

EvoSTEER: ONLINE SELF-EVOLVING GRAPH ORCHESTRATION VIA REFERENCE-ANCHORED CREDIT ASSIGNMENT

Anonymous authors

Paper under double-blind review

ABSTRACT

In recent years, LLM-based multi-agent systems have been widely applied to orchestrate tool-using agents into executable communication graphs. However, existing self-evolving orchestration still faces key challenges, including *post-hoc evolution* that revises the team only after the trajectory ends, *credit diffusion* that gives every action the same terminal advantage under confounded baselines, and *skill admission* that is uncalibrated and never retired. To address these challenges, we propose EvoSteer, a new paradigm of Online Self-Evolving Graph Orchestration—the orchestrator builds a running team and repairs its plausible but failing steps from execution features and a learned value estimate. To support this paradigm, we introduce *Anchored Trajectory Balance* (AnchorTB), a regression-style flow-matching loss that assigns each orchestration action a coefficient by balancing subtrajectories against a frozen reference. Built on the learned flow, we further propose *Validated Skill Admission*, in which a candidate skill is tried before promotion and promoted only if paired evidence passes a sequential test under a shared nominal testing budget. Moreover, AnchorTB combines measured task-level reference reward statistics with prefix-dependent corrections. Experimental results on twelve datasets show that EvoSteer significantly outperforms baselines across question answering, mathematical reasoning, code generation, and interactive decision making. Our code is available at anonymous.4open.science/r/evosteer-C312.

1 INTRODUCTION

In recent years, a variety of powerful LLM-based multi-agent systems have been applied to solve a wide range of complex tasks (Yao et al., 2022b; Hong et al., 2024; Zhuge et al., 2024), gradually moving beyond a single model call toward teams of tool-using agents that complete tasks end-to-end.

In this process, graph orchestration has become a key bridge from task goals to reproducible execution: by assigning each node a role and bound skills and each edge a communication protocol on an agent communication graph (Fig. 1, left), agents can complete complex tasks with improved controllability, compositionality, and reusability (Zhang et al., 2025b; 2026b). However, in practice, orchestration still evolves only between trajectories, by revising skills, prompts, or topology after a batch of runs has ended (Li & Ramakrishnan, 2026; Pan et al., 2026), making a misconfigured team costly to detect, as its outputs still look plausible, and impossible to repair while the workflow is still being executed.

To address these issues, three main lines of work have emerged, as shown in Figure 2. First, *self-evolving orchestration* adapts topologies, role prompts, and skill libraries from the outcomes of

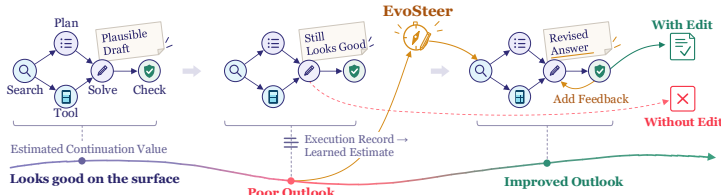


Figure 1: Outputs that look plausible can still be headed for failure. EvoSteer reads a low continuation estimate off the execution record and edits the running team, here routing the checker’s report back to the solver; dashed: the same team left unedited.

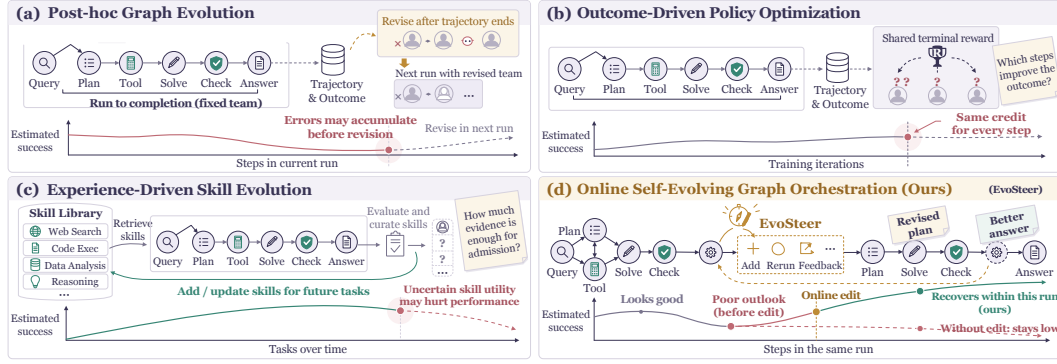


Figure 2: Three lines of work and ours. (a) Post-hoc evolution revises the team only after the run. (b) Outcome-driven optimization spreads one terminal reward over every step. (c) Skill evolution admits by judges or replay. (d) EvoSteer edits the team inside the same run, deciding from measured execution features and a reference value estimate, and admits a skill only after a sequential test.

completed trajectories (Li & Ramakrishnan, 2026; Pan et al., 2026; Zhang et al., 2026d), treating the team as an object of learning. Second, *credit assignment for multi-agent LLMs* converts a shared terminal reward into per-agent or per-message signals through learned critics, leave-one-out baselines, counterfactual replay, and flow-matching objectives (Chen et al., 2026c; Li et al., 2026b; Shah, 2026; Zhang et al., 2026c), telling which step was responsible for the outcome. Third, *skill libraries* let capability grow across tasks by extracting reusable procedures from trajectories and curating them with judges, replay checks, or posteriors (Wang et al., 2023; Yang et al., 2026a; Wu et al., 2026).

However, these methods still face three challenges. (i) **Post-hoc evolution.** Self-evolving orchestration revises the team after the trajectory ends, by rounds or batches (Pan et al., 2026; Li & Ramakrishnan, 2026); during execution the orchestrator can only keep adding agents, and the decisive error is locally indistinguishable from a recoverable step (Zhang et al., 2026a), so a wrong agent is neither rerun nor removed until the task has failed. (ii) **Credit diffusion.** Under a shared terminal reward, trajectory-level methods give every action the same coefficient of a single terminal advantage (Chen et al., 2026c; Zhang, 2026), and learned state values confound how often a task is solved, execution noise, and structural choice, so the orchestrator cannot tell whether a low score came from a task the reference rarely solves or from a badly organized team; counterfactual replay localizes the failing step (Shah, 2026; Bonagiri et al., 2026) but does not provide a training signal for every decision. (iii) **Uncalibrated skill admission.** Skills are admitted by LLM judgement or by success counts treated as reliable belief (Yang et al., 2026a; Wu et al., 2026); such point estimates cannot separate insufficient evidence from a confirmed effect, so an accidental strategy from one trajectory is consolidated (Chen et al., 2026a), and since no test tracks whether an admitted skill still helps on later tasks, it stays in the library and is never retired.

To address these challenges, we propose **EvoSteer** (Fig. 2d), a new paradigm of *Online Self-Evolving Graph Orchestration*, in which the orchestrator dynamically builds and repairs a running team from execution features and a learned value estimate—replacing the run-then-revise loop. The orchestrator issues one atomic edit per turn, rerun and drop included; each is executed at once, and its feedback and the value estimate enter the state, so a plausible but failing result can be redone mid-run. To support this paradigm, we introduce *Anchored Trajectory Balance* (AnchorTB), a regression-style flow-matching loss that assigns each orchestration action a coefficient by balancing subtrajectories against a frozen reference. It combines measured task-level reference reward statistics with prefix-dependent corrections. Built on the learned flow, we further propose *Validated Skill Admission*: a skill proposer distills candidates from scored experience, the orchestrator may try a candidate before promotion, and paired rollouts differing only in the candidate are judged by a sequential sign test under a shared nominal testing budget, which promotes, retires, or defers. Trained end-to-end on natural and paired trajectories with the reference frozen throughout, EvoSteer turns construction, repair, and skill growth into one set of learned decisions under a single training objective.

We evaluate on benchmarks across question answering, mathematical reasoning, interactive decision making, and code generation. Results show EvoSteer outperforms workflow search, flow- and RL-based orchestration training, and skill-evolution methods on every benchmark and improves all seven executors—a foundation for orchestration that evolves while it runs.

2 RELATED WORK

Agent Task Orchestration. Orchestration graphs are searched under execution feedback (Zhang et al., 2025b; 2026b). Later work evolves the team between runs (Dang et al., 2026; Hao et al., 2026), mutates topology between test-time rounds (Xu et al., 2026), repairs finished trajectories (Luan et al., 2026; Lu & Zhang, 2026; Zhao et al., 2026), or audits unfolding prefixes (Wang et al., 2026a; Jiang et al., 2026; Li et al., 2026d), but keeps the running team fixed. Credit comes from counterfactual replay (Chen et al., 2026b; Deshmukh et al., 2026), group-relative objectives (Mishra et al., 2026), or (sub)trajectory balance (Malkin et al., 2022; Madan et al., 2023; Venkatraman et al., 2024) and its extensions (Fawkes & Hartford, 2026; Liu et al., 2026; Wang et al., 2026b; Liang et al., 2026). EvoSteer reruns and drops agents in the running team and measures flows from a frozen reference.

Skill Self-Evolution. Skill libraries check candidates by replay (He & Yang, 2026), paired trajectories (Gao et al., 2026), leave-one-out attribution (Shen et al., 2026; Hu et al., 2026; Wang et al., 2026c), or success-count posteriors (Wu et al., 2026). Others couple per-skill credit to library edits (Yao et al., 2026), gate contaminated candidates (Zhang & Li, 2026), or gate self-modification with sequential or paired tests (Shawn, 2026; Sengupta, 2026; Shang & Yang, 2026). EvoSteer gates skills in a live episode under one testing budget per run, and candidates stay selectable while tested.

3 PRELIMINARIES

Definition 1: Typed Agent Communication Graph. A typed agent communication graph (henceforth communication graph) is a directed graph $G = (V, E, o)$ over agent nodes, where each node $v_i = (c_i, \mathcal{K}_i)$ carries a role c_i from a role catalogue and a set of bound skills $\mathcal{K}_i$ drawn from the visible active skill set, each edge (v_i, v_j, π_{ij}) carries a communication protocol π_{ij} that decides whether and how the source’s result revises the target, and $o \in V$ together with an output rule fixes the final answer (our typed edges and nodes extend the untyped topologies of Zhang et al. (2024; 2025a) and the computational graphs of Zhuge et al. (2024)).

Definition 2: Orchestration Trajectory. A complete sequence of T orchestration actions from the empty team to a legal stop defines an orchestration trajectory, in which every action is executed as soon as it is issued, before the next action is chosen:

$$\tau = \{(a_t, o_t^{\text{exec}})\}_{t=0}^{T-1} \Rightarrow r(x), \quad s_{t+1} = s_t \oplus (a_t, o_t^{\text{exec}}), \quad (1)$$

where a_t is an atomic edit with type in $\{\text{ADD_AGENT}, \text{ADD_EDGE}, \text{BIND_SKILL}, \text{SET_OUTPUT}, \text{RERUN_AGENT}, \text{DROP_AGENT}, \text{STOP}\}$, o_t^{exec} is the execution feedback returned by running the agent, communication, repair, or output that the action triggers, and x is the completed history $x = s_T$ with task reward $r(x) \in [0, 1]$ and $T \geq 1$. Action a_t is chosen at s_t and produces s_{t+1} , for $t = 0, \dots, T-1$. Because each state retains the full history, every non-root state has a unique parent. The deterministic feature record is suppressed here and made explicit in Eq. (3).

Problem Statement. Given a task q , a frozen executor with tools, a frozen reference policy ρ , and a skill library $\mathcal{S}$, we seek an orchestrator π_θ trained toward an ideal reward-proportional history distribution relative to ρ (Venkatraman et al., 2024):

$$P^*(x | C) = \frac{\rho(x | C) R_\beta(x)}{Z(C)}, \quad R_\beta(x) = 1 + (e^\beta - 1) r(x), \quad (2)$$

where $0 \leq \beta < \infty$, and C collects the task, tools, roles, and the skill and value-head snapshots of the current batch, $\rho(x | C)$ is the history measure induced by the reference policy and the executor’s stochastic transitions, and R_β keeps a positive mass for failures while weighting a full score e^β times a zero score. Equation (2) is the training target; any deterministic executor, and more generally one meeting the condition of Proposition A.10, lets action choices alone realize it.

4 METHODOLOGY: EVOSTEER

As illustrated in Figure 3, this section introduces the EvoSteer framework, including online self-evolving graph orchestration (Section 4.1), anchored trajectory balance (Section 4.2), and validated skill admission (Section 4.3), all built on the frozen reference ρ .

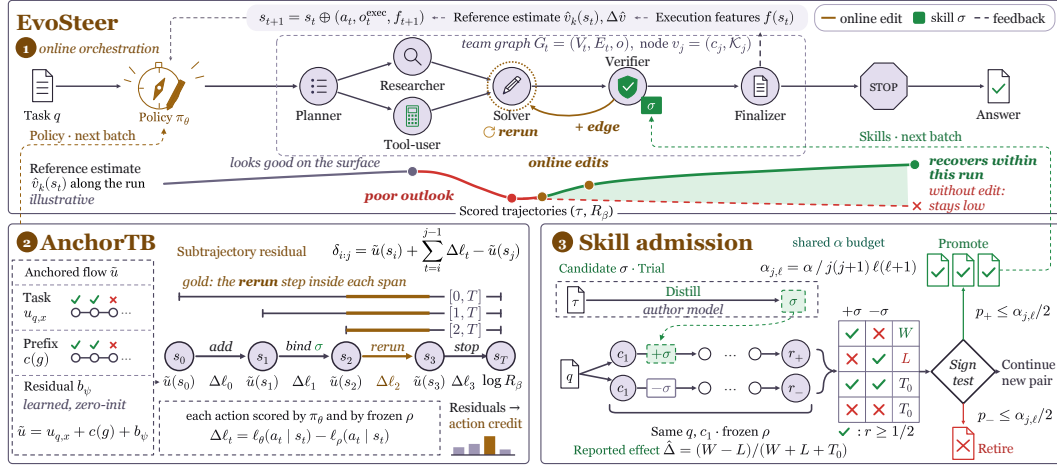


Figure 3: EvoSteer architecture. **Top:** π_θ builds and runs the team from execution features f_t and reference estimate $\hat{v}_k$; rerun and new edges are ordinary actions. **Bottom left:** AnchorTB balances subtrajectories against ρ ; $\tilde{u}, \delta$ abbreviate trajectory-specific flow estimates and residuals. **Bottom right:** paired rollouts and a sign test under a shared nominal α budget promote or retire candidate σ .

4.1 ONLINE SELF-EVOLVING GRAPH ORCHESTRATION

As shown in Figure 3 (top), EvoSteer follows an Orchestrator–Executor paradigm (Yao et al., 2022b; Zhu et al., 2026; Zhang et al., 2026b): a trainable orchestrator π_θ organizes tool-using agents, each on a frozen executor, inside a structured environment.

Structured Environment. The environment $\mathcal{E} = (\mathcal{P}, \mathcal{S}^{(k)}, \mathcal{R}_{\text{exec}}, \hat{v}_k)$ maintains the role catalogue $\mathcal{P}$, i.e., the roles with their tool and session permissions; the active skill set $\mathcal{S}^{(k)}$ visible in training step k (slot counts in Appendix C); the executor runtime $\mathcal{R}_{\text{exec}}$ with a shared resource budget; and the reference value head $\hat{v}_k$ of Eq. (5). The two snapshots $\mathcal{S}^{(k)}$ and $\hat{v}_k$ are fixed for a whole rollout batch, while the legal mask $\mathcal{A}(s)$ remains state-dependent.

Interleaved Execution. Each orchestrator action is validated, applied to the graph, and executed at once; its feedback and a fixed-length execution feature vector are appended to the state:

$$s_{t+1} = s_t \oplus (a_t, o_t^{\text{exec}}, f_{t+1}), \quad f_{t+1} = f(s_t, a_t, o_t^{\text{exec}}) \in \mathbb{R}^{30}, \quad (3)$$

where o_t^{exec} is the execution feedback of Definition 2 and f records graph size, action type, repair outcome, whether agents answered and agreed, communication revision, environment phase, and budget use. These features need no extra model call and cover the execution-side failure modes catalogued for multi-agent systems (Cemri et al., 2026); $f(s)$ denotes the latest vector in s .

Learnable Repair. Repair shares one action space and one policy $a_t \sim \pi_\theta(\cdot | s_t)$ with construction:

$$\mathcal{A}(s_t) \subseteq \mathcal{A}_{\text{con}} \cup \mathcal{A}_{\text{rep}} \cup \{\text{STOP}\}, \quad \mathcal{A}_{\text{rep}} = \{\text{RERUN_AGENT}, \text{DROP_AGENT}\}, \quad (4)$$

where $\mathcal{A}_{\text{con}} = \{\text{ADD_AGENT}, \text{ADD_EDGE}, \text{BIND_SKILL}, \text{SET_OUTPUT}\}$ holds the construction edits of Definition 2, and the mask $\mathcal{A}(s_t)$ (Li et al., 2026a) follows from the current graph, roles, skill slots, output modes, and budget, admitting repairs only when valid. One objective trains all three kinds of decision: how many agents a task needs, when a result is worth redoing, and when the answer in hand suffices. Team size and repair count follow from the policy under the shared budget, and the terminal reward $r(x)$ is computed once, after the stop, on the output node’s answer.

In-Loop Reference Value Estimate. A lightweight head uses execution features and task type to estimate reference continuation reward by squared-error regression:

$$\hat{v}_\omega(s) = \sigma(\text{MLP}_\omega[f(s); \text{onehot}(\text{tasktype}(q))]), \quad \mathcal{L}_v = \mathbb{E}_{(s,r) \sim \mathcal{D}_\rho} (\hat{v}_\omega(s) - r)^2, \quad (5)$$

where MLP_ω is a two-layer head on 30+6 inputs (Appendix B), $\mathcal{D}_\rho$ holds states of legal reference continuations with their terminal rewards, and $\hat{v}_k$ in $\mathcal{E}$ is the snapshot of $\hat{v}_\omega$ frozen for batch k . The estimate and its one-step change are shown to the orchestrator as feedback; both are deterministic readouts of the retained history under the frozen head and carry cross-task experience into in-loop decisions, including when to stop. Its population MSE optimum is the feature-conditional mean of the terminal reward under the training data law, and this optimum is calibrated (Proposition B.8).

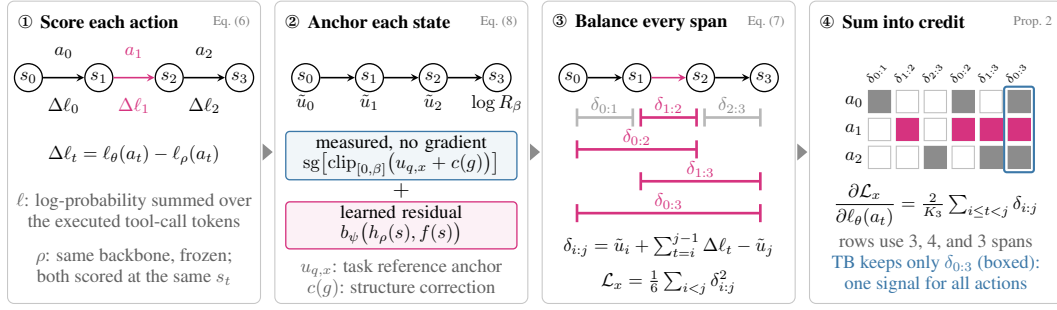


Figure 4: AnchorTB on a three-action history: ① score each action against the frozen reference; ② anchor each state with a measured, stop-gradient anchor (blue) and a learned residual (pink); ③ form all $K_3 = 6$ span residuals; ④ sum each action’s spans into its coefficient.

Proposition 1. *Executing every action as it is issued and placing rerun and drop in the action space lets the orchestrator edit and retry a running team through the same action interface. Proof.* Appendix C.6, via the history tree and legal actions of Lemma A.1.

4.2 ANCHORED TRAJECTORY BALANCE

As shown in Figure 3 (bottom left) and step by step in Figure 4, we train π_θ on the history tree of Section 3 toward the ideal reward-proportional target of Eq. (2), relative to the frozen reference.

Action Log-Ratio. An orchestrator action is a tool call of K_t tokens sampled under a grammar mask; its log-probability and its log-ratio to the reference are computed on the executed tokens:

$$\ell_\theta(a_t | s_t) = \sum_{j=1}^{K_t} \log P_\theta(a_{t,j} | s_t, a_{t,<j}), \quad \Delta \ell_t = \ell_\theta(a_t | s_t) - \ell_\rho(a_t | s_t), \quad (6)$$

where ρ is the same backbone without the trainable adapter, scored under the same context, tokens, and mask. Positions with a single legal token contribute zero, so only real choices enter the ratio.

Subtrajectory Residual and Loss. Every non-root state has a unique parent, so the backward policy is identically one. Motivated by reference-relative subtrajectory balance (Madan et al., 2023; Venkatraman et al., 2024), we regress each residual toward zero:

$$\delta_{i,j}^{(x)} = \tilde{u}_x(s_i) + \sum_{t=i}^{j-1} \Delta \ell_t - \tilde{u}_x(s_j), \quad \tilde{u}_x(s_T) = \log R_\beta(x), \quad \mathcal{L}_x = \frac{1}{K_T} \sum_{0 \leq i < j \leq T} (\delta_{i,j}^{(x)})^2, \quad (7)$$

where nonterminal flows use Eq. (8) with the same trajectory index x throughout (batch index suppressed), the terminal value overrides the learned parameterization, and $K_T = T(T+1)/2$ counts the equally weighted subtrajectories for $T \geq 1$, so each action’s coefficient sums the residuals of the spans containing it (Figure 4, ④).

Measured State Flows. The ideal relative log flow $u^*(s) = \log \mathbb{E}_\rho[R_\beta(x) | s, C]$ satisfies $u^*(s_0) = \log Z(C)$. A shared flow with zero residuals recovers u^* and P^* (Proposition A.8); this motivates reference anchors with a learned residual:

$$\tilde{u}_x(s) = \text{sg}[\text{clip}_{[0, \beta]}(u_{q,x} + c(g(s)))] + b_\psi(h_\rho(s), f(s)), \quad u_{q,x} = \log[1 + (e^\beta - 1) \hat{p}_q^{(-x)}], \quad (8)$$

For nonterminal s , $\hat{p}_q^{(-x)}$ is a task-level partial leave-one-out shrinkage estimate of the reference mean reward on the same task, computed from reference rollout outcomes (Appendix B); $c(g)$ is a correction read one batch behind from a bucket of canonical prefix structures g , $h_\rho(s)$ the frozen reference encoding, and b_ψ a zero-initialized MLP. Under leave-one-out, one prefix can differ across scored trajectories. The correction averages reward ratios over prefix visits before the logarithm,

$$y_x = \frac{R_\beta(x)}{e^{u_{q,x}}} - 1, \quad c(g) = \text{clip}_{[-c_{\max}, c_{\max}]} \log\left(1 + \frac{\sum_{(x,t) \in \mathcal{V}_g} y_x}{n_g + n_0}\right), \quad (9)$$

where $\mathcal{V}_g$ is the multiset of eligible reference prefix visits to structure g , paired prefixes included (Appendix C), $n_g = |\mathcal{V}_g|$, and $n_0 \geq 0$ with $n_g + n_0 > 0$. The pseudo-count shrinks rare structures toward zero correction and $c_{\max}$ bounds it (values in Appendix C), and the logarithm’s argument

stays positive (Lemma B.6). The measured terms are held fixed, so only π_θ and b_ψ are updated, with b_ψ carrying the remainder.

Proposition 2. *AnchorTB assigns each orchestration action a regression coefficient from all subtrajectories containing it, and combines a task-level reference anchor with prefix-dependent corrections. Proof.* Appendix C.6, via the coefficient of Proposition A.14 and the corrections of Appendix B.

4.3 VALIDATED SKILL ADMISSION

Prior work admits a skill from a single trajectory, by a judge, by counting successes, or by a trust hierarchy over verifiers (Yang et al., 2026a; Chen et al., 2026a; Wu et al., 2026; Shang et al., 2026), leaving underspecified *whether* a candidate helps and *when* it may be promoted. EvoSteer answers both inside the loop: a candidate is an ordinary entry of the action set in Eq. (4) that the orchestrator may bind while its evidence accumulates, and that evidence is measured under the frozen reference of Section 4.2 (Figure 3, bottom right), so credit and admission share one reference.

Skill Proposal and Action-Set Integration. In an author window, an independent author model distills at most one candidate σ , a structured procedure, from scored, side-effect-free trajectories (proposal rules and skill format in Appendix C). A candidate occupies a candidate slot of $\mathcal{S}^{(k)}$, so the action mask of Eq. (4) already exposes it to ADD_AGENT and BIND_SKILL before it is validated; the full skill text is rendered only when the bound agent executes.

Paired Counterfactual Rollouts. For every batch task in a family with a candidate, two extra trajectories are launched from the same record and the same forced first role, one binding σ and one not, and both are continued by the reference policy:

$$x^+ \sim \rho(\cdot \mid C^+, \text{ADD_AGENT}(c_1, \sigma)), \quad x^- \sim \rho(\cdot \mid C^-, \text{ADD_AGENT}(c_1, \emptyset)), \quad (10)$$

where c_1 is the forced first role. The arm contexts $C^\pm$ share the initial task, runtime, and value-head snapshot; C^- masks σ throughout. Pairs are scored by the terminal reward of Eq. (2); with $b^\pm = \mathbb{I}[r(x^\pm) \geq \frac{1}{2}]$, the comparisons accumulate as wins $W = \sum \mathbb{I}[b^+ > b^-]$, losses $L = \sum \mathbb{I}[b^+ < b^-]$, and ties $T_0 = \sum \mathbb{I}[b^+ = b^-]$ over complete, side-effect-free pairs, which also enter AnchorTB as off-policy paths, re-scored under each arm’s own context and legal mask (Appendix C).

Sequential Validation. At validation rounds between batches, each candidate with new wins or losses gets one binomial sign-test tail per direction, exact under the fixed-look sign model (Appendix C):

$$p_+ = \mathbb{P}\{\text{Bin}(D, \frac{1}{2}) \geq W\}, \quad p_- = \mathbb{P}\{\text{Bin}(D, \frac{1}{2}) \geq L\}, \quad D = W + L, \quad (11)$$

with $p_+ = p_- = 1$ when $D = 0$. A candidate whose p_+ clears its boundary is promoted to validated, one whose p_- clears it is retired, and otherwise it keeps accumulating. One budget covers the whole run on two levels, and the same counts give the reported effect:

$$\alpha_{j,\ell} = \frac{\alpha}{j(j+1)\ell(\ell+1)}, \quad \hat{\Delta} = \frac{W - L}{W + L + T_0}, \quad (12)$$

where $\hat{\Delta}$ is reported only when $W + L + T_0 > 0$ and measures the paired threshold-pass-rate difference. Here j globally indexes the registered comparison and ℓ its observation; each direction receives $\alpha_{j,\ell}/2$, and the budget is spent only when new evidence arrives, so the total nominal level is at most α (Appendix C). By a union bound, family-wise error is at most α whenever each directional p-value is valid for the selection and sampling rules in use, even for dependent tests.

Proposition 3. *Validated Skill Admission lets a candidate be tried before it is promoted, and promotes or retires it only when paired evidence passes a sequential test under a shared alpha-spending budget. Proof.* Appendix C.6, via the candidate-slot and status-transition rules, the nominal budget accounting of Proposition C.5, and the conditional FWER result of Proposition C.6.

5 EXPERIMENTS

We evaluate EvoSteer through the following research questions (RQs). **RQ1:** How does EvoSteer compare with the baselines in distribution? **RQ2:** Does it generalize to held-out benchmarks? **RQ3:** Does it transfer across executor backbones? **RQ4:** What does each component contribute, and how do fixed orchestration paradigms compare? **RQ5:** How does AnchorTB compare with other objectives, and do the measured anchor, validated admission, and in-run edits each behave as designed?

		Baseline		SFT	GRPO [†]		AFlow		Agent+RL		Skill evolution		Ours
Dataset	Metric	Qwen3.5	v4-flash	Qwen3.5	Qwen3.5	Qwen3.5	AgentFlow	FlowSteer	SkillFlow	SkillOpt	EvoSteer (Δ ↑)		
(a) In-Distribution (IID) benchmarks													
HotpotQA	Ans EM	57.66 ±1.28	70.94 ±0.35	63.59 ±1.18	69.22 ±0.43	87.34 ±0.86	86.09 ±0.86	89.22 ±0.65	89.06 ±0.00	88.28 ±0.00	92.34 ±0.35 (+34.7)		
	Ans F1	73.75 ±1.24	82.33 ±0.47	77.70 ±1.23	81.64 ±0.50	88.97 ±0.85	88.42 ±0.88	90.30 ±0.66	92.00 ±0.11	90.34 ±0.44	92.84 ±0.35 (+19.1)		
NQ-Open	Ans EM	23.59 ±0.86	38.59 ±1.31	25.78 ±0.00	24.22 ±1.10	76.56 ±0.78	81.41 ±1.28	79.53 ±1.28	82.66 ±0.86	82.34 ±1.31	85.00 ±0.65 (+61.4)		
	Ans F1	33.04 ±1.22	49.60 ±1.78	37.69 ±0.14	32.59 ±1.45	80.77 ±0.82	85.46 ±1.75	84.31 ±1.36	86.56 ±0.80	86.28 ±1.27	88.92 ±0.69 (+55.9)		
MedQA	Acc.	71.41 ±0.43	84.53 ±0.35	73.13 ±0.43	77.66 ±0.43	89.22 ±0.65	88.44 ±0.86	90.31 ±1.18	91.41 ±0.78	91.72 ±0.70	93.13 ±0.86 (+21.7)		
AIME 2026	Acc.	48.67 ±2.98	50.67 ±1.49	32.67 ±2.79	32.00 ±1.83	53.33 ±0.00	60.67 ±2.79	64.67 ±2.98	63.33 ±2.36	66.00 ±1.49	74.67 ±1.83 (+26.0)		
MBPP+	Pass@1	78.91 ±1.10	84.53 ±0.86	78.44 ±0.70	81.88 ±0.35	85.00 ±0.65	86.09 ±0.65	87.34 ±0.86	88.59 ±0.70	90.16 ±0.43	92.19 ±1.10 (+13.3)		
ALFWorld	SR	46.88 ±1.24	60.47 ±0.43	40.31 ±0.70	50.47 ±0.70	73.28 ±0.65	76.09 ±0.43	81.56 ±0.70	83.28 ±0.70	85.78 ±0.86	89.69 ±1.02 (+42.8)		
Avg. (IID)	Ans EM	40.63 ±0.77	54.77 ±0.68	44.69 ±0.59	46.72 ±0.59	81.95 ±0.58	83.75 ±0.77	84.38 ±0.72	85.86 ±0.43	85.31 ±0.65	88.67 ±0.37 (+48.0)		
	Ans F1	53.39 ±0.87	65.97 ±0.92	57.70 ±0.62	57.12 ±0.77	84.87 ±0.59	86.94 ±0.98	87.31 ±0.76	89.28 ±0.40	88.31 ±0.67	90.88 ±0.39 (+37.5)		
	Acc.	61.46 ±0.86	70.05 ±0.45	56.14 ±0.75	60.50 ±0.51	75.21 ±0.28	77.82 ±0.76	80.97 ±0.85	81.65 ±0.67	83.41 ±0.48	87.42 ±0.63 (+26.0)		
(b) Out-of-Distribution (OOD) benchmarks													
TriviaQA	Ans EM	44.22 ±0.43	70.31 ±0.00	46.09 ±0.55	47.34 ±1.18	90.94 ±1.05	89.06 ±0.00	91.25 ±0.35	90.16 ±1.05	92.34 ±1.28	95.63 ±1.18 (+51.4)		
	Ans F1	53.46 ±0.52	80.81 ±0.21	58.77 ±0.70	58.50 ±1.45	92.34 ±1.06	90.44 ±0.04	92.71 ±0.35	91.79 ±1.07	93.41 ±1.30	96.81 ±1.23 (+43.4)		
MuSiQue	Ans EM	39.22 ±0.86	50.00 ±0.78	39.06 ±0.55	40.63 ±0.55	77.66 ±0.70	79.06 ±1.16	80.00 ±1.18	80.94 ±0.43	81.09 ±0.65	84.53 ±0.86 (+45.3)		
	Ans F1	49.12 ±1.17	58.73 ±0.72	50.79 ±0.95	50.09 ±0.68	82.70 ±0.74	84.36 ±1.24	85.74 ±1.47	84.82 ±0.45	86.09 ±0.69	87.84 ±0.89 (+38.7)		
GPQA	Acc.	61.72 ±1.10	73.91 ±0.43	68.91 ±0.86	67.50 ±0.89	75.63 ±1.02	79.69 ±0.96	83.75 ±0.65	82.19 ±0.86	81.88 ±0.35	85.63 ±0.70 (+23.9)		
MATH-Hard	Acc.	89.06 ±0.78	92.19 ±0.55	88.59 ±1.31	87.50 ±0.78	91.09 ±1.05	92.66 ±0.70	93.44 ±0.70	92.34 ±1.02	94.69 ±0.65	95.47 ±0.86 (+6.4)		
SWE-Bench	Resolved	15.94 ±0.43	36.25 ±0.70	15.00 ±0.86	16.88 ±1.05	28.28 ±0.65	38.75 ±0.70	40.31 ±0.70	39.84 ±0.78	41.56 ±0.65	42.50 ±0.70 (+26.6)		
WebShop	SR	32.66 ±0.86	65.47 ±0.86	32.81 ±0.78	36.25 ±0.70	59.53 ±0.86	76.09 ±0.43	78.28 ±0.86	82.81 ±1.10	85.94 ±0.78	90.63 ±0.00 (+58.0)		
Avg. (OOD)	Ans EM	41.72 ±0.48	60.16 ±0.39	42.58 ±0.39	43.98 ±0.65	84.30 ±0.63	84.06 ±0.58	85.63 ±0.62	85.55 ±0.57	86.72 ±0.72	90.08 ±0.73 (+48.4)		
	Ans F1	51.29 ±0.64	69.77 ±0.38	54.78 ±0.59	54.29 ±0.80	87.52 ±0.65	87.40 ±0.62	89.22 ±0.76	88.31 ±0.58	89.75 ±0.74	92.32 ±0.76 (+41.0)		
	Acc.	49.84 ±0.41	66.95 ±0.33	51.33 ±0.49	52.03 ±0.43	63.63 ±0.45	71.80 ±0.36	73.95 ±0.37	74.30 ±0.47	76.02 ±0.31	78.55 ±0.33 (+28.7)		

Table 1: Main results (five-run mean $\pm$ std). All methods run on the Qwen3.5-9B executor except v4-flash (DeepSeek-V4-Flash); GRPO[†] trains the backbone; $\Delta \uparrow$: gain over Qwen3.5-9B.

5.1 EXPERIMENTAL SETUP

Datasets. Six in-distribution (IID) benchmarks supply the training tasks and IID tests: HotpotQA (Yang et al., 2018), NQ-Open (Kwiatkowski et al., 2019), MedQA (Jin et al., 2021), AIME 2026, MBPP+ (Liu et al., 2023), and ALFWorld (Shridhar et al., 2020). Six out-of-distribution (OOD) benchmarks are held out: TriviaQA (Joshi et al., 2017), MuSiQue (Trivedi et al., 2022), GPQA (Rein et al., 2023), MATH-Hard (Hendrycks et al., 2021), SWE-Bench Verified (Jimenez et al., 2024), and WebShop (Yao et al., 2022a). Each test set has 128 items (AIME 2026: 30) disjoint from training.

Baselines. We compare with direct prompting (Qwen3.5-9B, DeepSeek-V4-Flash), backbone training (SFT, GRPO[†] (Shao et al., 2024)), workflow search (AFlow (Zhang et al., 2025b)), RL-trained orchestration (AgentFlow (Li et al., 2026c), FlowSteer (Zhang et al., 2026b)), and skill evolution (SkillFlow (Zhang et al., 2026c), SkillOpt (Yang et al., 2026b)). All use Qwen3.5-9B as executor and trained model, the same training tasks, and their authors’ best configurations; DeepSeek-V4-Flash only writes EvoSteer’s candidate skills, and other authors move the averages by under 2 points. EvoSteer is tested with learned parts frozen, and every test score is a five-run mean (Appendix D).

5.2 MAIN RESULTS AND BACKBONE TRANSFER (RQ1–RQ3)

Main results. EvoSteer is best on all twelve benchmarks (Table 1), ahead of the strongest baseline by 2.81 EM and 4.01 accuracy points on IID averages and by 3.36 and 2.53 on OOD, with standard deviations at most 1.83. The lead is largest on AIME 2026 (8.67) and smallest on MATH-Hard (0.78), where the backbone is near its ceiling. Training the backbone alone (SFT, GRPO[†]) adds at most 6.09 points to the IID EM average, so the gain comes from orchestration over the same frozen executor, and with a 9B executor EvoSteer beats the larger DeepSeek-V4-Flash on every benchmark.

Backbone transfer. The trained orchestrator transfers unchanged to six other executors and improves all 72 backbone–benchmark scores (Figure 5a,c), so one trained policy serves every executor. Weaker executors gain more, and the IID spread across backbones shrinks from 17.5 to 4.3 points; gains are largest on interactive tasks and smallest on math, where frozen scores are already near the ceiling.

Variant	IID						OOD					
	HotpotQA	NQ-Open	MedQA	AIME	MBPP+	ALFWorld	TriviaQA	MuSiQue	GPQA	MATH	SWE	WebShop
	Ans EM	Ans EM	Acc.	Acc.	Pass@1	SR	Ans EM	Ans EM	Acc.	Acc.	Resolved	SR
Qwen3.5-9B (frozen)	57.66	23.59	71.41	48.67	78.91	46.88	44.22	39.22	61.72	89.06	15.94	32.66
EvoSteer arch., untrained	82.34	73.28	82.97	53.33	87.66	73.59	90.47	71.88	75.31	90.00	29.53	69.84
<i>Fixed orchestration paradigms</i>												
Single agent with tools	75.94	67.19	80.78	48.67	85.63	70.16	86.25	61.41	70.31	89.84	27.03	66.09
Fixed multi-agent template	82.03	74.84	85.47	56.67	83.28	66.72	89.84	70.31	73.75	90.47	24.06	54.84
Plan-then-execute	80.47	72.50	81.88	51.33	84.84	65.63	90.78	68.59	71.56	89.53	25.94	57.81
Searched workflow	87.19	76.41	88.91	54.67	85.16	73.13	91.56	77.50	75.63	90.78	28.28	59.06
<i>Online orchestration (Sec. 4.1)</i>												
– Interleaved execution	89.38	83.75	89.53	58.67	90.63	80.94	95.00	79.84	81.41	93.28	31.25	75.47
– Execution features	88.28	83.75	90.31	66.67	91.41	81.88	93.91	81.25	82.34	93.28	37.34	82.81
– Learnable repair	90.31	84.06	89.84	67.33	90.47	83.59	95.31	80.16	82.81	92.50	31.88	87.03
– Reference value head	83.28	84.22	91.09	68.00	89.84	80.16	94.22	79.22	81.09	92.81	39.22	82.66
<i>AnchorTB (Sec. 4.2)</i>												
– Measured flows	90.16	83.59	89.22	56.67	90.00	77.81	94.38	76.25	81.88	91.09	33.75	80.63
– Flow corrections	85.94	82.66	89.84	64.00	90.31	82.50	95.00	83.75	82.81	92.19	38.13	85.16
<i>Skill admission (Sec. 4.3)</i>												
– Skill evolution	88.28	80.94	90.16	68.67	84.53	82.81	94.38	80.63	83.28	92.81	38.59	85.16
– Sequential validation	90.47	81.72	89.22	66.00	90.00	78.59	94.84	81.25	83.28	94.53	35.00	86.09
EvoSteer (Full)	92.34	85.00	93.13	74.67	92.19	89.69	95.63	84.53	85.63	95.47	42.50	90.63

Table 2: Component ablation and paradigm comparison (untrained: initial π_θ). Paradigms (same executor and budget) fix the team before execution: a ReAct-style agent with all tools, a hand-designed template, a planner that writes the graph once, and a workflow searched offline on training tasks. Ablations: – interleaved execution builds the graph first; – execution features drops f from the state of π_θ ; – learnable repair masks RERUN and DROP; – reference value head withholds $\hat{v}_k$ and $\Delta\hat{v}$; – measured flows learns u_q as a free scalar; – flow corrections turns off $c(g)$ and b_ψ ; – skill evolution runs without any skills; – sequential validation admits after $k = 5$ successes.

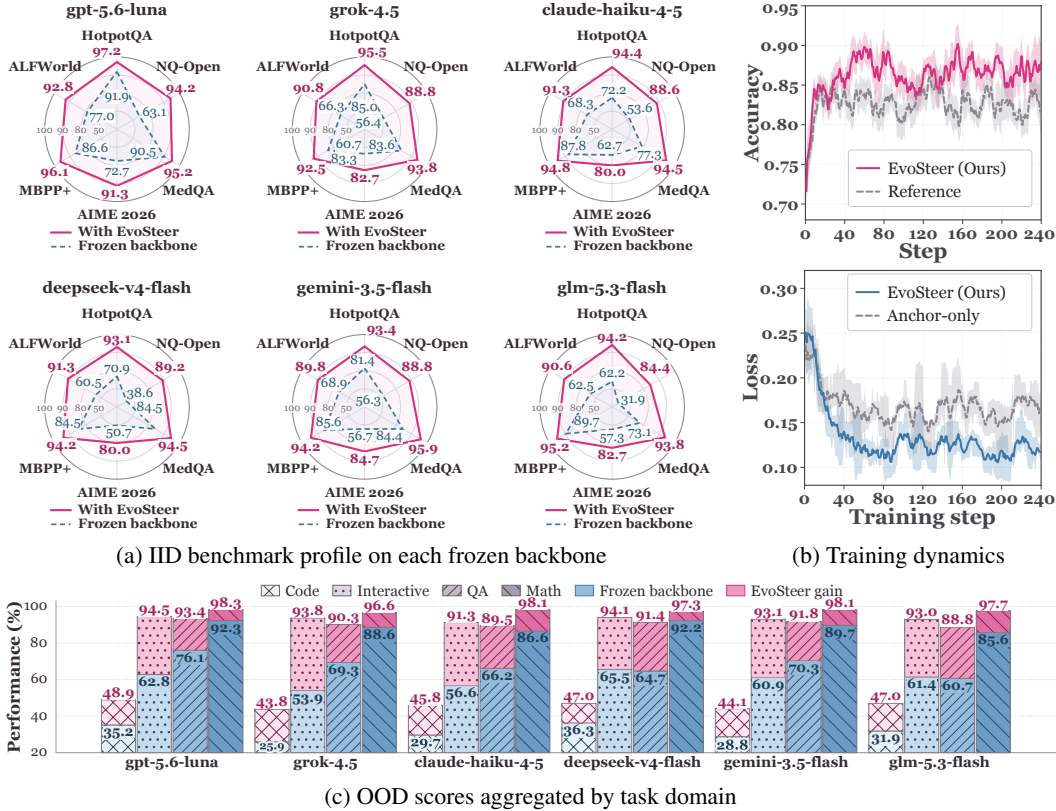


Figure 5: Backbone transfer and training dynamics. (a) IID scores of six other frozen executors (dashed) and with the same trained orchestrator (solid); the radius is linear over 20–80 on the inner half and 80–100 on the outer half. (b) Accuracy of π_θ and of the frozen ρ , and loss against the anchor-only level, over 240 steps. (c) OOD scores per task domain.

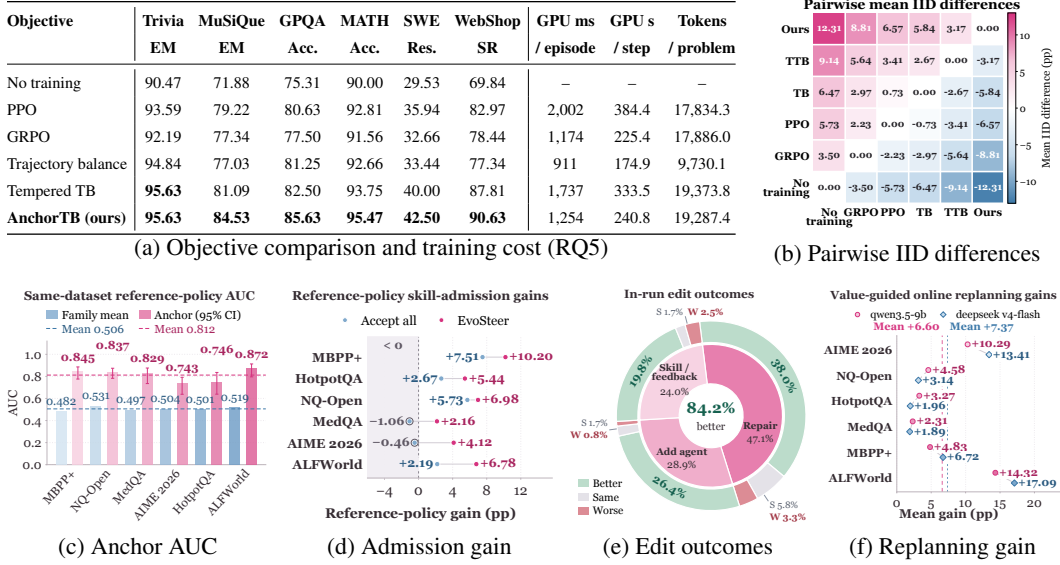


Figure 6: Objective comparison and mechanism analysis (definitions in Appendix D). (a) OOD scores and cost per training objective. (b) Row minus column, IID mean (pp); TTB: tempered TB. (c) AUC for predicting reference-rollout correctness. (d) Paired skill gain, admitting all or only validated candidates. (e) In-run edits by type and outcome. (f) Gain of value-guided replanning in training.

5.3 COMPONENT ANALYSIS (RQ4, RQ5)

Online orchestration (Section 4.1). Each of its mechanisms contributes (Table 2): building the graph before execution, dropping the execution features f , masking rerun and drop, and withholding the value estimate cost 5.69, 4.12, 3.57, and 5.07 IID points. Keeping interleaving, text feedback as in FlowSteer, and the value estimate but not f costs 3.91 OOD points, so f carries information the text does not. Fixed paradigms help decomposable questions but lag where results must be redone, and the best trails EvoSteer by 10.26 IID points; even untrained, the architecture leads all four on the OOD average. In-run edits mostly help: 84.2% raise the graded answer score and 6.6% lower it (Figure 6e). Repairs, the most frequent edit, raise the score in 80.7% of cases, and value-guided replanning helps both executors on every IID dataset, most on the interactive ALFWorld (Figure 6f).

AnchorTB (Section 4.2). Measured flows are the largest single contribution on IID: a free-scalar u_q costs 6.59 IID points, and removing $c(g)$ and b_ψ costs 5.29. The anchor is informative on its own, predicting reference outcomes before a rollout starts with AUC 0.812 against 0.506 for the task-family mean (Figure 6c), and the learned part lowers the loss below the anchor-only level (Figure 5b). With only the loss changed, AnchorTB beats tempered TB (Zhang et al., 2026c), TB (Malkin et al., 2022), PPO (Schulman et al., 2017), and GRPO by 3.17–8.81 IID points (Figure 6a,b); TB lags most on MuSiQue, SWE-Bench, and WebShop. AnchorTB needs 28% less update compute than tempered TB, at a rollout cost within 9% of PPO and GRPO, so its gains come at comparable compute.

Validated admission (Section 4.3). Removing skill evolution costs 5.27 IID and 3.26 OOD points, most on MBPP+ (7.66). A fixed success count in place of the sequential test costs 5.17 IID points, nearly the same for any count from 1 to 10 (Appendix D), and even trails the no-skill variant on MedQA, AIME 2026, and ALFWorld. Accepting every candidate helps on average (+2.76) but hurts MedQA and AIME 2026, whereas validated admission helps all six IID datasets (+5.95; Figure 6d).

Architecture and training. The untrained architecture adds 21.01 IID and 24.04 OOD points to the frozen executor, and AnchorTB training adds 12.31 and 11.22, so each supplies much of the gain.

6 CONCLUSION

EvoSteer unifies construction, repair, and skill growth in one learned loop: each action executes as it is issued, AnchorTB credits it against a frozen reference, and paired sequential tests decide which skills are kept. It beats all baselines on twelve benchmarks and improves every executor backbone.

AI USE STATEMENT

In this work, generative AI tools were used to edit and polish the text for clarity and readability, assist with L^AT_EX typesetting and page layout, and adjust the placement and sizing of existing figures and tables. They were also used for retrieval and discovery, to help search for and identify related work; all bibliographic entries were taken from Google Scholar. The authors manually reviewed and verified all AI-assisted text, figures, numerical values, citations, and formatting against the underlying implementation and experimental records. The authors take full responsibility for the final content of the paper and all artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

This work follows the ICLR Code of Ethics. We aim to conduct and report our research with scientific integrity, transparency, and reproducibility. Experimental results are reported without fabrication, falsification, or intentional misrepresentation, and sufficient implementation and evaluation details are provided to facilitate verification and reproduction. We acknowledge prior work and use existing benchmarks, models, and associated resources in accordance with their intended research purposes and applicable licenses. This study involves no human participants or personal information.

REPRODUCIBILITY STATEMENT

We support reproducibility by documenting the orchestration environment, action space, and value estimate in Section 4.1, the AnchorTB objective in Section 4.2, and the validated skill admission procedure in Section 4.3. Section 5.1 specifies the benchmarks, baselines, and evaluation setting. The appendix provides the assumptions and complete proofs of all theoretical results (Appendices A–C), the training algorithm and all hyperparameters (Appendix C, Table 3), and the benchmark splits, metrics, baseline settings, evaluation protocol, and compute (Appendix D). All test scores are five-run means, and Table 1 also reports their standard deviations. **Code availability.** Our code, the exact training and test splits, and the full configuration are publicly available at `anonymous.4open.science/r/evosteer-C312`.

REFERENCES

- Akash Bonagiri, Devang Borkar, Gerard Janno Anderias, Setareh Rafatirad, and Houman Homayoun. CausalfLOW: Causal attribution and counterfactual repair for llm agent failures. *arXiv preprint arXiv:2605.25338*, 2026.
- 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.
- Kunfeng Chen, Qihuang Zhong, Juhua Liu, and Bo Du. Skillcat: Contrastive assessment and topology-aware skill self-evolution for llm agents. *arXiv preprint arXiv:2606.13317*, 2026a.
- Xudong Chen, Yixin Liu, Hua Wei, and Kaize Ding. Lemon: Learning executable multi-agent orchestration via counterfactual reinforcement learning. *arXiv preprint arXiv:2605.14483*, 2026b.
- YanJun Chen, Yirong Sun, Hanlin Wang, Jinghan Wang, Xinming Zhang, Xiaoyu Shen, Wenjie Li, and Wei Zhang. Exact is easier: Credit assignment for cooperative llm agents. *arXiv preprint arXiv:2603.06859*, 2026c.
- Yufan Dang, Chen Qian, Xueheng Luo, Jingru Fan, Zihao Xie, Ruijie Shi, Weize Chen, Cheng Yang, Xiaoyin Che, Ye Tian, et al. Multi-agent collaboration via evolving orchestration. *Advances in neural information processing systems*, 38:165025–165059, 2026.
- Shripad Deshmukh, Jayakumar Subramanian, Raghavendra Addanki, and Nikos Vlassis. Cosac: Counterfactual credit assignment in sequential cooperative teams. *arXiv preprint arXiv:2604.17693*, 2026.
- Jake Fawkes and Jason Hartford. f -trajectory balance: A loss family for tuning gflownets, generative models, and llms with off-and on-policy data. *arXiv preprint arXiv:2605.15417*, 2026.
- Haowen Gao, Haoran Chen, Can Wang, Shasha Guo, Liang Pang, Zhaoyang Liu, Huawei Shen, and Xueqi Cheng. Skillaudit: Ground-truth-free skill evolution via paired trajectory auditing. *arXiv preprint arXiv:2606.14239*, 2026.

- Zhezhen Hao, Tianfu Wang, Huanshuo Dong, Ziyang Liu, Hong Wang, Xiankun Lin, Qiang Lin, Can Wang, Hande Dong, and Jiawei Chen. Evolve as a team: Collaborative self-evolution for llm-based multi-agent systems. *arXiv preprint arXiv:2605.29790*, 2026.
- Yu He and Weikai Yang. Skillcommit: Evolving agent skills through behaviorally validated scope expansion. *arXiv preprint arXiv:2608.15165*, 2026.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.
- Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Steven Yau, Zijuan Lin, Liyang Zhou, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, volume 2024, pp. 23247–23275, 2024.
- Wentao Hu, Zhendong Chu, Yiming Zhang, Junda Wu, Ming Jin, Xiangyu Zhao, Yilei Shao, Yanfeng Wang, and Qingsong Wen. Skillbrew: Multi-objective curation of skill banks for llm agents. *arXiv preprint arXiv:2605.29440*, 2026.
- Yanze Jiang, Mingxuan Li, Yuhao Wang, Shengfang Zhai, and Jiaheng Zhang. Don’t solve, just compare: Tiny advisors for runtime intervention in llm agents. *arXiv preprint arXiv:2608.21027*, 2026.
- Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024.
- Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421, 2021.
- Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, 2017.
- Tom Kwiakowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.
- Haoran Li, Shulun Chen, Shaoyuan Sun, and Hanchen Wang. Multi-agent coordination adaptation via structure-guided orchestration. *arXiv preprint arXiv:2605.25746*, 2026a.
- Sha Li and Naren Ramakrishnan. Experience as a compass: Multi-agent rag with evolving orchestration and agent prompts. *arXiv preprint arXiv:2604.00901*, 2026.
- Zhongyi Li, Wan Tian, Jinju Chen, Huiming Zhang, Yang Liu, Yikun Ban, and Fuzhen Zhuang. Counterfactual credit policy optimization for multi-agent collaboration. *arXiv preprint arXiv:2603.21563*, 2026b.
- 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, 2026c.
- Zongyue Li, Chengyue Yu, Lei Zang, Chenyi Zhuang, Linjian Mo, and Leilei Gan. Last step matters: Early uncertainty cannot predict failure in long-horizon agents. *arXiv preprint arXiv:2608.29685*, 2026d.
- Taoran Liang, Yang Liu, Shang Luo, Yingguang Yang, Rongrong Zhang, Yingzong Min, Yulin Huang, Jianshen Zhang, Yongzhi Qi, Kefu Xu, et al. Granularity-adaptive credit assignment for long-horizon llm agent reinforcement learning. *arXiv preprint arXiv:2609.12424*, 2026.
- Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in neural information processing systems*, 36:21558–21572, 2023.
- Xiaodong Liu, Michael Xu, Jack W Stokes, Paul Smolensky, Doug Burger, and Jianfeng Gao. Gflowrl: Scaling distribution-matching rl to large language models. *arXiv preprint arXiv:2607.13394*, 2026.
- Hanxiao Lu and Tianyi Zhang. Autonomous repair for multi-agent systems via monte-carlo tree search. *arXiv preprint arXiv:2607.29055*, 2026.
- Zhongwen Luan, Xiaoyu Zhang, Ming Hu, Yue Yang, Jiongchi Yu, and Xiaohong Chen. Repair or re-sample? rethinking failure debugging in llm multi-agent systems. *arXiv preprint arXiv:2608.25920*, 2026.

- Kanika Madan, Jarrod Rector-Brooks, Maksym Korablyov, Emmanuel Bengio, Moksh Jain, Andrei Cristian Nica, Tom Bosc, Yoshua Bengio, and Nikolay Malkin. Learning gflownets from partial episodes for improved convergence and stability. In *International Conference on Machine Learning*, pp. 23467–23483. PMLR, 2023.
- Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in gflownets. *Advances in Neural Information Processing Systems*, 35: 5955–5967, 2022.
- Amritansh Mishra, Supriyo Chakraborty, and Berkcan Kapusuzoglu. On the policy gradient foundations of group relative policy optimization: Credit assignment, gradient sparsity, and rank collapse. *arXiv preprint arXiv:2606.29238*, 2026.
- Shuai Pan, Yixiang Liu, Jiaye Gao, Te Gao, Weiwen Liu, Jianghao Lin, Zhihui Fu, Jun Wang, Weinan Zhang, and Yong Yu. Skillmas: Skill co-evolution with llm-based multi-agent system. *arXiv preprint arXiv:2605.09341*, 2026.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- Biswa Sengupta. Self-evolving agents with anytime-valid certificates. *arXiv preprint arXiv:2607.00871*, 2026.
- Jaineet Shah. Causal agent replay: Counterfactual attribution for llm-agent failures. *arXiv preprint arXiv:2606.08275*, 2026.
- Fangxin Shang and Yehui Yang. Hypothesis-driven skill optimization for llm agents. *arXiv preprint arXiv:2606.22330*, 2026.
- Linfang Shang, Ming Xu, Yiding Sun, Tianle Xia, Lingxiang Hu, Lan Xu, and Ning Zheng. When self-evolution backfires: Pre-commit gating against skill contamination in llm agents. *arXiv preprint arXiv:2608.05810*, 2026.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
- Zayx Shawn. Pace: Anytime-valid acceptance tests for self-evolving agents. *arXiv preprint arXiv:2606.08106*, 2026.
- Junhao Shen, Teng Zhang, Xiaoyan Zhao, and Hong Cheng. Dynamic skill lifecycle management for agentic reinforcement learning. *arXiv preprint arXiv:2605.10923*, 2026.
- Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.
- 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.
- Siddarth Venkatraman, Moksh Jain, Luca Scimeca, Minsu Kim, Marcin Sendera, Mohsin Hasan, Luke Rowe, Sarthak Mittal, Pablo Lemos, Emmanuel Bengio, et al. Amortizing intractable inference in diffusion models for vision, language, and control. *Advances in neural information processing systems*, 37:76080–76114, 2024.
- Chenyu Wang, Yunbo Lyu, Junda He, Zhou Yang, Chenxing Zhong, Yaniv Harel, and David Lo. Fail-fast, restart-smart: Early failure prediction and restart for swe agentic tasks. *arXiv preprint arXiv:2608.03222*, 2026a.
- Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlikar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.
- Tao Wang, Suhang Zheng, and Xiaoxiao Xu. Rtmc: Step-level credit assignment via rollout trees. *arXiv preprint arXiv:2604.11037*, 2026b.
- Yixuan Wang, Yiyang Zhou, Yiming Liang, Congyu Zhang, Fuxiao Liu, Jiawei Zhou, and Huaxiu Yao. Not all skills help: Measuring and repairing agent knowledge. *arXiv preprint arXiv:2606.15390*, 2026c.
- Xiaojun Wu, Cehao Yang, Honghao Liu, Xueyuan Lin, Wenjie Zhang, Zhichao Shi, Xuhui Jiang, Chengjin Xu, Jia Li, and Jian Guo. Bayesian-agent: Posterior-guided skill evolution for llm agent harnesses. *arXiv preprint arXiv:2606.08348*, 2026.

- Chen Xu, Yicheng Hu, Ruizi Wang, Xinyu Lin, Wenjie Wang, Dongrui Liu, and Fuli Feng. Tacomas: Test-time co-evolution of topology and capability in llm-based multi-agent systems. *arXiv preprint arXiv:2605.09539*, 2026.
- Shidong Yang, Ziyu Ma, Tongwen Huang, Xucong Wang, Renda Li, Yiming Hu, Yong Wang, and Xiangxiang Chu. Skillforge: Evolving verifiable skills for reinforcement learning agents. *arXiv preprint arXiv:2608.24747*, 2026a.
- Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, et al. Skillopt: Executive strategy for self-evolving agent skills. *arXiv preprint arXiv:2605.23904*, 2026b.
- 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.
- Huaiyuan Yao, Xiaoou Liu, Charles Fleming, Tianlong Chen, and Hua Wei. Maskills: Continual skills optimization for multi-agent llm systems. *arXiv preprint arXiv:2609.02094*, 2026.
- Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022a.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022b.
- Boxuan Zhang, Jianing Zhu, Zeru Shi, Dongfang Liu, and Ruixiang Tang. Agentforesight: Online auditing for early failure prediction in multi-agent systems. *arXiv preprint arXiv:2605.08715*, 2026a.
- Chenchen Zhang. Reinforcement learning for llm-based multi-agent systems through orchestration traces. *arXiv preprint arXiv:2605.02801*, 2026.
- Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. G-designer: Architecting multi-agent communication topologies via graph neural networks. *arXiv preprint arXiv:2410.11782*, 2024.
- Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Yu, and Tianlong Chen. Cut the crap: An economical communication pipeline for llm-based multi-agent systems. In *International Conference on Learning Representations*, volume 2025, pp. 75389–75428, 2025a.
- Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, et al. Aflow: Automating agentic workflow generation. In *International Conference on Learning Representations*, volume 2025, pp. 34040–34077, 2025b.
- Mingda Zhang, Wenjin Liu, Tiesunlong Shen, Qika Lin, Rui Mao, Erik Cambria, Xiaoying Tang, and Haoran Luo. Flowsteer: Towards agents designing agentic workflows via reinforced progressive canvas editing. *arXiv preprint arXiv:2602.01664*, 2026b.
- Mingda Zhang, Tiesunlong Shen, Haoran Luo, Wenjin Liu, Zikai Xiao, Erik Cambria, and Xiaoying Tang. Skillflow: Flow-driven recursive skill evolution for agentic orchestration. *arXiv preprint arXiv:2605.14089*, 2026c.
- Yan Zhang and Shibo Li. Consistencygate: Preventing memory contamination in llm agents via self-consistency admission control. *arXiv preprint arXiv:2607.22962*, 2026.
- Yaolun Zhang, Tianyi Xu, Shengyu Dai, Zhenwen Shao, Qingyun Wu, and Huazheng Wang. Evochamber: Test-time co-evolution of multi-agent system at individual, team, and population scales. *arXiv preprint arXiv:2605.11136*, 2026d.
- Chenyu Zhao, Shenglin Zhang, Wenwei Gu, Yongqian Sun, Dan Pei, Chetan Bansal, Saravan Rajmohan, and Minghua Ma. Agenttether: Graph-guided diagnosis and runtime intervention for reliable llm agent operation. *arXiv preprint arXiv:2607.06273*, 2026.
- Junze Zhu, Weihao Chen, Xuanwang Zhang, Zhen Wu, and Xinyu Dai. Recognize your orchestrator: An entropy dynamics perspective for llm multi-agent systems. *arXiv preprint arXiv:2606.01351*, 2026.
- Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. Language agents as optimizable graphs. *arXiv preprint arXiv:2402.16823*, 2024.

A ANCHORED TRAJECTORY BALANCE

This appendix treats three objects in turn: the reward-tilted history law, the regression loss evaluated on recorded trajectories, and the statistical procedure for skill validation. We first establish structural and distributional properties conditional on a frozen environment context, then analyze the implemented trajectory-indexed anchors and paired tests. Algebraic identities hold on every recorded trajectory, and each distributional or statistical result holds under the assumptions it states.

A.1 FORMAL SETTING, HISTORIES, AND LEGAL ACTIONS

Definition A.1 (Context and complete-history representation). Fix a task and a rollout-batch context C . The context includes the task, tools, role catalogue, executor configuration, skill-menu and value-head snapshots, and rules that generate legal masks. A state s_t retains the complete ordered history, including the current communication graph, execution records, features, and budget information. With s_0 the initial state,

$$s_{t+1} = s_t \oplus (a_t, o_t^{\text{exec}}, f_{t+1}), \quad a_t : s_t \longrightarrow s_{t+1}, \quad 0 \leq t < T, \quad (\text{A.1})$$

where $f_{t+1} = f(s_t, a_t, o_t^{\text{exec}})$ is the feature record produced by action a_t . Thus $f(s_{t+1}) = f_{t+1}$ denotes the latest feature record; the initial state has its interface-specified initial features. The displayed value estimate and its change are deterministic functions of this retained history and the frozen head in C ; they add no transition randomness. A complete history is $x = (s_0, a_0, s_1, \dots, a_{T-1}, s_T)$, identified with its terminal state s_T , which retains the whole record. The final STOP transition is included in $T \geq 1$. Write $x \succeq s$ when x extends prefix s , and let $|s|$ count appended action records.

Definition A.2 (Legal actions and fixed executor). Let $\mathcal{A}_C(s)$ be the nonempty legal action set at a nonterminal history. It is a state-dependent subset of the batch-frozen action vocabulary. Let $K_C(s' | s, a)$ be the normalized kernel of execution, tool observations, and history updates after a legal action. For a normalized history-dependent action policy p , its joint successor kernel is $p(a | s, C)K_C(s' | s, a)$. The executor is fixed: this conditional kernel is the same for every p , and its outcomes may be stochastic and history-dependent.

Lemma A.1 (Strict growth and unique parentage). *Under Definition A.1, the history graph is a rooted tree: each edge increases $|s|$ by one and every non-root state has a unique parent. Its backward policy on parents is therefore identically one, so no backward model needs to be learned.*

Proof. Equation (A.1) appends exactly one action record, so $|s_{t+1}| = |s_t| + 1$. A directed cycle would strictly increase this integer and return to its original value, a contradiction. Deleting the last action, observation, and feature record from a non-root history recovers its preceding prefix. Any parent must produce precisely this last appended record and the same preceding ordered history, so no second parent exists. A normalized distribution on this singleton parent set has probability one. Returning to an earlier topology keeps the intervening records, so the history still grows. $\square$

Assumption A.1 (Support, normalization, and termination). The serialized history space is finite or countable. At every reference-reachable nonterminal prefix, $\rho(\cdot | s, C)$ is normalized and positive on $\mathcal{A}_C(s)$. A policy compared to it in a log-ratio has the same positive action support and uses the same K_C . From every such prefix, both continuation processes reach a terminal history almost surely, and every terminal record has a reward $r(x) \in [0, 1]$, with $0 \leq \beta < \infty$.

Proposition A.2 (Normalized terminal and prefix laws). *Under Assumption A.1, terminal histories form a probability-one partition for $p \in \{\rho, \pi_\theta\}$. In particular,*

$$\sum_{x \in \mathcal{X}_C} P_p(x | C) = 1, \quad P_p(s | C) = \sum_{x \succeq s} P_p(x | C), \quad P_p(s_0 | C) = 1, \quad (\text{A.2})$$

where P_p denotes the history law induced by p and the executor.

Proof. Normalized local kernels define the successive action and observation probabilities. Different terminal histories describe disjoint events: a terminal record cannot be a proper prefix of a later executed history. Almost-sure termination makes their union a probability-one event. The same reasoning, restricted to the event of reaching s , partitions that event by its terminal descendants. Countable additivity proves both sums; the root is reached with probability one. $\square$

Remark A.1 (Legal actions and horizon). A frozen snapshot fixes the interface and the rules that generate masks, and the legal set $\mathcal{A}_C(s)$ at each state follows from them; for example, a rerun or drop

action requires an applicable agent and sufficient resources. The communication graph may contain cycles. Finite-horizon results below assume a bound $T \leq H$.

A.2 PATH LAWS AND POSITIVE AFFINE TILTING

Lemma A.3 (Path factorization and kernel cancellation). *Under Assumption A.1,*

$$P_p(x \mid C) = \prod_{t=0}^{T-1} p(a_t \mid s_t, C) K_C(s_{t+1} \mid s_t, a_t). \quad (\text{A.3})$$

For a supported continuation of $s = s_m$, define $\Delta \ell_t = \log \pi_\theta(a_t \mid s_t, C) - \log \rho(a_t \mid s_t, C)$. Then

$$\frac{P_\theta(x \mid s, C)}{P_\rho(x \mid s, C)} = \prod_{t=m}^{T-1} \frac{\pi_\theta(a_t \mid s_t, C)}{\rho(a_t \mid s_t, C)} = \exp\left(\sum_{t=m}^{T-1} \Delta \ell_t\right). \quad (\text{A.4})$$

Here $P_\theta = P_{\pi_\theta}$ and $P_\rho(x \mid C)$ is the quantity written $\rho(x \mid C)$ in Eq. (2).

Proof. Apply the probability chain rule, conditioning each action on the history before it and each observation on that history and action. This gives Eq. (A.3); starting at s_m gives the corresponding suffix product. On the common support every executor factor in numerator and denominator is equal and positive, so it cancels. The remaining product consists of the action ratios, whose logarithm is the displayed sum. This cancellation compares two policies on the same recorded history. $\square$

Lemma A.4 (Masked token likelihoods). *Suppose legal actions have unique finite token serializations, including any required action-ending symbol, and masked token generation terminates almost surely. Use the same legal token sets for both policies, with each policy normalized on each such set. The sum of executed token log-probabilities is the log-probability of that action. Singleton legal sets contribute zero to its log-ratio, since both policies give them probability one.*

Proof. For action $a = (a_1, \dots, a_K)$, the autoregressive chain rule gives

$$p(a \mid s, C) = \prod_{j=1}^K p(a_j \mid s, a_{<j}, C),$$

with any termination probability included in the serialization. Taking logarithms gives the sum, and taking the difference for the two policies gives the action log-ratio. If a legal token set is a singleton, both normalized probabilities equal one. Its two log-probabilities are zero. Each model's normalization constant on the common mask remains part of its normalized probability. $\square$

Proposition A.5 (Affine tilt, normalizer, and ideal reward gain). *Let $\mu = \mathbb{E}_\rho[r(X) \mid C]$ and $a_\beta = e^\beta - 1$. The target in Eq. (2) is normalized, has the same terminal support as P_ρ , and satisfies*

$$1 \leq R_\beta(x) \leq e^\beta, \quad Z(C) = 1 + a_\beta \mu \in [1, e^\beta], \quad (\text{A.5})$$

$$\mathbb{E}_{P^*}[r(X) \mid C] - \mu = \frac{a_\beta \text{Var}_\rho(r(X) \mid C)}{1 + a_\beta \mu} \geq 0. \quad (\text{A.6})$$

At $\beta = 0$, $P^* = P_\rho$. For $\beta > 0$, strict gain in Eq. (A.6) occurs exactly when the reference reward has positive variance, which holds whenever the reference both succeeds and fails.

Proof. The reward range follows from $r \in [0, 1]$ and $a_\beta \geq 0$. Summing $P_\rho(x)(1 + a_\beta r(x))$ gives $Z = 1 + a_\beta \mu$; it is finite and positive, so dividing by Z normalizes the target and keeps its support. Moreover,

$$\mathbb{E}_{P^*} r = \frac{\mathbb{E}_\rho r + a_\beta \mathbb{E}_\rho r^2}{1 + a_\beta \mu}.$$

Subtracting μ gives $a_\beta(\mathbb{E}_\rho r^2 - \mu^2)/(1 + a_\beta \mu)$, and the equality and strictness statements follow from $a_0 = 0$ and $a_\beta > 0$ for $\beta > 0$. $\square$

Remark A.2 (Meaning of the target). The tilt is affine in r and equals $\exp(\beta r)$ on binary rewards, where μ is the success probability; for continuous scores μ is a mean reward. Equation (A.6) describes the training target P^* , and skill validation uses the threshold-pass probability of Appendix C.1.

A.3 CONDITIONAL RELATIVE FLOWS AND UNIQUENESS

Definition A.3 (Unnormalized target flow and relative flow). For $P_\rho(s | C) > 0$, define

$$F^*(s | C) = \sum_{x \succeq s} P_\rho(x | C) R_\beta(x), \quad U^*(s | C) = \frac{F^*(s | C)}{P_\rho(s | C)}, \quad (\text{A.7})$$

and $u^*(s | C) = \log U^*(s | C)$. Thus F^* is unnormalized. The normalized target prefix probability is $P^*(s | C) = F^*(s | C)/Z(C)$, so $P^*(s | C)/P_\rho(s | C) = U^*(s | C)/Z(C)$.

Lemma A.6 (Prefix partition and conditional recursion). *Under Assumption A.1, at every supported prefix,*

$$U^*(s | C) = \mathbb{E}_\rho[R_\beta(X) | s, C] = 1 + a_\beta \mathbb{E}_\rho[r(X) | s, C] \in [1, e^\beta]. \quad (\text{A.8})$$

For nonterminal s , writing $\text{Ch}(s)$ for its supported children,

$$F^*(s) = \sum_{s' \in \text{Ch}(s)} F^*(s'), \quad (\text{A.9})$$

$$U^*(s) = \sum_{a \in \mathcal{A}_C(s)} \rho(a | s, C) \sum_{s'} K_C(s' | s, a) U^*(s'). \quad (\text{A.10})$$

The boundaries are $U^(x) = R_\beta(x)$ and $U^*(s_0) = Z(C)$.*

Proof. Divide the descendant sum defining $F^*(s)$ by the positive prefix probability. By Eq. (A.2), the resulting weights are the normalized reference continuation law, yielding Eq. (A.8). Descendants of a nonterminal s partition by their unique first child, proving Eq. (A.9). Each child retains its incoming action, so $P_\rho(s') = P_\rho(s) \rho(a | s, C) K_C(s' | s, a)$. Substitute $F^*(s') = P_\rho(s') U^*(s')$ in the child sum and divide by $P_\rho(s)$ to obtain Eq. (A.10). No additional transition weight belongs in the sum of $F^*(s')$, since it already includes the probability of reaching the child. A terminal has only itself as a terminal descendant; the root descendant sum is Z . $\square$

Proposition A.7 (Uniqueness with an appropriate horizon condition). *On a history tree with a uniform finite action horizon H , U^* is the unique finite-valued solution of Eq. (A.10) with terminal boundary $V(x) = R_\beta(x)$. On a possibly infinite-depth tree with almost-sure reference termination from every supported prefix, U^* is the unique bounded solution with this boundary.*

Proof. For the finite-horizon statement, terminal values coincide. Suppose $V = U^*$ at all supported successors of a nonterminal s . Applying the same recursion to both functions gives equality at s . Induct backward through the at most H remaining transitions. This proves equality at every supported history, also for countably many children since their already identified bounded values have well-defined weighted sums, so the backward induction goes through unchanged.

For the infinite-depth statement, start a reference continuation at s , and let τ be its remaining termination time. Extend the state process mathematically by retaining its terminal state after τ . The recursion and iterated conditional expectation give $V(s) = \mathbb{E}_\rho[V(S_{n \wedge \tau}) | s, C]$ for every integer n . Almost-sure termination implies $V(S_{n \wedge \tau}) \rightarrow R_\beta(X)$. Boundedness permits passage to the expectation, so $V(s) = \mathbb{E}_\rho[R_\beta(X) | s, C] = U^*(s)$. The same argument holds at each supported prefix. Equation (A.8) already ensures that U^* itself is bounded. $\square$

Remark A.3 (Why infinite-tree boundedness matters). Consider a single action that, at nonterminal depth n , terminates with probability $1/2$ and boundary value one, or advances to depth $n + 1$. Termination is almost sure. Nevertheless, for any $c > 0$, the unbounded finite-valued function $V_n = 1 + c2^n$ obeys $V_n = \frac{1}{2} + \frac{1}{2}V_{n+1}$ and the same terminal boundary. It differs from $U^* = 1$. Boundedness in Proposition A.7 is therefore what makes the solution unique.

A.4 IDEAL ZERO-RESIDUAL CONSISTENCY

Definition A.4 (Shared-flow residual). A shared flow is a real-valued function $u(s)$ assigning one value to a prefix independently of which later continuation is scored. Its terminal boundary is $u(s_T) = \log R_\beta(x)$. For $0 \leq i < j \leq T$, set

$$\delta_{i,j}^u(x) = u(s_i) + \sum_{t=i}^{j-1} \Delta \ell_t - u(s_j), \quad \mathcal{L}_x^u = \frac{1}{K_T} \sum_{i < j} (\delta_{i,j}^u(x))^2, \quad K_T = \frac{T(T+1)}{2}. \quad (\text{A.11})$$

This is Eq. (7) with a common state function; Appendix B treats the trajectory-indexed estimate.

Proposition A.8 (Ideal reference-relative consistency). *Under Assumption A.1, suppose a shared u satisfies the terminal boundary and $\delta_{t:t+1}^u(x) = 0$ on every supported complete history and every action in that history. Then*

$$e^{u(s)} = U^*(s \mid C), \quad u(s_0) = \log Z(C), \quad P_\theta(x \mid C) = P^*(x \mid C) \quad (\text{A.12})$$

on the reference support. Requiring zero residual for all subtrajectories is equivalent here to requiring it for all singleton intervals, so checking one-step residuals suffices.

Proof. Fix a supported prefix $s = s_m$. Adding the singleton equalities along any supported complete continuation gives

$$\sum_{t=m}^{T-1} \Delta \ell_t = \sum_{t=m}^{T-1} (u(s_{t+1}) - u(s_t)) = \log R_\beta(x) - u(s).$$

By Lemma A.3, $e^{u(s)} P_\theta(x \mid s, C) = P_\rho(x \mid s, C) R_\beta(x)$. Sum over all terminal descendants. The shared value $e^{u(s)}$ can be taken outside the sum, and the policy continuation probabilities sum to one by termination. Hence

$$e^{u(s)} = \sum_{x \succeq s} P_\rho(x \mid s, C) R_\beta(x) = U^*(s \mid C).$$

At terminal s , this equality follows directly from the boundary; at the root, it yields the normalizer $u(s_0) = \log Z(C)$, and substituting this into the root-to-terminal identity gives $P_\theta = P^*$. Finally, the same finite sum of singleton equalities telescopes on any interval $i : j$. The reverse implication follows because singleton intervals are included among all subtrajectories. $\square$

Corollary A.9 (Population zero loss under full coverage). *Suppose the preceding shared-flow and history-law assumptions hold and a fixed sampling law ν assigns positive mass to every reference-supported terminal history. If $\mathbb{E}_\nu[\mathcal{L}_X^u] = 0$, then Proposition A.8 applies.*

Proof. Every loss is nonnegative. A positive loss at a history of positive ν -mass would give a positive expectation, so each supported history has zero loss. Its finite sum of squared residuals then has every term zero, in particular each singleton term. The consistency proposition therefore applies. $\square$

A.5 FIXED-EXECUTOR REALIZABILITY AND ITS OBSTRUCTION

Proposition A.10 (Action-only realizability criterion). *Under the reference-law and reward premises of Assumption A.1, define, on its support,*

$$M^*(s, a) = \sum_{s'} K_C(s' \mid s, a) U^*(s'), \quad (\text{A.13})$$

$$Q^*(a, s' \mid s) = \rho(a \mid s, C) K_C(s' \mid s, a) \frac{U^*(s')}{U^*(s)}, \quad (\text{A.14})$$

$$\pi^*(a \mid s) = \rho(a \mid s, C) \frac{M^*(s, a)}{U^*(s)},$$

$$K^*(s' \mid s, a) = K_C(s' \mid s, a) \frac{U^*(s')}{M^*(s, a)}. \quad (\text{A.15})$$

Here Q^ is the target joint successor law and $Q^* = \pi^* K^*$. Some arbitrary complete-history action policy using the fixed executor realizes P^* if and only if*

$$U^*(s') = M^*(s, a) \quad K_C(\cdot \mid s, a)\text{-almost surely} \quad (\text{A.16})$$

for every supported (s, a) . If realizable, that action policy is uniquely π^ on the reference support. A deterministic executor satisfies Eq. (A.16) automatically.*

Proof. The target probability of a supported child divided by the target probability of its parent is $F^*(s')/F^*(s)$. Substitute the reference prefix factorization from Lemma A.6; since the child records its action, this gives Eq. (A.14). Summing over children for a fixed action gives π^* , and dividing by this positive action marginal gives K^* . Both normalize by M^* and the flow recursion.

For necessity, a policy π realizing the complete target law must also realize its prefix and joint successor probabilities. Its joint kernel is $\pi(a \mid s)K_C(s' \mid s, a)$. Equating it to Q^* and canceling a positive executor factor yields $\pi(a \mid s) = \rho(a \mid s)U^*(s')/U^*(s)$. The left side does not vary with s' , so $U^*(s')$ is constant on this successor support. Averaging that constant gives $M^*(s, a)$, proving Eq. (A.16) and $\pi = \pi^*$.

For sufficiency, under this condition $K^* = K_C$ and $\pi^*K_C = Q^*$. Multiplying these joint transitions on a complete history telescopes the U^* ratios, yielding $P_\rho(x)U^*(x)/U^*(s_0) = P^*(x)$. These terminal probabilities sum to one, so the constructed process terminates almost surely and realizes the target. Uniqueness follows from its already determined action marginals. A deterministic kernel has only one supported successor, which proves the last statement. $\square$

Proposition A.11 (Irreducible executor contribution to relative entropy). *Under the reference-law and reward premises of Assumption A.1, suppose the supported tree has a uniform finite horizon H . Let Π_C contain all normalized complete-history action policies supported on the reference legal actions, including policies with some zero action probabilities. For $\pi \in \Pi_C$, where $D_{\text{KL}}(p \parallel q) = \sum_z p(z) \log[p(z)/q(z)]$, zero- p terms contribute zero, and a positive- p , zero- q term gives infinite divergence,*

$$D_{\text{KL}}(P^* \parallel P_\pi) = \mathbb{E}_{P^*} \left[\sum_{t=0}^{T-1} D_{\text{KL}}(\pi^*(\cdot \mid S_t) \parallel \pi(\cdot \mid S_t)) \right] + \mathcal{I}_C, \quad (\text{A.17})$$

$$\mathcal{I}_C = \mathbb{E}_{P^*} \left[\sum_{t=0}^{T-1} D_{\text{KL}}(K^*(\cdot \mid S_t, A_t) \parallel K_C(\cdot \mid S_t, A_t)) \right], \quad (\text{A.18})$$

$$\inf_{\pi \in \Pi_C} D_{\text{KL}}(P^* \parallel P_\pi) = \mathcal{I}_C. \quad (\text{A.19})$$

The infimum is attained by using the action marginal π^* with the fixed executor. Moreover, $\mathcal{I}_C = 0$ exactly when Eq. (A.16) holds on the reference support.

Proof. Factor the target with π^*K^* and the comparison process with πK_C . Their log density ratio is

$$\sum_{t=0}^{T-1} \log \frac{\pi^*(A_t \mid S_t)}{\pi(A_t \mid S_t)} + \sum_{t=0}^{T-1} \log \frac{K^*(S_{t+1} \mid S_t, A_t)}{K_C(S_{t+1} \mid S_t, A_t)}.$$

For the first sum, condition on each nonterminal S_t and average over its target action law; the result is the action KL term in Eq. (A.17). For the second, condition on (S_t, A_t) and average over K^* ; the result is Eq. (A.18). Variable lengths cause no additional term: for this calculation only, histories may be padded to H with deterministic terminal coordinates of log-ratio zero.

These steps remain valid as extended expectations. For probability vectors p, q , the negative part of $\sum p \log(p/q)$ is bounded by $\sum_{p < q} q(p/q) \log(q/p) \leq 1/e$, since $z \log(1/z) \leq 1/e$ on $[0, 1]$. There are at most $2H$ conditional terms. Also $U^*, M^* \in [1, e^\beta]$, so the executor log-ratio is bounded in absolute value by β . No subtraction of two infinite positive expectations is required.

Each conditional KL is nonnegative. Taking $\pi = \pi^*$ makes every action term zero and proves the infimum formula; finite horizon makes this fixed-executor policy a terminating member of Π_C . The remaining sum is zero precisely when $K^* = K_C$ at every target-supported state-action pair. The target and reference have the same support, and Eq. (A.15) makes this equality equivalent to Eq. (A.16). $\square$

Remark A.4 (A stochastic-executor counterexample). Let there be one legal action, after which the executor terminates with $r = 0$ or $r = 1$, each with probability $1/2$. Every action policy has success probability $1/2$, whereas for $\beta > 0$ the target has success probability $e^\beta/(1 + e^\beta) > 1/2$. Target realizability therefore needs the condition of Proposition A.10 in addition to kernel cancellation.

A.6 SUBTRAJECTORY ALGEBRA AND CREDIT COEFFICIENTS

Fix a recorded trajectory x of length $T \geq 1$. All action log-ratios and endpoint values are finite. Write v_i for the value used at s_i on this record, allowing $v_i = \tilde{u}_x(s_i)$; reuse that same value in every interval of the record. Put $e_t = v_t + \Delta \ell_t - v_{t+1}$, $0 \leq t < T$, and $\delta_{i:j}^{(x)} = v_i + \sum_{t=i}^{j-1} \Delta \ell_t - v_j$.

Lemma A.12 (Within-trajectory telescoping). *For every interval $0 \leq i < j \leq T$, $\delta_{i:j}^{(x)} = \sum_{t=i}^{j-1} e_t$.*

Proof. Expand the sum of e_t . Each internal value $v_{i+1}, \dots, v_{j-1}$ appears once with each sign and cancels; only $v_i - v_j$ and the action log-ratios remain. No other trajectory is used. $\square$

Proposition A.13 (Matrix form, positive definiteness, and zero loss). *For all $0 \leq i < j \leq T$ and $0 \leq t < T$, let $A_T[(i, j), t] = \mathbf{1}\{i \leq t < j\}$. With $G_T = A_T^\top A_T$,*

$$\delta^{(x)} = A_T e, \quad \mathcal{L}_x = \frac{1}{K_T} e^\top G_T e, \quad (G_T)_{rs} = (\min(r, s) + 1)(T - \max(r, s)), \quad (\text{A.20})$$

where $0 \leq r, s < T$. The matrix G_T is positive definite and

$$\mathcal{L}_x \geq \frac{1}{K_T} \sum_{t=0}^{T-1} e_t^2, \quad \mathcal{L}_x = 0 \iff e = 0 \iff \delta_{i:j}^{(x)} = 0 \text{ for every } i < j. \quad (\text{A.21})$$

Proof. Lemma A.12 gives the matrix multiplication and hence the quadratic form. Entry (r, s) counts intervals containing both actions: there are $\min(r, s) + 1$ possible left endpoints and $T - \max(r, s)$ possible right endpoints. This proves the entry formula. The rows indexed by $(t, t + 1)$ form an identity matrix, so A_T has full column rank. For any nonzero e , $\|A_T e\|_2^2 > 0$, proving positive definiteness. Retaining only the singleton squares proves the lower bound and forces $e = 0$ if the loss is zero. Conversely, $e = 0$ gives every interval residual zero and therefore zero loss. $\square$

Proposition A.14 (Action coefficients and cumulative-sum identity). *Holding the endpoint values fixed when differentiating with respect to action log-ratios, the coefficient for action t is*

$$\Gamma_t(x) := \frac{\partial \mathcal{L}_x}{\partial \Delta \ell_t} = \frac{2}{K_T} \sum_{i \leq t < j} \delta_{i:j}^{(x)} = \frac{2}{K_T} (G_T e)_t. \quad (\text{A.22})$$

There are $(t + 1)(T - t)$ intervals in this sum. If $h_0 = 0$ and $h_j = \sum_{t=0}^{j-1} e_t$, then

$$K_T \mathcal{L}_x = (T + 1) \sum_{j=0}^T h_j^2 - \left(\sum_{j=0}^T h_j \right)^2. \quad (\text{A.23})$$

Proof. The partial derivative of $\delta_{i:j}^{(x)}$ with respect to $\Delta \ell_t$ is $\mathbf{1}\{i \leq t < j\}$. Differentiate the finite sum of squared residuals to obtain the first coefficient formula; $A_T^\top \delta^{(x)} = G_T e$ gives the second. An interval containing t has $t + 1$ possible starts and $T - t$ possible ends. For the cumulative identity, telescoping gives $\delta_{i:j}^{(x)} = h_j - h_i$. Therefore

$$\sum_{i < j} (h_j - h_i)^2 = \frac{1}{2} \sum_{i=0}^T \sum_{j=0}^T (h_j - h_i)^2 = (T + 1) \sum_{j=0}^T h_j^2 - \left(\sum_{j=0}^T h_j \right)^2,$$

where the last equality expands the two square terms and the cross term. $\square$

Remark A.5 (Local coverage). For $T = 2$, $e_0 = c$ and $e_1 = -c$ give a zero full-trajectory residual but $\mathcal{L}_x = 2c^2/3 > 0$ when $c \neq 0$. Singleton intervals expose such local errors. Equation (A.23) also allows an algebraic linear-time evaluation once e is known.

A.7 APPROXIMATE CONSISTENCY UNDER UNIFORM RESIDUAL CONTROL

Proposition A.15 (Root, log-density, and total-variation bounds). *Under Assumption A.1, let u be shared with the exact terminal boundary. If $|\delta_{0:T(x)}^u(x)| \leq \varepsilon$ for every reference-supported complete history, where $\varepsilon \geq 0$, then*

$$|u(s_0) - \log Z(C)| \leq \varepsilon, \quad (\text{A.24})$$

$$\left| \log \frac{P_\theta(x | C)}{P^*(x | C)} \right| \leq 2\varepsilon, \quad (\text{A.25})$$

$$\text{TV}(P_\theta, P^*) \leq \tanh(\varepsilon), \quad (\text{A.26})$$

where $\text{TV}(P, Q) = \frac{1}{2} \sum_x |P(x) - Q(x)|$.

Proof. Write $d(x) = \delta_{0:T(x)}^u(x)$. Equation (A.4) and the terminal boundary give

$$P_\theta(x)/P^*(x) = \exp(d(x) + \log Z - u(s_0)).$$

Summing against P^* and using normalization shows

$$u(s_0) - \log Z = \log \mathbb{E}_{P^*} e^{d(X)}.$$

Because $e^{-\varepsilon} \leq e^{d(X)} \leq e^\varepsilon$, this logarithm lies in $[-\varepsilon, \varepsilon]$, proving the root bound. Subtract it from $d(x)$ to obtain the log-density bound. In particular, for $g(x) = P_\theta(x)/P^*(x)$, $e^{-2\varepsilon} \leq g(x) \leq e^{2\varepsilon}$. For any positive z in this interval,

$$\frac{|z - 1|}{z + 1} \leq \frac{e^{2\varepsilon} - 1}{e^{2\varepsilon} + 1} = \tanh(\varepsilon).$$

The fraction increases for $z \geq 1$ and decreases for $z \leq 1$, so its maximum is at an interval endpoint. Integrating $|g - 1| \leq \tanh(\varepsilon)(g + 1)$ under P^* with $\mathbb{E}_{P^*} g = 1$ gives Eq. (A.26). $\square$

Corollary A.16 (A uniformly controlled all-subtrajectory loss). *If additionally $\eta \geq 0$, $T(x) \leq H$, and $\mathcal{L}_x^u \leq \eta^2$ for every supported complete history, the preceding bounds hold with $\varepsilon = \sqrt{H(H+1)/2} \eta$.*

Proof. The full-path square is one nonnegative summand of $K_T \mathcal{L}_x^u$, so Proposition A.15 applies with

$$|\delta_{0:T}^u| \leq \sqrt{K_T} \eta \leq \sqrt{H(H+1)/2} \eta = \varepsilon. \quad \square$$

A.8 MIXED-BEHAVIOR REGRESSION AND GRADIENT BOUNDS

Remark A.6 (Gradient coefficients and data regime). The action coefficient in Proposition A.14 is a partial derivative on a recorded trajectory. In the implemented objective, θ receives gradients through recomputed action log-probabilities and ψ through nonterminal residual-head values. The reference encodings and measured anchor statistics are detached. Trajectories are collected asynchronously, so the behavior adapter may lag the trained one by one update; paired rollouts have a forced first action.

Lemma A.17 (Record-conditioned gradient routing). *Condition on a recorded batch, its contexts, and detached statistics. Assume $b_\psi(h_\rho(s), f(s))$ has no θ -dependence. The AnchorTB sample gradients are*

$$\nabla_\theta \mathcal{L}_x = \sum_{t=0}^{T-1} \Gamma_t(x) \nabla_\theta \log \pi_\theta(a_t | s_t, C), \quad (\text{A.27})$$

$$\nabla_\psi \mathcal{L}_x = \frac{2}{K_T} \sum_{i < j} \delta_{i:j}^{(x)} (\nabla_\psi \tilde{u}_x(s_i) - \nabla_\psi \tilde{u}_x(s_j)), \quad (\text{A.28})$$

where $\nabla_\psi \tilde{u}_x(s_T) = 0$ because the terminal reward boundary overrides the learned head.

Proof. Differentiate each squared residual using the chain rule. For θ , the endpoint values and reference log-probabilities are fixed, leaving only the current action log-probabilities; collecting terms for the same action gives Eq. (A.27). For ψ , the action log-ratios are fixed and the two endpoints have opposite signs, giving Eq. (A.28). Stop-gradient sets the derivative of the measured part to zero, and the calculation conditions on the sampled history and its terminal reward. $\square$

Proposition A.18 (A bounded-Jacobian gradient second-moment bound). *Let ξ be any chosen trainable parameter vector, with the record and detached statistics held fixed. At differentiability points suppose*

$$\frac{1}{K_T} \sum_{i < j} \|\nabla_\xi \delta_{i:j}^{(x)}\|_2^2 \leq G^2$$

for a constant $G < \infty$. Then

$$\|\nabla_\xi \mathcal{L}_x\|_2^2 \leq 4G^2 \mathcal{L}_x. \quad (\text{A.29})$$

For a fixed distribution of recorded inputs satisfying this bound and $\mathbb{E} \mathcal{L}_X < \infty$,

$$\text{tr Cov}(\nabla_\xi \mathcal{L}_X) \leq \mathbb{E} \|\nabla_\xi \mathcal{L}_X\|_2^2 \leq 4G^2 \mathbb{E} \mathcal{L}_X. \quad (\text{A.30})$$

Proof. Apply vector Cauchy–Schwarz to the sample gradient:

$$\left\| \frac{2}{K_T} \sum_{i < j} \delta_{i:j}^{(x)} \nabla_\xi \delta_{i:j}^{(x)} \right\|_2^2 \leq 4 \left(\frac{1}{K_T} \sum_{i < j} (\delta_{i:j}^{(x)})^2 \right) \left(\frac{1}{K_T} \sum_{i < j} \|\nabla_\xi \delta_{i:j}^{(x)}\|_2^2 \right).$$

The first factor is the loss and the second is at most G^2 . Taking expectation proves the second-moment bound. The identity $\text{tr Cov}(V) = \mathbb{E} \|V\|_2^2 - \|\mathbb{E} V\|_2^2$ for square-integrable V gives the rest. $\square$

Sampling law. With a fixed data law ν , AnchorTB is a regression of recorded residuals under ν . Under full coverage its exact shared zero is characterized by Corollary A.9, and nonzero optima depend on the sampling weights. These sample-gradient identities concern the AnchorTB loss; the value head and the diagnostic heads are fitted separately.

B SHRINKAGE BASELINES, STRUCTURAL POOLING, AND THE VALUE HEAD

The ideal conditional mean in Eq. (A.8) is a property of a specified reference continuation law. Measured task statistics, lagged structural buckets, and a feature-based value head are different estimators with different data sources. We give their algebraic properties and population conditions.

B.1 IMPLEMENTED TRAJECTORY-INDEXED FLOWS

Definition B.1 (Implemented flow and terminal override). For batch k and scored trajectory x , put $u_{q,k,x} = \log[1 + a_\beta \hat{p}_{q,k}^{(-x)}]$. The flow estimate in Eq. (8), with its terminal boundary made explicit, is

$$\tilde{u}_{k,x}(s_i) = \begin{cases} \text{sg}[\text{clip}_{[0,\beta]}(u_{q,k,x} + c_{k-1}(g(s_i)))] + b_\psi(h_\rho(s_i), f(s_i)), & 0 \leq i < T, \\ \log R_\beta(x), & i = T. \end{cases} \quad (\text{B.1})$$

The second branch overrides the entire learned parameterization at the terminal, including b_ψ . We suppress the fixed batch index and write $\tilde{u}_x, u_{q,x}, \hat{p}_q^{(-x)}, c$ when unambiguous. The symbol $g(s)$ denotes the canonical structure bucket of the history.

Proposition B.1 (When a trajectory-indexed family is shared). *On a given collection of scored prefixes, the family $\{\tilde{u}_x(s)\}$ defines a single state function if and only if*

$$\tilde{u}_x(s) = \tilde{u}_{x'}(s) \quad \text{whenever } x \succeq s \text{ and } x' \succeq s \text{ belong to that collection.} \quad (\text{B.2})$$

Within every record, Lemma A.12 and Propositions A.13–A.14 apply using Eq. (B.1).

Proof. If a single function exists, evaluating it at s gives the same value for every continuation, which proves necessity. Conversely, define its value at each represented prefix by selecting any one continuation. Equation (B.2) makes the definition independent of that selection. This proves sufficiency on the specified collection. The within-record identities only require that a fixed endpoint value be reused in each interval of that record, which the definition provides. $\square$

Remark B.1 (Partial LOO is numerically trajectory dependent). For two same-task reference records with rewards one and zero, take $m_0 = 1/2$ and the configured $\kappa_q = 0.5$. Removing the successful record from its task-level statistic gives $\hat{p}_q^{(-x_1)} = 1/6$; removing the failure gives $\hat{p}_q^{(-x_2)} = 5/6$. With $c = 0$, $b_\psi = 0$, and $\beta > 0$, the common root receives two distinct values $\log(1 + a_\beta/6)$ and $\log(1 + 5a_\beta/6)$, both inside $[0, \beta]$. Stop-gradient leaves these numerical values unchanged.

B.2 HIERARCHICAL SHRINKAGE AND ITS RANGE

Definition B.2 (Hierarchical task baseline). Let S and N be reward sums and trajectory counts from the healthy, non-paired reference pool, with subscripts g, c, q for global, category, and task statistics. Here g in S_g, N_g, m_g means the global level, not a structure bucket. Define

$$m_g = \begin{cases} S_g/N_g, & N_g > 0, \\ 1/2, & N_g = 0, \end{cases} \quad m_c = \frac{S_c + \kappa_g m_g}{N_c + \kappa_g}, \quad (\text{B.3})$$

$$m_0 = \frac{n'_d p_d + \kappa_c m_c}{n'_d + \kappa_c}, \quad \hat{p}_q^{(-x)} = \frac{S_q^{(-x)} + \kappa_q m_0}{N_q^{(-x)} + \kappa_q}. \quad (\text{B.4})$$

The optional record prior is (p_d, n_d) , with $n'_d = \min(n_d, 20)$; absence of that prior gives $n'_d = 0$. The configured strengths are $(\kappa_g, \kappa_c, \kappa_q) = (2, 4, 0.5)$. If the scored record is in the eligible task pool, it is removed once from that task sum and count; otherwise no subtraction is made. Leave-one-out applies at the task level. The estimate is clipped to $[0, 1]$ before the reward transform. For bounded nonbinary rewards, $\hat{p}$ estimates a mean score, and for binary rewards a success probability.

Lemma B.2 (Range of the hierarchical estimate). *Assume nonnegative counts, $0 \leq S \leq N$ at every used level, a valid task-level deletion, $p_d \in [0, 1]$, $n'_d \geq 0$, and positive shrinkage strengths. Then $m_g, m_c, m_0, \hat{p}_q^{(-x)} \in [0, 1]$ and $0 \leq u_{q,x} \leq \beta$.*

Proof. A nonempty empirical mean S/N lies in $[0, 1]$, and the empty global convention does too. For $N_c > 0$, m_c is the convex combination of S_c/N_c and m_g with weights $N_c/(N_c + \kappa_g)$ and $\kappa_g/(N_c + \kappa_g)$; if $N_c = 0$ it equals m_g . The same argument applies to m_0 and the task estimate, including zero-count cases because their shrinkage denominators stay positive. Finally $p \mapsto \log(1 + a_\beta p)$ is nondecreasing on $[0, 1]$, with endpoint values 0 and β . This holds for any reward in $[0, 1]$. $\square$

Proposition B.3 (Fixed-prior shrinkage moments and sensitivity). *For fixed $n \geq 0$, $m \in [0, 1]$, $\kappa > 0$, and rewards $Y_i \in [0, 1]$, let $\hat{p} = (\sum_{i=1}^n Y_i + \kappa m)/(n + \kappa)$. If the Y_i are i.i.d. with mean μ and variance σ^2 , then*

$$\mathbb{E}\hat{p} - \mu = \frac{\kappa(m - \mu)}{n + \kappa}, \quad \text{Var}(\hat{p}) = \frac{n\sigma^2}{(n + \kappa)^2}, \quad (\text{B.5})$$

$$\mathbb{E}(\hat{p} - \mu)^2 = \frac{n\sigma^2 + \kappa^2(m - \mu)^2}{(n + \kappa)^2}. \quad (\text{B.6})$$

For a fixed record set, changing one bounded reward changes $\hat{p}$ by at most $1/(n + \kappa)$. Changing only m to m' changes it by exactly $\kappa|m - m'|/(n + \kappa)$. When $n \geq 1$, deleting record i while holding the prior fixed gives $\hat{p} - \hat{p}_{-i} = (Y_i - \hat{p}_{-i})/(n + \kappa)$, again of magnitude at most $1/(n + \kappa)$.

Proof. Linearity gives $\mathbb{E}\hat{p} = (n\mu + \kappa m)/(n + \kappa)$. Independence makes the variance of the sum $n\sigma^2$; division by the squared denominator gives its variance. Squared bias plus variance proves the MSE expression, also at $n = 0$. The replacement and prior sensitivities follow by subtracting the two numerators over their common denominator. For deletion, write $\sum_{h \neq i} Y_h + \kappa m = (n - 1 + \kappa)\hat{p}_{-i}$ and substitute in $\hat{p}$; subtracting $\hat{p}_{-i}$ gives the identity, bounded as $Y_i, \hat{p}_{-i} \in [0, 1]$. $\square$

Pooled estimand. The moment formulas above condition on a fixed prior and sample size. If records come from contexts C_h and pass a health event A_h , their conditional means are $\mathbb{E}[r \mid C_h, A_h]$, and for fixed mixture weights w_h the pooled population mean is $\sum_h w_h \mathbb{E}[r \mid C_h, A_h]$.

B.3 ANCHOR SENSITIVITY AND RESIDUAL STABILITY

Lemma B.4 (Reward-transform sensitivity and clipping). *For $p, p' \in [0, 1]$, let $f_\beta(p) = \log(1 + a_\beta p)$ and $A(p, c) = \text{clip}_{[0, \beta]}(f_\beta(p) + c)$. Then*

$$|f_\beta(p) - f_\beta(p')| \leq a_\beta |p - p'|, \quad |A(p, c) - A(p', c')| \leq a_\beta |p - p'| + |c - c'|. \quad (\text{B.7})$$

Proof. For $\beta > 0$, $f'_\beta(p) = a_\beta/(1 + a_\beta p) \leq a_\beta$, so integration between p and p' proves the first inequality. At $\beta = 0$, both sides of that inequality are zero. Projection onto a closed interval is nonexpansive: ordering the two inputs shows that clipping can only shorten their distance. This holds for $[0, 0]$ too. This and the triangle inequality give the second bound. $\square$

Proposition B.5 (Endpoint perturbations, loss, and credit stability). *Compare two endpoint arrays $v_i, \bar{v}_i$ on one fixed trajectory with the same action log-ratios, and suppose $\max_{0 \leq i \leq T} |\bar{v}_i - v_i| \leq \varepsilon$. Then*

$$|\bar{\delta}_{i,j} - \delta_{i,j}| \leq 2\varepsilon, \quad |\sqrt{\bar{\mathcal{L}}_x} - \sqrt{\mathcal{L}_x}| \leq 2\varepsilon, \quad (\text{B.8})$$

$$|\bar{\Gamma}_t - \Gamma_t| \leq \frac{4\varepsilon}{K_T}(t + 1)(T - t), \quad 0 \leq t < T. \quad (\text{B.9})$$

If only the measured anchor changes by inputs satisfying $|p - p'| \leq \varepsilon_p$, $|c - c'| \leq \varepsilon_c$, while b_ψ and terminal rewards are fixed, these bounds hold with $\varepsilon = a_\beta \varepsilon_p + \varepsilon_c$.

Proof. The difference of interval residuals is $(\bar{v}_i - v_i) - (\bar{v}_j - v_j)$, bounded by 2ε . There are K_T residuals, so the Euclidean distance between their vectors is at most $2\varepsilon\sqrt{K_T}$. Divide the reverse triangle inequality for their norms by $\sqrt{K_T}$ to obtain the loss bound. By Eq. (A.22), the coefficient difference is the sum of $(t + 1)(T - t)$ residual differences times $2/K_T$; bounding each summand gives Eq. (B.9). Lemma B.4 gives the rest, the terminal being fixed. $\square$

Remark B.2 (The residual head lies outside clipping). The measured part of Eq. (B.1) lies in $[0, \beta]$ and b_ψ is added outside it; a changing head adds its perturbation to ε in Proposition B.5.

B.4 RATIO-FIRST STRUCTURAL POOLING

Definition B.3 (Visit-multiplicity pooling). For a structural bucket g , let $\mathcal{V}_g$ be the multiset of eligible prefix visits (x, t) assigned to that bucket, and set $n_g = |\mathcal{V}_g|$. Each included occurrence is counted in both the sum and the denominator. Put $B_x = e^{u_{q,x}}$ and $z_{x,t} = R_\beta(x)/B_x$. For $n_0 \geq 0$, $c_{\max} \geq 0$, and $n_g + n_0 > 0$, define

$$A_g = 1 + \frac{\sum_{(x,t) \in \mathcal{V}_g} (z_{x,t} - 1)}{n_g + n_0}, \quad c(g) = \text{clip}_{[-c_{\max}, c_{\max}]}(\log A_g). \quad (\text{B.10})$$

These are the exact-arithmetic quantities before the positive-domain numerical safeguard in the implementation description. Buckets use a canonical graph and an open/stop marker, fall back to coarse node and edge counts below eight visits, and are read one batch behind. The configured values are $n_0 = 8$ and $c_{\max} = 0.25$.

Lemma B.6 (Positive denominator algebra and bounds). *Under Definition B.3, if each $z_{x,t} > 0$ is finite, then*

$$A_g = \frac{n_0 + \sum_{(x,t) \in \mathcal{V}_g} z_{x,t}}{n_g + n_0} > 0. \quad (\text{B.11})$$

If $R_\beta(x), B_x \in [1, e^\beta]$, then $A_g \in [e^{-\beta}, e^\beta]$ and $|c(g)| \leq \min(\beta, c_{\max})$. If $n_g = 0 < n_0$, then $A_g = 1$ and $c(g) = 0$, and the formula defines no value when $n_g = n_0 = 0$.

Proof. The number of subtracted ones is exactly n_g , so combining terms over the common denominator gives Eq. (B.11). If $n_g > 0$, its numerator contains a positive term; if $n_g = 0$, the positive denominator forces $n_0 > 0$. This proves strict positivity. The ratio is a weighted average of the values $z_{x,t}$ and the pseudo-observation one. Each lies in $[e^{-\beta}, e^\beta]$ under the additional bounds. The same interval contains their weighted average; logarithm and clipping give the stated bound. Substituting $n_g = 0$ into the ratio gives $A_g = 1$ and $c(g) = 0$, which proves the empty-bucket case. $\square$

Proposition B.7 (A visit-weighted population interpretation). *For this proposition only, suppose the tuples (X_h, B_h, V_h) are i.i.d. under a fixed law, where $z_h = R_\beta(X_h)/B_h > 0$ and V_h counts the eligible visits to a fixed bucket in record h . Assume $0 < \mathbb{E}V_h < \infty$ and $\mathbb{E}[V_h z_h] < \infty$. With a fixed finite n_0 , pooling the first m records satisfies*

$$\frac{n_0 + \sum_{h=1}^m V_h z_h}{n_0 + \sum_{h=1}^m V_h} \longrightarrow \frac{\mathbb{E}[V_h z_h]}{\mathbb{E}V_h} \quad \text{almost surely.} \quad (\text{B.12})$$

The log and clipped-log quantities converge to the corresponding transforms of this positive limit.

Proof. The strong law applied to $V_h z_h$ and V_h makes their averages converge to the two expectations. Divide numerator and denominator by m ; $n_0/m \rightarrow 0$, and the limiting denominator is strictly positive, so the quotient converges. Its limiting numerator is positive because $z_h > 0$ on the positive-probability event $V_h > 0$. Continuity of log at a positive limit and of clipping gives the rest. $\square$

Three sources of statistical discrepancy. First, ratio-first pooling targets an arithmetic ratio average, and the finite-sample Jensen gap remains: for a positive random average $\bar{z}$, $\mathbb{E} \log \bar{z} \leq \log \mathbb{E} \bar{z}$. Second, random denominators matter even before taking a logarithm:

$$\mathbb{E}[R/B] = \mathbb{E}[R]\mathbb{E}[B^{-1}] + \text{Cov}(R, B^{-1}), \quad (\text{B.13})$$

when these moments exist. This identity is the definition of covariance; with $R = 1$ and B equally likely to be one or two, $\mathbb{E}[R/B] = 3/4$ while $\mathbb{E}[R]/\mathbb{E}[B] = 2/3$. For $\beta \geq \log 2$ these denominators obey the preceding bounded range; shrinkage and clipping add further differences.

Third, visit pooling weights a record by V_h , so its limiting ratio in Eq. (B.12) is a visit-weighted average. Repeated visits share the terminal reward and can be strongly correlated. For a fixed number $n > 0$ of random visit ratios with finite second moments,

$$\text{Var}\left(\frac{1}{n} \sum_{v=1}^n z_v\right) = \frac{1}{n^2} \sum_{v=1}^n \sum_{w=1}^n \text{Cov}(z_v, z_w),$$

by bilinearity of covariance, so a visit count measures visits rather than independent tasks. Structural reads lag one batch, and buckets also include paired reference prefixes (Section C.5).

B.5 VALUE-HEAD PROJECTION AND CALIBRATION

Definition B.4 (Feature-level regression distribution). Let $\mathcal{D}$ be the actual distribution of scored training pairs (S, Y) for the value head, where X is the terminal continuation record from S and $Y = r(X) \in [0, 1]$ its reward. For Eq. (5), $\mathcal{D}$ denotes the actual scored-pair law represented by $\mathcal{D}_\rho$, including its stated data filters. The input is $z(S) = [f(S); \text{onehot}(\text{tasktype}(q))] \in \mathbb{R}^{30+6}$. Write $Z_f = z(S)$ for the random feature vector; it is unrelated to the normalizer $Z(C)$. Define $m_{\mathcal{D}}(Z_f) = \mathbb{E}_{\mathcal{D}}[Y \mid Z_f]$.

Proposition B.8 (Feature-conditional MSE projection and calibration). *Over measurable square-integrable functions of Z_f , the population risk in Eq. (5) decomposes as*

$$\mathbb{E}_{\mathcal{D}}(v(Z_f) - Y)^2 = \mathbb{E}_{\mathcal{D}}(v(Z_f) - m_{\mathcal{D}}(Z_f))^2 + \mathbb{E}_{\mathcal{D}} \text{Var}(Y \mid Z_f). \quad (\text{B.14})$$

Its unique minimizer up to $\mathcal{D}$ -null sets is $m_{\mathcal{D}}$, and

$$\mathbb{E}_{\mathcal{D}}[Y \mid m_{\mathcal{D}}(Z_f)] = m_{\mathcal{D}}(Z_f). \quad (\text{B.15})$$

For any such predictor $v(Z_f)$, its squared mean-calibration error is at most its excess risk above this unrestricted optimum:

$$\mathbb{E}_{\mathcal{D}}[(\mathbb{E}_{\mathcal{D}}[Y \mid v(Z_f)] - v(Z_f))^2] \leq \mathbb{E}_{\mathcal{D}}(m_{\mathcal{D}}(Z_f) - v(Z_f))^2. \quad (\text{B.16})$$

Proof. Write $v - Y = (v - m_{\mathcal{D}}) + (m_{\mathcal{D}} - Y)$ and expand its square. The cross term has conditional expectation zero given Z_f because $\mathbb{E}[Y - m_{\mathcal{D}} \mid Z_f] = 0$. The remaining conditional squared error is $\text{Var}(Y \mid Z_f)$, proving the risk identity. The first term is nonnegative and vanishes exactly when $v = m_{\mathcal{D}}$ almost surely, which proves the minimizer claim. Since $m_{\mathcal{D}}$ is measurable with respect to Z_f , the tower property gives Eq. (B.15). For the last bound, v is also a function of Z_f , and hence

$$\mathbb{E}[Y \mid v] - v = \mathbb{E}[m_{\mathcal{D}} - v \mid v].$$

Conditional Jensen’s inequality for the square followed by expectation gives Eq. (B.16). $\square$

Complete-history values and a finite sigmoid network. Equality of $m_{\mathcal{D}}(z(s))$ with the ideal reference value $\mu_\rho(s, C) = \mathbb{E}_\rho[r(X) \mid s, C]$ holds when the continuation-data law is compatible and the features are sufficient for that conditional mean; two histories that share features but have values $1/4$ and $3/4$ show why sufficiency is needed. Calibration above is with respect to $\mathcal{D}$.

The implemented head is the sigmoid MLP of Eq. (5) with $30 + 6$ inputs and hidden width 32: the 30 execution features and a six-slot task-type one-hot, one slot per IID benchmark. Finite logits produce outputs strictly between zero and one; an unrestricted sigmoid predictor approaches a conditional mean of zero or one by clipping it to $[\epsilon, 1 - \epsilon]$ and taking its logit, with squared excess risk at most ϵ^2 . The head estimates a success probability for binary rewards and an expected score otherwise. Its weights are exported per batch and evaluated on CPU with no extra encoder call. The two larger outcome heads on $[h_\rho(s); f(s)]$ serve as diagnostics.

B.6 THE IMPLEMENTED OPTIMUM AND THE IDEAL FLOW

Proposition B.9 (Squared-log regression and the arithmetic normalizer). *In the one-action executor of Remark A.4, take a shared scalar root value u_0 and the exact terminal boundary. Taking expectation under that reference executor law, for $\beta > 0$ the population AnchorTB loss is minimized at $u_0 = \beta/2$, not at the ideal root $\log[(1 + e^\beta)/2]$, and its minimum is $\beta^2/4 > 0$.*

Proof. There is one action and both policies assign it probability one, so $\Delta\ell_0 = 0$, $T = K_T = 1$. The terminal log reward is zero or β , each with probability $1/2$. Consequently

$$\mathcal{L}(u_0) = \frac{1}{2}u_0^2 + \frac{1}{2}(u_0 - \beta)^2 = (u_0 - \beta/2)^2 + \beta^2/4.$$

This proves the minimizer and minimum; also $Z = (1 + e^\beta)/2$. Strict concavity of $\log$ on the two distinct rewards gives $\mathbb{E} \log R_\beta = \beta/2 < \log \mathbb{E} R_\beta$ for $\beta > 0$, and both vanish at $\beta = 0$. $\square$

Remark B.3 (Why sharedness is needed). For any fixed pair of supported policies, define the artificial family $v_x(s_i) = \log R_\beta(x) - \sum_{t=i}^{T-1} \Delta\ell_t$ on each complete record. The empty sum at $i = T$ gives the exact terminal boundary, and subtraction shows $v_x(s_i) + \Delta\ell_i - v_x(s_{i+1}) = 0$ at every step. Thus all its within-record losses are zero whether or not $P_\theta = P^*$, and the family is generally not shared across continuations; distributional results therefore rely on sharedness.

The ideal $u^*(s \mid C)$, the measured task anchor $u_{q,x}$, the implemented $\tilde{u}_x(s)$, and a population regression minimizer are distinct objects. The first is identified by a conditional expectation and a recursion; the next two are specified estimators; the last on its function class and training law.

C SEQUENTIAL VALIDATION AND THE TRAINING LOOP

Paired validation compares two specified rollout regimes. We identify that effect, prove fixed-look directional tests and the error accounting of an adaptive run, and close with the training loop.

C.1 PAIRED INTERVENTION REGIMES AND EFFECT ESTIMANDS

Definition C.1 (Paired regimes, counts, and empty samples). For a registered candidate, the two arms start from the empty team with the same task record, runtime configuration, value-head snapshot, and forced first role. The candidate arm binds the candidate at its first `ADD_AGENT`; the control arm does not, and keeps the candidate unavailable later. Both arms subsequently use the reference policy under their respective histories. They share the other validated skills, and their later decisions and observations may differ. For complete pair i , set $b_i^\pm = \mathbf{1}\{r(x_i^\pm) \geq 1/2\}$ and define

$$\begin{aligned} W &= \sum_{i=1}^N \mathbf{1}\{b_i^+ = 1, b_i^- = 0\}, & L &= \sum_{i=1}^N \mathbf{1}\{b_i^+ = 0, b_i^- = 1\}, \\ T_0 &= \sum_{i=1}^N \mathbf{1}\{b_i^+ = b_i^-\}, & D &= W + L, & N &= W + L + T_0. \end{aligned} \quad (\text{C.1})$$

When $N > 0$, $\hat{\Delta} = (W - L)/N$. For $N = 0$ the empirical effect is undefined, rather than an estimate of zero. With $D = 0$, both directional p-values below are defined to be one, and there is no rejection.

Lemma C.1 (Paired effect identity). For $N > 0$,

$$\hat{\Delta} = \frac{1}{N} \sum_{i=1}^N (b_i^+ - b_i^-), \quad |\hat{\Delta}| \leq 1. \quad (\text{C.2})$$

For a specified pair distribution, let p_W and p_L be its win and loss probabilities. Its threshold-pass effect is

$$\Delta = \mathbb{P}(b^+ = 1) - \mathbb{P}(b^- = 1) = p_W - p_L. \quad (\text{C.3})$$

Neither identity requires independence of the two arms within a pair.

Proof. A win contributes one to $b_i^+ - b_i^-$, a loss contributes minus one, and a tie contributes zero. Summing proves the sample identity, and boundedness of each summand proves its range. Taking expectations gives the population identity without factoring joint arm probabilities. $\square$

Remark C.1 (The intervention and the reward threshold). The intervention begins at the first action of an empty team, and the control keeps the candidate unavailable throughout, so Δ is the full-regime threshold-pass effect, later decisions included. For continuous scores Δ counts threshold crossings on r : a reward moving from 0.49 to 0.51 gives $\Delta = 1$ but changes the mean by 0.02.

C.2 EXACT FIXED-LOOK DIRECTIONAL TESTS

Assumption C.1 (Registered fixed-look i.i.d. pair model). Let $\mathcal{G}_j$ denote information available at registration of comparison j . Conditional on it, the candidate, arm protocols, target pair distribution, and a finite nonnegative integer N of complete pairs for this look are fixed. The future pair outcomes are conditionally i.i.d. under that distribution. Within-pair dependence is allowed. No data used to choose the candidate are reused as its future validation outcomes under this assumption. Denote the conditional win and loss probabilities by p_W, p_L , and the corresponding mean threshold effect by $\Delta_j = p_W - p_L$.

Proposition C.2 (Both composite directional nulls at a fixed look). Under Assumption C.1, define

$$p_+ = \mathbb{P}\{B_D \geq W\}, \quad p_- = \mathbb{P}\{B_D \geq L\}, \quad B_D \sim \text{Bin}(D, 1/2), \quad (\text{C.4})$$

Here probabilities in the displayed tails are over the auxiliary binomial variable with the observed D, W, L held fixed; use the $D = 0$ convention of Definition C.1. For every $u \in [0, 1]$,

$$H_j^+ : \Delta_j \leq 0 \implies \mathbb{P}(p_+ \leq u \mid \mathcal{G}_j) \leq u, \quad (\text{C.5})$$

$$H_j^- : \Delta_j \geq 0 \implies \mathbb{P}(p_- \leq u \mid \mathcal{G}_j) \leq u. \quad (\text{C.6})$$

Each inequality holds on its null event, so each tail is valid for its whole composite directional null.

Proof. All probabilities in the proof condition on $\mathcal{G}_j$. Put $p_D = p_W + p_L$. If $p_D = 0$, then $D = 0$ almost surely and $p_+ = p_- = 1$, which satisfies both inequalities; the same conclusion holds if $N = 0$. Otherwise, for d of positive probability and $0 \leq w \leq d$, the multinomial law of wins, losses, and ties gives

$$\mathbb{P}(W = w, L = d - w, D = d) = \frac{N!}{w!(d-w)!(N-d)!} p_W^w p_L^{d-w} (1-p_D)^{N-d},$$

$$\mathbb{P}(D = d) = \binom{N}{d} p_D^d (1-p_D)^{N-d}.$$

Dividing, with the usual limiting conventions when a cell probability is zero, proves

$$W \mid D = d \sim \text{Bin}(d, \vartheta), \quad \vartheta = p_W/p_D, \quad L \mid D = d \sim \text{Bin}(d, 1 - \vartheta).$$

Under H_j^+ we have $p_W \leq p_L$, so $\vartheta \leq 1/2$. With independent uniform variables $U_1, \dots, U_d$, the sum $\sum_i \mathbf{1}\{U_i \leq \vartheta\}$ is pointwise at most $\sum_i \mathbf{1}\{U_i \leq 1/2\}$. Hence the former binomial is stochastically dominated by the latter. Define

$$k_u(d) = \min\{k \in \{0, \dots, d\} : \mathbb{P}(B_d \geq k) \leq u\},$$

using $k_u(d) = d + 1$ if the set is empty. The tail is nonincreasing in its observed count, so $\{p_+ \leq u\} = \{W \geq k_u(d)\}$ conditional on $D = d$, and stochastic domination then gives

$$\mathbb{P}(p_+ \leq u \mid D = d) \leq \mathbb{P}(B_d \geq k_u(d)) \leq u.$$

For $d = 0$, the assigned value one has the same validity property. Averaging over D proves Eq. (C.5).

Under H_j^- , $\vartheta \geq 1/2$, so $1 - \vartheta \leq 1/2$. Apply the same uniform coupling to the conditional law of L . The event $\{p_- \leq u\}$ is $\{L \geq k_u(d)\}$, whose probability is at most the corresponding fair-binomial upper tail and thus at most u . Averaging over D , including $D = 0$, proves Eq. (C.6). $\square$

Lemma C.3 (A stronger fixed-context conditional-sign model). *Fix a candidate and a prespecified finite collection of complete pairs. Condition on the full vector of contexts $\mathbf{C}$, the full discordance vector $\mathbf{D} = (\mathbf{1}\{b_i^+ \neq b_i^-\})_i$, and registration information. Suppose the retained win signs $Y_i = \mathbf{1}\{b_i^+ = 1, b_i^- = 0\}$ are conditionally independent. If every retained sign has conditional probability $p_i \leq 1/2$, then p_+ is conditionally super-uniform. If every $p_i \geq 1/2$, then p_- is conditionally super-uniform. If all $p_i = 1/2$, then $W \mid (\mathbf{C}, \mathbf{D}, \mathcal{G}_j)$ is exactly $\text{Bin}(D, 1/2)$.*

Proof. After conditioning, the retained index set and its size $D = d$ are fixed. For the positive direction construct independent uniforms and represent each independent sign as $Y_i = \mathbf{1}\{U_i \leq p_i\}$. When $p_i \leq 1/2$, each sign is bounded above by $\mathbf{1}\{U_i \leq 1/2\}$, so their sum is stochastically dominated by $\text{Bin}(d, 1/2)$. Using the cutoff $k_u(d)$ from the preceding proof gives the super-uniform upper tail. When $p_i \geq 1/2$, the independent loss signs $1 - Y_i$ instead have probabilities at most $1/2$; the same argument applies to p_- . When every $p_i = 1/2$, independence identifies the sum as precisely the fair binomial. The zero-discordance convention handles an empty set. $\square$

Corollary C.4 (Two-direction accounting at one look). *If each valid direction receives level $b/2$, the probability of any false directional rejection at that look is at most b . Also, for the p -values in Eq. (C.4), both directions cannot simultaneously satisfy $p_{\pm} \leq a$ when $a < 1/2$.*

Proof. Sum the error probabilities only over directions whose nulls are true. There are at most two, each bounded by $b/2$, so the union bound gives b , without independence. For the second claim, fair-binomial symmetry and $L = D - W$ imply $p_- = \mathbb{P}(B_D \leq W)$, so

$$p_+ + p_- = 1 + \mathbb{P}(B_D = W) \geq 1,$$

which no two numbers at most $a < 1/2$ satisfy. If $D = 0$, both p -values equal one. $\square$

Two null models. The i.i.d. pair proposition tests an average threshold effect under the registered pair distribution, possibly averaging over randomly sampled tasks. The fixed-context lemma assumes the stronger sign inequalities for every retained pair after conditioning on *all* contexts and discordance indicators; equally weighted contexts with certain wins in one and certain losses in the other show that an average zero effect can coexist with opposite deterministic conditional signs.

C.3 ADAPTIVE SELECTION, REPEATED PROBLEMS, AND REPEATED LOOKS

Remark C.2 (Correlated-sign counterexample). Draw one fair Bernoulli variable Y , and for $i = 1, \dots, 8$ set $(b_i^+, b_i^-) = (Y, 1 - Y)$. Every pair has zero marginal effect, but all eight signs agree. The observed direction always has p-value $2^{-8} = 0.00390625$. At the configured $\alpha = 0.05$ and the first allocation $j = \ell = 1$, its threshold is $\alpha/8 = 0.00625$. Thus one false directional rejection occurs with probability one, although the nominal spending formula is obeyed. Correlated duplicates therefore carry the evidence of one pair rather than eight, and validity depends on the conditional outcome and dependence model.

Remark C.3 (Candidate-selection counterexample). For integers $M \geq 2$ and $d \geq 1$, consider M true zero-effect candidates, each with d independent fair validation signs, independent across candidates. Use these same signs to select a candidate with all wins whenever one exists, and then report its ordinary fixed-candidate p-value. The event of at least one all-win candidate has probability $1 - (1 - 2^{-d})^M$, while the selected p-value on this event is 2^{-d} . For $M > 1$ this exceeds its nominal tail probability and can approach one. The invalidity arises from selection and reuse of screening outcomes. The protocol below selects from registration information and validates on future data.

Planned looks and cumulative data. Suppose a fixed candidate has valid p-values at prespecified cumulative sample sizes $n_1, n_2, \dots$, with deterministic levels $a_1, a_2, \dots$. Although the looks reuse data and their p-values are correlated,

$$\mathbb{P}\{\text{some true-null look rejects}\} \leq \sum_{\ell} \mathbb{P}(p_{\ell} \leq a_{\ell}) \leq \sum_{\ell} a_{\ell}.$$

Independence between looks is unnecessary, and early stopping only removes potential later rejections. What matters is that the *actual* look indexed by ℓ retains a valid null law; renumbering selected looks, choosing the sample size by favorable signs, reusing candidate-selection outcomes, or filtering complete pairs on outcomes can change that law. The registration-based protocol below conditions on registration information, which is all the union bound requires.

C.4 NOMINAL SPENDING AND CONDITIONAL COUNTABLE FWER

Proposition C.5 (Alpha-spending accounting). *Let $0 < \alpha < 1$, and assign each registered comparison $j \geq 1$ and look $\ell \geq 1$ two directional allocations*

$$a_{j,\ell,d} = \frac{\alpha}{2j(j+1)\ell(\ell+1)}, \quad d \in \{+, -\}. \quad (\text{C.7})$$

If each directional index is globally unique and used at most once, any realized run spends at most α .

Proof. For every integer $M \geq 1$, $\sum_{n=1}^M 1/[n(n+1)] = \sum_{n=1}^M (1/n - 1/(n+1)) = 1 - 1/(M+1)$. Therefore, for finite J, L ,

$$\sum_{j=1}^J \sum_{\ell=1}^L \sum_{d \in \{+, -\}} a_{j,\ell,d} = \alpha \left(1 - \frac{1}{J+1}\right) \left(1 - \frac{1}{L+1}\right).$$

Monotone limits give total allocation α , and the consumed indices, a subset, sum to at most α . $\square$

Proposition C.6 (Conditional-validity family-wise error bound). *For each registered j , let $H_{j,d}$ be the event that its directional null is true, measurable with respect to registration information $\mathcal{G}_j$. Let $E_{j,\ell,d}$ be the event that the directional test is actually performed and rejects a true null. Unregistered or unperformed tests have empty rejection events. Suppose the actual selection, sampling, and observation rules ensure*

$$\mathbb{P}(E_{j,\ell,d} \mid \mathcal{G}_j) \leq a_{j,\ell,d} \mathbf{1}_{H_{j,d}} \quad \text{almost surely for every } j, \ell, d. \quad (\text{C.8})$$

Then the probability of any false directional rejection is at most α . More generally, if a pre-run information field $\mathcal{F}_0$ is contained in every $\mathcal{G}_j$, the same bound holds conditional on $\mathcal{F}_0$.

Proof. By the tower property, Eq. (C.8) gives $\mathbb{P}(E_{j,\ell,d}) \leq a_{j,\ell,d}$. For a finite rectangle of indices, the probability of the union is at most their summed probabilities. Increasing these rectangles to all positive indices and using continuity of probability from below gives

$$\mathbb{P}\left(\bigcup_{j,\ell,d} E_{j,\ell,d}\right) \leq \sum_{j,\ell,d} \mathbb{P}(E_{j,\ell,d}) \leq \sum_{j,\ell,d} a_{j,\ell,d} = \alpha.$$

For the conditional version, the tower property with $\mathcal{F}_0 \subseteq \mathcal{G}_j$ first gives $\mathbb{P}(E_{j,\ell,d} \mid \mathcal{F}_0) \leq a_{j,\ell,d}$. Apply the finite conditional union bound and conditional monotone convergence to the same increasing sequence of unions. This proves the bound almost surely given $\mathcal{F}_0$. The argument uses no independence between comparisons or looks, so it covers dependent tests. $\square$

Assumption C.2 (A sufficient registration-and-future-data protocol). For each comparison j , the protocol consists of the following conditions. The candidate may be chosen adaptively from $\mathcal{G}_j$, but its content, arm protocols, response law, and target pair distribution are then fixed for that comparison. Conditional on $\mathcal{G}_j$, future complete pairs form the i.i.d. stream in Assumption C.1; past screening outcomes are not reused, and pair inclusion does not selectively retain favorable outcomes. A nondecreasing countable sequence of finite nonnegative integer cumulative sample sizes $n_{j,\ell}$ is $\mathcal{G}_j$ -measurable and fixed before those future outcomes are seen. Each potential p-value is computed from the first $n_{j,\ell}$ pairs, and keeps its original planned index and allocation. Whether a planned test is ultimately performed may depend on available information, but cannot change that potential test or its data law. Unperformed planned tests consume no level.

Corollary C.7 (FWER under the sufficient protocol). *Assumption C.2, the unique indices of Proposition C.5, and rejection at the allocations in Eq. (C.7) imply the bound of Proposition C.6.*

Proof. Conditional on $\mathcal{G}_j$, each $n_{j,\ell}$ is fixed. Apply Proposition C.2 to the corresponding future-data prefix to obtain, on $H_{j,d}$, $\mathbb{P}(p_{j,\ell,d} \leq a_{j,\ell,d} \mid \mathcal{G}_j) \leq a_{j,\ell,d}$. The actual false rejection event is a subset of $H_{j,d} \cap \{p_{j,\ell,d} \leq a_{j,\ell,d}\}$, even when the decision to execute a planned look uses earlier outcomes. Since $H_{j,d}$ is registration-measurable, this subset relation proves Eq. (C.8). The preceding proposition then gives the claimed bound, also when cumulative prefixes overlap. $\square$

Remark C.4 (Error accounting for the online procedure). The implemented observation index advances when new wins or losses exist. For this procedure we establish nominal accounting (Proposition C.5) and whole-run FWER under the premise of Eq. (C.8). Dependence *between* valid planned tests leaves the union bound intact. These results bound false directional rejections.

C.5 TRAINING ALGORITHM AND DATA ROUTING

Skill proposal rules. The author model runs only in an author window and distills at most one candidate per window from scored, side-effect-free trajectories, preferring task families with few skills and contrasting a high-scoring run with a same-task failure when one exists. A candidate is a structured procedure with name, description, trigger, plan, pitfall, and constraint fields, and is filtered for duplicates and answer leakage. Candidate slots are ordered by fewest paired observations. Paired rollouts enter AnchorTB and structural buckets as off-policy paths but not the task, category, and global baseline statistics, and the forced first action is scored at its true probability under θ and ρ .

The following algorithm describes the frozen implementation. It is distinct from the sufficient testing protocol in Assumption C.2. The same trajectory-indexed endpoint values are reused across all intervals of each scored trajectory. For paired records, “the same context” in the re-scoring step means that θ and ρ use that record’s actual arm context, C^+ or C^- , and its legal mask.

Remark C.5 (Implementation details). The paired design intervenes only at the first ADD_AGENT, and the active skill set exposes slot identifiers rather than full skill text.

Data routing. Natural healthy non-paired reference trajectories supply the hierarchical task, category, and global statistics. Natural and paired paths can supply the AnchorTB regression. Structural buckets also receive paired reference prefixes, without the forced-prefix filter of the value-head data and with their visit multiplicities retained. Conditional flow statements apply to a fixed C .

Algorithm 1 One training step of EvoSteer

1. Sample 32 task records balanced over sources.
2. Roll out $2\theta + 2\rho$ natural trajectories per record, plus one candidate and one control rollout per record in a family with a candidate, under fixed skill-menu and value-head snapshots; execute every action on issue and append feedback and features.
3. Gate the batch on completeness and risk; publish the current adapter to the sampling service and prefetch the next batch.
4. Re-score every action under θ and ρ with the same context, tokens, and grammar mask.
5. Fit the value head and auxiliary heads on legal reference states.
6. Build same-task shrinkage baselines; read structural buckets from earlier batches; accumulate this batch’s ratios.
7. Compute all subtrajectory residuals of Eq. (7) and update θ and b_ψ .
8. Accumulate paired wins, losses, and ties; at each validation round, run the author window and the sequential tests of Eq. (11).
9. Save a checkpoint with adapter, heads, optimiser, statistics, skills, and test state.

C.6 PROOFS OF THE MAIN-TEXT PROPOSITIONS

The first main-text proposition is procedural: legal repairs execute before the next decision and preserve the history representation (Lemma A.1). The second follows from the action coefficient in Proposition A.14 and the estimator components in Definition B.1. The third combines candidate-slot and status-transition rules with Proposition C.5’s nominal accounting; its statistical interpretation is fixed-look validity in Proposition C.2 and the conditional whole-run result of Proposition C.6.

D EXPERIMENTAL DETAILS

Benchmarks and splits. The six IID benchmarks (HotpotQA, NQ-Open, MedQA, AIME 2026, MBPP+, ALFWorld) supply the training tasks, and their test items are disjoint from those tasks; all 30 AIME 2026 problems are test items, and the mathematics training tasks are compiled from earlier AIME problems. The six OOD benchmarks (TriviaQA, MuSiQue, GPQA, MATH-Hard, SWE-Bench Verified, WebShop) are evaluation-only. Every other test set has 128 items, so one run’s 0/1 metric lies on a $k/128$ grid ($k/30$ for AIME 2026) and a reported five-run mean on a $k/640$ grid ($k/150$); GPQA uses 128 random Diamond questions, and MATH-Hard is the level-5 subset of MATH. The exact training and test splits and the full configuration are released in our anonymous code repository.

Metrics. Answer exact match (Ans EM) and token-level F1 (Ans F1) use SQuAD-style answer normalization and the maximum over reference aliases. Accuracy (Acc.) is the share of correct final answers: the chosen option on MedQA and GPQA and the final value on AIME 2026 and MATH-Hard. Pass@1 on MBPP+ and the success rate (SR) on WebShop follow the official evaluators; SR on ALFWorld is the share of episodes that complete the task within 50 steps. The resolved rate is the share of SWE-Bench Verified issues whose patch passes the official tests.

Baselines and fairness. Unless a column is labeled otherwise, every method uses Qwen3.5-9B both as the frozen executor and as the model it trains, and every trained method uses the same training tasks; SFT fits the reference answers of those tasks. Each baseline runs in the best configuration reported by its authors. The DeepSeek-V4-Flash column prompts that model directly, as a larger reference point. GRPO[†] in Table 1 fine-tunes the backbone itself, whereas GRPO in Figure 6(a,b) trains π_θ inside the EvoSteer architecture with the same harness, actions, and tools.

Ablation variants and fixed paradigms. Each ablation in Table 2 removes one mechanism and restores the usual alternative: — interleaved execution builds the whole graph before running it; — execution features keeps interleaved execution and the textual feedback o_t^{exec} but removes the feature vector f from the state of π_θ , so the orchestrator reads the textual history and the value estimate but not f , while the value head and b_ψ are unchanged; — learnable repair masks RERUN and DROP; — reference value head withholds $\hat{v}_k$ and $\Delta\hat{v}$ from the state; — measured flows learns u_q as a free

Group	Setting
Backbones	orchestrator, reference, executor: one shared base model (Qwen3.5-9B); skill author: DeepSeek-V4-Flash, called only in author windows
Adapter	LoRA rank 64, $\alpha = 128$, on q/k/v/o_proj, out_proj, and in_proj_qkv; AdamW ($\beta_1 = 0.95$), lr 5×10^{-6}
Heads	b_ψ : zero-initialized MLP on $[h_\rho; f]$; value head: two-layer MLP on 30+6 inputs (one task-type slot per IID benchmark)
AnchorTB	$\beta = 2$; all subtrajectories equally weighted; gradient-norm clip 1.0
Baselines	$(\kappa_q, \kappa_c, \kappa_g) = (0.5, 4, 2)$; prior success rate capped at 20 counts; bucket minimum 8; pseudo-count $n_0 = 8$; correction bound $c_{\max} = 0.25$
Skill slots	at most three validated skills and one candidate per task type
Batch	32 tasks per step, each with $2\theta + 2\rho$ natural trajectories, plus paired rollouts (192 episodes per step); asynchronous prefetch
Orchestration	team size, repair count, and episode length chosen by the policy; bounded only by the shared budget and the learned STOP
Budget	shared resource budget of 98,304 tokens, 50 tool calls, and 600 s per episode
Temperatures	orchestrator 1.0; executor 0.3
Skills	at most one new candidate per author window; library ≤ 60 ; ≤ 12 per family
Validation	sequential sign test at each validation round; $\alpha = 0.05$ with two-level alpha spending

Table 3: Configuration of the frozen implementation used for the experiments.

scalar instead of reading it from $\hat{p}_q$; – flow corrections turns off $c(g)$ and b_ψ ; – skill evolution removes the whole skill mechanism (no author window, no candidates, no paired rollouts, and an empty skill library), so the team uses roles and tools only; – sequential validation admits a candidate after a fixed number of successes. The four fixed paradigms decide the team before execution, on the same executor and budget as EvoSteer: a ReAct-style agent with all tools (Yao et al., 2022b), one hand-designed multi-agent template for all tasks, a planner that writes the graph once and then runs it, and a workflow searched offline on the training tasks and then frozen.

Test-time protocol and transfer. At test time EvoSteer runs the trained π_θ with the value head and the skill library frozen; no reference, paired, or validation rollouts are drawn. The transfer study of Figure 5 reuses this orchestrator unchanged and swaps only the executor, called through its API identifier: gpt-5.6-luna (GPT-5.6 Luna), grok-4.5 (Grok 4.5), claude-haiku-4-5 (Claude Haiku 4.5), deepseek-v4-flash (DeepSeek-V4-Flash), gemini-3.5-flash (Gemini 3.5 Flash), and glm-5.3-flash (GLM-5.3-Flash). Frozen-backbone scores prompt each model directly, as in the v4-flash column of Table 1. Each OOD benchmark uses its IID counterpart’s task type (e.g., MuSiQue that of HotpotQA, SWE-Bench that of MBPP+), so OOD results test generalization across matched task types.

Skill author. The skill author only writes candidate skills in author windows; validation, training, and testing are unchanged. Table 4 replaces DeepSeek-V4-Flash with a stronger author, GPT-5.6-Luna, and with the executor itself, Qwen3.5-9B, and reports the averages of Table 1. The stronger author raises every average, self-written skills lower them by at most 1.56 points, and all three settings stay above the strongest baseline of Table 1 in every column.

Skill author	IID avg.		OOD avg.	
	Ans EM	Acc.	Ans EM	Acc.
DeepSeek-V4-Flash (main setting, Table 1)	88.67	87.42	90.08	78.55
GPT-5.6-Luna	90.16	89.28	90.55	79.34
Qwen3.5-9B (the executor)	87.81	86.64	88.98	76.99

Table 4: EvoSteer with different skill authors: five-run means averaged as in Table 1 (Ans EM over the question-answering benchmarks, Acc. over the others).

Objectives and cost. In Figure 6(a,b) every objective trains the same untrained architecture of Table 2, with the harness, action space, tool set, and skill library unchanged. Tempered TB follows the implementation of Zhang et al. (2026c): the log reward is tempered as $\beta \log(R + \epsilon)$, $\log Z$ and the backward policy are learned, and each edge’s log-probability is normalized by its token count. GPU time per episode is the compute of the parameter update (PPO’s counts actor and critic, GRPO’s includes its KL term). Tokens per problem is the rollout cost, prefill plus decode; training rollouts

run the same orchestration as inference, so it is also the inference cost. Training EvoSteer for 240 steps used 95.98 H800 GPU hours and 1,580,691,323 tokens in total. The fixed-count admission of Table 2 admits a candidate after $k = 5$ successes; every k from 1 to 10 gives nearly the same results.

Diagnostics. In Figure 5(b), the reference curve is the accuracy of the frozen ρ on the natural reference rollouts of the same batch, which changes with the batch although ρ does not; the anchor-only level evaluates the AnchorTB loss with $b_\psi = 0$ and $\Delta\ell_t = 0$, so that each flow equals its measured part $\text{clip}_{[0,\beta]}(u_{q,x} + c(g))$. In Figure 6(c), the predictor is the leave-one-out task anchor $\hat{p}_q$ (equivalently u_q , a monotone transform with the same AUC), the target is the final correctness of every natural reference rollout after step 2, and the prediction is made before the episode starts; the task-family mean uses the same reference statistics at the same step. In Figure 6(d), the gain of a skill is its paired effect: the same task and initial state, the first action forced to bind the skill or not, and both arms continued by the frozen ρ . In Figure 6(e), an edit is better, the same, or worse when the task grader’s score of the designated answer rises, stays, or falls from just before to just after the edit on the same trajectory. In Figure 6(f), the orchestrator is trained once with each of the Qwen3.5-9B and DeepSeek-V4-Flash executors, and the gain of value-guided replanning is measured during training.

Runs and aggregation. Every test score is the mean over five independent runs. Table 1 also reports their standard deviation; for the Avg. rows it is $\sqrt{\sum_b \sigma_b^2 / n}$ over n benchmarks.

E CASE STUDY

E.1 CODE THAT PASSES THE VISIBLE TEST

We present a case from MBPP+ in which the executor returns code that looks finished: it is short, it runs, and it passes the only test given in the task. The code is wrong, and $\hat{v}$ falls by 0.216 at that point; the orchestrator reruns the agent instead of submitting, and the rerun passes every hidden test.

MBPP+ Problem 801 (training step 87)

Write a python function to count the number of equal numbers from three given integers. Your code should pass this test:

```
assert test_three_equal(1,1,1) == 3
```

Ground Truth (hidden tests; only the first is shown in the task): `test_three_equal(1,1,1) == 3, test_three_equal(-1,-2,-3) == 0, test_three_equal(1,2,2) == 2`

Final Team: `planner n0` with the skill below (output agent) (6 rounds, 4 executor calls)

Skill on the Menu: Implement small Python functions from behavioral specifications

Purpose: Translate the task wording into the simplest function that preserves the specified semantics, especially indexing, ordering, and duplicate-handling rules, then return executable code rather than planning or verification prose.

Use when: Use when asked to write a short Python function from a natural-language requirement and one or more example assertions.

Steps:

1. Extract the exact function name, parameters, return value, and indexing convention from the prompt.
2. Identify semantic keywords before coding: positional “kth element” means access the original sequence unless sorting is explicitly requested; [...]
3. Choose the direct operation or standard algorithm matching those semantics. [...]
4. Mentally test more than the supplied example, including empty or boundary positions where applicable, duplicates for insertion behavior, unsorted input for positional retrieval, and negative values for extrema.
5. Return a complete, syntactically valid function definition with the exact requested name. Keep the final response focused on executable code.

Pitfalls to avoid: Do not infer behavior solely from a single assertion: sorting an array can accidentally pass a positional-access example while changing the required semantics. Do not confuse right insertion with lower-bound insertion; [...] Do not output only a plan, review, or statement that code is correct. [...] Avoid adding arbitrary validation or changed exception behavior unless required by the specification.

Constraint: Preserve the input’s order unless the task explicitly requests sorting or an order statistic. Match the requested function name and argument order exactly. Provide closed, valid Python code and ensure the

function returns the result rather than merely printing it. Do not include source-specific answers or rely only on the visible test case.

ROUND-BY-ROUND INTERACTION LOG

Round 1-2: Planner n0 Writes a Plan

Round 1 - Observation: Empty team; $\hat{v} = 0.605$; 15 legal actions; the menu holds the skill above and one other Python skill.

Round 1 - Action: {"kind": "ADD_AGENT", "node_id": "n0", "role_id": "planner"}

Round 1 - Input Prompt:

Role: Write a short step-by-step plan for the public task. Do not solve it.

Task: <problem above>

Perform your role using the task. Return your result.

Round 2 - Observation: n0 holds a plan without code; $\hat{v} = 0.623$ ($\Delta\hat{v} = +0.018$); 20 legal actions.

Round 2 - Action: {"kind": "RERUN_AGENT", "node_id": "n0"}

Round 1 - Executor Output: (951 characters; excerpt)

[...] 3. **Compare Pairs:** Check if the first number equals the second. [...] *Note: Depending on the specific interpretation of "equal numbers" in the context of the test case*
test_three_equal(1, 1, 1) == 3, the logic likely needs to count every matching pair found. [...]

Round 2 - Executor Output: (981 characters; excerpt)

[...] *Note: To satisfy the test case test_three_equal(1, 1, 1) == 3, the logic must count every unique pair comparison that results in equality.* 4. **Increment Logic:** If any of the pair comparisons are true, increment the count variable by 1. [...]

Round 1-2 - Feedback: [Status]: answered, no code | [Tokens]: 86 in, 243 out; 338 in, 256 out

Team State: (empty) → planner n0: plan

Grader (hidden): ✗ no code

Round 3: Bind the Skill to n0; the Code Looks Finished

Round 3 - Observation: n0 holds a plan that counts equal pairs; $\hat{v} = 0.569$ ($\Delta\hat{v} = -0.054$); 20 legal actions.

Round 3 - Action: {"kind": "BIND_SKILL", "node_id": "n0", "skill_id": "evosteer-2744c58b1d3dd27b8cb7"}

Round 3 - Input Prompt: (the role asks for a plan and the skill asks for executable code)

Role: Write a short step-by-step plan for the public task. Do not solve it.

Task: <problem above>

Procedure to follow (skill evosteer-2744c58b1d3dd27b8cb7): <skill above>

Your previous output: <the plan of Round 2>

Perform your role using the task, the procedure and your previous output. Return your result.

Round 3 - Executor Output: (177 characters; full)

```
def test_three_equal(a, b, c):
    count = 0
    if a == b:
        count += 1
    if b == c:
        count += 1
    if a == c:
        count += 1
    return count
```

Round 3 - Feedback: [Status]: answered | [Answer]: code | [Tokens]: 872 in, 65 out

Surface Check: passes the visible test (1, 1, 1) → 3; also (-1, -2, -3) → 0

Grader (hidden): ✗ wrong: (1, 2, 2) → 1, expected 2

Round 4: $\hat{v}$ Falls, and n0 Is Rerun Instead of Submitted

Round 4 - Observation: n0 holds the code above in full, with an answer present and no failure; $\hat{v} = 0.353$ ($\Delta\hat{v} = -0.216$), the largest drop of the episode; 19 legal actions, including SET_OUTPUT.

Round 4 - Action: {"kind": "RERUN_AGENT", "node_id": "n0"}

Round 4 - Input Prompt:

Role: *<planner instruction>*

Task: *<problem above>*

Procedure to follow (skill evosteer-2744c58b1d3dd27b8cb7): *<skill above>*

Your previous output: *<the code of Round 3>*

Perform your role using the task, the procedure and your previous output. Return your result.

Round 4 - Executor Output: (162 characters; full)

```
def test_three_equal(a, b, c):
    if a == b == c:
        return 3
    elif a == b or b == c or a == c:
        return 2
    else:
        return 0
```

Round 4 - Feedback: [Status]: answered | [Answer]: code | [Tokens]: 681 in, 59 out

Team State: planner n0: corrected code

Grader (hidden): ✓ correct (3 of 3 tests)

Round 5–6: Set Output and Stop

Round 5 - Observation: $\hat{v} = 0.396$ ($\Delta\hat{v} = +0.043$); 19 legal actions.

Round 5 - Action: {"kind": "SET_OUTPUT", "node_id": "n0"}

Round 6 - Observation: n0 is the output agent; $\hat{v} = 0.521$ ($\Delta\hat{v} = +0.125$); 18 legal actions.

Round 6 - Action: {"kind": "STOP"}

Final Status: [Output agent]: planner n0 | [Tests]: 3 of 3 | [Reward]: 1.0

Key Observations: At Round 4 every visible sign says the task is done: the call returned an answer without failure, the code is short and valid, and it passes the only test in the task. The code is still wrong, because it counts equal pairs and returns 1 on (1, 2, 2). $\hat{v}$, which reads execution features and the task type, not the code, falls by 0.216 to 0.353 at this state, its largest drop in the episode, and the orchestrator reruns the agent instead of setting it as the output. The rerun rewrites the function to count equal numbers and passes all three hidden tests, and $\hat{v}$ rises by 0.043 and 0.125. The rewrite follows the bound skill’s first pitfall, which warns against inferring behavior from a single assertion. Four of the other five rollouts of this problem in the same batch fail: three, including both reference rollouts, submit code that passes the visible test and fails a hidden one, and one returns a Boolean. The only other rollout that passes is the paired one that binds the same skill.