HIERARCHICAL PLANNING WITH CROSS-LEVEL CREDIT ASSIGNMENT FOR LONG-HORIZON TOOL USE

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon tasks require large language model (LLM) agents to plan hierarchically, in which a global plan fixes task-wide objectives and local subgoals adapt to observations revealed only during execution. When to place these local subgoals depends on observations that emerge during execution, so that we formulate it as a learning problem which however poses a credit assignment challenge. Prior hierarchical planning methods supply this credit through a learned value model or a prescribed process reward, which provides neither the after-execution evidence the global plan needs nor the task-wide reference a local subgoal lacks. We introduce HiCLA (Hierarchical Planning with Cross-Level credit Assignment), which supplies both through cross-level on-policy self-distillation: after a trajectory terminates, the policy (1) generates terminal evidence that rescores the global plan (i.e., *execution-to-planning*) and (2) produces a revised plan that in turn rescores each recorded state and decision (i.e., *planning-to-execution*), supplementing GRPO and used only during training. Across three Qwen backbones, HiCLA achieves the highest τ^2-bench mean success over four evaluation seeds among the compared methods and exceeds the strongest baseline on held-out BFCL-V3 with the 7B and 4B models by up to 5.25%.

1 INTRODUCTION

Large language model (LLM) agents learn to interact with environments through multi-turn reinforcement learning (RL) (Jin et al., 2025; Wang et al., 2025; Cheng et al., 2026). Long-horizon tasks involve multiple dependent stages, and the information needed to complete them is often revealed only as execution proceeds. Hierarchical planning (Hu et al., 2025b; Peng et al., 2026; Diao et al., 2026) addresses this difficulty by decomposing a task into intermediate objectives that guide lower-level decisions. These intermediate objectives are either placed in advance as a global plan (Xue et al., 2026), or selected during execution as local subgoals that respond to new observations and persist over several decisions (Peng et al., 2026; Diao et al., 2026). Each level, unfortunately, poses a complementary limitation: a global plan fixed in advance cannot specify choices that depend on future observations, whereas a local subgoal chosen reactively from the current observation lacks a task-wide reference for judging whether that choice serves the remaining objectives.

Deciding *where and when to place local subgoals*, thus, is not a matter of a p blacketermined decomposition or reactive rule. Instead, we propose to learn it from experiences, including (1) which subgoal to pursue now, (2) whether to hold it across decisions, or (3) to switch as observations emerge. This learning problem poses a *c blackit assignment* challenge. Task-level GRPO (Shao et al., 2024), which assigns the same outcome-based advantage to every decision in a rollout, gives little guidance on how to better place each local subgoal that spans several states and decisions. Prior hierarchical methods conqure this by building an external scorer, either from a learned value model (Peng et al., 2026) or a prescribed process reward (Diao et al., 2026) placed on top of the trajectory. Yet such a scorer is limited in two ways that mirror the two planning levels. First, it is fit or specified before the interaction completes, so it cannot use the evidence that only a finished trajectory exposes to judge whether the global plan itself was adequate. Second, it scores each step in isolation, without a task-wide reference, so it cannot tell whether a local decision served the remaining objectives rather than only its immediate observation.

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.

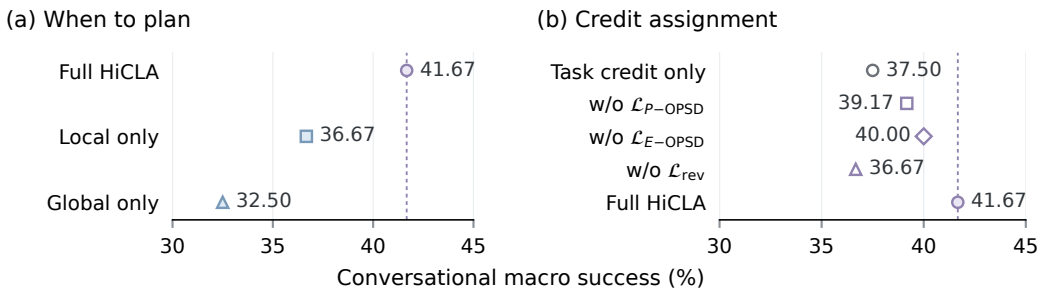

Figure 1: **When to plan and credit assignment.** Results on τ^2-bench with Qwen3-4B-Instruct-2507. (a) Full HiCLA and variants with only global or local planning; the latter use task-only GRPO (Shao et al., 2024). (b) Loss ablations (Eq. 10) with both planning levels retained. Dashed lines mark full HiCLA (41.67%). Training settings and environment rollout budgets are matched.

In this paper, we introduce *HiCLA* (Hierarchical Planning with Cross-Level c blackit Assignment) to supply both missing pieces from the terminal evidence, which is a summary and a reflection conditioned only on the observed outcome once a trajectory terminates. Inspi black by the recent success of on-policy self-distillation (OPSD) (Zhao et al., 2026; Hübotter et al., 2026; Wu et al., 2026a), HiCLA turns terminal evidence into token-level supervision through cross-level OPSD in which the sha black policy rescores its own outputs at one planning level using evidence from the other in two directions. (1) In addressing the first missing piece, *execution-to-planning OPSD* supervises the global plan, where the terminal evidence rescores the sampled global plan tokens. (2) In addressing the second missing piece, *planning-to-execution OPSD* supervises the execution with structure from planning: the policy first revises the global plan into a revised plan under the same terminal evidence; rescoring each recorded state and decision under this revised plan and the fixed interaction history gives each local decision the task-wide reference it lacked and turns its likelihood change into a signal for whether it served the remaining objectives. Note that terminal evidence and revised plans are used only during training, and inference uses only the initial global plan and adaptive local subgoals.

Across three Qwen backbones, HiCLA achieves the state-of-the-art τ^2-bench mean success over four evaluation seeds among the compa black methods on held-out BFCL-V3 with the 7B and 4B models (Table 1). On the 4B backbone, task-only ablations support retaining both global plans and local subgoals, and cross-level supervision adds a further 4.17% increase (Figure 1). Analysis of successful rollouts shows that HiCLR is coupled with higher correspondence between local states and subgoals, alongside more frequent recovery from tool errors during training (Section 5.4).

2 Related Work

Reinforcement learning and local credit. GRPO (Shao et al., 2024) compares task rewards without a value model. Finer credit exploits repeated states in GiGPO (Feng et al., 2025), historical-context groups (He et al., 2026b), or state-transition graphs (Wang et al., 2026c). Other approaches use answer-based values, environmental feedback, or reference-model scores to assign or reweight credit (Tao et al., 2026; Huo et al., 2026; He et al., 2026a). BEACON (Wang et al., 2026e) uses verifiable milestones to partition trajectories, shape rewards, and combine trajectory- and segment-level advantages. PhGPO (Li et al., 2026) reuses successful tool transitions as a graph prior that guides subsequent rollout generation. HiCLA supplements task credit with separate state and decision signals from likelihood changes under a revised plan, allowing different magnitudes and signs.

Hierarchical planning and context. Hierarchical approaches organize execution through local subgoals in HiPER (Peng et al., 2026) or initial strategies in StraTA (Xue et al., 2026), alongside task decomposition and progress supervision (Hu et al., 2025b; Li et al., 2025; Wang et al., 2026b). ToolScope (Deng et al., 2026) combines global navigation with local perception and tool execution in a training-free visual question-answering framework. HIPIF (Diao et al., 2026) and related approaches manage long interaction histories through summaries or compressed contexts (Hu et al., 2025a; Sun et al., 2025; Wu et al., 2026b). HiCLA retains continuous execution history, combining

a persistent global plan with adaptive local subgoals. Terminal evidence evaluates the initial plan, and a learned revised plan guides local behavior.

Self-distillation with additional training information. On-policy distillation supervises student-generated prefixes with teacher distributions (Agarwal et al., 2024). Self-teachers receive solutions (Zhao et al., 2026) or feedback in SDPO (Hübotter et al., 2026). SEED (Wu et al., 2026a), SDAR (Lu et al., 2026), and related methods (Wang et al., 2026a) distill training-only skill guidance; other variants retain skill access at inference (Pan et al., 2026). Additional information can supervise separate output fields (Zhang et al., 2026) or guide rollout generation followed by same-context GRPO (Su et al., 2026). SGCD (Ding et al., 2026) and CrEST (Wang et al., 2026d) reweight task advantages without changing their signs. HiCLA instead supplies independently signed auxiliary feedback to states and decisions through a learned revised plan. Evidence scores the original plan, and revised plans score the executor; all updates retain their generation contexts.

3 Problem Formulation

Interactive task setting. A task $X \sim \mathcal{D}$ consists of a task specification and the admissible initial context, including the initial observation, available tools, and operational constraints. At interaction j, the agent produces a decision d_j, either a structured tool action or a natural-language response. The environment, including user interaction when applicable, returns an observation v_j. A completed execution ξ receives an externally evaluated outcome $o(\xi) \in \{0, 1\}$. We seek a policy that maximizes task success under the task's constraints:

$$J(\theta) = \mathbb{E}_{X\sim\mathcal{D}}\mathbb{E}_{P,\xi\sim p_\theta(\cdot|X)}[o(\xi)]. \tag{1}$$

Hierarchical policy. We organize the policy's outputs into an initial global plan P, a local state s_j, and a decision d_j. The plan represents task-wide objectives and dependencies. The local state is generated text specifying the current subgoal and progress, distinct from the environment observation v_j. For an execution of T interactions, write $\xi = (s_j, d_j, v_j)_{j=1}^T$ and $h_j = (s_\ell, d_\ell, v_\ell)_{\ell<j}$, with the initial observation included in X. The trajectory distribution factorizes as

$$p_\theta(P, \xi \mid X) = \pi_\theta(P \mid X) \prod_{j=1}^{T} \pi_\theta(s_j \mid X, P, h_j)\pi_\theta(d_j \mid X, P, h_j, s_j)\, p_{\text{env}}(v_j \mid X, h_j, d_j). \tag{2}$$

Here p_{env} includes environment feedback and termination. All three policy conditionals share parameters θ. A plan spans the task, while a subgoal may persist across several state–decision steps.

Credit assignment. The outcome evaluates the execution but does not identify which choices to reinforce. Plan credit depends on the executions a plan induces, while local credit is conditioned on the preceding history and plan. Appendix B.1 defines these conditional credit targets; Section 4 develops the task and auxiliary signals used to train the hierarchy.

4 Hierarchical Planning and Credit Assignment

HiCLA connects global and local planning through two supervision paths (Figure 2): terminal evidence re-evaluates the initial plan, while a learned revised plan supplies feedback on local planning and decisions. All roles share one policy; a frozen snapshot π_{old} generates outputs and computes signals for updating π_θ.

4.1 Global Plans and Adaptive Local Planning

Planning from the latest observation alone can overlook task-wide dependencies. HiCLA generates a global plan P from X with subgoals and observable completion conditions. This persistent reference guides actions as information emerges during execution.

At step j, the policy combines P with visible history to generate a state s_j, then a decision d_j, following Eq. (2). The state expresses what to address and why; the decision executes that intent

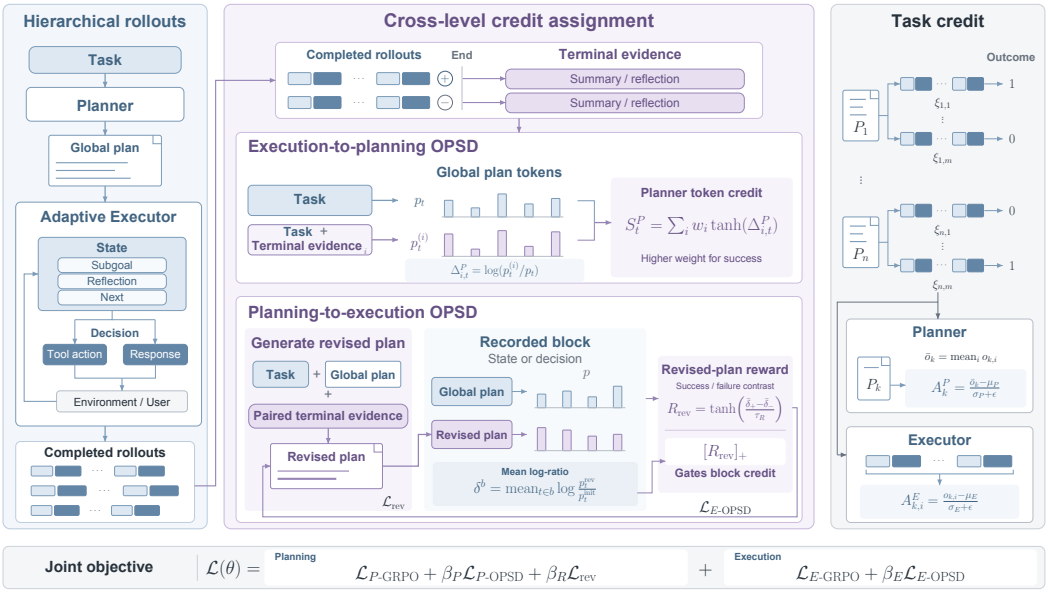

Figure 2: **HiCLA overview.** Global plans guide adaptive execution. Terminal evidence supervises global planning and conditions revised-plan generation. Rescoring successful and failed executions trains the revised plan and gates planning-to-execution OPSD. GRPO (Shao et al., 2024) assigns task credit at both levels. Profiles and outcomes are schematic; advantages omit format penalties.

through a tool action or response. Within the state, *subgoal* maintains the current objective, *reflection* assesses progress and errors using environmental feedback, and *next* formulates the upcoming interaction. Each new observation informs the next state, closing the local planning cycle.

Keeping the global plan fixed preserves task-wide direction while local subgoals adapt to feedback, persist, or revisit unmet dependencies. The history retains all states, decisions, and observations.

4.2 Execution-to-Planning OPSD

For each task, we sample n plans and m independent executions per plan, giving $G = nm$ environment rollouts. After execution i under plan k, the shared policy generates terminal evidence $H_{k,i}$: a summary and reflection conditioned on the observed outcome. These summaries compress long histories to fit the planning context budget. Environmental observations and task results supply information unavailable when P was generated; summarization organizes it without access to a reference answer or evaluator explanation.

We select successful and unsuccessful executions of the same plan (Appendix A.3), keeping evidence relevant to P and its induced behavior. Combining outcomes addresses complementary biases: success-only summaries may reaffirm the plan, whereas failure-only reflections may attribute diverse execution mistakes to planning. Let H_+, H_- denote the selected evidence with outcomes $o_+ = 1, o_- = 0$, suppressing the plan index.

To train planning before execution, we compare each sampled plan token's likelihood with and without evidence, retaining its prefix. With i indexing evidence items and t plan tokens, the token signal is

$$S_t^P = \sum_{i\in\{+,-\}} w_i \tanh\left[\log \frac{\pi_{\text{old}}(P_t \mid X, H_i, P_{<t})}{\pi_{\text{old}}(P_t \mid X, P_{<t})}\right], \qquad w_i = \frac{e^{(2o_i-1)/\tau}}{\sum_{l\in\{+,-\}} e^{(2o_l-1)/\tau}}. \tag{3}$$

During development, failure evidence could produce stronger negative likelihood shifts than the positive shifts supported by success evidence. We use success-weighted fusion to temper this imbalance while retaining both views; τ controls their relative weights. These positive weights combine evi-

dence sources without prescribing either signal's sign. The detached S_t^P reinforces or discourages the original token under X, so the planner learns to generate P without terminal evidence.

4.3 PLANNING-TO-EXECUTION OPSD

Executor supervision uses evidence-conditioned revised plans to rescore local behavior. We generate at most one revised plan Q from (X, P, H_+, H_-) per eligible pair, prompted to correct evidence-supported assumptions or dependencies while preserving subgoal granularity. An execution mistake need not imply a plan defect, so an adequate plan may remain unchanged.

States and decisions receive separate scores under Q: in development runs, states were longer and more sensitive to context changes than decisions, especially tool calls. A pooled step score could obscure decision feedback. Let $\mathcal{C}_{i,j}^b$ contain the content-token positions of block b at valid step j in trajectory i, excluding structural tags. For recorded token $z_{i,t}$, let $c_{i,t}^Z$ retain its complete sampled causal prefix with plan Z. The mean log-likelihood change is

$$\delta_{i,j}^b = \frac{1}{|\mathcal{C}_{i,j}^b|} \sum_{t \in \mathcal{C}_{i,j}^b} \log \frac{\pi_{\text{old}}(z_{i,t} \mid c_{i,t}^Q)}{\pi_{\text{old}}(z_{i,t} \mid c_{i,t}^P)}, \qquad b \in \{s, d\}. \tag{4}$$

The contexts differ only in the plan; tags remain in the prefixes, and decisions retain the sampled state. Positive $\delta_{i,j}^b$ indicates increased support under Q; negative values indicate the reverse. Block normalization removes token-count weighting. We average over trajectory i's M_i valid steps:

$$D_i(Q) = \frac{1}{2M_i} \sum_{j=1}^{M_i} \left(\delta_{i,j}^s + \delta_{i,j}^d\right). \tag{5}$$

Evaluating and learning revised plans. The Q-versus-P difference reduces dependence on baseline likelihood. Comparing these changes across outcomes evaluates discriminative guidance: raising both trajectories' scores by 0.2 gives zero contrast, whereas $+0.2$ for success and -0.1 for failure gives 0.3. Uniform likelihood gains earn no reward. For a valid, scoreable revised plan,

$$R_{\text{rev}}(Q) = \tanh\left(\frac{D_{\text{success}}(Q) - D_{\text{failure}}(Q)}{\tau_R}\right). \tag{6}$$

Here τ_R controls reward scale. The reward measures relative likelihood support on recorded executions, not success under Q. Proposition 1 does not guarantee this normalized score's sign or calibration (Appendix B.2).

We train Q with the signed reward under (X, P, H_+, H_-), learning revised-plan generation. Execution-to-planning OPSD instead trains P under X, before evidence is available. Revised-plan training updates future executor guidance, complementing initial-plan learning.

Transferring feedback to local behavior. A nonpositive contrast does not establish that reversing every local score would yield useful guidance. We conservatively withhold auxiliary executor supervision from such revised plans and use the positive part of the reward as a gate:

$$\widetilde{S}_{i,j}^b = \text{sg}\left[\eta(Q)\tanh(\delta_{i,j}^b)\right], \qquad \eta(Q) = \max(0, R_{\text{rev}}(Q)), \qquad b \in \{s, d\}. \tag{7}$$

Here sg stops gradients. The gate determines whether to transmit guidance; each block's likelihood change determines its sign. A failed execution can contain positive feedback and a successful one negative feedback; state and decision can also receive different signs. If $Q = P$, all likelihood changes vanish: retaining an adequate plan introduces no executor signal.

Each sampled block is updated under its original context containing P; Q supplies scoring only. Planner supervision from H_+, H_- remains independent of revised-plan quality.

4.4 JOINT POLICY OPTIMIZATION

Task-level GRPO (Shao et al., 2024) compares the n plans by mean success over their executions and the nm trajectories by individual outcomes, with bounded format penalties. Separate groups match credit to planning choices and completed executions, yielding advantages A_k^P and $A_{k,i}^E$. All original samples contribute, irrespective of evidence eligibility. Appendix A.1 defines reward normalization.

Policy losses and update contexts. All objectives use the same clipped token loss $\ell_t(A)$, with policy ratios evaluated under actual generation contexts (Appendix A.1). Signals and old-policy probabilities are detached. GRPO and execution-to-planning OPSD use A_k^P and S_t^P to update P under X; GRPO and planning-to-execution OPSD use $A_{k,i}^E$ and $\widetilde{S}_{i,j}^b$ under X, the original P, and causal history. Every token retains its sampled prefix.

Both planner losses average token losses within each plan, then across the n original plan slots and tasks; ineligible plans contribute zero to OPSD. Executor GRPO averages online token losses within trajectories, then across all nm trajectories and tasks. Each trajectory shares one task advantage; terminal evidence and external observations are excluded.

Planning-to-execution OPSD averages losses separately by block. Let $\mathcal{B}_{i,j}^b \supseteq \mathcal{C}_{i,j}^b$ contain all token positions in block b, including structural tags. Using exactly $\widetilde{S}_{i,j}^b$ from Eq. (7),

$$\mathcal{L}_i^{E\text{-OPSD}} = \frac{1}{2M_i} \sum_{j=1}^{M_i} \sum_{b\in\{s,d\}} \frac{1}{|\mathcal{B}_{i,j}^b|} \sum_{t\in\mathcal{B}_{i,j}^b} \ell_t(\widetilde{S}_{i,j}^b). \tag{8}$$

Each block first averages token losses, then the roles and valid steps are averaged. Separate feedback is preserved without giving longer blocks more weight solely from token count. The full loss averages over all nm original trajectories and tasks, counting inactive trajectories as zero, rather than normalizing by eligibility.

Learning the revised plan. The revised-plan loss applies the signed reward directly to the sampled tokens of Q. For a batch of B original tasks,

$$\mathcal{L}_{\text{rev}} = \frac{1}{B} \sum_q \frac{1}{L_{\text{cap}}} \sum_{t\in Q_q} \ell_t(\text{sg}[R_q]). \tag{9}$$

Here q indexes generated revised plans with available rewards; valid, scoreable revised plans use $R_q = R_{\text{rev}}(Q_q)$. Each update retains (X, P, H_+, H_-). Normalization uses the original task count B and fixed allowance L_{cap}, not revised-plan count or length. Negative rewards remain active when the executor gate closes; absent signals contribute zero (Appendix A.2).

The resulting plan and execution objectives jointly update the shared policy:

$$\begin{aligned} \mathcal{L}_{\text{plan}} &= \mathcal{L}_{P\text{-GRPO}} + \beta_P \mathcal{L}_{P\text{-OPSD}} + \beta_R \mathcal{L}_{\text{rev}}, \\ \mathcal{L}_{\text{exec}} &= \mathcal{L}_{E\text{-GRPO}} + \beta_E \mathcal{L}_{E\text{-OPSD}}, \\ \mathcal{L} &= \mathcal{L}_{\text{plan}} + \mathcal{L}_{\text{exec}}. \end{aligned} \tag{10}$$

Training begins with task-only warm-up before activating the three auxiliary losses. Terminal evidence receives no direct loss, including during warm-up. Inference uses only initial planning and execution with adaptive local states. Appendix A gives the format penalties, validity conditions, and numerical settings.

5 Experiments

5.1 Experimental Setup

We train in the official τ^2-bench gym environment (Barres et al., 2025) on its official training split and evaluate on its official Airline, Retail, and Telecom test splits. We separately evaluate on the full BFCL-V3 multi-turn subset (Patil et al., 2025). We compare GRPO (Shao et al., 2024), GiGPO (Feng et al., 2025), HiPER (Peng et al., 2026), and SEED (Wu et al., 2026a) on Qwen2.5-7B-Instruct, Qwen3-4B-Instruct-2507, and Qwen3-8B non-thinking.

HiCLA uses GRPO task losses with $n = m = 2$, matching four environment rollouts per task. HiCLA and SEED exclude BFCL from training, skill construction, and checkpoint selection. Scores are percentages; τ^2 Overall averages domains uniformly. Appendix C details the training settings, evaluation protocols, and aggregation.

Table 1: **Task performance (%).** The **best** and second-best scores include ties. avg@4 averages four evaluation seeds; the domain scores and Overall retain the separate original evaluation.

	BFCL-V3 multi-turn					τ^2-bench				
Method	**Base**	**Missing Func.**	**Missing Param.**	**Long Context**	**Overall**	**Airline**	**Retail**	**Telecom**	**Overall**	**Avg@4**
Qwen2.5-7B-Instruct										
Backbone	21.00	14.00	14.50	13.00	15.62	11.25	5.63	3.13	6.67	7.29
GRPO	19.00	9.00	14.00	9.00	12.75	33.75	5.63	5.63	15.00	17.92
GiGPO	**31.50**	**16.00**	**20.00**	14.50	20.50	25.00	13.13	**28.75**	22.29	19.97
HiPER	16.50	7.00	11.50	8.00	10.75	30.00	5.00	17.50	17.50	9.17
SEED	4.50	4.50	2.00	5.50	4.12	30.00	5.00	15.00	16.67	11.25
HiCLA	30.00	10.50	18.50	**28.50**	**21.88**	**35.00**	**22.50**	22.50	**26.67**	**23.96**
Qwen3-4B-Instruct-2507										
Backbone	33.00	10.00	**24.50**	27.00	23.62	21.25	40.00	9.38	23.54	7.08
GRPO	30.00	10.50	18.50	**28.50**	21.88	33.75	46.88	17.50	32.71	35.42
GiGPO	34.50	12.50	21.00	**28.50**	24.12	33.75	45.63	21.25	33.54	33.54
HiPER	32.00	18.50	12.50	23.50	21.62	30.00	47.50	27.50	35.00	27.50
SEED	10.00	1.00	8.00	6.50	6.38	**40.00**	20.00	22.50	27.50	27.71
HiCLA	**38.00**	**33.00**	21.00	25.50	**29.37**	**40.00**	**55.00**	**30.00**	**41.67**	**37.50**
Qwen3-8B non-thinking										
Backbone	22.00	4.00	16.50	15.00	14.38	5.00	10.00	15.63	10.21	10.21
GRPO	46.00	**40.50**	31.50	26.50	36.12	25.00	26.25	18.75	23.33	23.13
GiGPO	**47.00**	38.00	**34.00**	**28.00**	**36.75**	25.63	25.00	21.88	24.17	24.38
HiPER	44.00	11.00	16.00	17.00	22.00	20.00	40.00	**30.00**	30.00	21.25
SEED	2.50	0.00	1.50	1.50	1.38	**35.00**	40.00	7.50	27.50	24.58
HiCLA	44.50	**40.50**	31.50	24.50	35.25	**35.00**	**45.00**	12.50	**30.83**	**28.75**

5.2 Main Results

Under our exeperimental settings, HiCLA achieves the highest τ^2-bench avg@4: 23.96%, 37.50%, and 28.75% on the 7B, 4B, and 8B backbones, respectively (Table 1). In the original 4B domain evaluation, HiCLA matches or exceeds the best baseline in each domain (Figure 6).

On BFCL-V3, HiCLA scores 21.88% and 29.37% on the 7B and 4B models, respectively, compared with 20.50% and 24.12% for the strongest baseline, GiGPO (Feng et al., 2025) (Figure 7).

Validation dynamics. HiCLA's validation success improves from early to late training across all three backbones (Figure 3; protocol in Appendix C.2).

Training efficiency. On Qwen3-4B-Instruct, HiCLA outperforms GiGPO, HiPER, and SEED on both benchmarks with less recorded time per training step on average. Its 242.5-second mean exceeds GRPO's 165.8 seconds, mainly due to rollout and actor optimization (Appendix D.6).

5.3 Ablation Studies

Table 2 separates planning structure from supervision with matched settings and rollout budgets on Qwen3-4B-Instruct-2507. Structural ablations use task-only GRPO; others modify full HiCLA.

Global and local planning. Under task-only GRPO, removing local states lowers success from 37.50% to 32.50%, while removing the global plan lowers it to 36.67%. These comparisons favor retaining both components; Appendix E.1 illustrates their interaction.

Cross-level supervision. With the hierarchy fixed, auxiliary supervision raises success from 37.50% to 41.67%; removing execution-to-planning or planning-to-execution OPSD (Zhao et al.,

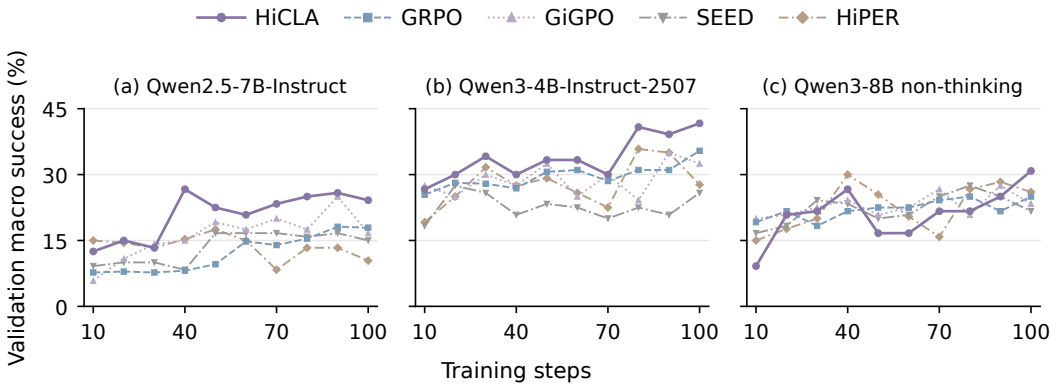

Figure 3: **Validation dynamics on τ^2-bench.** Macro-average success across Airline, Retail, and Telecom, evaluated every 10 training steps.

Table 2: **Component ablations on τ^2-bench.** Original 4B ablation evaluation (%), separate from main-table avg@4. Upper rows ablate the full method; lower rows remove planning components under task-only training.

Variant	Airline	Retail	Telecom	Overall
HiCLA (full)	**40.00**	**55.00**	**30.00**	**41.67**
Without execution-to-planning OPSD	35.00	52.50	30.00	39.17
Without planning-to-execution OPSD	40.00	55.00	25.00	40.00
Without revised-plan loss	30.00	50.00	30.00	36.67
Without explicit roles	30.00	55.00	30.00	38.33
Task credit only	30.00	55.00	27.50	37.50
Without global plan	30.00	52.50	27.50	36.67
Without local state	30.00	50.00	17.50	32.50

2026) reduces it to 39.17% or 40.00%. OPSD can supervise equal-reward plans using mixed terminal evidence. Appendix D.5 reports archive diagnostics of coverage, blockwise signals, and supervision at zero task advantage; the ablations assess the feedback's utility.

Learning useful revised plans. Keeping revised-plan generation, scoring, and gating fixed, removing only its training loss lowers success from 41.67% to 36.67%. This comparison tests the benefit of training the feedback source (Appendix D.4).

Explicit planning and execution roles. Removing explicit planner and executor roles lowers success to 38.33% while preserving structured outputs and auxiliary objectives, supporting distinct roles within the shared policy.

5.4 Global Guidance and Local Adaptation

Persistence under local subgoals. HiPER (Peng et al., 2026) learns SWITCH/KEEP decisions to let local subgoals guide multiple actions. In our τ^2-bench runs, Qwen3-8B non-thinking selects SWITCH at every sampled valid decision late in training, while the fraction of adjacent subgoal texts that differ rises from 55.89% early in training to 94.07% (Figure 8). Qwen2.5-7B-Instruct also switches at 98.97% of sampled valid late-stage decisions, yet only 20.37% of adjacent subgoal texts differ, and tool use ceases late in training.

Global plan guidance. In successful HiCLA executions, mean state–plan alignment is 75.94% and subgoal coverage is 79.07% (Figure 4; protocol in Appendix C.5). This alignment is consistent with task-wide guidance.

Local feedback adaptation. A correction attempt requires error-related reflection, an interaction purpose matching the next decision, and a changed decision. The *error recovery rate*, the fraction of these attempts followed by error-free tool feedback, rises from 14.53% to 32.98% over training (Figure 5). Rates cover tool feedback in successful rollouts; Appendix E illustrates corrections.

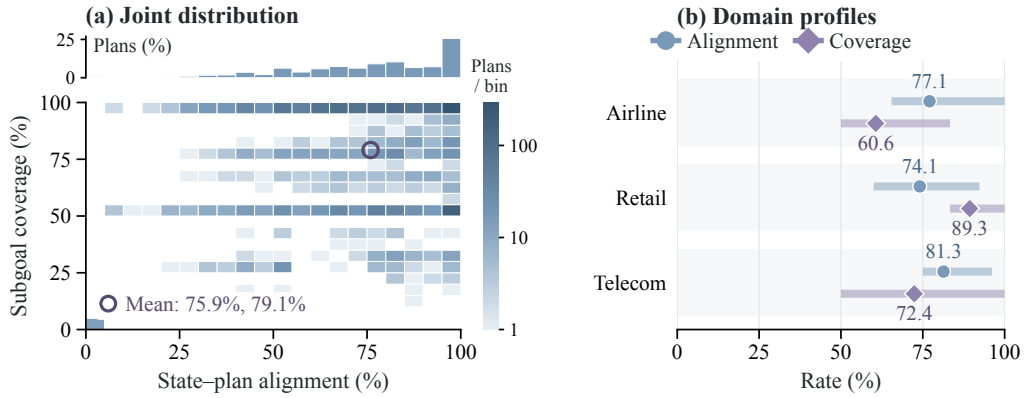

Figure 4: **Global plan guidance.** Successful executions, averaged per plan. (a) Joint distribution and histogram with 5%-wide bins and log counts. (b) Domain means and interquartile ranges.

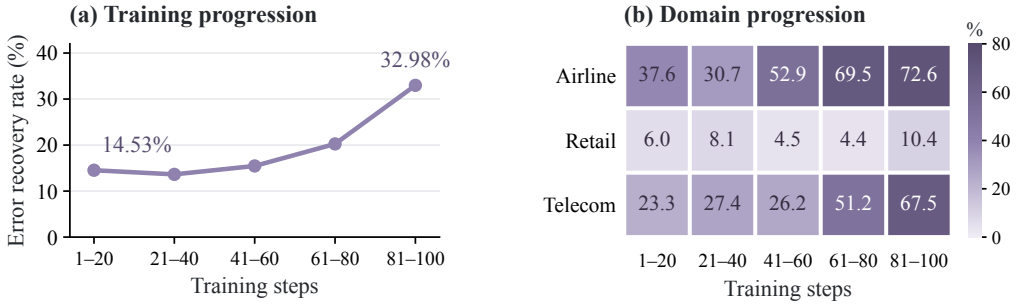

Figure 5: **Local feedback adaptation during training.** Error recovery is the fraction of correction attempts followed by error-free tool feedback within successful rollouts. (a) Overall and (b) domain rates, aggregated over 20-step windows.

6 Conclusion

We introduced HiCLA, which connects global planning with adaptive local subgoals through cross-level OPSD. The global plan provides task-wide direction, while the agent learns to retain or change local subgoals as observations arrive. Terminal evidence supervises the global plan, while revised plans provide feedback for recorded states and decisions. Results on τ^2-bench and held-out BFCL-V3, with 4B ablations, support both the planning hierarchy and cross-level supervision. These findings suggest that completed interactions can help agents learn how to plan and adapt.

Limitations

HiCLA's auxiliary supervision requires both a successful and a failed execution of the same plan. When all sampled executions succeed or all fail, the model learns only from task rewards. The signals also rely on model-generated terminal evidence: inaccurate summaries or reflections can affect plan revision and the resulting supervision. Finally, we assess revised plans through their effects on the likelihood of recorded executions, without executing them in the environment. This provides a training signal, but does not directly establish whether following a revised plan would improve task success.

Reproducibility Statement

Section 4 defines the interaction protocol and training signals. Appendices A and B provide the algorithm details, theoretical assumptions, and proofs. Appendix C specifies the training settings, evaluation metrics, and aggregation procedures used in the behavioral analysis.

Ethics Statement

Tool-use agents can take actions with external consequences. The interaction protocol includes domain constraints and user permission requirements. Deployment requires evaluating action safety beyond task success, and releasing trajectories requires reviewing potentially sensitive information.

AI Use Statement

The authors were responsible for identifying the research motivation, designing the method and experiments, conducting the experiments, and interpreting the observations and results. AI tools provided assistance with writing and code implementation. The authors retain full responsibility for the scientific content and conclusions.

References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-Policy Distillation of Language Models: Learning from Self-Generated Mistakes, 2024. URL https://arxiv.org/abs/2306.13649.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment, 2025. URL https://arxiv.org/abs/2506.07982.

Mingyue Cheng, Shuo Yu, Daoyu Wang, Qingchuan Li, Xiaoyu Tao, Jie Ouyang, Yucong Luo, Yitong Zhou, Qi Liu, and Enhong Chen. Agent-R1: A unified and modular framework for agentic reinforcement learning, 2026. URL https://arxiv.org/abs/2511.14460. Version 2, revised May 30, 2026.

Mengjie Deng, Guanting Dong, and Zhicheng Dou. ToolScope: An Agentic Framework for Vision-Guided and Long-Horizon Tool Use. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 211–225. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.11. URL https://aclanthology.org/2026.findings-acl.11/.

Juncheng Diao, Zhicong Lu, Peiguang Li, Yongwei Zhou, Changyuan Tian, Qingbin Li, Rongxiang Weng, Jingang Wang, and Xunliang Cai. HIPIF: Hierarchical Planning and Information Folding for Long-Horizon LLM Agent Learning, 2026. URL https://arxiv.org/abs/2606.10507.

Tianyu Ding, Jianhong Xin, and Juan Pablo De la Cruz Weinstein. Keep policy gradient in charge: Sibling-guided credit distillation for long-horizon tool-use agents, 2026. URL https://arxiv.org/abs/2606.12634.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-Group Policy Optimization for LLM Agent Training, 2025. URL https://arxiv.org/abs/2505.10978.

Langzhou He, Junyou Zhu, Yue Zhou, Zhengyao Gu, Junhua Liu, Wei-Chieh Huang, Henry Peng Zou, David Wipf, Philip S. Yu, and Qitian Wu. Resolving action bottleneck: Agentic reinforcement learning informed by token-level energy, 2026a. URL https://arxiv.org/abs/2605.14558.

Shuo He, Lang Feng, Qi Wei, Xin Cheng, Lei Feng, and Bo An. Hierarchy-of-groups policy optimization for long-horizon agentic tasks, 2026b. URL https://arxiv.org/abs/2602.22817.

Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. HiAgent: Hierarchical working memory management for solving long-horizon agent tasks with large language model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32779–32798, Vienna, Austria, July 2025a. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.1575. URL https://aclanthology.org/2025.acl-long.1575/.

Zican Hu, Wei Liu, Xiaoye Qu, Xiangyu Yue, Chunlin Chen, Zhi Wang, and Yu Cheng. Divide and conquer: Grounding LLMs as efficient decision-making agents via offline hierarchical reinforcement learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 24570–24590. PMLR, 2025b. URL https://proceedings.mlr.press/v267/hu25q.html.

Yifu Huo, Shunjie Xing, Chenglong Wang, Peinan Feng, Qiaozhi He, Yan Ding, Anxiang Ma, Yuxin Gao, Tongran Liu, Tong Xiao, and Jingbo Zhu. Learning from environmental feedback: Credit assignment across multiple timescales for agentic reinforcement learning, 2026. URL https://arxiv.org/abs/2608.08255.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement Learning via Self-Distillation, 2026. URL https://arxiv.org/abs/2601.20802.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning, 2025. URL https://arxiv.org/abs/2503.09516.

Yu Li, Guangfeng Cai, Shengtian Yang, Han Luo, Shuo Han, Xu He, Dong Li, and Lei Feng. PhGPO: Pheromone-Guided Policy Optimization for Long-Horizon Tool Planning, 2026. URL https://arxiv.org/abs/2602.13691.

Zhuofeng Li, Haoxiang Zhang, Seungju Han, Sheng Liu, Jianwen Xie, Yu Zhang, Yejin Choi, James Zou, and Pan Lu. In-the-flow agentic system optimization for effective planning and tool use, 2025. URL https://arxiv.org/abs/2510.05592.

Zhengxi Lu, Zhiyuan Yao, Zhuowen Han, Zi-Han Wang, Jinyang Wu, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Self-Distilled Agentic Reinforcement Learning, 2026. URL https://arxiv.org/abs/2605.15155.

Tianjun Pan, Yuan Li, Hongda Wang, Linbo Jin, Mengfei Song, Lei Gao, Qiming Shi, Shaokang Fu, Jiarong Zhao, Chengyu Wang, and Chengfu Huo. Bidirectional Context Self-Distillation for Reinforcement Learning of Skill-Based LLM Agents, 2026. URL https://arxiv.org/abs/2608.09555.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48371–48392. PMLR, 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

Jiangweizhi Peng, Yuanxin Liu, Ruida Zhou, Charles Fleming, Zhaoran Wang, Alfredo Garcia, and Mingyi Hong. HiPER: Hierarchical Reinforcement Learning with Explicit Credit Assignment for Large Language Model Agents, 2026. URL https://arxiv.org/abs/2602.16165.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Xingyu Su, Abhishek Kumar, Qing Ping, Youzhi Luo, Jonathan Buck, Zach Zhang, Subramanian Chidambaram, and Vinayak Arannil. From Self-Distillation to Self-Practice: Privileged Information for Multi-Turn Agents, 2026. URL https://arxiv.org/abs/2609.29051.

Weiwei Sun, Miao Lu, Zhan Ling, Kang Liu, Xuesong Yao, Yiming Yang, and Jiecao Chen. Scaling long-horizon LLM agent via context-folding, 2025. URL https://arxiv.org/abs/2510.11967.

Leitian Tao, Baolin Peng, Wenlin Yao, Tao Ge, Hao Cheng, Mike Hang Wang, Jianfeng Gao, and Sharon Li. TRACE: Turn-level reward assignment via credit estimation for long-horizon agents, 2026. URL https://arxiv.org/abs/2607.13988.

Hao Wang, Guozhi Wang, Han Xiao, Yufeng Zhou, Yue Pan, Jichao Wang, Ke Xu, Yafei Wen, Xiaohu Ruan, Xiaoxin Chen, and Honggang Qi. Skill-SD: Skill-Conditioned Self-Distillation for Multi-turn LLM Agents, 2026a. URL https://arxiv.org/abs/2604.10674.

Taiyi Wang, Sian Gooding, Florian Hartmann, Oriana Riva, and Edward Grefenstette. A subgoal-driven framework for improving long-horizon LLM agents, 2026b. URL https://arxiv.org/abs/2603.19685.

Yunan Wang, Minghui Song, Zihan Zhang, Shaohan Huang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, and Qi Zhang. Group-graph policy optimization for long-horizon agentic reinforcement learning, 2026c. URL https://arxiv.org/abs/2606.22995.

Zechuan Wang, Siyuan Lu, Hongxuan Zhang, Linjian Mo, Chenyi Zhuang, and Leilei Gan. Teach the magnitude, not the direction: Verifier-bounded credit assignment for multi-turn multi-step LLM agents, 2026d. URL https://arxiv.org/abs/2608.13179.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. RAGEN: Understanding self-evolution in LLM agents via multi-turn reinforcement learning, 2025. URL https://arxiv.org/abs/2504.20073.

Zixuan Wang, Yuchen Yan, Hongxing Li, Teng Pan, Dingming Li, Ruiqing Zhang, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Milestone-Guided Policy Learning for Long-Horizon Language Agents, 2026e. URL https://arxiv.org/abs/2605.06078.

Jinyang Wu, Shuo Yang, Zhengxi Lu, Fan Zhang, Yuhao Shen, Lang Feng, Haoran Luo, Zheng Lian, Shuai Zhang, Zhengqi Wen, and Jianhua Tao. SEED: Self-Evolving On-Policy Distillation for Agentic Reinforcement Learning, 2026a. URL https://arxiv.org/abs/2607.14777.

Xixi Wu, Kuan Li, Yida Zhao, Liwen Zhang, Litu Ou, Huifeng Yin, Zhongwang Zhang, Xinmiao Yu, Dingchu Zhang, Yong Jiang, Pengjun Xie, Fei Huang, Minhao Cheng, Shuai Wang, Hong Cheng, and Jingren Zhou. ReSum: Unlocking long-horizon search intelligence via context summarization, 2026b. URL https://arxiv.org/abs/2509.13313.

Xiangyuan Xue, Yifan Zhou, Zidong Wang, Shengji Tang, Philip Torr, Wanli Ouyang, Lei Bai, and Zhenfei Yin. StraTA: Incentivizing Agentic Reinforcement Learning with Strategic Trajectory Abstraction, 2026. URL https://arxiv.org/abs/2605.06642.

Yan Zhang, Daiqing Wu, Huawen Shen, Liang Li, Gang Cao, Zhi Gong, Wei Dai, Xiaode Zhang, Can Ma, and Yu Zhou. Learn How to Act from Your Own Interactions: On-Policy Self-Distillation for GUI Agents, 2026. URL https://arxiv.org/abs/2609.27307.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-Distilled Reasoner: On-Policy Self-Distillation for Large Language Models, 2026. URL https://arxiv.org/abs/2601.18734.

A Algorithm Details

This appendix specifies the rewards, loss reductions, and training procedure. The planner and executor share parameters. We use π_{old} to denote the frozen policy that collects and scores the current training data. Training on initial plans and executor outputs retains their original generation contexts, with detached signals computed from terminal evidence and revised plans, respectively. Revised

plans are trained under their evidence-conditioned generation contexts. These objectives distinguish planning before execution from revising a plan after observing its results. Online sampling follows

$$P \sim \pi_{\text{old}}(\cdot \mid X), \qquad s_j \sim \pi_{\text{old}}(\cdot \mid X, P, h_j), \qquad d_j \sim \pi_{\text{old}}(\cdot \mid X, P, h_j, s_j). \tag{11}$$

A.1 Task rewards and loss reductions

For a task, let $k \in \{1, \dots, n\}$ index initial plans and $i \in \{1, \dots, m\}$ index the executions under each plan. The planner reward and executor reward are

$$\begin{aligned} r_k^P &= \frac{1}{m} \sum_{i=1}^{m} o_{k,i} - 0.1\mathbf{1}[\text{invalid plan}], \\ r_{k,i}^E &= o_{k,i} - 0.1\mathbf{1}[\text{high-level format error}] - 0.1\mathbf{1}[\text{action format error}]. \end{aligned} \tag{12}$$

Each error category incurs at most one penalty per episode. Invalid plans are not executed and receive zero success. Let (μ_P, σ_P) and (μ_E, σ_E) denote means and population standard deviations over the task's n plan rewards and nm execution rewards. For nondegenerate groups,

$$A_k^P = \frac{r_k^P - \mu_P}{\sigma_P + \epsilon}, \qquad A_{k,i}^E = \frac{r_{k,i}^E - \mu_E}{\sigma_E + \epsilon}. \tag{13}$$

We use $\epsilon = 10^{-6}$; a group with standard deviation at most ϵ receives zero advantage. Each episode's executor advantage applies to all tokens generated during online interaction.

For the planner, let $\bar{r}^P = n^{-1} \sum_k r_k^P$ and $\sigma_P^2 = n^{-1} \sum_k (r_k^P - \bar{r}^P)^2$. Its advantage is

$$A_k^P = \begin{cases} (r_k^P - \bar{r}^P)/(\sigma_P + \epsilon), & \sigma_P > \epsilon, \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

Each plan is thus evaluated by its own m executions and compared with other plans for the same task. Nonzero plan advantages require $n \geq 2$. The experiments use $n = m = 2$ to match a budget of $G = nm = 4$ environment rollouts per task.

For token y_t with actual generation context c_t, the shared clipped surrogate is

$$\begin{aligned} \rho_t &= \frac{\pi_\theta(y_t \mid c_t)}{\pi_{\text{old}}(y_t \mid c_t)}, \\ \ell_t^0(A) &= \max\{-\rho_t A, -\text{clip}(\rho_t, 1 - \epsilon_l, 1 + \epsilon_h)A\}, \\ \ell_t(A) &= \begin{cases} \min\{\ell_t^0(A), -cA\}, & A < 0, \\ \ell_t^0(A), & A \geq 0. \end{cases} \end{aligned} \tag{15}$$

Here ϵ_l, ϵ_h define the clipping interval and $c > 1$ caps the loss for negative signals. We use $\epsilon_l = \epsilon_h = 0.2$, giving the ratio clipping interval $[0.8, 1.2]$, and dual-clip coefficient $c = 3.0$. Privileged contexts determine the detached signals; the policy ratio still uses the original context c_t. Both planner losses average token losses within each plan, then average across the n plans and across tasks. Ineligible plans contribute zero to execution-to-planning OPSD.

Let $\mathcal{T}_{k,i}$ contain online generated token positions and $N_{k,i} = |\mathcal{T}_{k,i}|$. The executor task loss is

$$\mathcal{L}_{E\text{-GRPO}} = \mathbb{E}_X \left[\frac{1}{nm} \sum_{k=1}^{n} \sum_{i=1}^{m} \frac{1}{N_{k,i}} \sum_{t \in \mathcal{T}_{k,i}} \ell_t(A_{k,i}^E) \right]. \tag{16}$$

It covers online policy outputs, including malformed generations, and excludes terminal evidence and external observations. A trajectory with no online generated tokens contributes zero.

Equation (8) averages planning-to-execution OPSD within blocks and across valid steps, then over all nm original trajectories and tasks. Inactive trajectories, including those with no valid steps, contribute zero. Only valid state and decision blocks with nonempty content sets receive OPSD; terminal evidence and malformed outputs are excluded. The score in Eq. (4) averages over $\mathcal{C}_{i,j}^b$, while the loss applies to the complete block $\mathcal{B}_{i,j}^b$, including structural tags.

A.2 LEARNING FROM REVISED PLANS

Equation (5) aggregates the separately normalized state and decision lifts, denoted by $\bar{\delta}_{\pm}$ in Figure 2. Equation (6) contrasts the resulting successful and failed execution scores to obtain $R_{\mathrm{rev}}(Q)$, with $\tau_R = 1$. The reward uses raw likelihood lifts before applying $\tanh$ or the executor gate. Revised-plan training changes the source of planning guidance used for executor supervision in subsequent training steps.

The revised-plan loss in Eq. (9) uses each candidate's actual generation context (X, P, H_+, H_-), with fixed token allowance $L_{\mathrm{cap}} = 768$. Tasks without revised-plan supervision contribute zero. Negative rewards are retained, and a generated candidate with invalid plan format receives $R_q = -1$. Missing evidence or absent revised-plan generation does not receive a fabricated reward.

For valid, scoreable revised plans, $\eta(Q) = \max(0, R_{\mathrm{rev}}(Q))$; otherwise it is zero. This gate affects planning-to-execution OPSD only. Execution-to-planning OPSD uses eligible terminal evidence independently of the revised-plan reward. All scores, rewards, and aggregation weights are detached; no gradient flows through evidence generation or reward computation.

Algorithm 1 HiCLA training

for each training batch **do**
  Freeze the current policy as π_{old}.
  Initialize auxiliary signals to zero.
  for each task X **do**
    Sample n global plans; execute each m times with adaptive local subgoals.
    Compute task rewards and GRPO credit for sampled plans and executions.
    Generate terminal evidence conditioned on the external outcome.
    if the auxiliary learning phase is active **then**
      for each plan with an eligible success/failure pair **do**
        Re-score the initial plan given each terminal evidence item to obtain S^P.
        Generate at most one revised plan Q from (X, P, H_+, H_-).
        if Q is valid and scoreable **then**
          Re-score recorded blocks to obtain $\delta^s_{i,j}, \delta^d_{i,j}$.
          Aggregate lifts into $D_i(Q)$ using Eq. (5).
          Compute $R_{\mathrm{rev}}(Q)$ and gated scores using Eqs. (6)–(7).
        else if Q was generated with invalid format **then**
          Assign revised-plan reward -1; keep planning-to-execution OPSD inactive.
        end if
      end for
    end if
  end for
  Optimize the shared policy using Eq. (10).
end for

A.3 EVIDENCE AND TRAINING SETTINGS

During online execution, the policy observes the task, initial plan, and causal interaction history. After execution, it generates terminal evidence conditioned on the outcome indicator, without access to a reference answer or evaluator explanation. Summaries and reflections compress long execution records into feedback that fits the planning context budget. The prompt requests feedback on planning and execution that could guide a fresh attempt. Evidence generation receives no direct loss.

For auxiliary supervision, we select a successful and an unsuccessful execution of the same initial plan, each with valid terminal evidence and at least one valid state/decision step. Evidence from other plans is excluded to keep feedback specific to the plan being evaluated and the behavior it induces. All original rollouts contribute to task learning. Training begins with 20 task-only training steps to establish planning and interaction formats, followed by joint optimization of the five objectives in Eq. (10). No direct summary or reflection training objective is introduced during warm-up.

Table 3: HiCLA experimental settings. The allocation $n = m = 2$ matches the four-rollout budget; the method samples n plans and m executions per plan. Optimizer and rollout-sampling settings are specified in Appendix C.

Setting	Value
Initial plans n / executions per plan m	2 / 2
Planner / executor task loss	GRPO
Outcome weighting temperature τ	1
Revised-plan reward temperature τ_R	1
Task-only warm-up	20 training steps
$\beta_P, \beta_E, \beta_R$ after warm-up	0.01 each
Revised-plan token cap L_{cap}	768
Terminal-evidence token allowance	512
Revised-plan sampling temperature	0.7

B Theoretical Details

B.1 Conditional credit with shared parameters

We define separate conditional advantages for the plan, state, and decision. Let $c_j = (X, P, h_j)$ and abbreviate $o(\xi)$ as o. The ideal advantages are

$$\begin{aligned} \mathcal{A}^P &= \mathbb{E}_\theta[o \mid X, P] - \mathbb{E}_\theta[o \mid X], \\ \mathcal{A}^s_j &= \mathbb{E}_\theta[o \mid c_j, s_j] - \mathbb{E}_\theta[o \mid c_j], \\ \mathcal{A}^d_j &= \mathbb{E}_\theta[o \mid c_j, s_j, d_j] - \mathbb{E}_\theta[o \mid c_j, s_j]. \end{aligned} \tag{17}$$

The plan advantage averages over possible executions. State and decision advantages condition on the plan and the information available when each output is generated.

Let $J(\theta) = \mathbb{E}_\theta[o]$. Partition the policy's online outputs into causally ordered blocks Y_b, each generated from context C_b. Assume a finite horizon, differentiable normalized generation laws with locally fixed support, and integrable scores that permit differentiation under expectation. Environment transitions and stopping depend on history but not directly on θ. Define

$$U(Y, C) = \nabla_\theta \log \pi_\theta(Y \mid C), \qquad A(Y, C) = \mathbb{E}_\theta[o \mid C, Y] - \mathbb{E}_\theta[o \mid C].$$

The likelihood-ratio identity gives $\nabla_\theta J = \mathbb{E}_\theta[o \sum_b U(Y_b, C_b)]$. Since $\mathbb{E}_\theta[U(Y, C) \mid C] = 0$, conditioning on (C, Y) and subtracting the context-dependent baseline yields

$$\nabla_\theta J = \mathbb{E}_\theta \left[\sum_b A(Y_b, C_b) U(Y_b, C_b) \right]. \tag{18}$$

All policy factors are differentiated, so the identity also holds when the roles share parameters. The advantages serve as learning signals and are not differentiated separately.

For the plan–state–decision factorization, let $c_j = (X, P, h_j)$ and write u_P, u^s_j, u^d_j for the score gradients of the three conditionals. When these blocks cover the generated outputs, Eq. (18) becomes

$$\nabla_\theta J = \mathbb{E}_\theta \left[\mathcal{A}^P u_P + \sum_{j=1}^{T} \left(\mathcal{A}^s_j u^s_j + \mathcal{A}^d_j u^d_j \right) \right], \tag{19}$$

using the advantages in Eq. (17) and trajectory length T. Any additional outputs retain their terms in Eq. (18). This conditional-baseline decomposition motivates distinct credit for planning and execution, as in the hierarchical policy-gradient analysis of HiPER (Peng et al., 2026). HiCLA uses group-relative task advantages and auxiliary signals from evidence-conditioned and revised-plan-conditioned scoring rather than estimating these conditional expectations directly.

B.2 Interpreting Revised-Plan Credit

The revised-plan reward compares how a plan changes the likelihood of successful and failed executions. To relate this contrast to task success, fix a task, an executor, and two plans P, Q.

Let μ, ν be the trajectory distributions induced by P, Q under the same environment and evaluator. Write their success probabilities as p, q and their distributions conditional on $o = r$ as μ_r, ν_r. Define

$$\Lambda = \log \frac{d\nu}{d\mu}, \qquad C_\Lambda = \mathbb{E}_{\mu_1}[\Lambda] - \mathbb{E}_{\mu_0}[\Lambda], \qquad K_r = D_{\mathrm{KL}}(\mu_r \| \nu_r).$$

Proposition 1 (Revised-plan contrast and success odds) *If μ, ν are mutually absolutely continuous, $0 < p, q < 1$, and Λ is conditionally absolutely integrable, then*

$$\log \frac{q}{1-q} - \log \frac{p}{1-p} = C_\Lambda + K_1 - K_0. \tag{20}$$

The identity relates the change in success odds to the likelihood contrast and a residual term, $K_1 - K_0$, that accounts for changes within each outcome class. HiCLA uses a normalized contrast on sampled executions as an auxiliary signal. This score measures relative likelihood support for successful behavior; it does not measure the success rate of executing Q. GRPO (Shao et al., 2024) supplies credit from external task outcomes. The proof follows below, and Appendix B.4 relates the identity to the practical score.

B.3 Proof of Proposition 1

Fix X, P, Q and the executor. Let μ, ν be its complete trajectory distributions under P, Q, including observations, stopping, and the evaluated outcome. Both use the same initial environment distribution, transition law, and evaluator. Under the assumptions of Proposition 1, write $p_1 = p, p_0 = 1 - p$ and define q_r analogously. On the outcome class $\{o = r\}$,

$$\frac{d\nu}{d\mu} = \frac{q_r}{p_r} \frac{d\nu_r}{d\mu_r}, \qquad \mathbb{E}_{\mu_r} \Lambda = \log \frac{q_r}{p_r} - D_{\mathrm{KL}}(\mu_r \| \nu_r).$$

Subtracting the failure equation from the success equation proves Eq. (20). The common environment and evaluation factors cancel, so Λ is the sum of executor-token log-likelihood ratios under their causal prefixes. The initial plans are fixed, and terminal evidence is generated after execution; neither enters this ratio.

B.4 Connection to the practical revised-plan score

Let $S_Z(\xi)$ be the normalized content-token log likelihood of a recorded execution ξ under plan Z, using the block and step weights in Eq. (5). Then $D(\xi; Q) = S_Q(\xi) - S_P(\xi)$, and the contrast used by the revised-plan reward satisfies

$$D(\xi_+; Q) - D(\xi_-; Q) = [S_Q(\xi_+) - S_Q(\xi_-)] - [S_P(\xi_+) - S_P(\xi_-)].$$

The reward therefore measures how the revised plan changes the successful execution's score relative to the failed execution. The practical score normalizes within blocks and aggregates across roles and steps, so longer blocks do not receive greater weight solely because they contain more tokens. This sampled, normalized contrast is an auxiliary training signal and differs from the population contrast C_Λ in Proposition 1.

C Experimental Details

C.1 Training and evaluation protocol

We use the backbones in Section 5.1. Training uses seed 42 and eight NVIDIA H800 GPUs. DeepSeek-v4-flash serves as the user simulator and natural-language evaluator.

We train HiCLA for 10 epochs with 32 tasks per training batch, a learning rate of 10^{-6}, a rollout temperature of 0.7, and top-$p = 0.95$. Revised-plan generation uses the same temperature with top-$p = 1$ and no top-k restriction. We validate every 10 training steps with 16 tasks per batch. Setting $n = m = 2$ matches four environment rollouts per task; terminal-evidence generation and rescoring add model-side computation. Appendix A.3 lists the auxiliary training settings.

Baseline initialization. For HiPER (Peng et al., 2026), we initialize the policy and a three-head critic from the corresponding backbone and train the hierarchical policy and value estimates jointly. For SEED (Wu et al., 2026a), we use supervised initialization followed by RL and skill-conditioned distillation; its trajectory analyzer shares the current policy snapshot. We use DeepSeek-v4-flash to generate trajectory-derived skill annotations during initialization. The behavioral analyses cover the subsequent RL phase.

Evaluation metrics. The τ^2-bench avg@4 column in Table 1 averages success over four different evaluation seeds for each fixed checkpoint; it is not pass@4. The domain columns and their Overall retain the original evaluation, while the ablation study retains its separately reported results. BFCL-V3 uses four evaluation trials per task and the checkpoint with the best training performance. Overall scores are unweighted means across the three τ^2 domains or four BFCL multi-turn categories. All results use a single training run per configuration; averages across evaluation seeds do not measure variation across independently trained policies.

Component ablations. All variants in Table 2 use Qwen3-4B-Instruct-2507 with matched training settings and environment rollout counts. Task-only training sets $\beta_P = \beta_E = \beta_R = 0$. Structural ablations remove either the initial global plan or the entire local state (subgoal, reflection, and next) from this variant. The OPSD ablations remove the loss for the named direction while retaining revised-plan training. The revised-plan-loss ablation sets $\beta_R = 0$, retaining revised-plan generation, scoring, and gated executor supervision; other objectives still optimize the shared policy. The role ablation retains plan, state, decision, and terminal-evidence outputs and the auxiliary objectives without assigning distinct planner and executor roles.

C.2 Validation trajectories

Figure 3 follows 150 evaluation checkpoints across the 15 method–backbone trajectories, sampled at 10-step intervals through training step 100. Each value is the unweighted mean of the three domain success rates, converted to a percentage. Resumed checkpoints continue the same training trajectory. Lines connect the recorded evaluations without smoothing, imputation, or extrapolation.

The curves retain the validation estimators reported in the source summaries. GRPO (Shao et al., 2024) and GiGPO (Feng et al., 2025) records contain four executions per task; HiPER (Peng et al., 2026) summaries with trial-count metadata report one or four trials. HiCLA and SEED (Wu et al., 2026a) summaries supply domain-level aggregates. Thus, these trajectories describe the reported validation performance, while Table 1 uses the final evaluation protocol above.

C.3 Local-subgoal analysis

We examine the switching decisions and subgoal persistence of HiPER (Peng et al., 2026) on all three backbones over 100 training steps. We sample trajectories every five training steps and compare training steps 1–20 with 81–100, pooling counts within each window. SWITCH frequency is the fraction of valid SWITCH/KEEP decisions that select SWITCH. Subgoal change is the fraction of adjacent extracted subgoal texts within the same episode that differ after case and whitespace normalization. This statistic measures textual variation rather than semantic change. The tool-use observation in Section 5.4 uses training-batch logs: both the tool-call ratio and mean calls per episode are zero late in training for Qwen2.5-7B-Instruct.

C.4 SEED behavioral analysis

We analyze SEED (Wu et al., 2026a) over 100 training steps per backbone, with 128 episodes per training step. We compare training steps 1–20 and 81–100. Tool and response fractions are computed over all generated actions; the no-tool rate is the fraction of episodes containing no strictly formatted tool call. Success is averaged over training batches. The distillation objective is active throughout these runs; skill extraction excludes all-failure groups for the 7B and 8B models.

C.5 HiCLA behavioral analysis

Section 5.4 examines successful executions over 100 training steps. State–plan alignment uses strict matching between local states and plan subgoals; subgoal coverage is the fraction of plan subgoals matched during execution. Scores are averaged first over successful executions of each plan, then equally across plans. The analysis includes 2,181 plans and 3,364 successful executions. Figure 4 shows empirical interquartile ranges.

For explicit tool errors in successful rollouts, a correction attempt requires an error-related reflection, an immediate interaction purpose aligned with the subsequent decision, and a change from the preceding decision. Error recovery rate is the fraction of these attempts followed by tool feedback reporting no error; this does not establish that the original error was resolved. Attempts followed by a user response remain in the denominator but do not count as error-free tool follow-ups. We pool events in consecutive 20-step windows; early and late denote the first and last. These windows include 4,044 attempts and 768 error-free tool outputs.

D Additional Results and Analyses

D.1 Conversational domain profile

Figure 6 breaks down the original conversational evaluation by domain, separately from the four-seed avg@4 results. The 4B model improves or matches the strongest comparison in every domain. For 7B and 8B, improvements are concentrated in Airline and Retail.

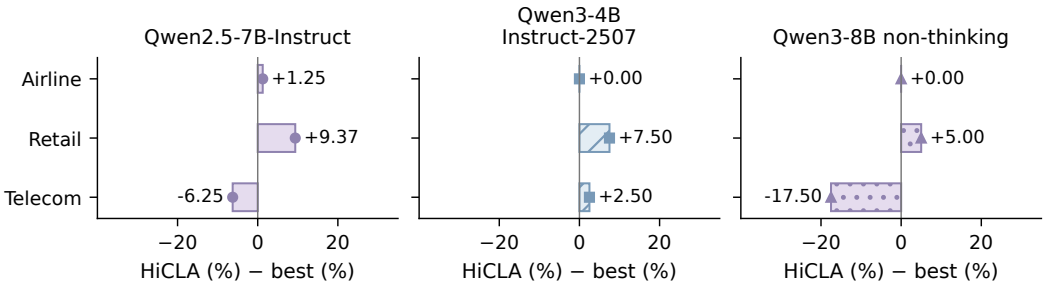

Figure 6: **Conversational gains across domains.** Each bar subtracts the strongest backbone or baseline success rate (%) from HiCLA's rate in the original domain evaluation; this is not a breakdown of avg@4. Positive and negative bars indicate gains and losses, respectively.

D.2 Function-calling profile

Figure 7 reports accuracy by function-calling category. The 7B model leads in Long Context, while the 4B model leads in Base and Missing Functions. These category-level strengths contribute to their overall gains on BFCL-V3. For the 8B model, overall accuracy is 35.25%, compared with 36.75% for GiGPO (Feng et al., 2025).

D.3 Local-subgoal persistence

Figure 8 gives the full trajectories of HiPER (Peng et al., 2026) discussed in Section 5.4. Appendix C.3 defines the sampling and text-comparison protocol.

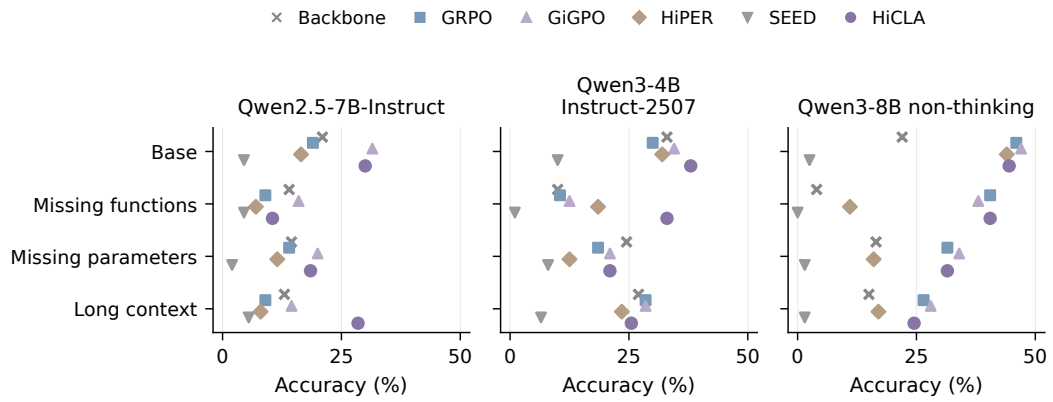

Figure 7: **BFCL-V3 category profiles across backbones.** Points show accuracy for each method and category. Shapes distinguish methods, with vertical offsets for readability.



Figure 8: **HiPER's switching decisions and subgoal changes.** SWITCH frequency and textual change between adjacent subgoals in our runs, sampled every five training steps.

D.4 Skill supervision and tool behavior

In our experiments with SEED (Wu et al., 2026a), Qwen2.5-7B-Instruct stops using tools late in training despite 15.55% training task success (Figure 9). The 4B and 8B runs continue calling tools.

SEED (Wu et al., 2026a) supervises skill generation during initialization and applies its RL-stage objectives to action tokens; shared parameters mean that policy optimization also changes the analyzer. HiCLA explicitly trains generated revised plans using their relative likelihood support for successful and failed executions. Removing this loss reduces success from 41.67% to 36.67% (Table 2).

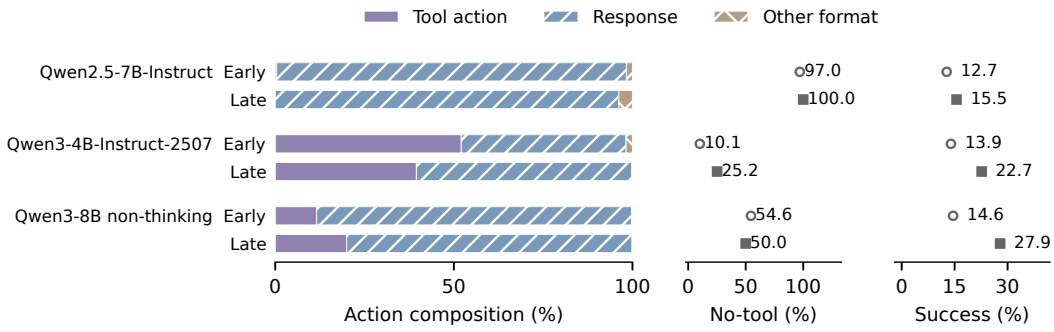

Figure 9: **Tool use and success under SEED.** Early (circles): training steps 1–20; Late (squares): 81–100. All rates are measured in our τ^2-bench training runs.

D.5 AUXILIARY SIGNAL DIAGNOSTICS

We analyze scalar summaries and paired block records from the available 4B training archive during the auxiliary-training phase, following task-only warm-up. Plan-coverage denominators include all sampled plan slots in this phase; conditional signal summaries use observations with defined auxiliary scores.

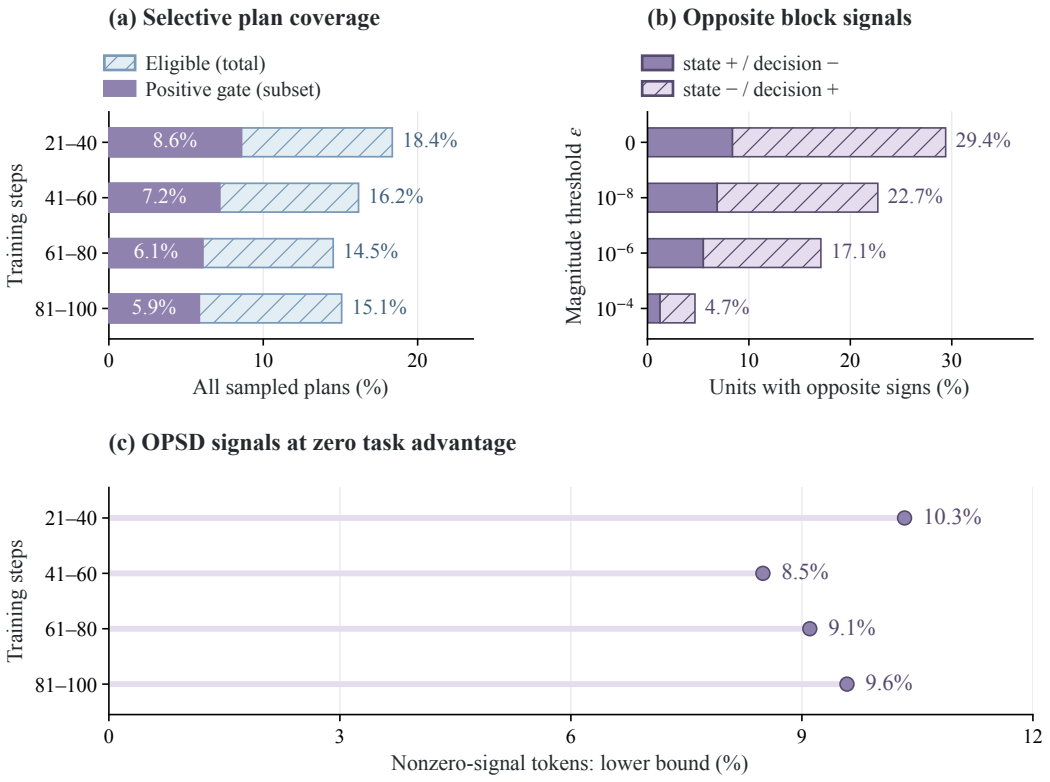

Figure 10: **Coverage and complementary OPSD signals during the auxiliary-training phase.** (a) Eligible plans and their positive-gated subset, each divided by all sampled plan slots per window. Eligibility precedes revised-plan validity checks. (b) Opposite-sign state/decision applied scores, split by sign direction. Both magnitudes must exceed the displayed threshold; every bar uses all positive-gated units as its denominator. (c) Lower bounds on nonzero execution-to-planning OPSD signals as percentages of zero-plan-task-advantage tokens within that objective's mask in each window. Horizontal stems are reading guides, not uncertainty intervals. The points establish nonzero supervision in each window, not the exact coverage or its trend. These are descriptive archive diagnostics.

Selective coverage across training. During the auxiliary-training phase, 16.04% of sampled global plans are eligible and 6.93% pass the positive gate. Figure 10(a) separates eligibility from this gated subset. From the first to the last training window, positive-gate coverage decreases from 8.59% to 5.86%, reflecting both a lower scored-plan fraction (18.36% to 15.08%) and a lower positive fraction among scored plans (46.81% to 38.86%). With domain sampling proportions fixed, Airline accounts for 82.86% of the decrease in positive-gated plan instances. However, mask-token coverage is 8.81%, 7.94%, 9.62%, and 8.51% across the four windows. Thus the decline in plan coverage does not imply a monotonic decline in token coverage or establish training convergence.

Blockwise signals at a shared task advantage. Within each positive-gated execution unit, state and decision share the same episode task advantage, yet their applied OPSD scores have opposite signs in 29.40% of units. Requiring both magnitudes to exceed 10^{-6} reduces this fraction to 17.10%; at 10^{-4}, it is 4.68% (Figure 10(b)). Separate block scoring therefore preserves differences that a single shared scalar cannot represent, although the prevalence of opposite signs depends on the magnitude threshold. This observation does not validate the direction of individual signals or measure the performance effect of separating the blocks.

When task advantages vanish. If both plans each produce one successful and one failed execution, their rewards are both 0.5, giving $A_1^P = A_2^P = 0$. Each plan nevertheless retains its own mixed evidence for token scoring. During the auxiliary-training phase, 75.37% of planner token positions have zero task advantage, including positions in ineligible plans.

We restrict the following analysis to zero-advantage tokens inside the execution-to-planning OPSD mask. Exact joint token records are unavailable, but per-step subset counts bound their nonzero supervision. Let N_u^{mask}, $N_u^{\neq 0}$, and $N_u^{A^P=0}$ count mask tokens, their nonzero-signal subset, and their zero-task-advantage subset at training step u. Summing the intersection lower bound $\max(0, N_u^{\neq 0} + N_u^{A^P=0} - N_u^{\text{mask}})$ and dividing by $\sum_u N_u^{A^P=0}$ gives an overall nonzero-signal proportion of at least 9.48%. The four window-specific lower bounds are 10.33%, 8.49%, 9.10%, and 9.59% (Figure 10(c)). These bounds establish extra supervision at zero task advantage, not its exact coverage or a nonzero net parameter change. Uniformly successful or failed plan executions remain ineligible. Executor auxiliary coverage requires mixed outcomes, under which its trajectory advantages are nonzero for the reward ranges in Appendix A.1.

Association with revised plans. Across auxiliary-stage training steps with defined signal summaries, edited plan positions have larger mean absolute signals than unchanged positions, with a mean per-step ratio of 2.30 (Figure 11). Changed positions correspond to original-plan tokens deleted or replaced by the revised plan, or marked at insertion boundaries. Positions within eight tokens of an edit form the near-edit category; the remaining positions are unchanged. This aggregate association can reflect differences between plans and does not establish discrimination within an individual plan. Model-generated edits are not independent error annotations.

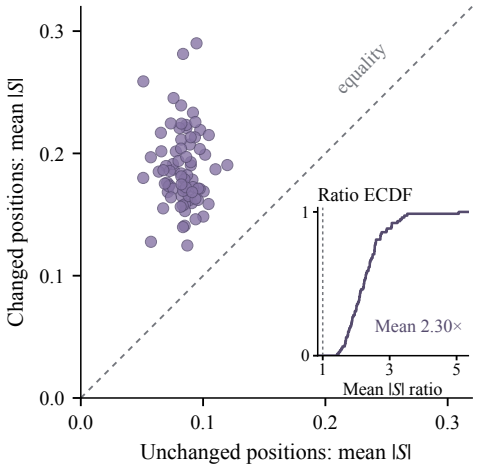

Figure 11: **Aggregate association with revised plans.** Each point compares mean absolute planner signals at changed and unchanged positions in an auxiliary-stage training step with defined signal summaries. The inset gives the empirical ratio distribution. These summaries do not resolve within-plan credit.

Descriptive distributions and evidential limits. Figure 12 separates signal frequency from magnitude. Within the planner OPSD mask, changed tokens have a higher negative-signal rate than unchanged tokens (51.85% versus 28.30%), and a higher positive rate (27.37% versus 18.56%). These are pooled token frequencies, not frequencies inferred from the sign of a per-step mean. The categories describe edits, not independently verified mistakes. The paired block records above establish differences at a shared task advantage, while the token-count bounds establish additional supervision at some zero-advantage positions. The ablations assess utility; these diagnostics do not establish causal attribution or validate individual signals.

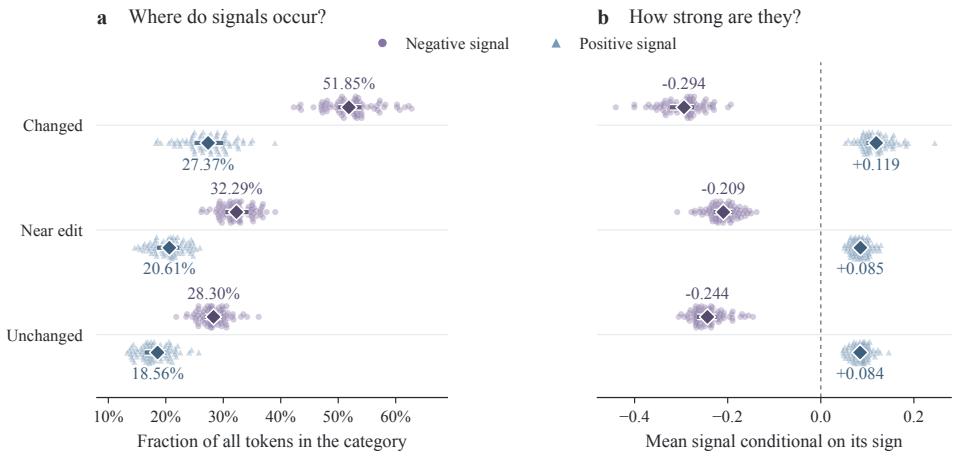

Figure 12: **Planner signal distributions by edit location.** (a) Positive and negative rates include zero-score positions in each denominator. (b) Conditional means among positive or negative positions, before coefficient scaling. Small points represent training steps; diamonds pool token positions, and intervals show empirical interquartile ranges across training steps, not confidence intervals.

D.6 Training efficiency

We compare 100 training steps from one Qwen3-4B-Instruct-2507 run per method on eight NVIDIA H800 GPUs. Figure 13 includes only timed training stages, excluding evaluation, checkpoint saving, and unattributed intervals. Stage shares use summed recorded time, with nested timers counted only once. Rollout time includes interaction and waiting. HiCLA uses task-only training for the first 20 steps.

The methods differ in where they spend training time. For HiCLA, rollout accounts for roughly two-thirds of the recorded time, followed by actor optimization. HiPER completes rollout faster (116.3 versus 163.4 seconds), but also spends about a quarter of its time on critic evaluation and optimization. This helps explain why a shorter rollout does not translate into a shorter overall training step. The gap is modest after HiCLA's task-only warm-up: over steps 21–100, HiCLA and HiPER average 244.0 and 256.1 seconds, respectively.

SEED's largest cost is actor optimization, which takes nearly half of its recorded time. Its batches contain substantially more input and response tokens than HiCLA's (12.95 million versus 1.75 million on average), consistent with a larger optimization workload. The timings do not isolate how much individual losses or teacher evaluation contribute, and token counts should not be interpreted as FLOPs.

Mean times also depend on occasional long training steps. For example, HiCLA has a lower mean than GiGPO but a higher median (215.1 versus 199.2 seconds). Figure 13 therefore shows the full distributions alongside the means. These describe variation within one run, rather than uncertainty across training seeds; the partial timings do not establish total GPU-hour or monetary costs.

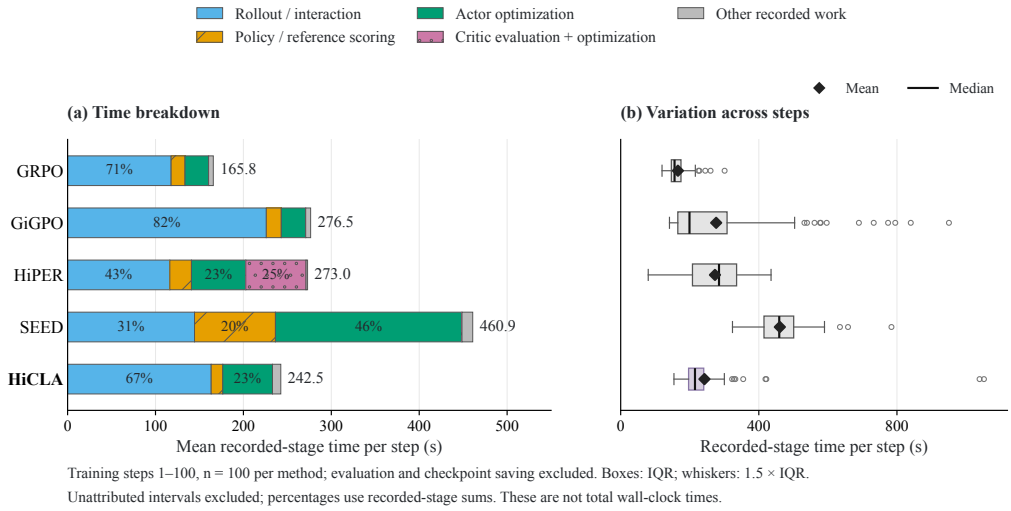

Figure 13: **Recorded training-stage times on Qwen3-4B-Instruct.** (a) Mean stage durations over 100 training steps; bar labels show shares of recorded time. (b) Per-step distributions from one run per method: boxes show interquartile ranges, lines show medians, and diamonds show means. Whiskers extend to observations within 1.5 interquartile ranges; all outliers are retained. Evaluation, checkpoint saving, and unattributed intervals are excluded.

E CASE STUDIES

The first three cases show how plans, local states, and decisions relate in successful training rollouts. We preserve the source wording: [...] marks omissions, bracketed identifiers are aliases, and bold text highlights relevant changes. Tool actions and outputs contain selected fields; context rows summarize prior interactions.

E.1 CANCEL TWO ORDERS, THEN RETURN THE WATCH

Table 4: A subgoal spans several decisions. The agent retains the cancellation subgoal while the *Next* field changes between orders, then resumes the watch return specified in the global plan.

Source	Trajectory excerpt
Task and global plan	
User	Hi, I need help canceling all my pending orders and also returning a watch I just received. I'd like to know how much I can get back in total.
Plan	**Subgoal 2.** [...] Retrieve all pending orders for the user, confirm their status, and cancel each with reason 'no longer needed' or 'ordered by mistake' after explicit user confirmation. Then, retrieve details of the delivered order containing the watch to initiate its return and calculate total refund. [...]
Context	Authentication and order lookups identify pending orders [A] and [B], and delivered order [C] containing the watch. The user confirms the cancellations:
User	Yes, please cancel both pending orders. For both of them, the reason is "no longer needed." [...]
1. Cancel the first pending order	
Subgoal	Cancel both pending orders with reason 'no longer needed' after explicit user confirmation.
Next	Cancel pending **order [A]** with reason 'no longer needed' after explicit user confirmation.
Action	`cancel_pending_order` {"order_id": "[A]", "reason": "no longer needed"}
Tool	status: "cancelled"; payment history: refund $3,131.10
2. Keep the subgoal; act on the remaining order	
Subgoal	Cancel both pending orders with reason 'no longer needed' after explicit user confirmation.
Reflection	Pending order [A] has been successfully canceled with reason 'no longer needed'. Next, we need to cancel pending order [B] with the same reason.
Next	Cancel pending **order [B]** with reason 'no longer needed' after explicit user confirmation.
Action	`cancel_pending_order` {"order_id": "[B]", "reason": "no longer needed"}
Tool	status: "cancelled"; payment history: refund $4,777.75
3. The cancellations are complete; the watch return remains	
Reflection	Both pending orders [A] and [B] have been successfully canceled with reason 'no longer needed'. **Next, we need to initiate the return of the watch** from delivered order [C] and calculate the total refund amount for all actions.
Next	Initiate the return of the watch (item_id: [watch]) from delivered order [C] and calculate the total refund amount.
Action	`return_delivered_order_items` {"order_id": "[C]", "item_ids": ["[watch]"]}
Tool	status: "return requested"
Outcome	**Next:** Calculate the total refund amount. `calculate`(3131.1 + 4777.75 + 367.38) → 8276.23

E.2 AN INVALID REMOVAL BECOMES A CONFIRMED REPLACEMENT

The tool rejects an empty replacement list because the numbers of original and replacement items must match. The agent acknowledges this constraint, obtains the user's approval to find a cheaper replacement, and changes the arguments of its next modification request.

Table 5: Revising an operation after tool feedback. Following a rejected removal request, reflection and user confirmation precede a one-to-one item replacement.

Source	Trajectory excerpt
Context: a new retail request emerges during the conversation	
Context	The initial user question concerns a poem. The initial plan includes the subgoal below, and the agent declines that question. The user then turns to a retail request.
Plan	[...] Deny the user's request as it does not pertain to any supported retail task and is outside the policy domain. [...]
User	[...] Let's move on to my order. I recently bought a bluetooth speaker from you. [...]
Context	After authentication and order lookup, the agent reports a pending order containing a Bluetooth speaker priced at $302.67. The user asks to leave the rest of the order unchanged.
User	**I don't want to cancel my entire order—just remove the bluetooth speaker from it.** The reason is that I thought it was cheaper, so please mark it as "no longer needed." Could you also let me know if there are any bluetooth speakers under $300 available?
1. Attempt to remove the speaker without a replacement	
Next	Proceed with modifying the order items, removing the Bluetooth speaker.
Action	`modify_pending_order_items` {"item_ids": ["[old speaker]"], "new_item_ids": []}
Tool	**Error: The number of items to be exchanged should match**
2. Use the error to propose a feasible alternative	
Reflection	[...] However, the policy does not support removing an item without replacement. We must instead clarify that item removal is not supported directly; **modification requires a replacement of the same product type.**
Next	Inform the user that we cannot remove the Bluetooth speaker without replacing it with another option, as per the policy. Suggest alternatives under $300.
Response	I'm sorry, but we cannot remove the Bluetooth speaker from your order without replacing it with another item of the same product type. [...] Please confirm if you'd like me to proceed with that.
User	Yes, please check for any bluetooth speakers under $300. If there are some available, **please add the cheapest one to my order instead.**
3. Find the cheapest variant and change the operation	
Subgoal	Find available Bluetooth speakers under $300 and identify the cheapest one.
Action	`get_product_details` {"product_id": "[speaker product]"}
Context	The product lookup returns available variants. The agent identifies the cheapest available speaker at $271.89; [old speaker] and [new speaker] below denote the original and replacement item identifiers.
Next	Proceed with modifying the order to replace the current Bluetooth speaker with this cheaper option.
Action	`modify_pending_order_items` {"item_ids": ["[old speaker]"], "new_item_ids": ["[new speaker]"]}
Tool	status: "pending (item modified)"; payment history: refund $30.78

E.3 EXCELLENT SIGNAL DOES NOT YET SATISFY THE SPEED REQUIREMENT

The plan specifies excellent mobile data speed as a completion condition. After the user reports excellent signal but a speed test below the requested level, the agent incorporates the user's VPN observation into its reflection and requests another test after the VPN is disconnected.

Table 6: The completion condition is checked against observed speed. The user supplies the device reports, the VPN suggestion, and both speed measurements; the agent guides the intervening actions.

Source	Trajectory excerpt
Task and completion condition	
User	Hi there—my mobile data on my phone keeps cutting out or is really slow. I need to get excellent internet speed on my phone. Can you help me fix this? I don't have Wi-Fi to use.
Plan	**Subgoal 2, completion.** [...] Resolve the issue by adjusting network settings, toggling data, or optimizing network preferences, **ensuring the user has excellent mobile data speed.** [...]
Context	The user has enabled mobile data, disabled Data Saver, and changed the network preference to 4G/5G. The excerpt starts with their report of the new device status.
1. Check whether improved settings meet the goal	
User	Done. I set the network mode to 4G/5G preferred, and now the status bar shows: - Signal: excellent - Network: 5G - Data: enabled - VPN: connected [...]
Next	Instruct the user to run a speed test to confirm excellent internet speed.
Response	[...] To confirm the internet speed is excellent, please run the following command on your device: `run_speed_test()`. This will measure your current download speed and let us know if it meets excellent standards. Share the result with me.
2. The first measurement is below the requested level	
User	I ran the speed test, and it shows **27.50 Mbps**, rated as "Good." That's not excellent, so I'm still not satisfied—I really need excellent internet speed. Also, I noticed a VPN is connected, which could be slowing things down. Would you like me to disconnect the VPN and rerun the speed test? [...]
Reflection	The speed test result is "Good" (27.50 Mbps), which is **below the user's expectation of "excellent."** A connected VPN may be affecting performance. Disabling the VPN could improve speed. The user is willing to follow steps one at a time.
Next	Guide the user to disconnect the VPN and rerun the speed test.
Response	The VPN connection might be slowing down your internet speed. Let's disconnect it to see if it improves performance. Please run the following command on your phone: `disconnect_vpn()`. After that, let me know the result so I can guide you to rerun the speed test.
3. Confirm the action, then measure again	
User	VPN is now disconnected. [...] Ready for the next step. Would you like me to run the speed test again now?
Reflection	The VPN has been disconnected, and the signal is now on excellent 5G. The next step is to verify if the speed has improved to "excellent" after removing the VPN.
Next	Instruct the user to run a speed test to confirm excellent internet speed.
Response	Great! The VPN is now disconnected. Let's check your internet speed again. Please run the following command on your device: `run_speed_test()`. This will measure your current download speed and confirm if it now meets excellent standards. Share the result with me.
User	Great news—I ran the speed test, and it shows **275 Mbps**, rated as "Excellent." **That resolves my issue.** Thanks for the help!

E.4 INITIAL AND REVISED PLANS

Table 7 compares selected subgoals from two Retail plans and their revised plans at training step 37. Each revised plan uses generated terminal evidence from one successful and one failed execution of the same initial plan. These are training-time rescoring cases; Q is not executed in the environment.

Table 7: **Initial plans and evidence-conditioned revised plans.** Plan rows quote the indicated subgoal or completion field; bold text highlights selected changes and [...] marks omitted evidence. Rewards and gates are recorded values, rounded for display.

Source	Excerpt or recorded signal
A. T-shirt exchange: make the confirmation requirements explicit	
Evidence	H_+: The user's T-shirt exchange request was successfully processed, with the red XXL cotton T-shirt exchanged for a black XL cotton T-shirt. [...] H_-: The exchange of the T-shirt was successfully processed, but the evaluator reports that the task was not completed successfully.
Initial P	**Subgoal 4.** Obtain user confirmation for the exchange details and payment method for any price difference.
Revised Q	**Subgoal 4.** Obtain explicit user confirmation for the exchange details, **including the new size**, payment method for price difference, and **assurance that all items to be exchanged are included**.
Signal	$R_{\text{rev}} = +0.00378$, $\eta = 0.00378$: a small positive gate enables planning-to-execution OPSD.
B. Earbud exchange: elaboration receives a negative revised-plan reward	
Evidence	H_-: [...] However, the exchange confirmation step was not completed due to an incomplete response.
Initial P	**Subgoal 4, completion.** Present the exchange details, including item IDs, price difference, and payment method, and obtain explicit user confirmation ("yes") to proceed.
Revised Q	**Subgoal 4, completion.** Clearly outline the exchange options, **calculate any price difference**, specify the payment method for the difference, and request explicit user confirmation ("yes") **before proceeding with the exchange**.
Signal	$R_{\text{rev}} = -0.00262$, $\eta = 0$: the revised plan receives a negative training signal and supplies no planning-to-execution OPSD.

In Case A, Q makes the size and item scope explicit in the confirmation subgoal; the feedback does not establish the cause of failure. In Case B, P already requires confirmation, and the recorded failure contains an incomplete response. Elaborating this requirement does not receive positive relative likelihood support. Thus, generation alone does not activate executor supervision.

An additional hiking-boot and puzzle exchange case retains $Q = P$ verbatim, with $R_{\text{rev}} = \eta = 0$. Together, the cases illustrate editing, preservation, and reward-gated use of Q as a scoring context for executor feedback.