EXECUTABLE FEEDBACK IS A MEASUREMENT: BUDGETED ACQUISITION AND SELECTION-CORRECTED CREDIT FOR CODING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning for coding agents depends on executable feedback that is costly to obtain and unreliable once obtained. Branching methods reduce cost by observing counterfactual continuations only at chosen states, but the measurements they buy are then a selected sample of the eligible credit units, and the verifier at each leaf can be wrong: on a uniform audit of validation leaves, the raw verifier disagrees with a higher-fidelity reference on 12.1% of leaves with determinate reference verdicts. We introduce BANC, a training-time layer that treats executable feedback as a budgeted, randomized measurement process. BANC purchases branch and audit measurements by their expected effect on the policy gradient against shadow-priced resource costs, calibrates raw verdicts with a selection-corrected reliability model, and recovers action-level credit with an augmented inverse-propensity estimator. Because every purchase probability is randomized, bounded below, and logged, adaptive acquisition changes which measurements are bought but not the credit estimand. Tested separately, the three mechanisms cut normalized gradient error from 0.50 to 0.34, reduce harmful update mass under 20% verifier noise from 24.7% to 9.1%, and reduce the selection bias of adaptive acquisition from 0.148 to 0.021 at 44% lower RMSE than inverse propensity scoring. Under matched token and sandbox caps, BANC improves resolved rate on a temporally held-out SWE-rebench suite by 2.7 points over BPO+DIFF ($p = 0.019$) while using 7.4% less sandbox time than APPO, with gains concentrated on long, costly tasks and preserved on 16B and 32B backbones.

1 INTRODUCTION

Reinforcement learning has become the main route to stronger coding agents: a policy edits a repository, runs commands, and is rewarded by whether the hidden tests pass (Jimenez et al., 2024; Pan et al., 2024; Jain et al., 2025; Wei et al., 2025). The reward is grounded in execution, but execution is also the bottleneck. A single trajectory can take minutes of sandbox time, and outcome-only training assigns the same scalar to every action. Recent methods buy finer credit by *branching*: they restore an intermediate state, sample alternative continuations, and compare their outcomes (Kazemnejad et al., 2024; He et al., 2026; Wang et al., 2026b). Because branching multiplies execution cost, it is applied selectively at states a heuristic deems promising.

Two consequences of this practice are left unmodeled. First, selective branching makes the observed local credits a *selected sample*. A heuristic that branches where continuations are likely to diverge observes high-credit-variance units more often than low-variance units; treating the observed credits as uniform absorbs that preference into the gradient. Second, the verdict at every leaf is produced by a verifier that can be wrong. On a uniform audit of validation leaves against a higher-fidelity reference protocol, the raw verifier disagrees on 12.1% of leaves with determinate reference verdicts because of flaky tests, timeouts, and incomplete assertions (Sec. 4.4). Rerunning the same tests detects transient failures but not the other causes, and stronger checks cost additional execution. Existing branching methods take the branch set as given and the verdict as truth.

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

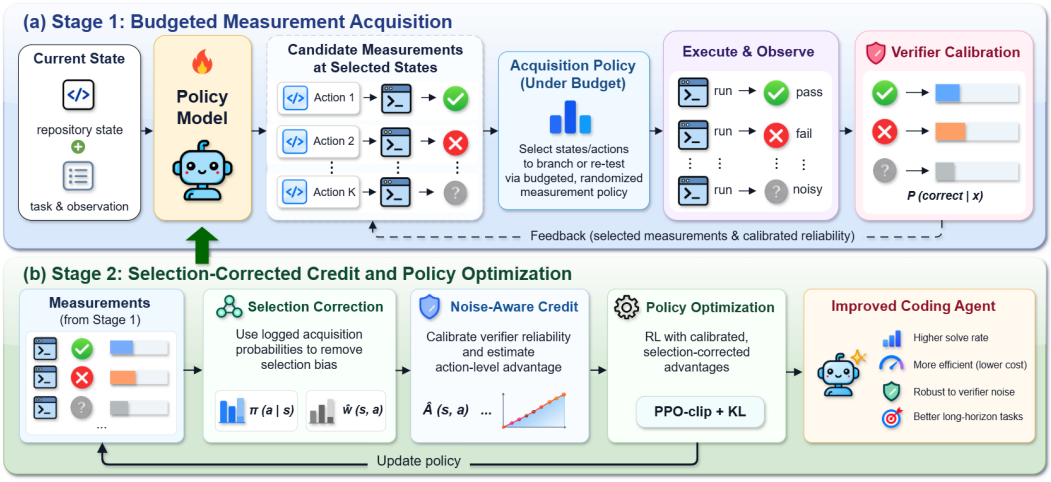

Figure 1: Overview of the BANC training-time measurement and credit pipeline.

We argue that both problems have one source and one fix: executable feedback in coding-agent RL is a *measurement*, purchased under a budget, returned with noise, and purchased more often where it looks informative. This raises three questions: which measurements are worth buying, how much each returned verdict should be trusted, and how selectively observed measurements can be turned into credit without changing what is being estimated. We introduce BANC (Budgeted Adaptive Noise-Calibrated Credit), a training-time layer that answers all three. BANC scores candidate BRANCH and AUDIT purchases by their expected effect on the policy gradient against shadow-priced resource costs, samples purchases from randomized probabilities bounded below and logged exactly, calibrates raw verdicts with a reliability model corrected for adaptive audit selection, and forms action-level credit with an augmented inverse-propensity estimator that targets the same protocol-conditioned estimand as full-feedback training under adaptive acquisition. Figure 1 summarizes the pipeline.

The parts are individually familiar; the claim is that they are jointly necessary. Our factorial ablation makes this concrete: gradient-value acquisition without correction is worse than uniform acquisition, whereas the same acquisition with calibration and propensity correction is the best configuration on every metric (Sec. 4.6).

Our contributions are:

- **A measurement view of executable feedback.** We model branches and audits as randomized, logged, budgeted purchases, allowing aggressive acquisition to coexist with an unchanged credit estimand (Sec. 3).
- **Gradient-value acquisition.** An acquisition score that targets reduction in policy-gradient error rather than divergence among continuations, controlled by a multi-resource dual controller. On an exhaustive diagnostic set it ranks candidates at Spearman 0.54 versus 0.38 for the strongest prior score at 43% lower cost per unit of error reduction (Sec. 4.3).
- **Selection-corrected reliability calibration.** A calibrator fit under inverse audit propensities. Under 20% injected verifier noise it cuts harmful update mass from 24.7% to 9.1% and retains 93% of clean performance (Sec. 4.4).
- **Selection-corrected credit.** An AIPW credit estimator with a conditional unbiasedness guarantee under stated overlap, randomization, and calibration assumptions; it reduces selection bias from 0.148 to 0.021 at 44% lower RMSE than inverse propensity scoring (Sec. 4.5).

Under matched token and sandbox caps on a temporally held-out SWE-rebench suite, BANC improves resolved rate by 2.7 points over BPO+DIFF ($p = 0.019$) while using 7.4% less sandbox time than APPO; the gain is concentrated on long, costly tasks and carries over to 16B and 32B backbones. The method offers little on short tasks with fast, reliable feedback and cannot repair verifier errors shared by its audit protocol.

2 Related Work

RL for coding agents and action-level credit. Executable environments built from real repositories (Jimenez et al., 2024; Pan et al., 2024; Jain et al., 2025; Badertdinov et al., 2025; Zan et al., 2025) made verifiable rewards available for long-horizon agents, and outcome-only optimizers such as PPO, RLOO, and GRPO (Schulman et al., 2017; Ahmadian et al., 2024; Shao et al., 2024) remain strong on them (Wei et al., 2025; Pu et al., 2026). Finer credit has been pursued along two lines. Rollout-based methods estimate the value of an intermediate state by sampling continuations from it (Wang et al., 2024; Kazemnejad et al., 2024), and the sandbox-native variants BPO and APPO choose branch points inside agent trajectories (He et al., 2026; Wang et al., 2026b). Code-structured methods instead change the unit of credit to a diff hunk (Wang et al., 2026c) or make group-relative advantage estimation robust to noisy rollout rewards and outliers(Zhang et al., 2026). In all of these, the set of branched states is chosen by a heuristic and then treated as a uniform sample, and the leaf verdict is used as ground truth. BANC keeps their credit units and branch mechanics but replaces the heuristic with an acquisition rule tied to the policy gradient and corrects for the selection the rule induces.

Imperfect verifiers. Test flakiness is well documented in software engineering (Luo et al., 2014), and it survives into agent benchmarks: SWE-bench Verified was constructed specifically to remove tasks with unreliable tests (OpenAI, 2024). In RL, verifier error is one form of reward misspecification and enables reward hacking (Skalse et al., 2022; Gao et al., 2023); verification quality can also degrade as policy capability grows, motivating reward mechanisms that co-evolve with the generator (Wang et al., 2026a). Our reference protocol draws on property-based testing (MacIver & Hatfield-Dodds, 2019) and mutation analysis (Jia & Harman, 2011) to produce a higher-fidelity verdict on demand, and we treat the raw verifier's agreement with that verdict as a quantity to be estimated under the same selection correction as everything else.

Active measurement and selection correction. Active learning selects which examples to label (Settles, 2009; Houlsby et al., 2011; Ash et al., 2020), gradient-norm importance sampling selects which labeled examples to train on (Katharopoulos & Fleuret, 2018), and bandits with knapsacks select under several resource constraints (Badanidiyuru et al., 2013; Agrawal et al., 2016). Active testing shows that adaptive selection biases the resulting estimate unless the acquisition probabilities are accounted for (Kossen et al., 2021; Farquhar et al., 2021), and the same issue arises when reward observation is costly (Krueger et al., 2020). The standard remedy is inverse propensity weighting (Horvitz & Thompson, 1952) and its augmented, doubly robust form (Robins et al., 1994; Dudík et al., 2011; Jiang & Li, 2016; Thomas & Brunskill, 2016), with cross-fitting to reuse data for the outcome model (Chernozhukov et al., 2018). Those estimators correct for which *actions* a logging policy took; BANC applies the same construction to which *measurements* were purchased, with the logged purchase probability as propensity, a lagged credit predictor as outcome model, and purchases randomized rather than top-k so that the correction applies. The acquired measurements then train a sequential policy, so the acquisition score is the expected reduction in policy-gradient error rather than an evaluation metric.

3 Method

Overview. BANC operates on a trajectory after it has been produced and before the policy is updated. It (i) parses the trajectory into environment-affecting credit units (Sec. 3.1); (ii) calibrates each leaf verdict against a training-only reference protocol (Sec. 3.2); (iii) purchases Branch and Audit measurements from randomized, logged probabilities under a multi-resource budget (Sec. 3.3); and (iv) forms selection-corrected local credit that enters a standard clipped policy objective (Secs. 3.4–3.5). The single fact that makes the pieces fit together is stated first.

Why adaptive acquisition needs correction. Let $S_i^{\mathrm{B}} \in \{0, 1\}$ indicate whether the reference local credit Δ_i^{ref} of unit i is measured, and let $\mathcal{F}_i$ be the history available before the purchase. Because the purchase probability depends on $\mathcal{F}_i$, units with different credit distributions are observed at different rates, so in general $\mathbb{E}[\Delta_i^{\mathrm{ref}} \mid S_i^{\mathrm{B}} = 1] \neq \mathbb{E}[\Delta_i^{\mathrm{ref}}]$. BANC randomizes the purchase given $\mathcal{F}_i$, which yields $S_i^{\mathrm{B}} \perp \Delta_i^{\mathrm{ref}} \mid \mathcal{F}_i$ under the logged acquisition policy. Acquisition therefore decides

what is informative to buy, and propensity correction prevents that preference from redefining the credit being estimated.

3.1 Problem Setup and Credit Units

A task is an executable environment $e = (\mathcal{R}, \xi, \mathcal{H}, \mathcal{V})$ with repository snapshot $\mathcal{R}$, issue ξ, agent harness $\mathcal{H}$, and training-time verifier $\mathcal{V}$. A policy π_θ produces $\tau = (s_0, u_0, o_0, \ldots, s_T, u_T, o_T, y_0)$, where s_t is the harness-visible state, u_t a model-generated action span, o_t the environment response, and $y_0 \in \{0, 1\}$ the raw terminal verdict. Benchmark-hidden evaluation outcomes are never used in training.

A deterministic parser Γ, applied before any additional execution, maps τ to eligible credit units $\mathcal{I}(\tau) = \{i = (s_i, u_i, d_i, v_i)\}_{i=1}^{N_\tau}$, where d_i is the repository delta induced by u_i and v_i holds pre-acquisition metadata only (action type, changed files, test history, timeout signatures, sandbox status); we write x_i for the unit features. Default units are environment-affecting edit, test, recovery, and submit actions; token spans without an observable action receive no local credit. Unit construction is held fixed across all compared acquisition policies.

Two-stage measurement. At the unit level, $S_i^{\mathrm{B}} \in \{0, 1\}$ decides whether to purchase $\textsc{Branch}(K)$: restore the snapshot before u_i, sample K sibling actions from $\pi_{\theta_{\mathrm{old}}}$, and roll them out to termination. Sibling continuations are used only for credit estimation; their tokens count against the token budget but are not added to the policy-update batch. After raw verdicts are available, every observed terminal leaf ℓ becomes an audit candidate, and $S_\ell^{\mathrm{A}} \in \{0, 1\}$ decides whether to purchase $\textsc{Audit}(r)$, which reruns or strengthens the verifier with recipe r (an independently recreated environment, a disjoint training-only test suite, or a mutation-based diagnostic). Repeating the same test process detects transient failures but is not treated as an independent oracle. Because audits follow branches, a branch outcome can itself be audited. Each candidate c has a predicted resource vector $\widehat{\mathbf{c}}_c \in \mathbb{R}_+^D$ and a realized vector $\mathbf{c}_c$; the two constrained coordinates in our experiments are generated policy tokens and sandbox seconds, with GPU time, test time, restoration time, and retries logged separately. The controller constrains expected acquisition resources and leaves the deployment-time reward unchanged.

3.2 Reliability as an Audited Posterior

The raw verdict y_ℓ of leaf ℓ is a noisy observation of a reference verdict $z_\ell^{\mathrm{ref}} \in \{0, 1\}$ produced by a protocol fixed before training (Sec. 4.1): clean environment recreation, a disjoint training-only test suite, and a deterministic adjudication rule. The reference is a higher-fidelity operational target, not an assumption of agreement with human intent. An audit reveals $r_\ell = \mathbb{I}[y_\ell = z_\ell^{\mathrm{ref}}]$ for the audited leaf only. When the protocol cannot establish a verdict, the audit returns $\textsc{Abstain}$: the leaf is excluded from the calibration loss, its cost stays in the resource ledger, and for credit purposes it is treated as unaudited, i.e. it receives the calibrated utility of Eq. equation 2 rather than a reference verdict.

We fit a calibrator $q_\eta(y_\ell, w_\ell) = \Pr(r_\ell = 1 \mid y_\ell, w_\ell)$, where w_ℓ contains unit-level context (task ID, action type) and post-verifier execution features (runtime, exit codes, crash dumps). Because suspicious leaves are audited more often, unweighted calibration is itself selection biased. With logged audit propensity p_ℓ^{A}, the calibrator minimizes

$$\mathcal{L}_{\mathrm{cal}}(\eta) = -\sum_\ell \frac{S_\ell^{\mathrm{A}}}{p_\ell^{\mathrm{A}}} \Big[r_\ell \log q_\eta(y_\ell, w_\ell) + (1 - r_\ell) \log \big(1 - q_\eta(y_\ell, w_\ell)\big) \Big]. \tag{1}$$

What the calibrator estimates. Audit candidates are leaves that exist: backbone leaves, always observed, and sibling leaves, which exist only when unit i was branched. The calibrator therefore targets reliability on the population of observed leaves, $\Pr(r_\ell = 1 \mid y_\ell, w_\ell,\ \ell \text{ observed})$, and $1/p_\ell^{\mathrm{A}}$ corrects only for which observed leaves were audited. This is exactly the population on which q_- is applied, since every leaf entering Eq. equation 6 is observed. Branch-stage selection is corrected separately by $1/p_i^{\mathrm{B}}$ in Eqs. equation 7–equation 8; the two corrections compose because branch selection depends only on pre-acquisition history and audit selection only on post-verdict (y_ℓ, w_ℓ). A marginal reliability over all potential leaves would require the joint probability $p_i^{\mathrm{B}} p_\ell^{\mathrm{A}}$; BANC does not need that quantity.

Policy-update targets use a lagged or cross-fitted calibrator q_-. The calibrated leaf utility is

$$\bar{z}_\ell = \begin{cases} z_\ell^{\mathrm{ref}}, & S_\ell^{\mathrm{A}} = 1 \text{ and the audit is determinate,} \\ q_-(y_\ell, w_\ell)\, y_\ell + \big(1 - q_-(y_\ell, w_\ell)\big)(1 - y_\ell), & \text{otherwise,} \end{cases} \tag{2}$$

and if q_- is calibrated relative to the reference protocol, both cases have conditional mean $\mathbb{E}[z_\ell^{\mathrm{ref}} \mid y_\ell, w_\ell]$.

3.3 Gradient-Value Acquisition under a Budget

Prior branching criteria ask where continuations are likely to diverge; GVA asks where an additional measurement is expected to change the policy update. For token j in unit i, let $g_{ij} = \nabla_{\theta_{\mathrm{old}}} \log \pi_{\theta_{\mathrm{old}}}(u_{ij} \mid h_{ij})$ and define the gradient-magnitude proxy $G_i = \frac{1}{|u_i|} \sum_{j \in u_i} \|g_{ij}\|_2^2$, estimated with a stop-gradient diagonal-Fisher approximation so that no per-candidate gradient is stored.

Utility. Under a local linearization of the update, replacing the unobserved credit of unit i by its conditional mean costs $\|\bar{g}_i\|_2^2 \operatorname{Var}(\Delta_i^{\mathrm{ref}} \mid \mathcal{F}_i)$ in squared gradient error, where $\bar{g}_i$ is the aggregated score (Appendix B.2); an exact measurement would remove it. GVA replaces the two factors by G_i and by $\widehat{\sigma}_i^2$, the cross-fitted residual variance of the credit model $m_\phi(x_i)$ of Sec. 3.4, and attenuates the product by the reliability the verdict is predicted to have:

$$U_i^{\mathrm{B}} = G_i\, \widehat{\sigma}_i^2\, \widehat{q}_i^{\mathrm{pre}}, \qquad \widehat{q}_i^{\mathrm{pre}} = \mathbb{E}_{(y,w) \sim \widehat{P}(\cdot \mid x_i)}\big[q_-(y, w)\big], \tag{3}$$

where $\widehat{P}(\cdot \mid x_i)$ predicts verdict and features before the branch runs, so no post-purchase feature enters the decision. This is a first-order surrogate; its ranking and factorization are tested directly in Sec. 4.3. For audits, let Q_- be the lagged cross-fitted ensemble of calibrators; the audit utility is

$$U_\ell^{\mathrm{A}} = G_\ell\, \widehat{\operatorname{Var}}_{q \sim Q_-}\big[q(y_\ell, w_\ell)\, y_\ell + (1 - q(y_\ell, w_\ell))(1 - y_\ell)\big], \tag{4}$$

where G_ℓ sums G_i over units whose credit would change if leaf ℓ's verdict were revised.

Randomized purchase and budget control. For candidate c with utility U_c, BANC samples a purchase from

$$p_c = p_{\min} + (p_{\max} - p_{\min})\, \sigma\!\left(\frac{U_c - \boldsymbol{\lambda}^\top \widehat{\mathbf{c}}_c}{\tau_{\mathrm{acq}}}\right), \qquad \boldsymbol{\lambda} \leftarrow \Big[\boldsymbol{\lambda} + \eta_\lambda\big(\textstyle\sum_c S_c \mathbf{c}_c - \mathbf{B}_{\mathrm{batch}}\big)\Big]_+, \tag{5}$$

where $p_{\min} > 0$ provides overlap and bounds inverse-propensity weights by $1/p_{\min}$, and the nonnegative duals $\boldsymbol{\lambda}$ are updated once per batch to enforce the caps in expectation. In the first iteration, before q_- and m_- exist, purchases are sampled uniformly at the budget-feasible rate. Batches are never truncated after outcomes are observed, since truncation would invalidate the logged probabilities. We log the complete candidate set, p_c, predicted and realized costs, purchase type, and outcome.

3.4 Selection-Corrected Local Credit

When unit i is branched, with $\ell = 0$ the original leaf and $\ell = 1, \ldots, K$ the siblings, the reliability-calibrated local credit is

$$\widetilde{\Delta}_i = \bar{z}_{i0} - \frac{1}{K} \sum_{k=1}^{K} \bar{z}_{ik}, \tag{6}$$

which compares the observed action with counterfactual siblings from the same restored state rather than including it in its own baseline. With logged branch propensity $p_i^{\mathrm{B}} \geq p_{\min}$, the credit model is trained by

$$\mathcal{L}_{\mathrm{credit}}(\phi) = \sum_i \frac{S_i^{\mathrm{B}}}{p_i^{\mathrm{B}}} \big(\widetilde{\Delta}_i - m_\phi(x_i)\big)^2, \tag{7}$$

and the policy target uses a lagged or cross-fitted predictor m_{-i}:

$$\widehat{\Delta}_i^{\mathrm{AC}} = m_{-i}(x_i) + \frac{S_i^{\mathrm{B}}}{p_i^{\mathrm{B}}} \big(\widetilde{\Delta}_i - m_{-i}(x_i)\big). \tag{8}$$

Eq. equation 8 is an augmented inverse-propensity pseudo-outcome in the sense of doubly robust off-policy estimation (Jiang & Li, 2016): unbranched units receive model-based credit, branched units add a residual correction weighted by the exact logged propensity.

Estimator property. Let $D_i = 1$ if the reference protocol returns a determinate verdict for the original leaf and all K siblings of unit i, and define the reference credit on that event, $\Delta_i^{\mathrm{ref}} = z_{i0}^{\mathrm{ref}} - \frac{1}{K}\sum_{k=1}^{K} z_{ik}^{\mathrm{ref}}$. Assume (i) eligibility and acquisition features are fixed before measurement; (ii) conditional on $\mathcal{F}_i$, branch selection is randomized with $p_i^{\mathrm{B}} = \Pr(S_i^{\mathrm{B}} = 1 \mid \mathcal{F}_i) \geq p_{\min} > 0$; (iii) $\mathbb{E}[\bar{z}_\ell \mid y_\ell, w_\ell] = \mathbb{E}[z_\ell^{\mathrm{ref}} \mid y_\ell, w_\ell, \text{determinate}]$; and (iv) abstention is independent of r_ℓ given (y_ℓ, w_ℓ), so that an abstaining audit reverts to Eq. equation 2 without changing the conditional mean. Then

$$\mathbb{E}\big[\widehat{\Delta}_i^{\mathrm{AC}} \mid \mathcal{F}_i,\, D_i = 1\big] = \mathbb{E}\big[\Delta_i^{\mathrm{ref}} \mid \mathcal{F}_i,\, D_i = 1\big]. \tag{9}$$

The result does not require a correctly specified credit model, but it does require exact logged propensities and a lagged or cross-fitted m_{-i}; if q_- is within ϵ_q of the protocol reliability, the bias of $\widetilde{\Delta}_i$ is at most $2\epsilon_q$ (proof and bound in Appendix B.4). The floor bounds the residual weight by $1/p_{\min}$, so no clipping is needed; candidates with zero purchase probability would fall outside the estimand. Because G_i is larger for low-probability actions, acquisition tilts toward actions the policy rarely takes; the floor and the correction keep this an efficiency question rather than a bias. The statement is relative to the protocol and to $\pi_{\theta_{\mathrm{old}}}$, not to human intent or to the causal effect of a hidden reasoning token.

3.5 Policy Update

The calibrated terminal advantage is $A_\tau^{\mathrm{term}} = \bar{z}_\tau - b_e$, with task-group baseline b_e. For an eligible unit i,

$$A_i^{\mathrm{BANC}} = \mathrm{Norm}_e\big(A_\tau^{\mathrm{term}} + \alpha_{\mathrm{loc}}\widehat{\Delta}_i^{\mathrm{AC}}\big), \tag{10}$$

projected onto the model-generated tokens of u_i; tool outputs and observations are masked. The policy is updated with the standard PPO-clip objective plus a KL penalty to π_{ref} (ϵ and β in Appendix A.2). Appendix B.3 lists one full iteration.

4 Experiments

We test the claim that executable feedback in coding-agent RL should be treated as an adaptively acquired, noisy, and costly measurement. Each mechanism is tested before end-to-end evaluation: **(RQ1)** Does GVA buy more useful measurements against shadow-priced resource costs and improve the frontier? **(RQ2)** Does selection-corrected calibration reduce harmful update mass under noise? **(RQ3)** Does AIPW credit recover the full-feedback reference under adaptive observation? **(RQ4)** Do these mechanisms translate into issue resolution, and how do they interact?

4.1 Experimental Protocol

Training environments and data audit. We train on tasks from SWE-Gym, R2E-Gym, and the SWE-rebench training partition (Pan et al., 2024; Jain et al., 2025; Badertdinov et al., 2025). Tasks are retained only if base commits build reproducibly, fail-to-pass tests fail before and pass after the reference patch, and verifiers finish within timeouts. We freeze a manifest of repository URLs, base commits, issue IDs, container digests, and commands, removing all overlap with evaluation suites via normalized text and AST hunk hashes. The final training corpus contains $N = 2{,}184$ tasks.

Audit-reference protocol and deterministic adjudication. The reference protocol operates only on training-split repositories without accessing benchmark-hidden tests or external LLMs. Synthesized properties are type, invariant, and crash-freedom checks. Synthesis and filtering happen once during manifest construction before any policy trajectory exists: for each task, the pipeline generates properties from the pre-commit interface using a frozen Hypothesis 6.108 configuration, executes them on the reference patch, and drops any property that fails or is flaky there. The surviving suite is frozen in the manifest. A leaf is PASS if the clean disjoint suite passes with invariants retained and mutation score ≥ 0.80, FAIL if any test fails with invariants retained, and ABSTAIN on timeouts, crashes, or mutation score < 0.80 (full rule in Appendix Table 9).

Evaluation suites. The primary confirmatory suite is a temporally held-out SWE-rebench snapshot ($N = 640$, cutoff March 2025) (Badertdinov et al., 2025). We also report SWE-bench Verified ($N = 500$) for historical comparability (OpenAI, 2024), and a fixed 400-task subset of Multi-SWE-bench (180 Java, 140 TypeScript, 80 Go) for language transfer (Zan et al., 2025). The primary metric is task-level resolved rate ($\mathrm{resolve@1}$). Each seed-specific checkpoint is evaluated once on every task with the official harness under identical limits of 8,192 generated tokens, 30 tool calls, and wall-clock time; reported means average over five training seeds; confidence intervals use the paired two-way bootstrap over seeds and tasks described in Appendix A.5.

Models, budgets, and overhead. The primary model is Qwen2.5-Coder-7B-Instruct (LoRA rank 64, $\alpha = 128$). Every method trains under two simultaneous caps: $B_{\mathrm{tok}} = 8.0 \times 10^7$ generated policy tokens and $B_{\mathrm{env}} = 1.2 \times 10^6$ sandbox seconds (including audits and restoration), stopping when either is exhausted. Runs execute on a single node with $8\times$ NVIDIA H100 SXM5 80GB GPUs. BANC's online scoring occupies 7.3% of node GPU time (APPO: 3.3%) and its run is 4.2 wall-clock hours longer (Appendix Table 10); no human or LLM-based audit supervision is used. Acquisition hyperparameters are listed in Appendix A.2.

Mechanism metrics. *Normalized GradErr* is $\|\widehat{g} - g^*\|_2 / \|g^*\|_2$, where g^* is the validation gradient computed with full branching and audited verdicts. *Harmful update mass* pools every eligible unit-level gradient contribution $\widehat{g}_i$ from 250 validation batches ($N_{\mathrm{contrib}} = 12{,}840$) and reports the percentage with negative cosine similarity to g^*. Normalized frontier area is computed via the trapezoidal rule:

$$\mathrm{nAUC} = \frac{1}{0.75} \int_{0.25}^{1} \mathrm{resolve@1}(b)\, db. \tag{11}$$

4.2 Baselines

We compare against SFT, RLOO, GRPO, BPO (He et al., 2026), APPO (Wang et al., 2026b), and DiDPO (Wang et al., 2026c). BPO+Diff is a compositional control: BPO's branching applied to BANC's diff-anchored units, without calibration, GVA, or AIPW. It isolates whether gains originate from unit granularity or from the adaptive measurement framework.

4.3 RQ1: Acquisition Quality and the Performance–Cost Frontier

Frontier. Appendix Table 11 reports resolved rate at four budget fractions. At 25% budget, BANC leads APPO by 0.6 points (17.4 vs. 16.8), reflecting the initial samples required for $\boldsymbol{\lambda}$ and q_η to stabilize. The margin widens at full budget to 3.6 over APPO and 2.7 over BPO+Diff, achieving the highest nAUC (24.4 vs. 22.3 and 21.7).

Exhaustive-reference diagnostic. On 512 held-out units with $K = 32$ continuations and audited verdicts for every leaf, the true reduction in squared policy-gradient error from measuring each unit is known. Table 1 compares acquisition rules against it. GVA ranks units at Spearman $\rho = 0.54$ versus 0.38 for the APPO score, reaches 63.8% top-10% precision, and lowers normalized gradient error to 0.34 (APPO score: 0.50); per unit of error reduction it spends 12.4 CPU-hours, 43% less than the APPO score (21.6). Variance-only and gradient-only rows test the factorization: each factor alone is worse than the APPO score, and only their product beats it. Across the 5 training runs, realized batch cost stays within 2.8% of the cap, propensities respect $p_{\min} = 0.025$, residual weights never exceed 40, and the effective sample size of the corrected credit is $0.74\, N_{\mathrm{elig}}$.

4.4 RQ2: Reliability under Imperfect Verifiers

Disagreement rate. On 500 validation leaves audited offline, the raw verifier and reference protocol are non-agreeing or unresolved on $71/500$ leaves (14.2%; 95% CI $[11.3\%, 17.5\%]$). Among the 488 leaves with determinate verdicts, 59 (12.1%) are confirmed disagreements: flaky execution 27, timeouts 19, and incomplete assertions 13. The remaining $12/500$ leaves (2.4%) are abstentions—eight crashes and four indeterminate mutation scores—and are not counted as verifier errors. We evaluate natural disagreement, a stress test with 20% state-dependent flaky corruption applied only to the training verifier (generator in Appendix A.7), and the clean unmodified evaluation protocol.

Table 1: Acquisition quality evaluated on the held-out exhaustive diagnostic set ($N = 512$ units, $K = 32$ leaves). Standard errors over 10 independent replay subsamples are reported in parentheses.

Rule	Spearman $\rho \uparrow$	Top-10% Precision (%) $\uparrow$	Normalized GradErr $\downarrow$	Cost / ΔErr (CPU-h) $\downarrow$
Uniform	0.05 (0.02)	19.2 (1.4)	0.79 (0.03)	39.8 (2.1)
Entropy	0.26 (0.03)	38.4 (1.8)	0.61 (0.02)	27.2 (1.4)
APPO-Score	0.38 (0.03)	49.1 (1.9)	0.50 (0.02)	21.6 (1.1)
Variance-only	0.32 (0.02)	43.7 (1.7)	0.56 (0.02)	24.9 (1.3)
Gradient-only	0.29 (0.03)	41.0 (1.6)	0.58 (0.03)	25.7 (1.2)
GVA (ours)	**0.54** (0.02)	**63.8** (1.6)	**0.34** (0.01)	**12.4** (0.7)

Calibration metrics are scored on a second, disjoint uniform sample of 1,024 determinate audited validation leaves that no calibrator was trained on (Appendix A.6).

Appendix Table 12 shows two effects. Because suspicious leaves are audited preferentially, the unweighted calibrator is biased (Brier 0.124, ECE 0.098); weighting by logged audit propensities (Eq. 1) lowers these to 0.086 and 0.061, most of the way to the full-audit reference (0.065, 0.042). Majority retest is below the raw verifier on the clean run because each retest triples verifier execution per leaf, so under the matched sandbox cap it trains on fewer trajectories. Under 20% injected noise, the raw-verifier policy puts 24.7% of gradient mass in the harmful direction and loses 7.0 points relative to its own clean run (26.2 $\rightarrow$ 19.2); majority retesting recovers only to 21.0. The IPW calibrator lowers harmful mass to 9.1% and loses 1.9 points (28.3 $\rightarrow$ 26.4), retaining 93% of its clean performance.

4.5 RQ3: Credit Recovery under Adaptive Observation

Appendix Table 13 evaluates credit recovery on 512 units under four adaptive masking policies. The exhaustive set contains only units with determinate verdicts at every leaf, matching the estimand of Eq. equation 9. Treating selected purchases as uniform introduces 0.148 selection bias. Standard IPS substantially reduces bias (0.018) but incurs heavy variance (RMSE 0.389). AIPW preserves near-zero bias (0.021) while cutting RMSE by 44% to 0.218, matching nominal 95% coverage (94.6%) and elevating gradient cosine similarity to 0.84.

4.6 RQ4: End-to-End Quality, Ablations, and Transfer

Main comparison. Table 2 presents matched-cap resolution. On SWE-rebench, BANC reaches 28.3% (181.2/640 on average; per-seed counts: 184, 178, 185, 177, 182; baseline counts in Appendix Table 15), leading BPO+Diff (25.6%) by 2.7 points; the prespecified paired sign-flip test rejects the null at $p = 0.019$. The BANC–BPO+Diff comparison on SWE-rebench is the sole prespecified confirmatory contrast; all other comparisons are secondary. The secondary contrast against APPO yields 3.6 points ($p = 0.003$). On SWE-bench Verified, BANC solves 166.0/500 tasks (33.2%; per-seed: 168, 163, 169, 164, 166) versus 149.0/500 (29.8%) for APPO. BANC uses identical token budgets and 7.4% less sandbox time than APPO (1.108 vs. 1.196×10^6 s) under a common ledger charging rollouts, audits, and restoration to all methods (Appendix Table 8).

Factorial ablation. Table 3 reports the 2^3 factorial on Seeds 1, 2, and 5 of the main runs. GVA without correction $(1, 0, 0)$ yields no gain (24.6% vs. 25.0% for uniform acquisition) and raises harmful update mass to 18.9%, consistent with adaptive sampling shifting the evidence distribution that an uncorrected learner then absorbs. Adding AIPW $(1, 0, 1)$ recovers 1.6 points, and the full configuration $(1, 1, 1)$ is best on all three metrics, with GradErr 0.34 against 0.62 for the uniform baseline; the components are complementary rather than additive.

Where the gain comes from. Stratifying evaluation tasks by two quantities fixed before training, the reference-suite execution time and the SFT checkpoint's median turn count, locates the gain. On short, cheap tasks (< 5 turns, < 10 s), BANC is level with APPO (37.8% vs. 38.2%, overlapping CIs): feedback there is fast and reliable and there is nothing to buy. On long, costly tasks (> 15

Table 2: End-to-end issue resolution under matched caps: mean resolved rate [95% CI] over 5 seeds. Tokens and environment time are terminal usage; environment time for BANC includes audits and container restoration.

Method	SWE-bench Verified	SWE-rebench	Multi-SWE (400)	Tokens (10^7)	Env. time (10^6 s)
SFT Checkpoint	18.2 [16.6, 19.8]	14.4 [12.9, 15.9]	11.8 [10.2, 13.3]	–	–
RLOO	21.8 [20.1, 23.4]	17.5 [15.9, 19.1]	14.2 [12.5, 15.8]	8.0	0.94
GRPO	23.0 [21.3, 24.7]	18.8 [17.1, 20.4]	15.0 [13.3, 16.7]	8.0	0.96
BPO	29.2 [27.4, 30.9]	24.1 [22.4, 25.7]	19.5 [17.7, 21.3]	8.0	1.18
APPO	29.8 [28.0, 31.6]	24.7 [23.0, 26.3]	20.2 [18.4, 22.1]	8.0	1.20
DiDPO	29.0 [27.2, 30.7]	24.0 [22.3, 25.6]	19.2 [17.4, 21.0]	8.0	1.15
BPO+DIFF	30.4 [28.6, 32.2]	25.6 [23.9, 27.2]	21.0 [19.2, 22.8]	8.0	1.19
BANC	**33.2** [31.4, 35.0]	**28.3** [26.6, 30.0]	**23.2** [21.3, 25.1]	8.0	**1.11**

Table 3: 2^3 factorial ablation on SWE-rebench across Seeds 1, 2, and 5. Harmful update mass and GradErr use the clean training verifier. CIs reflect the two-way paired bootstrap over the 3-seed subset.

GVA	Cal	AC	SWE-rebench resolve@1 (%) ↑	Harmful update mass (%) ↓	Normalized GradErr ↓
0	0	0	25.0 [23.0, 27.0]	16.4 [14.6, 18.3]	0.62 (0.02)
0	0	1	25.4 [23.3, 27.5]	15.8 [14.0, 17.6]	0.58 (0.02)
0	1	0	26.1 [24.1, 28.1]	11.2 [9.5, 13.0]	0.52 (0.02)
0	1	1	26.8 [24.8, 28.8]	9.8 [8.1, 11.5]	0.47 (0.01)
1	0	0	24.6 [22.6, 26.6]	18.9 [16.9, 20.8]	0.65 (0.03)
1	0	1	26.2 [24.2, 28.2]	13.5 [11.7, 15.3]	0.49 (0.02)
1	1	0	26.9 [24.9, 28.9]	8.6 [7.0, 10.3]	0.41 (0.01)
1	**1**	**1**	**28.3** [26.3, 30.3]	**7.3** [5.7, 9.0]	**0.34** (0.01)

turns, > 120 s) it leads by 6.8 points (25.4% vs. 18.6%). On Multi-SWE-bench it leads BPO+DIFF on Java (23.4 vs. 20.6) and TypeScript (24.7 vs. 21.6) but trails slightly on Go (20.2 vs. 20.7, within overlapping CIs). The training corpus is Python-only, so all cross-language performance is zero-shot transfer of the trained policy; we observe no mechanism-level explanation for the small Go gap and report it as an unexplained negative result.

On two larger backbones with 3 seeds, BANC reaches 34.1% [32.4, 35.8] on SWE-rebench with DeepSeek-Coder-V2-Lite versus 31.0 [29.3, 32.7] for BPO+DIFF, and 40.8 [38.9, 42.6] with Qwen2.5-Coder-32B versus 37.5 [35.7, 39.3] (Appendix Table 14). The margin over the strongest baseline (3.1 and 3.3 points) is at least as large as at 7B; we read this as a transfer check, not a scaling law.

5 CONCLUSION

Executable feedback in coding-agent RL is a measurement: costly to obtain, noisy once observed, and selectively acquired where it appears informative. BANC makes the resulting three choices explicit. Acquisition is scored by its expected effect on the policy gradient against resource costs; verifier reliability is estimated with selection correction; and local credit is formed by an augmented inverse-propensity estimator that preserves the target under adaptive acquisition. Experiments show that the three mechanisms are complementary: adaptive acquisition improves measurement efficiency, while calibration and propensity correction prevent the resulting selection from biasing learning. The gains are largest on long, costly tasks and persist across larger backbones. The same view also defines the limits: when feedback is cheap and reliable there is little to gain from selective measurement, and when the audit protocol shares the verifier's blind spots, calibration cannot recover the missing information.

Reproducibility Statement

We provide the dataset construction and leakage audit, optimization and acquisition hyperparameters, baseline tuning protocol, statistical procedures, injected-noise generator, and environment-time accounting in the appendix. The appendix further specifies the masking policies, auxiliary models, cross-fitted training loop, estimator analysis, additional results, and per-seed solved counts. The task manifest, property suites, acquisition logs, and code required to reproduce the experiments will be released with the implementation.

Ethics Statement

This work uses software-engineering benchmark tasks and does not involve human-subject experiments or the collection of private or sensitive personal data. Training-time reference tests are constructed only from the designated training repositories, and benchmark-hidden evaluation outcomes are not used during training. Dataset licenses are retained in the task manifest.

AI-Use Statement

Generative AI tools were used for language polishing, manuscript editing, and LaTeX formatting. The authors reviewed all AI-assisted edits and take full responsibility for the methodology, experiments, analysis, results, and final content of the paper.

References

Shipra Agrawal, Nikhil R. Devanur, and Lihong Li. An efficient algorithm for contextual bandits with knapsacks, and an extension to concave objectives. In *Conference on Learning Theory*, 2016.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, and Ahmet Üstün. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. *arXiv preprint arXiv:2402.14740*, 2024.

Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. In *International Conference on Learning Representations*, 2020.

Ashwinkumar Badanidiyuru, Robert Kleinberg, and Aleksandrs Slivkins. Bandits with knapsacks. In *IEEE Symposium on Foundations of Computer Science*, 2013.

Ibragim Badertdinov, Alexander Golubev, Maksim Nekrashevich, et al. SWE-rebench: An automated pipeline for task collection and decontaminated evaluation of software engineering agents. *arXiv preprint arXiv:2505.20411*, 2025.

Victor Chernozhukov, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins. Double/debiased machine learning for treatment and structural parameters. *The Econometrics Journal*, 21(1):C1–C68, 2018.

Miroslav Dudík, John Langford, and Lihong Li. Doubly robust policy evaluation and learning. In *International Conference on Machine Learning*, 2011.

Sebastian Farquhar, Yarin Gal, and Tom Rainforth. On statistical bias in active learning: How and when to fix it. In *International Conference on Learning Representations*, 2021.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, 2023.

Bowei He, Yankai Chen, Xiaokun Zhang, and Xue Liu. Branching policy optimization: Sandbox-native language agent reinforcement learning. *arXiv preprint arXiv:2607.14171*, 2026.

Daniel G. Horvitz and Donovan J. Thompson. A generalization of sampling without replacement from a finite universe. *Journal of the American Statistical Association*, 47(260):663–685, 1952.

Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

Naman Jain, Jaskirat Singh, Manish Shetty, Liang Zheng, Koushik Sen, and Ion Stoica. R2E-Gym: Procedural environments and hybrid verifiers for scaling open-weights SWE agents. *arXiv preprint arXiv:2504.07164*, 2025.

Yue Jia and Mark Harman. An analysis and survey of the development of mutation testing. *IEEE Transactions on Software Engineering*, 37(5):649–678, 2011.

Nan Jiang and Lihong Li. Doubly robust off-policy value evaluation for reinforcement learning. In *International Conference on Machine Learning*, 2016.

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? *International Conference on Learning Representations*, 2024.

Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *International Conference on Machine Learning*, 2018.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. Vineppo: Refining credit assignment in rl training of llms. *arXiv preprint arXiv:2410.01679*, 2024.

Jannik Kossen, Sebastian Farquhar, Yarin Gal, and Tom Rainforth. Active testing: Sample-efficient model evaluation. In *International Conference on Machine Learning*, 2021.

David Krueger et al. Active reinforcement learning: Observing rewards at a cost. *arXiv preprint*, 2020.

Qingzhou Luo, Farah Hariri, Lamyaa Eloussi, and Darko Marinov. An empirical analysis of flaky tests. In *ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2014.

David R. MacIver and Zac Hatfield-Dodds. Hypothesis: A new approach to property-based testing. *Journal of Open Source Software*, 4(43):1891, 2019.

OpenAI. Introducing SWE-bench Verified. `https://openai.com/index/introducing-swe-bench-verified/`, 2024. Human-validated SWE-bench subset.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with swe-gym. *arXiv preprint arXiv:2412.21139*, 2024.

Liming Pu, Xiaoxia Li, Yifu Liu, Teng Cao, and Bin Yang. Explore more, drift less: Outcome-only reinforcement learning can suffice for long-horizon interactive agents. *arXiv preprint arXiv:2609.01245*, 2026.

James M. Robins, Andrea Rotnitzky, and Lue Ping Zhao. Estimation of regression coefficients when some regressors are not always observed. *Journal of the American Statistical Association*, 89(427):846–866, 1994.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Burr Settles. Active learning literature survey. Technical report, University of Wisconsin–Madison, 2009.

Zhihong Shao, Peiyi Wang, Qihao Zhu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward hacking. In *Advances in Neural Information Processing Systems*, 2022.

Philip S. Thomas and Emma Brunskill. Data-efficient off-policy policy evaluation for reinforcement learning. In *International Conference on Machine Learning*, 2016.

Binghai Wang, Chenlong Zhang, Dayiheng Liu, Jiajun Zhang, Jiawei Chen, Mingze Li, Mouxiang Chen, Rongyao Fang, Siyuan Zhang, Xuwu Wang, et al. The verification horizon: No silver bullet for coding agent rewards. *arXiv preprint arXiv:2606.26300*, 2026a.

Peiyi Wang, Lei Li, Zhihong Shao, et al. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

Xucong Wang, Ziyu Ma, Yong Wang, et al. APPO: Agentic procedural policy optimization. *arXiv preprint arXiv:2606.12384*, 2026b.

Xucong Wang, Zhe Zhao, Liheng Yu, Di Wu, Xiaofeng Cao, and Pengkun Wang. Didpo: Diff-in-diff policy optimization for coding agent training. *arXiv preprint arXiv:2608.07147*, 2026c.

Yuxiang Wei, Zhiqing Sun, Emily McMilin, Jonas Gehring, David Zhang, Gabriel Synnaeve, Daniel Fried, Lingming Zhang, and Sida Wang. Toward training superintelligent software agents through self-play SWE-RL. *arXiv preprint arXiv:2512.18552*, 2025.

Daoguang Zan, Zhirong Huang, Wei Liu, et al. Multi-SWE-bench: A multilingual benchmark for issue resolving. *arXiv preprint arXiv:2504.02605*, 2025.

Jianqing Zhang, Zhezheng Hao, Wei Xia, Hande Dong, Hong Wang, Chenxing Wei, Yuyan Zhou, Yubin Qi, Qiang Lin, and Jian Cao. Gapo: Robust advantage estimation for real-world code llms. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 17663–17673, 2026.

A Reproducibility and Experimental Protocol

A.1 Dataset Construction and Leakage Audit

Table 4 reports the filtering funnel. A task is retained only if the property-synthesis pipeline produces at least five properties that pass and are non-flaky on the reference patch (three consecutive clean runs) and the mutation harness completes within its timeout. This restricts training to tasks on which the reference protocol is defined.

Table 4: Task filtering funnel. Counts are tasks remaining after each stage.

Stage	Tasks
Source pool (SWE-Gym 2,438; R2E-Gym 8,101; SWE-rebench train 13,297)	23,836
Reproducible build with pinned container digest	15,912
Fail-to-pass check (fails before / passes after reference patch)	9,847
Raw verifier completes within 900 s	8,516
Cross-source deduplication (repo, base commit, issue)	8,203
Evaluation-suite overlap removal	7,941
Audit-protocol viability (≥ 5 surviving properties, mutation harness runs)	**2,184**

Overlap removal matches on repository and base commit, issue ID and normalized issue text (MinHash, Jaccard ≥ 0.8), and SHA-256 AST-hunk hashes over file path and α-renamed patch subtrees. Any collision with a task in an evaluation suite removes the training candidate. All retained issues predate the March 2025 evaluation cutoff; source-dataset licenses are recorded per task in the manifest.

A.2 Hyperparameters

Table 5: Policy optimization and generation settings, identical across methods unless noted.

Policy optimization		Generation and rollout	
Optimizer	AdamW	Temperature / top-p	0.7 / 0.95 (no top-k)
Learning rate	1.0×10^{-5}, constant	Max generated tokens / task	8,192
Warmup	20 steps	Max tool calls / task	30
PPO clip ϵ	0.2	Max turns	40
KL coefficient β	0.02, fixed	Task timeout	900 s
PPO epochs	1	Branch-rollout timeout	600 s
Rollout batch	32 tasks $\times$ 4 trajectories	Sibling sampling	same as backbone
Minibatch	16 seq/GPU, grad. accum. 2	Token accounting	backbone + siblings
Max grad norm	1.0		vs. B_{tok}
Advantage norm	per environment group (Norm_e)		
LoRA	rank 64, $\alpha = 128$, dropout 0.05		
LoRA targets	q,k,v,o,gate,up,down proj.		
Reference policy	frozen SFT checkpoint		
Checkpoint	at budget exhaustion (no selection)		
Seeds	17, 23, 42, 1009, 2024	Ablation seeds	17, 23, 2024

Acquisition uses $K = 4$, $p_{\min} = 0.025$, $p_{\max} = 0.40$, $\tau_{\text{acq}} = 0.20$, $\eta_\lambda = 10^{-3}$, and $\alpha_{\text{loc}} = 0.5$. A full-budget run comprises approximately 195 iterations, 2.5×10^4 backbone trajectories, 2.1×10^3 branches (mean $p_c = 0.029$), and 1.08×10^3 audits (mean audit cost 74 sandbox seconds).

A.3 BANC Auxiliary Models

The calibrator q_η, credit model m_ϕ, and outcome predictor $\widehat{P}(y, w \mid x)$ are 3-layer MLPs (hidden width 128, GELU) over a shared 41-dimensional feature vector containing action type, diff statistics, test history, execution features, trajectory position, unit length, policy log-probability, and entropy. Task identity enters through a 16-dimensional hashed embedding. Features use running z-

normalization frozen at iteration boundaries. $\widehat{P}$ has logistic verdict and categorical execution heads; $\widehat{q}_i^{\mathrm{pre}}$ marginalizes q_- over 32 samples.

All auxiliary models use AdamW (lr 10^{-3}, batch 512, 200 refresh steps) with five task-level cross-fitting folds. Predictions at iteration t use fold-excluded models fitted through $t-1$; iteration 1 uses uniform budget-feasible purchases and GVA begins at iteration 2. Logged propensities are never recomputed after refreshes.

A.4 Baseline Tuning and Matched-Budget Protocol

Each method was tuned at 25% budget on a fixed 64-task validation split disjoint from the evaluation suites, using the search spaces in Table 6. The configuration with the highest validation resolve rate was fixed before full-budget runs.

Table 6: Baseline search spaces and selected configurations.

Method	Search space	Trials	Selected
RLOO	rollouts $\{4, 8\}$	2	8
GRPO	group size $\{4, 8\}$	2	8
BPO	branch rate $\{0.05, 0.1, 0.2\} \times K \in \{2, 4\}$	6	0.1, $K = 4$
APPO	granularity {turn, action}$\times$ rate $\{0.05, 0.1, 0.2\}$	6	action, 0.1
DiDPO	diff unit {hunk, file} $\times\ \beta_{\mathrm{dpo}} \in \{0.1, 0.3\}$	4	hunk, 0.1
BPO+DIFF	branch rate $\{0.05, 0.1, 0.2\} \times K \in \{2, 4\}$	6	0.1, $K = 4$
BANC	$\alpha_{\mathrm{loc}} \in \{0.25, 0.5, 1.0\} \times \tau_{\mathrm{acq}} \in \{0.1, 0.2, 0.4\}$	9	0.5, 0.2

A.5 Statistical Inference

Bootstrap. Unless otherwise noted, result-table CIs are percentile intervals from $B = 10{,}000$ two-way paired bootstrap replicates: seeds and tasks are resampled with replacement using identical indices across methods in a table. The SFT row uses task resampling only. Parenthetical diagnostic values are standard errors across ten replay or re-masking replicates, not bootstrap CIs.

Confirmatory test. For task t, let $d_t = \frac{1}{5}\sum_s (Y_{t,s}^{\mathrm{BANC}} - Y_{t,s}^{\mathrm{BPO+DIFF}})$ over the 640 SWE-rebench tasks. We studentize its mean and use $100{,}000$ Rademacher sign-flip replicates for a two-sided test. This contrast is the sole prespecified confirmatory hypothesis; all other comparisons are secondary and descriptive.

A.6 Audit Samples and Calibration Metrics

Two uniform random samples of validation leaves were audited offline with the full reference protocol, and neither is used to fit any calibrator. The *prevalence sample* (500 leaves) was drawn before training to estimate verifier disagreement and to set the intercept of the injected-noise generator; its breakdown is given in Sec. 4.4. The *calibration evaluation set* was drawn after training, disjoint from the prevalence sample: $1{,}049$ leaves, of which 25 (2.4%) abstained and $1{,}024$ received determinate verdicts; 125 of these (12.2%) are confirmed disagreements. Calibrators in Table 12 are fitted only on the adaptively audited training leaves of their own run and then scored on this set as predictors of $r_\ell = \mathbb{I}[y_\ell = z_\ell^{\mathrm{ref}}]$. Brier is the mean squared error, NLL the mean negative log-likelihood (natural log), and ECE uses ten equal-mass bins. The raw verifier ($q \equiv 1$) is a constant, not a probabilistic predictor: its Brier and ECE reduce to the disagreement rate and its NLL is unbounded, so these columns are not reported for that row; its remaining columns describe the policy trained without a calibrator.

A.7 Injected Verifier Noise

The stress condition corrupts the *training* verifier only. For each observed leaf, y_ℓ is inverted independently with probability

$$p_{\mathrm{flip}}(\ell) = \mathrm{clip}\big(0.08 + 0.24\hat{t}_\ell + 0.18\hat{f}_\ell,\ 0.02,\ 0.50\big),$$

where $\hat{t}_\ell = \min(\text{suite runtime}/300\,\text{s}, 1)$ and $\hat{f}_\ell$ is the fraction of filesystem- or network-fixture tests. The intercept targets a 20% expected rate. Evaluation is never corrupted, and the same within-seed realization is shared by compared variants.

Table 7: Realized corruption per noise-condition run (3 seeds).

Seed	Eligible leaves	Flipped	Rate	$1 \to 0$	$0 \to 1$
17	30,412	6,102	20.1%	1,873	4,229
23	29,876	5,911	19.8%	1,842	4,069
2024	30,655	6,178	20.2%	1,905	4,273

A.8 Environment-Time Accounting

Sandbox seconds cover rollout and test execution, restoration, container startup, and failed containers at realized consumption; scoring GPU time is reported separately. Every method exhausted the token cap.

Table 8: Ledger decomposition (10^6 s, mean over 5 seeds).

Method	Backbone	Branch	Audits	Restore	Failed containers	Total
APPO	0.640	0.436	0	0.086	0.034	1.196
BPO+DIFF	0.648	0.424	0	0.086	0.034	1.192
BANC	0.660	0.278	0.080	0.062	0.028	1.108

B Method Details

B.1 Masking Policies

The credit-recovery experiment applies four masking policies to the same exhaustive set. Each policy is randomized and assigns nonzero probability to every eligible unit:

- **Uniform at matched rate:** $p_c = \bar{p}$, where $\bar{p}$ is chosen to match the expected number of purchases made by GVA.
- **Entropy:** $p_c = p_{\min} + (p_{\max} - p_{\min})\sigma((H_c - \lambda^\top \widehat{\mathbf{c}}_c)/\tau_{\text{acq}})$, where H_c is predictive entropy.
- **APPO score:** $p_c = p_{\min} + (p_{\max} - p_{\min})\sigma((D_c - \lambda^\top \widehat{\mathbf{c}}_c)/\tau_{\text{acq}})$, where D_c is the APPO branching score.
- **GVA:** $p_c = p_{\min} + (p_{\max} - p_{\min})\sigma((U_c - \lambda^\top \widehat{\mathbf{c}}_c)/\tau_{\text{acq}})$, using the utility in Eqs. equation 3 and equation 4.

The realized Bernoulli draws and their probabilities are logged for every candidate and used unchanged by the IPS and AIPW estimators.

B.2 GVA Derivation and Implementation

Let $\bar{g}_i$ be the score aggregated over the tokens of unit i and $m^*(\mathcal{F}_i) = \mathbb{E}[\Delta_i^{\text{ref}} \mid \mathcal{F}_i]$. Under a local linearization of the policy update, the contribution of unit i is $\bar{g}_i \Delta_i^{\text{ref}}$, and using the conditional mean in place of the unobserved credit incurs $\mathbb{E}[\|\bar{g}_i \Delta_i^{\text{ref}} - \bar{g}_i m^*(\mathcal{F}_i)\|_2^2 \mid \mathcal{F}_i] = \|\bar{g}_i\|_2^2 \operatorname{Var}(\Delta_i^{\text{ref}} \mid \mathcal{F}_i)$. Eq. equation 3 replaces $\|\bar{g}_i\|_2^2$ by the per-token proxy G_i (the aggregated score would require a backward pass per candidate) and the conditional variance by $\widehat{\sigma}_i^2$. Finite-K Monte Carlo error, calibration error, and interactions among simultaneous purchases are not captured, which is why the surrogate is evaluated empirically rather than relied on as an identity.

B.3 Cross-Fitted Training Loop

Algorithm 1 One training iteration of BANC

Require: Policy π_θ, lagged q_- and m_-, batch budget $\mathbf{B}_{\text{batch}}$, duals $\boldsymbol{\lambda}$

1: Sample backbone trajectories; construct $\mathcal{I}(\tau) = \Gamma(\tau)$
2: Compute U_i^{B} (Eq. 3); sample and execute branches with logged p_i^{B} (Eq. 5)
3: Compute U_ℓ^{A} (Eq. 4); sample and execute audits with logged p_ℓ^{A}
4: Compute $\bar{z}_\ell$ (Eq. 2) and $\widehat{\Delta}_i^{\text{AC}}$ (Eq. 8)
5: Form advantages (Eq. 10); update π_θ
6: Fit q_η, m_ϕ (Eqs. 1, 7); update $\boldsymbol{\lambda}$

B.4 Estimator Statement and Proof

Condition on $\mathcal{F}_i$ and $D_i = 1$. By (i) the features x_i and the predictor m_{-i} are $\mathcal{F}_i$-measurable, and by (ii) $\mathbb{E}[S_i^{\text{B}}/p_i^{\text{B}} \mid \mathcal{F}_i] = 1$ with S_i^{B} independent of $\widetilde{\Delta}_i$ given $\mathcal{F}_i$. Hence $\mathbb{E}[\widehat{\Delta}_i^{\text{AC}} \mid \mathcal{F}_i, D_i{=}1] = m_{-i}(x_i) + \mathbb{E}[\widetilde{\Delta}_i - m_{-i}(x_i) \mid \mathcal{F}_i, D_i{=}1] = \mathbb{E}[\widetilde{\Delta}_i \mid \mathcal{F}_i, D_i{=}1]$. Each leaf utility $\bar{z}_\ell$ is either the determinate reference verdict (audited, no abstention) or the calibrated utility (unaudited, or audited with abstention); by (iii) and (iv) both have conditional mean $\mathbb{E}[z_\ell^{\text{ref}} \mid y_\ell, w_\ell, \text{determinate}]$, so $\mathbb{E}[\widetilde{\Delta}_i \mid \mathcal{F}_i, D_i{=}1] = \mathbb{E}[\Delta_i^{\text{ref}} \mid \mathcal{F}_i, D_i{=}1]$. For the bound, if $|q_-(y,w) - \Pr(y = z^{\text{ref}} \mid y, w)| \le \epsilon_q$ then $|\mathbb{E}[\bar{z}_\ell \mid y_\ell, w_\ell] - \mathbb{E}[z_\ell^{\text{ref}} \mid y_\ell, w_\ell]| \le \epsilon_q$ for every leaf, and $\widetilde{\Delta}_i$ is a difference of two such terms, giving absolute bias at most $2\epsilon_q$.

C Additional Results

C.1 Additional Result Tables

Table 9: Deterministic adjudication rule for the reference verdict z_ℓ^{ref}.

Clean disjoint suite	Execution invariants	Mutation score	z_ℓ^{ref}
All pass	Retained	≥ 0.80	Pass ($z = 1$)
Any fail	Retained	–	Fail ($z = 0$)
Any	Violated (crash, exit $\neq 0$)	–	Abstain
All pass	Retained	< 0.80	Abstain

Table 10: Compute and supervision overhead at full budget on one 8×H100 node. Node GPU-h = 8× wall-clock; scoring GPU-h tracks online Fisher/utility estimation.

Method	Node GPU-h	Wall-clock h	Scoring GPU-h	Scoring share	Human h	Audit LLM tokens
APPO	284.8	35.6	9.4	3.3%	0	0
BPO+Diff	291.2	36.4	11.8	4.1%	0	0
BANC	318.4	39.8	23.2	7.3%	0	0

Table 11: Matched-cap performance frontier on SWE-rebench ($N = 640$). Point entries report mean resolved rate [95% two-way paired-bootstrap CI] over 5 seeds. nAUC is deterministically computed from displayed budget means via Eq. equation 11.

Method	25% Budget	50% Budget	75% Budget	100% Budget	nAUC ↑
GRPO	13.1 [11.8, 14.5]	15.6 [14.1, 17.0]	17.4 [15.8, 18.9]	18.8 [17.1, 20.4]	16.3
BPO	16.2 [14.8, 17.7]	20.4 [18.9, 21.9]	22.8 [21.1, 24.3]	24.1 [22.4, 25.7]	21.1
APPO	16.8 [15.3, 18.2]	20.9 [19.3, 22.4]	23.3 [21.7, 24.9]	24.7 [23.0, 26.3]	21.7
DiDPO	15.9 [14.4, 17.3]	19.8 [18.2, 21.3]	22.4 [20.7, 24.0]	24.0 [22.3, 25.6]	20.7
BPO+DIFF	16.9 [15.4, 18.4]	21.5 [19.9, 23.0]	24.2 [22.6, 25.8]	25.6 [23.9, 27.2]	22.3
BANC	**17.4** [15.9, 18.9]	**23.6** [22.0, 25.2]	**26.7** [25.0, 28.3]	**28.3** [26.6, 30.0]	**24.4**

Table 12: Reliability calibration on the held-out evaluation set ($N = 1{,}024$ determinate audited leaves, Appendix A.6) and robustness to 20% training-verifier noise. The raw verifier is a constant predictor, so proper-scoring metrics are not reported for it. Resolve columns use the same 3 seeds as Table 3. The Clean Resolve entries for the raw-verifier and IPW-calibrator variants correspond to rows $(1, 0, 1)$ and $(1, 1, 1)$ of that table, respectively; Harmful Update Mass here is measured under the 20% verifier-noise condition. The final row reports a full-audit diagnostic evaluated separately from the matched training budget.

Variant	Brier ↓	NLL ↓	ECE ↓	Harmful Update Mass (%) ↓	Clean Resolve (%) ↑	Shifted Resolve (%) ↑
Raw verifier (no calibrator, $q \equiv 1$)	–	–	–	24.7 [22.8, 26.6]	26.2 [24.2, 28.2]	19.2 [17.2, 21.2]
Majority retest	0.149	0.448	0.134	18.2 [16.4, 20.0]	25.1 [23.1, 27.1]	21.0 [19.0, 23.0]
Unweighted calibrator	0.124	0.382	0.098	12.1 [10.5, 13.8]	26.4 [24.4, 28.4]	23.5 [21.5, 25.5]
IPW calibrator (ours)	**0.086**	**0.274**	**0.061**	**9.1** [7.6, 10.7]	**28.3** [26.3, 30.3]	**26.4** [24.4, 28.4]
Full-audit reference	0.065	0.210	0.042	4.2 [3.1, 5.4]	29.1 [27.1, 31.0]	28.0 [26.0, 30.0]

Table 13: Credit recovery under adaptive observation on the exhaustive set ($N = 512$, averaged over four masking policies). Standard errors in parentheses.

Estimator	Absolute Bias ↓	RMSE ↓	Spearman ρ ↑	95% Coverage (%) ↑	GradCos Similarity ↑
Selected only	0.148 (0.008)	0.421 (0.012)	0.44 (0.02)	68.4 (1.5)	0.51 (0.02)
Model only	0.082 (0.005)	0.334 (0.009)	0.61 (0.02)	82.1 (1.2)	0.68 (0.02)
IPS	**0.018** (0.006)	0.389 (0.015)	0.58 (0.03)	89.2 (1.1)	0.62 (0.02)
AIPW (ours)	0.021 (0.004)	**0.218** (0.007)	**0.76** (0.01)	**94.6** (0.8)	**0.84** (0.01)

Table 14: Larger-backbone transfer on SWE-rebench across 3 seeds. Point estimates report mean [95% CI] from the two-way paired bootstrap.

Backbone	APPO	BPO+DIFF	BANC
DeepSeek-Coder-V2-Lite (16B/2.4B)	30.2 [28.5, 31.9]	31.0 [29.3, 32.7]	**34.1** [32.4, 35.8]
Qwen2.5-Coder-32B	36.2 [34.4, 38.0]	37.5 [35.7, 39.3]	**40.8** [38.9, 42.6]

C.2 Per-Seed Solved Counts

Table 15: Integer solved counts per seed. Each seed is an independent training run; each checkpoint is evaluated once per task.

Suite	Method	Seed 1	Seed 2	Seed 3	Seed 4	Seed 5
SWE-rebench (640)	RLOO	113	110	114	109	114
	GRPO	122	119	123	118	120
	BPO	156	152	157	151	155
	APPO	160	155	161	154	160
	DiDPO	155	152	156	151	154
	BPO+DIFF	166	161	167	160	165
	BANC	184	178	185	177	182
SWE-bench Verified (500)	APPO	151	146	152	145	151
	BPO+DIFF	154	149	155	148	154
	BANC	168	163	169	164	166
Multi-SWE (400)	APPO	82	79	83	78	82
	BPO+DIFF	85	82	86	82	85
	BANC	94	91	95	90	94

Table 16: Per-seed solved counts on SWE-rebench (640) for the larger-backbone runs (seeds 1, 2, and 5).

Backbone	Method	Seed 1	Seed 2	Seed 5
DeepSeek-Coder-V2-Lite	APPO	196	190	194
	BPO+DIFF	201	195	199
	BANC	221	215	219
Qwen2.5-Coder-32B	APPO	234	228	233
	BPO+DIFF	243	237	240
	BANC	264	258	261

C.3 Stratum Sizes

Short/cheap stratum (< 5 SFT-median turns, suite < 10 s): 176 tasks; long/costly (> 15 turns, > 120 s): 142; remainder: 322 (total 640). Multi-SWE languages: 180 Java, 140 TypeScript, and 80 Go.

D Qualitative and Failure Analysis

D.1 Reward Hacking Caught by an Audit

In training task `pydantic-2841` (SWE-Gym partition), a patch introduced a circular type-hint import that raised `ImportError` inside a `conftest.py` guard, causing pytest to skip the entire model-validation module; with all tests skipped, the runner exited with status 0 and the raw verifier rewarded the leaf. Abnormal runtime (1.4 s vs. an expected 18.2 s) and high calibrator-ensemble variance produced a large audit utility and an audit was sampled; the clean container ran the disjoint suite, which failed, setting $z_\ell^{\text{ref}} = 0$.

D.2 False Negative Corrected in the Opposite Direction

In a `moto` task, a leaf failed raw verification because a time-window assertion in the public suite is flaky near minute boundaries. The lagged calibrator assigned $q_- = 0.91$ to the *fail* verdict, so the soft utility stayed near 0. A sampled audit returned $z_\ell^{\text{ref}} = 1$: the disjoint suite passed in the clean container. The verdict was corrected for that update and later calibrators assigned low q to short-runtime failures with timer fixtures.

D.3 AUDIT ABSTENTION

In a `dvc` task, the patched code spawned a background daemon that kept the container's exit status nonzero after the suite finished. Execution invariants were violated, the audit returned ABSTAIN, the leaf reverted to its calibrated utility, and the audit's 61 sandbox seconds remained on the ledger. Roughly 2.4% of audits abstain (Sec. 4.4), bounding what auditing can resolve.

D.4 AN ACQUISITION FAILURE

GVA assigned top-decile branch utility to a unit editing a large configuration file: G_i was high (low-probability tokens) and $\widehat{\sigma}_i^2$ was overestimated in its feature stratum. All four siblings reached verdicts identical to the backbone and the realized credit was ≈ 0. Across the diagnostic set, 9% of GVA's top-decile purchases yield $|\widetilde{\Delta}_i| < 0.05$, part of the gap between $\rho = 0.54$ and a perfect ranking; the purchase is wasted budget but, by Eq. equation 9, not a source of bias.