60 iterations for StructAgent, EPC-AW, and SupervisorAgent, respectively. Relative to these three baselines, the corresponding reductions are 23.7%, 30.9%, and 32.6%, respectively. The mean trajectory length of the three baselines is 6.29 iterations, corresponding to a 29.2% reduction relative to this mean.

C.3 Redundant Iteration Analysis

To further characterize the source of shorter trajectories, we analyze redundant iteration loops in the telemetry traces. We define a redundant iteration loop as *two consecutive MAS iterations that focus on the same subgoal, invoke the same tool with highly similar parameters according to a fixed similarity criterion, and obtain no target information from the latter iteration*. Thus, a trajectory is considered to contain a redundant iteration loop if at least one pair of consecutive iterations satisfies these conditions.

For a trajectory x, let $\mathrm{RedLoop}(x) \in \{0, 1\}$ indicate whether x contains at least one redundant iteration loop. For a benchmark dataset D, we define the redundant-loop rate as the fraction of trajectories containing at least one redundant loop:

$$\bar{\ell}(D) = \frac{1}{|D|} \sum_{x \in D} \mathrm{RedLoop}(x). \tag{12}$$

Thus, each trajectory contributes at most once to the rate, regardless of how many redundant loops it contains. The same criterion is applied uniformly across all methods and benchmarks.

Table 7: Redundant iteration loop rate $\bar{\ell}$ on six benchmarks. Lower values indicate fewer trajectories containing redundant iteration loops.

Method	Bamboogle	2Wiki	HotpotQA	Musique	GAIA	MedQA	Avg.
StructAgent	15.0%	16.0%	18.2%	31.0%	59.5%	17.8%	26.3%
EPC-AW	32.4%	21.0%	27.2%	31.3%	37.2%	25.6%	29.1%
SupervisorAgent	22.4%	17.8%	31.4%	23.0%	40.4%	31.8%	27.8%
HIVE	**0.1%**	**7.8%**	**0.7%**	**3.9%**	**3.6%**	**0.2%**	**2.7%**

HIVE has an average redundant-loop rate of only 2.7%, compared with 26.3%, 29.1%, and 27.8% for StructAgent, EPC-AW, and SupervisorAgent, respectively. The mean redundant-loop rate of the three baselines is 27.7%, making the rate for HIVE 90.3% lower.

These results provide complementary evidence for the shorter trajectories of HIVE. When the current decision is insufficiently supported, HIVE probes an alternative decision rather than repeating an essentially unchanged iteration. The hierarchical probing order first considers alternative parameterizations and then expands to alternative tools or subgoals when necessary.

Taken together, the three analyses characterize HIVE from different perspectives. LLM-call accounting shows that HIVE does not introduce additional LLM-call overhead relative to Clear MAS. Average trajectory length shows that HIVE requires fewer MAS iterations under the same model and tool configuration. The redundant iteration analysis further shows that substantially fewer HIVE trajectories contain repeated iterations with essentially unchanged decision configurations.

D QUALITATIVE CASE STUDIES

We present five qualitative cases to illustrate HIVE's hierarchical sibling probing process and its effect on execution trajectories. Cases A1-A3 illustrate successful repair through parameter-, tool-, and subgoal-level interventions, respectively. Cases B1-B2 illustrate the resulting trajectory-level effects through targeted repair and early stopping. Each case also reports the updated knowledge produced by the sibling contrast, including the persisted constraint or preference and the sibling evidence supporting the update.

Table 8: Overview of the five qualitative cases. A1-A3 illustrate hierarchical sibling probing, while B1-B2 illustrate trajectory-level effects. Trace lengths are instance-level observations and are not intended as estimates of average intervention overhead.

Case	Source	Focus	Trace length	Contrast
A1	Bamboogle, pid=25	Parameter intervention	2	–
A2	Musique, pid=33	Tool intervention	4	–
A3	2Wiki, pid=97	Subgoal intervention	7	–
B1	HotpotQA, pid=18	Targeted repair	3	StructAgent: 10, incorrect
B2	Bamboogle, pid=20	Targeted repair	4	EPC-AW: 7, incorrect

D.1 HIERARCHICAL INTERVENTION

Case A1: Parameter-level repair. The Bamboogle instance asks "The husband of Lady Godiva was Earl of which Anglic kingdom?" with gold answer Mercia. HIVE initially identifies Lady Godiva but fails to obtain the husband/title relation. It therefore changes only the query parameter while keeping both the subgoal and tool fixed. The sibling retrieves the missing relation, after which the final answer is verified.

A1 Parameter-level repair **Bamboogle, pid=25**

Task. "The husband of Lady Godiva was Earl of which Anglic kingdom?"

Ground-truth answer: Mercia

Execution trace

Step	Decision	Outcome
τ_1	Same subgoal; `query="Lady Godiva"`	Lady Godiva identified, but information about her husband/title remains missing.
τ_2	**L1:** same subgoal and tool; `query="Lady Godiva husband"`	Leofric identified as Earl of Mercia; subgoal complete.

Downstream result. Leofric → Earl of Mercia → *Mercia* verified → **STOP**.

Key observation. The successful sibling changes only the parameter, preserving the established subgoal and tool while supplying the missing information.

Evolved knowledge (recorded by the online update)

Record	Content
Preference	*"When the subgoal requires retrieving a specific relationship or attribute (e.g., 'husband', 'title') of an entity, prefer queries that explicitly include the target relationship or attribute [. . .]."*
Avoid	`query="Lady Godiva"` (the failed bare-entity formulation)
Successful sibling	(P, winner): *parameter query: Lady Godiva → Lady Godiva husband*; *SUBGOAL_INCOMPLETE vs. SUBGOAL_COMPLETE*

Case A2: Tool-level repair. The Musique instance asks "When did the spouse of Parysatis II die?" with gold answer 323 BC. HIVE first identifies Alexander the Great as the spouse, but the subsequent Wikipedia invocation fails to retrieve the required information. An L1 sibling changes the query

while retaining the same tool, but encounters the same information gap. HIVE therefore escalates to L2, switching the tool while preserving the information need. The new tool returns the required evidence, allowing the downstream answer to be completed.

A2 Tool-level repair **Musique, pid=33**

Task. "When did the spouse of Parysatis II die?"

Ground-truth answer: 323 BC

Execution trace

Step	Decision	Outcome
τ_1	Google Search: identify spouse	Alexander the Great identified.
τ_2	Wikipedia Search: retrieve death date	returns an empty response.
τ_3	**L1:** same tool; `query="Alexander Great death date"`	returns an empty response again.
τ_4	**L2:** switch to Google Search	Evidence gives death in 323 BC; subgoal complete.

Downstream result. Alexander the Great → death date 323 BC → **STOP**.

Key observation. The failed L1 sibling provides an additional unsuccessful attempt under the current tool; switching the tool while preserving the information need enables downstream completion.

Evolved knowledge (recorded by the online update)

Record	Content
Preference	*"When the parameter contains multiple information needs, prefer Google Search [. . .]."*
Avoid	Repeatedly invoking `Wikipedia_Search_Tool` for the unresolved death-date retrieval after the observed unsuccessful siblings.
Failed sibling	(P, non-winner): same tool with a revised query; *SUBGOAL_INCOMPLETE vs. SUBGOAL_INCOMPLETE*, showing that the parameter-level sibling did not resolve the information need under the current tool.
Tool sibling	(T, winner): *Wikipedia_Search_Tool* → *Google_Search_Tool*; *"Prefer Google_Search_Tool for this information need; do not revert to Wikipedia_Search_Tool."*

Case A3: Subgoal-level repair. The 2Wiki instance asks whether Big Pal and The Invisible Man Attacks are from the same country, with gold answer No. The initial bundled information request fails. Neither an L1 query revision nor an L2 tool change obtains reliable evidence. HIVE therefore escalates to L3 and changes the decomposition of the information request. The first L3 intervention separates the film-level information needs while retaining the established Big Pal premise. When the remaining evidence is still incomplete, lower-level interventions again fail to resolve the missing evidence, and a second L3 intervention focuses on The Invisible Man Attacks. This produces the missing country evidence and enables the final comparison.

A3 Subgoal-level repair **2Wiki, pid=97**

Task. "Are Big Pal and The Invisible Man Attacks from the same country?"

Ground-truth answer: No

Execution trace

Step	Decision	Outcome
τ_1	Bundled request for both films	Insufficient evidence for the country comparison.
τ_2	**L1:** revised bundled request	Still insufficient evidence.
τ_3	**L2:** alternative search tool	`ABSENCE`; no reliable country evidence.
τ_4	**L3:** decompose into film-level requests	Big Pal retrieved as U.S. production, but the comparison remains unresolved.
τ_5, τ_6	L1/L2 probes on unresolved evidence	Lower-level interventions remain unsuccessful.
τ_7	**L3:** retain Big Pal premise; target The Invisible Man Attacks	Argentine production identified; country comparison completed.

Downstream result. Big Pal → USA; The Invisible Man Attacks → Argentina → *No* → **STOP**.

Key observation. When lower-level interventions fail to resolve the missing evidence, HIVE escalates to a finer-grained subgoal decomposition while preserving the established premise, enabling the unresolved comparison to complete.

Evolved knowledge (recorded by the online update)

Record	Content
Preference	*"When a bundled request leaves one entity unresolved, decompose the request into entity-specific retrieval subgoals while preserving the established premise [. . .]."*
Avoid	`query="Big Pal origin country Invisible Man Attacks country origin"` (the stalled bundled formulation)
Failed siblings	(P and T rounds): *SUBGOAL_INCOMPLETE vs. SUBGOAL_INCOMPLETE*; lower-level interventions do not resolve the missing comparison evidence.
Subgoal sibling	(G, winner): bundled two-film request → single-film hop with carried-over premise; *"Prefer the revised subgoal decomposition; do not restore the stalled information need."*

D.2 Trajectory-Level Effects

Case B1: Early stopping after targeted repair. The HotpotQA instance asks when the defending titlist of the 2009-10 Biathlon World Cup Pursuit Men was born, with gold answer 27 January 1974. HIVE first identifies Ole Einar Bjørndalen, but the subsequent Wikipedia invocation fails to retrieve the birth date. A targeted L1 re-execution uses the full name "Ole Einar Bjørndalen" as the query parameter and successfully retrieves the date, allowing the trajectory to terminate. StructAgent performs ten steps, repeatedly revisiting the identification milestone, and returns an incorrect answer.

B1 Targeted repair and early stopping **HotpotQA, pid=18**

Task. "When was the defending titlist of 2009-10 Biathlon World Cup - Pursuit Men born?"

Ground-truth answer: 27 January 1974

Execution trace

Step	Decision	Outcome
τ_1	Google Search: identify defending titlist	Ole Einar Bjørndalen identified.
τ_2	Wikipedia Search: retrieve birth date	Entity identified, but fails to retrieve the birth date.
τ_3	**L1:** re-execute with the full name "Ole Einar Bjørndalen"	27 January 1974 retrieved; subgoal complete → **STOP**.

Length contrast. HIVE: **3 steps**, correct. StructAgent: **10 steps**, incorrect.

Key observation. HIVE preserves the established entity and repairs only the missing slot, avoiding repeated attempts at an already completed milestone.

Evolved knowledge (recorded by the online update)

Record	Content
Preference	*"Prefer the full, unabbreviated name of the identified person when retrieving the corresponding Wikipedia page [. . .]."*
Prefer	`query="Ole Einar Bjrndalen"` (recorded as `preferred_parameters`)
Successful sibling	(P, winner): same tool re-executed; *SUBGOAL_INCOMPLETE vs. SUBGOAL_COMPLETE*; the verified re-execution is persisted as the preferred invocation for this information need.

Case B2: Tool repair without re-planning the established entity. The Bamboogle instance asks which city the Filipino statesman who established the government-in-exile during World War II also served as mayor of, with gold answer Quezon City. HIVE first identifies Manuel L. Quezon as the relevant entity, but two successive Wikipedia-based attempts fail to retrieve information about his mayoral service. HIVE then performs an L2 tool change, switching to Web Search while preserving Manuel L. Quezon as the established entity. Although the Wikipedia result does not directly provide the required information, it contains relevant URLs that point to additional evidence. Web Search follows these links and retrieves evidence that Quezon served as acting Mayor of Quezon City, directly resolving the information need and yielding the correct answer. EPC-AW performs seven steps and returns an incorrect answer after identifying Sergio Osmeña as the relevant entity.

B2 Tool repair without re-planning the established entity **Bamboogle, pid=20**

Task. The Filipino statesman who established the government-in-exile during the outbreak of World War II was also the mayor of what city?"

Ground-truth answer: Quezon City

Execution trace

Step	Decision	Outcome
τ_1	Google Search: identify statesman	Manuel L. Quezon identified.
τ_2	Wikipedia Search: retrieve mayoral record	Quezon is identified, but the result does not provide information about his mayoral service; relevant URLs are available.
τ_3	**L1:** re-execute with a different parameter using the same tool	Still fails to retrieve the required information.
τ_4	**L2:** switch to Web Search	Web Search follows a relevant URL from the Wikipedia result and retrieves evidence that Quezon served as acting Mayor of Quezon City; information need resolved → **STOP**.

Evolved knowledge (recorded by the online update)

Record	Content
Preference	*"When Wikipedia Search does not retrieve the required information but provides relevant URLs, use Web Search to further retrieve information from these URLs for the same information need [. . .]."*
Tool preference	*Prefer `Web_Search_Tool` to follow relevant URLs from a Wikipedia result when Wikipedia Search fails to retrieve the required information.*
Failed sibling	(P, non-winner): same tool re-executed; *SUBGOAL_INCOMPLETE vs. SUBGOAL_INCOMPLETE*.
Tool sibling	(T, winner): *Wikipedia_Search_Tool → Web_Search_Tool*; Web Search follows a relevant URL from the Wikipedia result and retrieves the mayoral evidence, yielding *SUBGOAL_COMPLETE*.

The update records a tool-level preference: when the current tool fails to retrieve the required information but exposes relevant URLs, use an alternative search tool to follow those URLs while preserving the established entity and information need.

Cross-case observation. Across A1-A3, HIVE follows a consistent intervention pattern: it preserves established information, first changes the least disruptive decision, and escalates to higher-level decisions only when lower-level siblings fail to resolve the missing evidence. Cases B1-B2 illustrate

the trajectory-level consequence of this behavior: targeted repair avoids revisiting established milestones, while successful completion triggers immediate stopping. The evolved-knowledge records also illustrate how sibling contrasts are converted into level-specific preferences or constraints, with Successful sibling evidence retained alongside each update.

E Related Work

E.1 Failure and Repair in LLM-based Multi-agent Systems

LLM-based multi-agent systems (MAS) have been increasingly adopted for complex decision making, tool use, and long-horizon task execution (Zhou et al., 2026a; Ronanki, 2025; Lin et al., 2025). Despite their modularity and complementary agent capabilities, recent studies consistently reveal that MAS remain highly susceptible to failures arising from reasoning errors, coordination breakdowns, inconsistent information use, and unreliable interactions among agents (Cemri et al., 2026; Hammond et al., 2025; Zhang et al., 2026). Importantly, failures in MAS are often not isolated events: an error made by one agent can be propagated through subsequent interactions and become amplified across agents and execution steps, leading to cascading failures in long-horizon tasks (Li et al., 2026a; Rabanser et al., 2026). These observations have motivated increasing efforts to understand, diagnose, and mitigate failures in LLM-based MAS.

Existing failure-handling approaches can be broadly divided into *post-hoc diagnosis and repair* and *online intervention during execution*. Post-hoc approaches analyze completed trajectories to identify responsible agents, problematic steps, or failure mechanisms. For example, Who&When (Zhang et al., 2025) studies which agent causes a task failure and at which point in the trajectory, while AGDebugger (Epperson et al., 2025) enables developers to inspect and modify multi-agent interaction traces after execution. DoVer (Ma et al., 2026) further introduces targeted interventions across repeated executions to test failure hypotheses and validate their effects, providing a systematic mechanism for cross-run debugging. Although these approaches provide increasingly fine-grained explanations of failures, they primarily operate after a failure has manifested and therefore cannot directly prevent an unsupported decision from causing downstream errors.

In contrast, online approaches intervene while the MAS is executing. COCO (Liang et al., 2025) continuously monitors execution and employs rollback and reflective correction to mitigate execution drift, while MASC (Shen et al., 2026) uses history-conditioned signals to detect problematic steps and apply local corrections before errors propagate. SupervisorAgent (Lin et al., 2026) introduces adaptive runtime supervision that determines when and how to intervene at potentially risky agent interactions, including guidance, observation correction, and verification. Other test-time approaches similarly modify agent behavior in response to problematic interactions (Kim et al., 2026b; Shen et al., 2026). These methods demonstrate the effectiveness of runtime mechanism for preventing or correcting execution failures. Another line of work improves MAS reliability by grounding agent decisions and execution in structured, verified system information. StructAgent (Wu et al., 2026) explicitly maintains and exploits structured system information and verified states to constrain downstream agent decisions and execution. By ensuring that agent behavior is grounded in information that has already been validated, StructAgent reduces failures arising from inconsistent or unreliable system knowledge. However, verified information alone does not guarantee that a particular decision is sufficiently supported: the available information may be correct yet still lack the information needed to justify an upstream choice.

This limitation exposes a distinct failure mode that is not explicitly addressed by existing failure diagnosis, runtime intervention, or verified-information grounding approaches. The execution itself may be locally correct, while the upstream decision that produced it is unsupported by the agent's available information, causing the resulting outcome to violate the intended subgoal. In such cases, simply correcting the execution or constraining it with existing verified information is insufficient, because the missing factor lies in the knowledge supporting the decision itself. This setting motivates us to study *epistemic failure*: failures caused by insufficient information for making a decision, rather than by incorrect execution of that decision.

E.2 Epistemic Reasoning and Calibration in Multi-agent Systems

A growing body of work studies whether LLMs can recognize and appropriately reason about the limits of their own knowledge. At the model level, prior research investigates confidence calibration, uncertainty estimation, and overconfidence in LLM predictions (Rodionov & Assylbekov, 2026; Yadkori et al., 2024; Woodruff & Hewitt, 2026; Shao et al., 2026). These approaches seek to align an individual model's expressed confidence with its actual knowledge or predictive reliability, using techniques such as post-hoc calibration, uncertainty-aware inference, and calibration-oriented training. While such methods establish the importance of epistemic awareness for reliable LLM decisions, they primarily consider an individual model and do not account for how epistemic deficiencies interact with role specialization, information asymmetry, and dependencies in a multi-agent workflow.

Recent work has begun to study epistemic reasoning in multi-agent settings. EPC-AW (Wang et al., 2026) explicitly identifies *epistemic gaps* as a source of planning failure, where a planner makes a decision without sufficient information to support it, and improves planning by evaluating the epistemic support of candidate plans. Epistemic Context Learning (Zhou et al., 2026b) further considers epistemic reasoning across agents by constructing explicit profiles of collaborators from interaction history, allowing an agent to assess the reliability of peer outputs rather than treating all collaborators equally. Another line of work studies the epistemic value of multi-agent communication, decomposing uncertainty into epistemic and aleatoric components and measuring the information gained through interaction (Qiao et al., 2026). Together, these studies establish that reliable multi-agent decision making depends not only on the quality of individual reasoning, but also on whether decisions are adequately supported by the information available across the system.

However, existing epistemic approaches primarily *evaluate*, *estimate*, or *aggregate* epistemic support from information that is already available. They therefore provide ways to assess whether an agent has sufficient support, but do not directly specify how an LLM-based MAS should acquire missing support after a decision failure. HIVE instead treats verifier-confirmed failure as an actionable epistemic signal, probing the decision hierarchy to acquire missing information and converting the resulting evidence into reusable, role-specific knowledge.

E.3 Active Information Seeking and Exploration in LLM Agents

Beyond failure diagnosis and repair, recent work has increasingly studied how LLM agents can actively acquire information through exploration and information-seeking interactions. One line of work focuses on acquiring informative observations from the environment. InfoSeeker (Lee et al., 2026) integrates information seeking with task-oriented planning, allowing an agent to identify uncertainty from previous trajectories, actively collect informative observations, and revise its internal dynamics before replanning. SeekerGym (Kim et al., 2026a) formulates agentic information seeking as a POMDP and evaluates an agent's ability to identify, retrieve, and integrate missing information through targeted queries. These approaches demonstrate that actively acquiring information, rather than relying solely on the observation history, can substantially improve downstream decision making.

Another line of work focuses on improving the exploration policy itself. LaMer (Jiang et al., 2026) uses meta-RL and test-time reflection to induce more effective exploration from environment feedback, while Strategy-Guided Exploration (Szot et al., 2026) moves exploration from low-level actions to higher-level language strategies to obtain more structured and diverse trajectories. Related efforts also explore information acquisition for skill discovery and agent self-improvement, where exploratory trajectories are converted into reusable skills or experience for subsequent behavior (Yang et al., 2025b). In parallel, recent epistemic accounts of tool use argue that external interaction should be invoked when the information required for reliable task completion cannot be obtained through internal reasoning alone (Hongru et al.). These methods primarily ask how an agent can explore more effectively, discover useful behaviors, or acquire environment knowledge under limited feedback.

However, existing approaches do not address epistemic gap exploration in LLM-based MAS, which faces two MAS-specific challenges: epistemic gaps can accumulate across hierarchical decisions, while explored trajectories entangle information across decision levels and roles. HIVE instead treats verifier-confirmed task failure as an epistemic signal and asks which upstream decision lacks sufficient information support. It therefore probes alternative decisions upward from parameters to tools to subgoals, then compares sibling trajectories to identify decision-specific information requirements and distills them into reusable, role-specific online knowledge.