TEST WHAT MATTERS: LEARNING TO VERIFY UPDATES IN SELF-EVOLVING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-evolving language agents improve by updating instructions, memories, and skills, but identifying beneficial updates incurs substantial verification cost. When candidates are evaluated using fixed task sets or random sampling, the selected tasks may not match those actually affected by the update. This mismatch can waste evaluations on unaffected tasks and leave critical improvements or regressions undetected, leading to incorrect update decisions. We introduce BRACE, Bayesian Risk-Aware Candidate Evaluation, which selects validation tasks to inform candidate-specific deployment decisions. BRACE learns shared response structure offline from historical paired outcomes to guide validation task selection. During online verification, BRACE updates predictions from new paired observations and uses Decision Value of Information (DVI) to prioritize tests by their expected reduction in deployment risk. A calibrated posterior-risk threshold governs stopping. We evaluate BRACE in SkillOpt and EvoSkill on SpreadsheetBench-Verified and DocVQA across four language-model configurations. With Qwen3.5-4B, BRACE reduces fresh online validation episodes by 54.0–82.3% relative to validation with a determinantal point process (DPP) coreset while achieving higher mean terminal-run success rates. Across models, BRACE achieves higher mean terminal-run success than host-native validation. Controlled replay further shows that DVI uses fewer pairs than Random and Value-variance selection.

1 INTRODUCTION

Language agents increasingly use interaction feedback to make persistent changes to their instructions, memories, skills, and operating procedures, enabling self-improvement (Zhang et al., 2025; Yang et al., 2026; Zhang et al., 2026; Zhao et al., 2024; Wang et al., 2025; Majumder et al., 2023). These updates allow agents to carry useful experience into future tasks and continue adapting over time. They also make evolution path dependent. Once accepted, an update can improve future behavior or carry regressions into subsequent tasks, while influencing further updates generated from the modified state. Reliable update validation is therefore essential to sustained beneficial evolution, determining which candidate modifications become part of persistent agent state.

To decide whether to accept candidate updates, existing approaches commonly follow a propose-and-verify loop, first generating candidate changes from interaction feedback and then assessing their effects through additional task executions. This paradigm is widely used in prompt and language-model program optimization (Yang et al., 2024; Opsahl-Ong et al., 2024; Agrawal et al., 2025; Khattab et al., 2024; Yüksekgönül et al., 2025) and has extended to skill evolution (Yang et al., 2026; Wang et al., 2024) and agent harness repair (Chen et al., 2026; Zhang et al., 2026). These methods typically verify candidate changes using fixed task sets (Lin et al., 2026; Guo et al., 2024), randomly sampled minibatches (Opsahl-Ong et al., 2024; Pryzant et al., 2023), or preselected task subsets (Pan et al., 2026).

Fixed or randomly sampled task sets do not adapt to differences in candidate effects. They can waste evaluations on unaffected tasks while missing improvements or regressions that change the deployment decision. Existing work reduces verification cost through task sampling (Opsahl-Ong et al., 2024; Pryzant et al., 2023), preselected subsets (Pan et al., 2026), staged screening (Luo et al., 2026), or sequential stopping (Shawn, 2026). Behavior-aware verification further targets a modification's intended effects and regression risks (Xu et al., 2026). The central question is

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.

how historical outcomes and online evidence can guide test selection by its expected reduction in deployment-decision loss.

We introduce BRACE, *Bayesian Risk-Aware Candidate Evaluation*, to verify candidate updates for deployment in self-evolving agents. The method comprises offline learning and online verification. Offline, BRACE learns shared response structure from historical paired outcomes of candidate and incumbent versions to predict new updates' effects across tasks. Online, it combines these predictions with acquired evidence to infer deployment value and uses *Decision Value of Information* (DVI) to select tests by their expected reduction in deployment risk. Predictions and testing priorities adapt as results arrive. A calibrated posterior-risk threshold governs stopping, after which BRACE selects the commit or preserve action with lower expected loss.

We integrate BRACE into SkillOpt and EvoSkill and evaluate it on SpreadsheetBench-Verified and DocVQA. With Qwen3.5-4B, BRACE reduces fresh online validation episodes by 54.0–82.3% relative to validation with a determinantal point process (DPP) coreset that balances task difficulty and diversity, while achieving higher mean terminal-run success rates. Including startup, retained host evaluation, and checkpoint checks, BRACE reduces amortized validation rollout cost by 24.9–40.1% relative to DPP, excluding configuration-search costs for both methods. Controlled replay and ablations show that the historical response structure allows observed outcomes to refine predictions for untested tasks, while decision-directed task selection reaches comparable mean deployment-decision regret with fewer paired evaluations. Across all four language-model configurations, BRACE achieves higher mean terminal-run success than host-native validation, while controlled replay shows fewer paired evaluations than Random and Value-variance selection (Appendix D).

Our contributions are threefold:

- We formulate persistent-update verification as a deployment decision and value each paired evaluation by its expected reduction in the loss from an incorrect commit or preserve action.
- We introduce BRACE, a validation plug-in that combines historical response transfer with DVI-guided online task selection for existing skill-evolution systems.
- We demonstrate BRACE's quality–cost gains in two skill-evolution systems on two benchmarks, with success and acquisition comparisons spanning four language-model configurations. Controlled replay and ablations reveal the contributions of historical response transfer and decision-directed task selection.

2 Related Work

Self-evolving agents and persistent-state updates. Language agents can retain interaction feedback as changes to prompts, context, and skills while keeping model weights fixed. GEPA uses trajectory reflection to propose and evaluate prompt changes, while ACE organizes accumulated strategies into evolving contextual playbooks (Agrawal et al., 2025; Zhang et al., 2025). SkillOpt generates bounded edits to a skill document and accepts them when they improve held-out performance (Yang et al., 2026). EvoSkill creates or revises skill folders and maintains a frontier of programs from which further candidates are proposed (Alzubi et al., 2026). BRACE receives the host's proposed candidate and guides its validation across a target workload, leaving candidate generation, task execution, and scoring unchanged.

Candidate-update verification and task allocation. Update verification methods differ in how they select comparison tasks, screen candidates, and determine when evidence supports acceptance. RHO selects a task coreset for difficulty and diversity before comparing proposed harnesses (Pan et al., 2026). HarnessBank applies validity, activation, significance, and gain gates on a sampled subset, reserving full evaluation for candidates that pass this screening (Luo et al., 2026). PACE uses an anytime-valid paired sequential test to control each candidate's false-commit probability while allowing early stopping (Shawn, 2026). Trajectory-grounded diagnosis guides repair validation (Chen et al., 2026), while behavior-aware verification selects tasks around a modification's intended effects and regression risks (Xu et al., 2026). BRACE learns shared response structure from historical candidate–incumbent paired effects to predict a new candidate's impact on unevaluated tasks. It uses these predictions to prioritize paired evaluations by their expected reduction in Bayes deployment risk.

Active evaluation offers related mechanisms for choosing informative tests. Active Testing adaptively acquires test labels while correcting selection bias (Kossen et al., 2021), and Active Evaluation Acquisition (AEA) learns response dependencies to predict unobserved outcomes and guide sequential testing for performance estimation (Li et al., 2025). Information-value theory and decision-directed active learning assess observations through their consequences for downstream actions (Howard, 1966; Javdani et al., 2014; Krause & Guestrin, 2009; Golovin et al., 2010). This decision-directed perspective underlies BRACE's asymmetric loss for accepting harmful updates or rejecting beneficial ones.

3 Task Selection for Update Validation

Update validation requires tasks that help judge whether a candidate is worth committing. We analyze 40 candidate updates from four SkillOpt evolution runs with Qwen3.5-4B. Each candidate is compared with its incumbent on the same 40 SpreadsheetBench tasks from the host's validation catalog, yielding 1,600 paired observations. Success differences are $\Delta_{ij} \in \{-1, 0, +1\}$. The full-catalog reference decision is to commit the candidate when its mean paired effect across all tasks is positive and to preserve the incumbent otherwise. One task-level paired comparison counts as one budget unit.

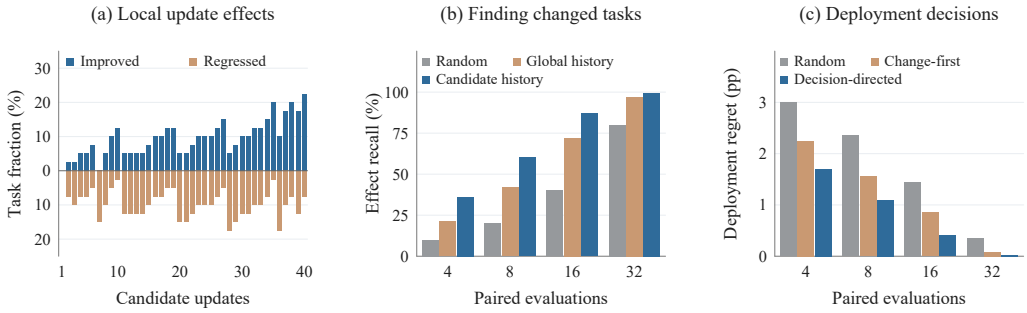

Figure 1: **Task selection for update validation.** (a) Improved (upward blue) and regressed (downward brown) task fractions for 40 candidate updates, ordered by total affected fraction. (b) Changed-task recall (higher is better). (c) Deployment-decision regret (lower is better). Panels (b,c) compare three strategies at budgets of 4, 8, 16, and 32 paired evaluations (10%, 20%, 40%, and 80% of the task pool). pp denotes percentage points.

Updates affect a small subset of tasks, often with both gains and losses. Figure 1(a) shows that 38 of the 40 candidate updates produce both improvements and regressions. The median update changes success on eight of the 40 tasks, leaving the remaining tasks unaffected. A small validation subset may therefore miss the affected tasks or capture only one side of an update's effects.

Candidate-related history helps locate affected tasks. Figure 1(b) compares random selection, global historical priorities shared across candidates, and candidate-related priorities based on similarity to historical updates. Rankings are fixed before testing and exclude current-candidate outcomes. Recall measures the fraction of changed tasks discovered. With eight paired evaluations (20% of the task pool), the three strategies achieve recalls of 20%, 42%, and 60%, respectively. The advantage over global priorities shows the value of conditioning task selection on the candidate update, beyond identifying tasks that are generally sensitive to changes.

Decision-directed selection turns evidence into better deployment decisions. Finding affected tasks does not by itself determine whether an update should be committed: improvements and regressions can offset one another. Figure 1(c) holds the historical predictor and action rule fixed and compares random, change-first, and decision-directed selection at matched evaluation budgets. Regret measures mean task reward lost relative to the full-catalog reference action. At 16 paired evaluations, the three strategies yield regrets of 1.45, 0.85, and 0.40 percentage points, respectively. The improvement over change-first selection shows the benefit of prioritizing evidence by its consequences for the deployment decision. Together, these findings motivate learning candidate-specific response patterns from history and using them to select tests that reduce deployment-decision risk.

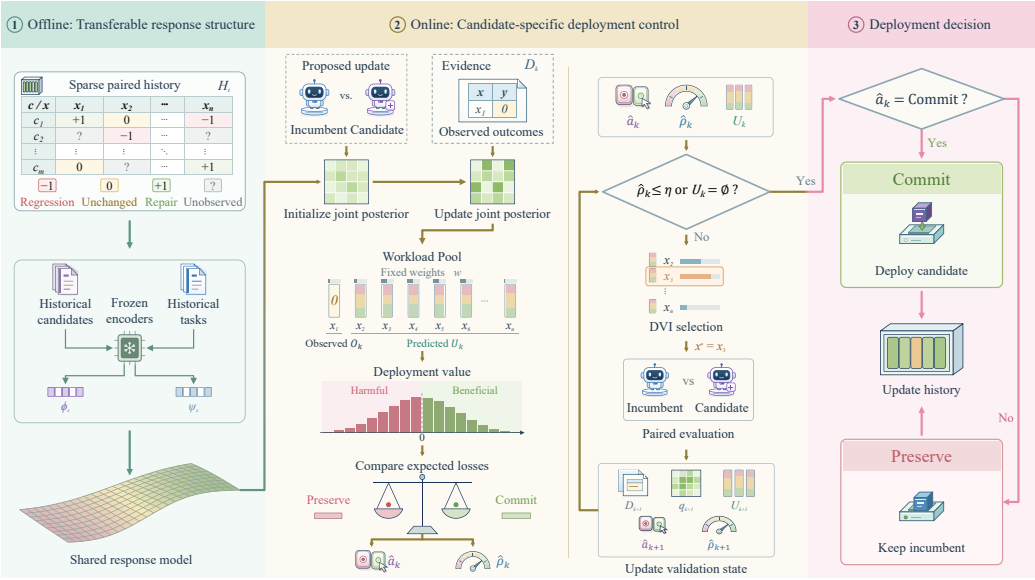

Figure 2: **Overview of BRACE.** (1) Offline, BRACE learns shared response structure from historical candidate–incumbent paired outcomes. (2) Online, the model and candidate context initialize a joint posterior. Paired observations update deployment-value and risk estimates. DVI selects the next task by its expected risk reduction. Validation continues until the stopping criterion is met. (3) BRACE commits the candidate or preserves the incumbent according to posterior expected loss, and adds acquired observations to the historical bank.

4 METHOD

BRACE decides whether a candidate update should enter persistent agent state. BRACE learns a shared response structure from historical paired outcomes to capture how an update's effects on different tasks are related. For a new candidate, the response model uses its update description and incumbent context to infer its position in this structure and predict its effects across tasks. DVI uses these predictions to select the next task with the greatest expected reduction in deployment risk, adapting test priorities as paired outcomes are observed. Posterior deployment risk determines when validation stops, after which the expected losses of commit and preserve are compared to make the deployment decision (Figure 2).

4.1 PERSISTENT UPDATES AS DEPLOYMENT DECISIONS

At evolution step t, the agent's current persistent state is K_t, and applying a proposed update u_t produces the candidate version $K_t \oplus u_t$. We denote this candidate–incumbent comparison by $c = (K_t, u_t)$ to assess whether the update should be committed.

Let $\mathcal{X}_t = \{x_1, \ldots, x_N\}$ be the validation pool, where each workload condition x consists of a task instance and its execution settings. The incumbent and candidate are evaluated on the same task under matched execution conditions. With $R(x \mid K) \in \{0, 1\}$ indicating whether version K succeeds on task x, the paired effect is

$$y_{c,x} = R(x \mid K_t \oplus u_t) - R(x \mid K_t) \in \{-1, 0, +1\}, \tag{1}$$

Here -1, 0, and $+1$ denote regression, no change, and improvement, respectively. Each matched comparison counts as one logical pair. Replay budgets count pairs. System costs count new rollout episodes, excluding cached branch outcomes.

Deployment depends on an update's aggregate effect. Let $w_x \geq 0$ denote task x's workload share, fixed before evaluation with $\sum_x w_x = 1$. The finite-pool deployment value is

$$\widehat{V}_c = \sum_{x \in \mathcal{X}_t} w_x y_{c,x}. \tag{2}$$

A positive value means the candidate is better overall under the given weights, while a negative value indicates an overall regression. Online validation aims to decide whether the update is worth deploying using a small number of targeted paired evaluations. Since effects on unevaluated tasks remain unknown, we need to quantify the cost of a wrong decision based on partial evidence.

We distinguish two decision errors: committing a harmful update and preserving the incumbent while missing a beneficial update. For action $a \in \{\text{COMMIT}, \text{PRESERVE}\}$ and deployment value V, define the loss

$$\ell(a, V) = \begin{cases} \lambda_{\mathrm{H}}[-V]_+, & a = \text{COMMIT}, \\ \lambda_{\mathrm{M}}[V]_+, & a = \text{PRESERVE}. \end{cases} \tag{3}$$

Here $[v]_+ = \max(v, 0)$, and the positive coefficients λ_{H} and λ_{M} control the penalties for harmful commits and missed beneficial updates, respectively. The full-information optimal action has zero loss, so $\ell(a, \widehat{V}_c)$ also measures the deployment-decision regret of action a relative to that reference.

4.2 Learning Shared Response Structure

In the offline stage, we learn shared patterns of update effects across tasks from historical candidate descriptions, task information, and observed paired outcomes. The historical bank $\mathcal{H}_t = \{(c, x, y_{c,x}) : (c, x) \in \Omega_t\}$ stores paired records acquired before the current candidate's validation begins, where Ω_t indexes observed candidate–task pairs. Each record links an update, a validation task, and its observed effect, providing evidence for learning the shared response structure.

For each historical candidate c, a frozen encoder maps its incumbent summary, update diff, and pre-validation trajectories or feedback to candidate features $\boldsymbol{\phi}_c$. Task instructions and metadata are likewise encoded as task features $\boldsymbol{\psi}_x$. The encoder extracts semantic information from descriptions, while the model learns its relationship to update effects from historical paired outcomes.

We represent a candidate's response features by a low-dimensional vector $\mathbf{z}_c \in \mathbb{R}^d$ and a task's relationship to these features by $\mathbf{h}_x \in \mathbb{R}^d$. The candidate's latent response score on the task is

$$s_{c,x} = b_c + b_x + \mathbf{z}_c^\top \mathbf{h}_x + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, 1), \tag{4}$$

Here b_c captures the candidate's overall effect tendency, b_x captures the task's overall effect tendency under historical updates, and $\mathbf{z}_c^\top \mathbf{h}_x$ captures candidate–task response differences. The same candidate vector enters effect predictions across tasks, linking these predictions.

To apply this structure to new candidates, we jointly learn mappings from description features to response parameters:

$$\boldsymbol{\theta}_c = [b_c, \mathbf{z}_c^\top]^\top \sim \mathcal{N}(W_c \boldsymbol{\phi}_c, \mathbf{S}_0), \quad \mathbf{h}_x = W_x \boldsymbol{\psi}_x + \mathbf{r}_x, \quad b_x = \mathbf{v}^\top \boldsymbol{\psi}_x + \beta_x. \tag{5}$$

Here W_c maps candidate descriptions to the prior mean of their response parameters, and $\mathbf{S}_0$ represents uncertainty around this mean. W_x and $\mathbf{v}$ map task descriptions to task response parameters. The residuals $\mathbf{r}_x$ and β_x have zero-mean Gaussian priors and use historical outcomes to capture task differences beyond descriptions.

Historical paired outcomes comprise regression, no change, and improvement. An ordered-probit model (McKelvey & Zavoina, 1975) converts the latent response score in Equation (4) into probabilities of the three outcomes. Two thresholds separate regression from no change and no change from improvement. Appendix A.1 gives the probability calculation.

Training uses observed historical paired outcomes to learn shared parameters, denoted by Ψ, including the description-to-response mappings, task response parameters, and thresholds. Variational inference (Salakhutdinov & Mnih, 2007; Paquet et al., 2012) yields an approximate posterior $q_t(\Psi)$, retaining uncertainty about parameter values given the historical data. The active checkpoint is fitted on $\mathcal{H}_t^{\text{fit}} \subseteq \mathcal{H}_t$, excluding held-out calibration records. Missing pairs remain unobserved and are not labeled as no change. The learned shared model is used to predict new candidates' effects during online validation.

4.3 Online Prediction Updates and Deployment-Value Inference

For a new candidate, the context-conditioned prior in Equation (5) and historical posterior $q_t(\Psi)$ predict task outcomes before paired evidence. Let $D_k = \{(x_i, y_{c,x_i})\}_{i=1}^k$ contain k revealed pairs,

including cached complete pairs, $\mathcal{O}_k$ the observed tasks, and $\mathcal{U}_k = \mathcal{X}_t \setminus \mathcal{O}_k$ the remaining tasks. The target joint posterior is

$$\widetilde{p}_k(\boldsymbol{\theta}_c, \Psi) \propto q_t(\Psi)\, p(\boldsymbol{\theta}_c \mid \boldsymbol{\phi}_c, \Psi) \prod_{(x,y)\in D_k} p(y \mid x, \boldsymbol{\theta}_c, \Psi). \tag{6}$$

where $p(y \mid x, \boldsymbol{\theta}_c, \Psi)$ is the ordered-probit likelihood from Section 4.2. Variational inference gives $q_k(\boldsymbol{\theta}_c, \Psi) \approx \widetilde{p}_k(\boldsymbol{\theta}_c, \Psi)$. The historical posterior $q_t(\Psi)$ stays fixed within the candidate's validation, while acquired evidence updates q_k. Shared candidate response parameters allow an observation on one task to refine predictions for unevaluated tasks.

Online inference also conditions on available cached branch outcomes (Appendix A.1). To infer deployment value, we retain observed effects and jointly sample remaining effects under q_k, preserving cross-task dependence. Aggregating with the fixed weights approximates $p(\widehat{V}_c \mid D_k, w)$ for risk estimation and task selection.

4.4 Deployment-Risk-Directed Task Selection

DVI values a test by its expected reduction in deployment-decision loss. Given the deployment-value posterior, the current Bayes risk is the smaller expected loss of commit and preserve:

$$\rho(D_k) = \min_{a\in\{\text{COMMIT},\text{PRESERVE}\}} \mathbb{E}[\ell(a, \widehat{V}_c) \mid D_k]. \tag{7}$$

For an unevaluated task x, the response model predicts $p(y \mid x, D_k)$ for each paired outcome y. Each hypothetical observation updates the posterior as in Section 4.3. The Decision Value of Information (DVI) (Howard, 1966) is current risk minus the expected risk after this update:

$$\mathrm{DVI}(x \mid D_k) = \rho(D_k) - \sum_{y\in\{-1,0,+1\}} p(y \mid x, D_k)\, \rho(D_k \cup \{(x, y)\}). \tag{8}$$

When validation continues, BRACE selects the task with the largest estimated risk reduction:

$$x_{k+1} = \arg\max_{x\in\mathcal{U}_k} \widehat{\mathrm{DVI}}(x \mid D_k). \tag{9}$$

Its matched candidate–incumbent outcome is added to D_{k+1}, updating the posterior and subsequent testing priorities. Appendix A.2 describes posterior-sample estimation by calibrated reweighting or refitting and its computational cost.

4.5 Stopping and Deployment Decisions

DVI selects the next task, while current deployment risk determines when to stop. A nonnegative tolerance η, selected by independent calibration and fixed before evaluation, bounds the acceptable posterior expected loss. Checking at initialization and after each update gives the stopping rule

$$k^* = \min\{k \in \{0, \ldots, N\} : \rho(D_k) \le \eta \text{ or } \mathcal{U}_k = \varnothing\}. \tag{10}$$

At stopping, BRACE chooses the lower-loss action:

$$a^* = \arg\min_{a\in\{\text{COMMIT},\text{PRESERVE}\}} \mathbb{E}[\ell(a, \widehat{V}_c) \mid D_{k^*}]. \tag{11}$$

Commit applies the update, while preserve retains the incumbent and resolves ties. Exhausting the pool makes deployment value fully observed. In implementation, early stopping requires independent Monte Carlo confirmation. Appendix A.3 gives the numerical procedure and calibration protocol.

All acquired pairs enter the historical bank, and shared checkpoints refresh between candidates under that protocol. Appendix A.4 gives the state transition and complete procedure.

5 Experiments

We evaluate whether BRACE can make persistent-update validation more efficient while preserving deployed-agent quality.

5.1 Experimental Setup

Hosts and validation policies. We evaluate BRACE with SkillOpt (Yang et al., 2026) and EvoSkill (Alzubi et al., 2026) on SpreadsheetBench-Verified (Ma et al., 2024) and DocVQA (Mathew et al., 2021) using Qwen3.5-4B. Appendix D reports results for three additional model configurations. Native retains each host's validation and update procedure. Fixed panel reuses a fixed task subset, Random minibatch samples a new subset for each candidate, and DPP coreset selects tasks for difficulty and diversity. These alternatives and BRACE operate at the same host-specific validation boundary, with matched initial states, proposer configurations, and proposal counts. End-to-end candidates may diverge following different deployment decisions. Controlled replay instead fixes candidate–incumbent events to isolate validation effects.

Evaluation protocol. Each policy is evaluated in four independent campaigns of 100 host runs. Host state and rollout caches reset between runs, while BRACE's history and active checkpoint persist within each campaign. Validation uses a 40-task catalog. Panel budgets and BRACE's stopping tolerance are selected on separate calibration runs. We evaluate the final agent from run 100 of each campaign on 100 audit tasks and report mean success and sample standard deviation across campaigns. Audit outcomes are excluded from response-model training. Appendix B details the host adaptations, data splits, and implementation settings.

Metrics and cost. We report terminal-run success and amortized validation rollout episodes per host run. Spreadsheet success requires output-cell matches, and DocVQA success uses $\mathrm{ANLS} \geq 0.5$. One episode executes one agent version on one condition. Paired evaluation reuses cached branches. Cost includes history acquisition, activation, screening, retained host evaluation, fresh validation branches, and checkpoint checks over 100 runs. Configuration search is excluded for every policy.

5.2 Main Results: Quality and Validation Cost

BRACE achieves the highest mean terminal-run success and the lowest validation cost across both hosts and benchmarks (Table 1). Relative to DPP, it reduces amortized validation episodes by 24.9–40.1% under the accounting above. DPP already selects tasks for difficulty and diversity, so the comparison goes beyond replacing exhaustive evaluation with a smaller task set. BRACE uses fewer episodes with slightly higher mean success.

Table 1: Terminal-run success (four-campaign mean $\pm$ sample SD) and amortized validation cost over runs 1–100. Cost includes startup and checks but excludes configuration search for all policies. Appendices B.3 and C give denominators and the ledger.

		SpreadsheetBench		DocVQA	
Host	Deployment policy	Success (%)	Rollout ep./run	Success (%)	Rollout ep./run
SkillOpt	Native	17.50 ± 1.29	600.0	87.50 ± 2.08	600.0
SkillOpt	Fixed panel	18.25 ± 1.71	424.0	87.75 ± 2.06	424.0
SkillOpt	Random minibatch	18.50 ± 1.73	480.0	88.25 ± 1.71	480.0
SkillOpt	DPP coreset	18.75 ± 1.71	381.6	88.00 ± 1.41	381.6
SkillOpt	BRACE	19.00 ± 1.41	228.6	88.50 ± 1.29	264.6
EvoSkill	Native	18.50 ± 2.08	520.0	87.50 ± 1.91	520.0
EvoSkill	Fixed panel	19.25 ± 1.71	360.0	87.75 ± 1.71	360.0
EvoSkill	Random minibatch	19.25 ± 1.50	378.0	87.75 ± 1.89	381.0
EvoSkill	DPP coreset	19.50 ± 2.08	321.6	88.00 ± 1.63	321.6
EvoSkill	BRACE	20.00 ± 1.63	207.6	88.25 ± 1.71	241.6

Figure 3 extends the comparison beyond the selected configurations. Larger panels generally improve baseline success at higher cost. Across the plotted operating points, BRACE reaches comparable or higher success with fewer episodes, supporting candidate- and decision-specific evaluation rather than larger validation sets.

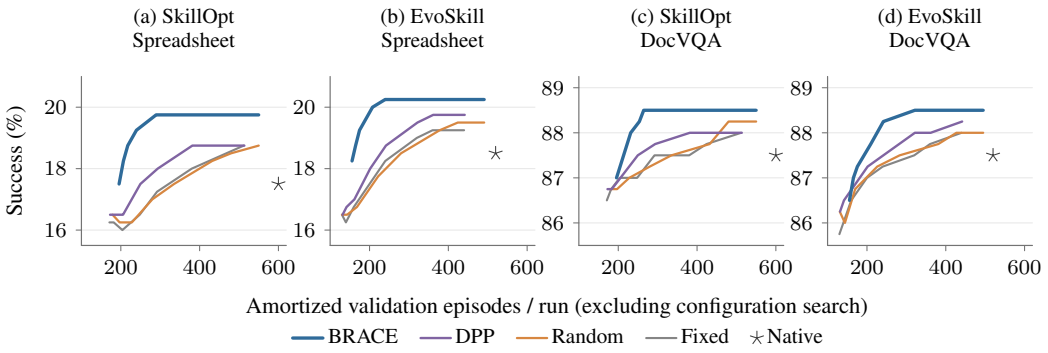

Figure 3: Terminal-run quality–cost trade-offs, excluding configuration search. Lines connect eight budget points for each panel policy and seven operating points for BRACE, including three supplementary measurements. Stars mark Native. Higher success at lower cost is better. Table 1 reports selected configurations and their success dispersion.

5.3 Cost Savings and Historical Reuse

BRACE reduces fresh online validation episodes by 54.0–82.3% relative to DPP (Figure 4). Retained host evaluation, startup, and checkpoint checks make the amortized reduction smaller, but BRACE retains its cost advantage over the 100-run horizon.

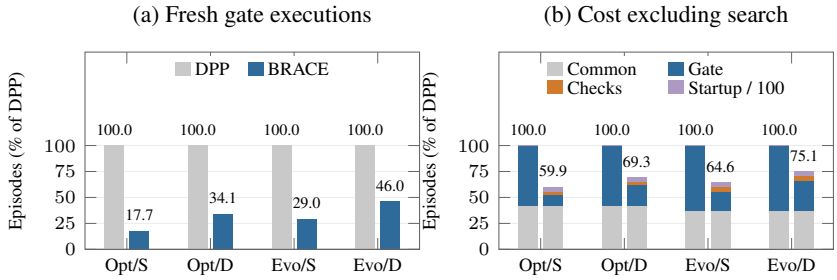

Figure 4: Validation costs relative to DPP (100%). Each pair compares DPP (left) with BRACE (right). (a) Newly executed validation branches. (b) Amortized cost excluding configuration search, split into retained evaluation/initialization (Common), new validation branches (Gate), checkpoint checks (Checks), and amortized startup (Startup/100). Opt/Evo denote SkillOpt/EvoSkill and S/D denote SpreadsheetBench/DocVQA. Labels show cost percentages, not savings.

BRACE reuses paired outcomes across candidates while charging history acquisition and checkpoint checks. Its recurring online savings exceed these additional costs over the reported horizon (Appendix C).

5.4 Decision-Directed Evidence Acquisition

We isolate task selection using four controlled cohorts of 40 candidates each. All selectors share the initial response model, posterior-update procedure, and terminal Bayes rule. Fixed-budget replay disables early stopping, so differences reflect which tasks are evaluated rather than how many evaluations are performed.

At ten paired evaluations, DVI reduces deployment-decision regret by 32.7% relative to Value-variance (Figure 5(a)). Value-variance seeks a more precise estimate of aggregate deployment value, whereas DVI prioritizes evidence expected to reduce the loss of an incorrect commit or preserve decision. The gap supports selecting tests for their consequences for the deployment action, rather than for predictive precision alone. DVI also achieves higher decision accuracy and fewer harmful commits at this budget (Appendix E.1), linking the regret reduction to improved deployment decisions.

Under a common posterior-risk tolerance of $\eta = .010$ and the same stopping-confirmation rule, DVI uses half as many pairs as Random, with both reporting mean realized regret of 0.70 percentage points (Table 9). Cohort-level regrets differ, so this agreement holds at the aggregate level. Prior-only decisions incur higher regret, showing that historical predictions do not replace online verification.

5.5 Response Transfer from Historical Updates

Figure 5(b) evaluates response modeling under identical random task prefixes on held-out candidates, separating prediction quality from task selection. Replacing candidate-conditioned initialization with Population prior increases deployment-value error even before testing, supporting the use of candidate information to distinguish the effects of different updates. Context-only retains history-trained semantic mappings but removes task-specific residuals. Its higher error indicates that task-specific historical responses provide information beyond these semantic mappings.

Independent effects isolates the contribution of cross-task posterior updates. It starts with the same predictions as BRACE and receives the same observations, but updates predictions only for tested tasks. After ten paired evaluations, its deployment-value MAE is 0.035, compared with BRACE's 0.021. Both variants incorporate observed outcomes into the aggregate value estimate. BRACE additionally uses the learned response structure to revise predictions for untested tasks, allowing the same evidence to improve the estimate of an update's overall effect. This comparison supports historical response transfer as a source of improved prediction beyond candidate-conditioned initialization. Together with the task-selection results, it identifies two complementary contributions to efficient verification: DVI determines which task to test, while the response model propagates the resulting evidence across tasks.

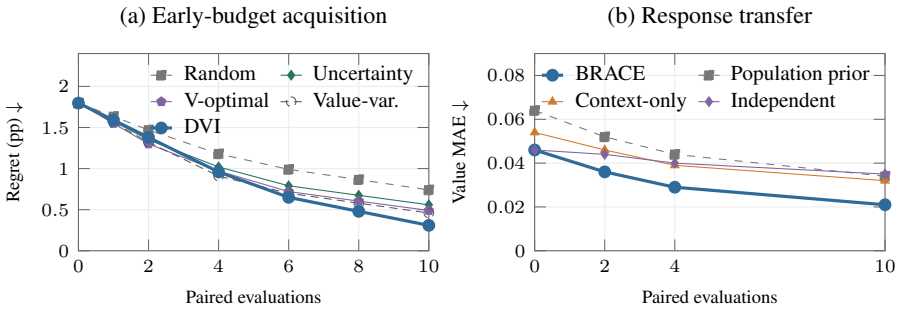

Figure 5: Task-selection and response-model comparisons across four controlled cohorts. (a) Task selectors share a predictor, with deployment-decision regret in percentage points. (b) Response-model variants use identical random task prefixes, with MAE for deployment value in $[-1, 1]$. Markers denote evaluated budgets. Ten pairs cover 25% of the catalog. Both metrics are lower-is-better.

6 Conclusion

We presented BRACE, a Bayesian approach to deployment-oriented update verification in self-evolving agents. BRACE transfers historical response structure to new candidates and selects paired tests by their expected reduction in deployment risk, with a calibrated posterior-risk threshold governing stopping. In SkillOpt and EvoSkill on SpreadsheetBench-Verified and DocVQA with Qwen3.5-4B, it reduces fresh online validation episodes by 54.0–82.3% relative to DPP coreset validation while achieving higher mean terminal-run task success. Controlled replay and ablations show that cross-task response transfer improves predictions for untested tasks and that decision-directed selection reaches comparable mean deployment-decision regret with fewer evaluations. Across four language-model configurations, BRACE also improves mean terminal-run success relative to host-native validation, with controlled replay supporting its acquisition efficiency across the evaluated models. Together, historical transfer and decision-directed testing reduce the evidence needed to decide which persistent updates to retain.

AI USE STATEMENT

We used generative AI tools to assist with language polishing, code development, and LaTeX formatting corrections. The authors reviewed the outputs and take full responsibility for the final content.

REPRODUCIBILITY STATEMENT

Appendix A describes the response model, posterior updates, DVI estimation, and stopping confirmation. Appendix B describes the host adaptations, data separation, and repeated-use evaluation protocol. Appendix C gives the validation-cost ledger, including initialization and checkpoint checks and excluding configuration search for all policies. Appendices D and E report evaluations across language models and controlled comparisons of task selection and response modeling.

REFERENCES

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Daniel Klein, Matei Zaharia, and Omar Khattab. GEPA: reflective prompt evolution can outperform reinforcement learning. *CoRR*, abs/2507.19457, 2025.

Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. Evoskill: Automated skill discovery for multi-agent systems, 2026. URL https://arxiv.org/abs/2603.02766.

Mengzhuo Chen, Junjie Wang, Zhe Liu, Yawen Wang, and Qing Wang. From failed trajectories to reliable LLM agents: Diagnosing and repairing harness flaws. *CoRR*, abs/2606.06324, 2026.

Daniel Golovin, Andreas Krause, and Debajyoti Ray. Near-optimal bayesian active learning with noisy observations. In *NIPS*, pp. 766–774, 2010.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *ICLR*, 2024.

Ronald A. Howard. Information value theory. *IEEE Trans. Syst. Sci. Cybern.*, 2(1):22–26, 1966.

Shervin Javdani, Yuxin Chen, Amin Karbasi, Andreas Krause, Drew Bagnell, and Siddhartha S. Srinivasa. Near optimal bayesian active learning for decision making. In *AISTATS*, volume 33 of *JMLR Workshop and Conference Proceedings*, pp. 430–438, 2014.

Justin Kay, Grant Van Horn, Subhransu Maji, Daniel Sheldon, and Sara Beery. Consensus-driven active model selection. In *ICCV*, pp. 4594–4604, 2025.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. Dspy: Compiling declarative language model calls into state-of-the-art pipelines. In *ICLR*, 2024.

Jannik Kossen, Sebastian Farquhar, Yarin Gal, and Tom Rainforth. Active testing: Sample-efficient model evaluation. In *ICML*, volume 139 of *PMLR*, pp. 5753–5763, 2021.

Andreas Krause and Carlos Guestrin. Optimal value of information in graphical models. *J. Artif. Intell. Res.*, 35:557–591, 2009.

Yang Li, Jie Ma, Miguel Ballesteros, Yassine Benajiba, and Graham Horwood. Active evaluation acquisition for efficient LLM benchmarking. In *ICML*, volume 267 of *PMLR*, 2025.

Jiahang Lin, Shichun Liu, Chengjun Pan, Lizhi Lin, Shihan Dou, Xuanjing Huang, Hang Yan, Zhenhua Han, and Tao Gui. Agentic harness engineering: Observability-driven automatic evolution of coding-agent harnesses. *CoRR*, abs/2604.25850, 2026.

Xiaotian Luo, Dizhan Xue, Fengxingyu Wang, Chuanrui Hu, and Yafeng Deng. Harnessbank: Semantic gene-bank search with gated verification for agent-harness self-evolution. *CoRR*, abs/2607.13683, 2026.

Zeyao Ma, Bohan Zhang, Jing Zhang, Jifan Yu, Xiaokang Zhang, Xiaohan Zhang, Sijia Luo, Xi Wang, and Jie Tang. Spreadsheetbench: Towards challenging real world spreadsheet manipulation. In *NeurIPS*, volume 37, 2024.

Bodhisattwa Prasad Majumder, Bhavana Dalvi Mishra, Peter A. Jansen, Oyvind Tafjord, Niket Tandon, Li Zhang, Chris Callison-Burch, and Peter Clark. CLIN: A continually learning language agent for rapid task adaptation and generalization. *CoRR*, abs/2310.10134, 2023.

Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. Docvqa: A dataset for VQA on document images. In *WACV*, pp. 2199–2208, 2021.

Richard D McKelvey and William Zavoina. A statistical model for the analysis of ordinal level dependent variables. *Journal of mathematical sociology*, 4(1):103–120, 1975.

Krista Opsahl-Ong, Michael J. Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. In *EMNLP*, pp. 9340–9366, 2024.

Wenbo Pan, Shujie Liu, Chin-Yew Lin, Jingying Zeng, Xianfeng Tang, Xiangyang Zhou, Yan Lu, and Xiaohua Jia. Evolving agents in the dark: Retrospective harness optimization via self-preference. *CoRR*, abs/2606.05922, 2026.

Ulrich Paquet, Blaise Thomson, and Ole Winther. A hierarchical model for ordinal matrix factorization. *Statistics and Computing*, 22(4):945–957, 2012.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *EMNLP*, pp. 7957–7968, 2023.

Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *NIPS*, volume 20, pp. 1257–1264, 2007.

Zayx Shawn. PACE: anytime-valid acceptance tests for self-evolving agents. *CoRR*, abs/2606.08106, 2026.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Trans. Mach. Learn. Res.*, 2024, 2024.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *ICML*, volume 267 of *PMLR*, 2025.

Skyler Wu, Yash Nair, and Emmanuel J. Candès. Efficient evaluation of LLM performance with statistical guarantees. *CoRR*, abs/2601.20251, 2026.

Jinghan Xu, Yikai Zhang, Aili Chen, Weiyuan Li, Jiaqing Liang, and Deqing Yang. Verify smarter, evolve further: Efficient harness evolution through behavior-aware verification. *arXiv preprint arXiv:2608.27311*, 2026.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *ICLR*, 2024.

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, Kai Qiu, Yuqing Yang, Dongdong Chen, Xue Yang, and Chong Luo. Skillopt: Executive strategy for self-evolving agent skills. *CoRR*, abs/2605.23904, 2026.

Mert Yüksekgönül, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative AI by backpropagating language model feedback. *Nat.*, 639(8055):609–616, 2025.

Hangfan Zhang, Shao Zhang, Kangcong Li, Chen Zhang, Yang Chen, Yiqun Zhang, Lei Bai, and Shuyue Hu. Self-harness: Harnesses that improve themselves. *CoRR*, abs/2606.09498, 2026.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. *CoRR*, abs/2510.04618, 2025.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: LLM agents are experiential learners. In *AAAI*, pp. 19632–19642, 2024.

Jin Peng Zhou, Christian K. Belardi, Ruihan Wu, Travis Zhang, Carla P. Gomes, Wen Sun, and Kilian Q. Weinberger. On speeding up language model evaluation. In *ICLR*, 2025.

A BRACE Inference and Validation Details

A.1 Response Model and Sparse-History Training

Candidate features encode the incumbent summary, proposed edit, and generating trajectory or feedback available before validation. Workload features encode task instructions and available metadata. Both use frozen BGE-large-en-v1.5 embeddings with 1,024 dimensions, a maximum input length of 512 tokens, CLS pooling, and L_2 normalization. Encoder checkpoints and input serialization are fixed before test evaluation, while $W_c, W_x, \mathbf{v}$ are learned from historical paired effects.

For the contextual coordinates in Equation (5), the workload residual priors are

$$\mathbf{r}_x \sim \mathcal{N}(\mathbf{0}, \sigma_r^2 I), \qquad \beta_x \sim \mathcal{N}(0, \sigma_b^2). \tag{A.1}$$

Unobserved conditions retain these residual priors. We parameterize $\mathbf{S}_0$ as a positive diagonal matrix, with $\mathbf{L}_0 = \mathbf{S}_0^{1/2}$. The shared quantities Ψ comprise context projections, workload residuals and biases, ordinal thresholds, and prior parameters. Historical candidate coordinates form Θ_{H}.

Write $m_{c,x} = b_c + b_x + \mathbf{z}_c^\top \mathbf{h}_x$. With $s_{c,x} = m_{c,x} + \varepsilon$, $\varepsilon \sim \mathcal{N}(0,1)$, the ordered-probit rule is

$$y_{c,x} = \begin{cases} -1, & s_{c,x} \le \tau_-, \\ 0, & \tau_- < s_{c,x} \le \tau_+, \\ +1, & s_{c,x} > \tau_+, \end{cases} \qquad \tau_- < \tau_+. \tag{A.2}$$

If Φ is the standard normal CDF, this gives

$$\begin{aligned} p(-1 \mid c, x) &= \Phi(\tau_- - m_{c,x}), \\ p(0 \mid c, x) &= \Phi(\tau_+ - m_{c,x}) - \Phi(\tau_- - m_{c,x}), \\ p(+1 \mid c, x) &= 1 - \Phi(\tau_+ - m_{c,x}). \end{aligned} \tag{A.3}$$

The unit noise variance fixes the latent scale. Thresholds are parameterized by a location and a positive gap. The unconstrained global vector γ contains the projections, threshold location, log gap, and log prior scales. Each entry of W_c, W_x, and $\mathbf{v}$ has a zero-mean Gaussian prior with standard deviation 0.5. The ordinal-threshold location has prior $\mathcal{N}(0,1)$. The log threshold gap, $\log \sigma_r$, $\log \sigma_b$, and the log candidate-prior standard deviations $\log \sqrt{(\mathbf{S}_0)_{jj}}$ have priors $\mathcal{N}(0, 0.5^2)$. These prior settings are fixed before evaluation.

Paired outcomes are conditionally independent given candidate coordinates and shared quantities Ψ. Their marginal dependence arises through these latent variables. Each pool entry identifies a task, initial settings, and a preassigned paired execution draw or seed configuration. Its outcome is revealed once and retained. Additional stochastic replicates receive separate condition identifiers and predeclared weights, so $\widehat{V}_c$ refers to the realized pool rather than an average over future repetitions.

Historical queries may depend on recorded contexts and previously observed outcomes. Conditional on this information, the query policy is assumed independent of unobserved outcomes and unknown response parameters. Every queried pair is retained irrespective of its result or the eventual acceptance decision. These conditions make the query factors ignorable in the conditional response likelihood. External logs with outcome-dependent retention require a selection model or are excluded from this training objective.

The active checkpoint uses a training history $\mathcal{H}_t^{\mathrm{fit}} \subseteq \mathcal{H}_t$ with observed index set Ω_t^{fit}. Held-out calibration records are excluded from this fit. The corresponding sparse-history posterior is

$$q_t(\Psi, \Theta_{\mathrm{H}}) \approx p(\Psi, \Theta_{\mathrm{H}} \mid \mathcal{H}_t^{\mathrm{fit}}) \propto p(\Psi, \Theta_{\mathrm{H}}) \prod_{(c,x)\in\Omega_t^{\mathrm{fit}}} p(y_{c,x} \mid \boldsymbol{\theta}_c, \Psi). \tag{A.4}$$

We fit the variational posterior by maximizing

$$\mathcal{L}(q) = \sum_{(c,x)\in\Omega_t^{\mathrm{fit}}} \mathbb{E}_q[\log p(y_{c,x} \mid \boldsymbol{\theta}_c, \Psi)] - \mathrm{KL}(q(\Psi, \Theta_{\mathrm{H}}) \,\|\, p(\Psi, \Theta_{\mathrm{H}}))\,. \tag{A.5}$$

Missing edges do not enter this objective. Minibatch likelihood estimates account for edge-sampling probabilities so that balanced sampling preserves the stated objective.

Variational representation. We use standardized latent coordinates to separate context predictions from candidate and task residuals:

$$\boldsymbol{\theta}_c = W_c \boldsymbol{\phi}_c + \mathbf{L}_0 \boldsymbol{\xi}_c, \qquad \mathbf{r}_x = \sigma_r \mathbf{e}_{x,1:d}, \qquad \beta_x = \sigma_b e_{x,d+1}, \tag{A.6}$$

where d is the response rank and $\boldsymbol{\xi}_c, \mathbf{e}_x \sim \mathcal{N}(\mathbf{0}, I)$. In these coordinates, the offline family is $q_t(\gamma)\prod_x q_t(\mathbf{e}_x)\prod_c q_t(\boldsymbol{\xi}_c)$, and the online family is $q_k(\gamma)\prod_x q_k(\mathbf{e}_x) q_k(\boldsymbol{\xi}_c)$. The global factor is diagonal Gaussian. Each candidate or task factor has a full covariance represented by a Cholesky factor with positive diagonal. Different residual blocks factorize, while the shared transformations induce dependence between candidate parameters and Ψ and across task predictions. The notation $q_k(\boldsymbol{\theta}_c, \Psi)$ denotes the induced joint distribution.

Initialization copies the stored historical factors and sets $q_0(\boldsymbol{\xi}_c) = \mathcal{N}(\mathbf{0}, I)$, recovering the context-informed prior exactly relative to q_t. A task without observed historical outcomes retains $q_t(\mathbf{e}_x) = \mathcal{N}(\mathbf{0}, I)$. Positive scales use exponential transforms, with log-scale parameters initialized at zero. Adam optimizes the reparameterized ELBO with response rank $d = 8$, $(\beta_1, \beta_2) = (0.9, 0.999)$, and $\epsilon = 10^{-8}$. Offline fitting uses 2,000 gradient steps at learning rate 10^{-3}. Each online update and hypothetical branch refit uses 200 steps at 3×10^{-4}. Each ELBO gradient estimate uses 16 Monte Carlo samples. Additional weight decay is zero, with regularization supplied by the prior and KL term. These settings are frozen before evaluation.

During online validation, $q_t(\Psi)$ remains the fixed historical reference. Each actual update fits q_{k+1} by minimizing $\mathrm{KL}(q \,\|\, \widetilde{p}_{k+1})$ within the online family, using all accumulated evidence in Equation (6). The previous fit provides a warm start. Each observation enters the likelihood once.

A hypothetical branch has the target $\widetilde{p}_{k,x,y} \propto \widetilde{p}_k\, p(y \mid x, \boldsymbol{\theta}_c, \Psi)$. Branch refreshes use the same historical reference, variational family, and fitting configuration as actual updates. Between candidates, the expanded historical objective counts each recorded pair once.

Conditioning on cached branch outcomes. Online inference conditions on any cached outcome matching the agent state, executor, task, and execution draw. An incumbent success restricts the paired effect to $\{-1, 0\}$, while an incumbent failure restricts it to $\{0, +1\}$. A known candidate success or failure gives the corresponding supports $\{0, +1\}$ or $\{-1, 0\}$. If both outcomes are available, their difference enters D as an observed effect without a new rollout. Single-sided results condition the predictive distribution but do not count as completed pairs. Predictive probabilities, branch likelihoods, deployment-value samples, and stopping confirmation use this same conditioning information, suppressed in the online notation. Infeasible effects have zero probability and are excluded from lookahead. Cache processing precedes the first risk check and is shared across selectors.

A.2 Posterior-Sample DVI Estimation

Draw S independent samples $(\boldsymbol{\theta}_c^{(s)}, \Psi^{(s)})$ from the current joint posterior q_k. Conditional on each sample, independently draw predictive outcomes $\widetilde{y}_{c,x}^{(s)}$ for unevaluated conditions. Their common latent variables preserve marginal cross-condition dependence. Current deployment-value samples are

$$\widehat{V}_c^{(s)} = \sum_{x\in\mathcal{O}_k} w_x y_{c,x} + \sum_{x\in\mathcal{U}_k} w_x \widetilde{y}_{c,x}^{(s)}, \qquad \widehat{\rho}_k = \min_a \frac{1}{S}\sum_{s=1}^{S} \ell(a, \widehat{V}_c^{(s)}). \tag{A.7}$$

For a hypothetical result y on condition x, reuse these latent samples with weights proportional to the likelihood of that result:

$$L_s^{x,y} = p(y \mid x, \boldsymbol{\theta}_c^{(s)}, \Psi^{(s)}), \qquad \widehat{p}(y \mid x, D_k) = \frac{1}{S}\sum_s L_s^{x,y}, \qquad \omega_s^{x,y} = \frac{L_s^{x,y}}{\sum_j L_j^{x,y}}. \tag{A.8}$$

In each value sample, replace the predicted outcome of the queried condition x with the hypothetical observation y:

$$\widehat{V}_c^{(s;x,y)} = \widehat{V}_c^{(s)} + w_x\big(y - \widetilde{y}_{c,x}^{(s)}\big), \qquad \widehat{\rho}_{x,y} = \min_a \sum_s \omega_s^{x,y}\ell(a, \widehat{V}_c^{(s;x,y)}). \tag{A.9}$$

Reweighting updates predictions for other conditions through the shared latent samples. Conditional independence permits reuse of their sampled outcomes while the queried outcome is fixed at y. The resulting branch risks give

$$\widehat{\mathrm{DVI}}(x \mid D_k) = \widehat{\rho}_k - \sum_{y\in\{-1,0,+1\}} \widehat{p}(y \mid x, D_k)\widehat{\rho}_{x,y}. \tag{A.10}$$

Branch approximation and refresh. Both scoring paths approximate the branch target $\widetilde{p}_{k,x,y}$ defined above. The fast path replaces $\widetilde{p}_k$ by q_k before likelihood reweighting. The refit path minimizes $\mathrm{KL}(q \,\|\, \widetilde{p}_{k,x,y})$ with the same family and iteration schedule as a real update. Likelihoods and weights are evaluated in log space, omitting zero-probability branches from the sum in Equation (A.10). If any positive-probability branch for task x has effective sample size $(\sum_s(\omega_s^{x,y})^2)^{-1} < \kappa S$, all feasible branches for that task use refitting. Their risks use new predictive samples with the queried outcome fixed at y. The pre-test probabilities $\widehat{p}(y \mid x, D_k)$ remain unchanged. Hypothetical fits never modify actual evidence or the historical checkpoint.

ESS measures weight concentration. We use $S = 4{,}096$ joint posterior samples for DVI scoring and an ESS threshold $\kappa = 0.5$. Configuration-selection replay compares fast scores with scores obtained by refitting all branches, recording top-task disagreement and the reference-score gap of the selected task. Tasks whose reference scores differ by at most 0.01η count as tied. Fast scoring requires a top-task disagreement rate no greater than 5% after accounting for these ties and a selected-task reference-score gap no greater than 0.05η. Otherwise all tasks use refit scoring. The mode and tolerances are frozen before the checkpoint check in Appendix A.3. Refit scores provide a numerical reference for variational approximation, while increasing S reduces sampling error.

Under consistent exact conditioning, $0 \le \mathrm{DVI}(x \mid D_k) \le \rho(D_k)$: retaining the current action bounds expected post-observation risk by current risk, and losses are nonnegative. Approximate estimates can be negative through sampling or posterior approximation. Material negative estimates trigger renewed inference and sampling for at most two retries, using 8,192 and 16,384 samples in sequence, after which the computed scores are used for ranking. Acquisition ties use catalog order. DVI is a one-step acquisition criterion, while total evaluation savings are assessed empirically.

For latent dimension d, evaluating pool likelihoods costs $O(S|\mathcal{U}_k|d)$ once workload coordinates are available. With predictive samples and weighted totals cached, scoring up to three outcomes and two actions costs $O(S|\mathcal{U}_k|)$ and uses $O(S|\mathcal{U}_k|)$ cache memory. If F_k tasks require refitting and one variational fit costs $C_{\mathrm{VI}}(k+1)$, branch fitting adds at most $3F_kC_{\mathrm{VI}}(k+1)$ plus predictive sampling. A real observation adds one fit, and confirmation uses $\sum_j S_j^{\mathrm{conf}}$ samples over the rounds attempted. Each selected logical pair compares an incumbent and a candidate outcome. Only uncached branches require new agent episodes, as specified in Appendix C.

A.3 Stopping Confirmation, Calibration, and Cost Accounting

Monte Carlo stopping confirmation. Equation (10) specifies the ideal posterior-risk rule. Its numerical implementation separates action selection from loss confirmation. Pilot samples give $\widehat{\rho}_k$ and an action $\widehat{a}_k$ minimizing sampled expected loss, with ties preserving the incumbent. If $\widehat{\rho}_k \le \eta$, freeze q_k and $\widehat{a}_k$ for confirmation. For each round $j \in \{1, \ldots, J\}$, draw a fresh batch of S_j^{conf} independent value samples from the predictive distribution induced by q_k, independently of the pilot

and earlier batches. With $L_{\max} = \max(\lambda_{\mathrm{H}}, \lambda_{\mathrm{M}})$, define

$$\begin{aligned} \overline{L}_{k,j} &= \frac{1}{S_j^{\mathrm{conf}}} \sum_{s=1}^{S_j^{\mathrm{conf}}} \ell(\widehat{a}_k, \widehat{V}_c^{(s;k,j)}), \\ U_{k,j} &= \min\left\{ L_{\max},\ \overline{L}_{k,j} + L_{\max}\sqrt{\frac{\log(NJ/\delta_{\mathrm{MC}})}{2S_j^{\mathrm{conf}}}} \right\}. \end{aligned} \tag{A.11}$$

Here $\delta_{\mathrm{MC}} \in (0,1)$ is a per-candidate numerical error tolerance. We fix $\delta_{\mathrm{MC}} = 0.01$, $J = 3$, and $(S_1^{\mathrm{conf}}, S_2^{\mathrm{conf}}, S_3^{\mathrm{conf}}) = (2^{16}, 2^{18}, 2^{20})$ before test evaluation. These are independent batch sizes rather than cumulative sample counts. Stop at the first confirmation with $U_{k,j} \le \eta$ and execute the same $\widehat{a}_k$. If all J rounds fail, acquire another pair. At pool exhaustion, select the exact full-pool action without Monte Carlo confirmation.

For $N = 40$ and unit loss coefficients, the smallest confirmation margin under this budget is approximately 0.00212. A smaller tolerance cannot pass confirmation before pool exhaustion, even when sampled loss is zero. The four BRACE tolerance-sweep settings reported in Appendix B.3 satisfy $\eta \ge 0.005$.

Because $\widehat{V}_c \in [-1, 1]$, losses lie in $[0, L_{\max}]$. Conditional on the pilot and current evidence, the one-sided bounded-sample inequality gives

$$\Pr_{\mathrm{MC}}\Big\{ \mathbb{E}_{q_k}[\ell(\widehat{a}_k, \widehat{V}_c)] > U_{k,j} \Big\} \le \frac{\delta_{\mathrm{MC}}}{NJ}.$$

There are at most N non-exhausted decision states and J confirmation rounds per state. A union bound therefore limits erroneous confirmation of an action with q_k-expected loss above η to probability δ_{MC} per candidate. This controls independent Monte Carlo sampling error relative to q_k. Model misspecification and variational error are evaluated through predictive calibration. The confirmed action approximates the terminal Bayes action in Equation (11) and is not reselected using the confirmation samples.

Calibration and activation. Complete host trajectories are assigned to history, calibration, and evaluation splits. Within each trajectory, replay preserves time order and uses only records available before the current candidate. Calibration trajectories are partitioned into configuration-selection and checkpoint-check blocks. A checkpoint is fitted without the candidates in its check block. Replay reveals only outcomes selected by the validation policy. We use fully evaluated reference outcomes to compute realized regret, class-conditional predictive log loss, and deployment-value interval coverage. Each class-specific log loss averages $-\log p(y \mid x, D_k)$ over unevaluated reference outcomes of that class at declared replay states. Predictive diagnostics average over declared states within each candidate, then give equal weight to candidates within a trajectory and to trajectories. Separate deployment tasks and execution draws measure deployed-skill success.

The loss coefficients, response configuration, and numerical settings are fixed before selecting η. For each host–benchmark pair, configuration-selection runs choose the lowest-cost tolerance meeting a common deployed-skill success target. The chosen tolerance is then checked on the separate checkpoint-check block, including numerical stopping confirmation. Each block contains one candidate from each of four independent source trajectories. Activation requires at least ten observed outcomes in each class, represented in at least two candidates. Class-conditional predictive log loss must not exceed $\log 3$ nats for any outcome class. Empirical coverage of nominal 90% deployment-value intervals must be at least 80%, using the replay-state and trajectory aggregation described above. Mean deployment-decision regret must not exceed 0.01 (one percentage point), and none of the four checkpoint-check candidates may incur a harmful commit. Deployment success must meet the host–benchmark target in Table 4. Harmful-commit frequency counts harmful commits among all evaluated candidates. These are empirical activation criteria. The experimental allocation of history and checkpoint checks is given in Appendices B.2 and C.2. Table 10 reports the selected tolerances. The posterior threshold η is measured in expected-loss units.

Before BRACE is activated, the host's existing validator decides whether to accept each update, while we collect stratified paired observations under matched conditions to build the historical bank. If no setting passes the checkpoint check, host validation continues. New paired records

are retained regardless of acceptance. History refitting occurs between candidates, but an updated checkpoint replaces the active model only after passing the same criteria on a fresh check block that has not been used to select or fit it. Refresh checkpoints and check-block allocation are declared in advance. If no fresh block is available, the active checkpoint is retained. Changes to the host executor or workload definition return decisions to host validation until a new checkpoint passes. Paired observations collected during online validation can be used for later model training under this refresh schedule. Unqueried reference outcomes and calibration or final deployment labels remain excluded. A zero-pair decision at $D_0 = \varnothing$ also requires Monte Carlo confirmation.

Current risk, rather than marginal DVI, governs stopping. If risk remains above η, acquisition continues even when one-step DVI is small. At pool exhaustion, all components of $\widehat{V}_c$ are observed and its Bayes risk is zero. Repeated execution draws assess the stability of the finite-pool reference action. Disjoint deployment workloads assess its relation to future agent performance.

Cost accounting. System-level validation cost counts newly executed rollout episodes, separating startup, retained host evaluation, fresh candidate–incumbent branches, and checkpoint checks. Configuration-search costs are excluded for all policies. Startup retains historical-data acquisition, activation, and screening. Reusing historical records or cached branch outcomes incurs no new rollout. Appendix C defines these components and their amortization over 100 host runs. Research-only audit and full-catalog replay episodes are outside this validation metric. Monte Carlo stopping confirmation requires model inference but no agent episodes, whereas collecting checkpoint-check labels incurs rollout cost. System comparisons report terminal-run task success and amortized validation episodes at calibration-selected operating points. Replay reports realized deployment-decision regret and acquired logical pairs.

Acquisition comparisons. Historical low-rank predictions support active performance estimation (Wu et al., 2026) and model selection (Zhou et al., 2025). To isolate the acquisition objective, we specify two controls with the same paired-effect posterior and stopping rule: expected reduction in deployment-value variance, and expected reduction in the entropy of the full-information action $\mathbf{1}\{\widehat{V}_c > 0\}$. The second follows the decision-identity objective of active model selection (Kay et al., 2025). DVI instead accounts for the magnitude and asymmetry of wrong-action losses. These are objective-level ablations, distinct from reproducing the complete published methods.

A.4 Complete Online Procedure

Algorithm 1 applies after activation. The active historical posterior is fixed within each candidate's validation. Updated checkpoints follow the promotion rule in Appendix A.3. DVI estimates use the checkpoint's calibrated scoring mode from Appendix A.2, and stopping confirmation uses Equation (A.11).

The selected action determines the next persistent state:

$$K_{t+1} = \begin{cases} K_t \oplus u_t, & a^* = \textsc{commit}, \\ K_t, & a^* = \textsc{preserve}. \end{cases} \tag{A.12}$$

Algorithm 1 BRACE validation of a new candidate after activation

Require: $c = (K_t, u_t)$, history $\mathcal{H}_t$, posterior $q_t(\Psi)$, pool $\mathcal{X}_t$, weights w, losses $(\lambda_{\mathrm{H}}, \lambda_{\mathrm{M}})$, tolerance η, confirmation settings $(J, \{S_j^{\mathrm{conf}}\}, \delta_{\mathrm{MC}})$

Encode contexts and initialize the context-informed joint prior
Initialize D from cached complete pairs and $\mathcal{U}$ from the remaining tasks
Condition on D and available single-sided outcomes, set $a^* \leftarrow \varnothing$
while $\mathcal{U} \neq \varnothing$ **do**
    Draw pilot samples; estimate $\widehat{\rho}(D)$ and minimizing action $\widehat{a}$
    if $\widehat{\rho}(D) \leq \eta$ **then**
        for $j = 1, \ldots, J$ **do**
            Draw fresh confirmation samples; compute $U_{k,j}$ with $k = |D|$
            if $U_{k,j} \leq \eta$ **then**
                $a^* \leftarrow \widehat{a}$; **break**
            end if
        end for
        if $a^* \neq \varnothing$ **then**
            break
        end if
    end if
    Estimate $\widehat{\mathrm{DVI}}(x \mid D)$ for every $x \in \mathcal{U}$
    $x^* \leftarrow \arg\max_{x \in \mathcal{U}} \widehat{\mathrm{DVI}}(x \mid D)$
    Execute uncached matched branches and observe y_{c,x^*}
    $D \leftarrow D \cup \{(x^*, y_{c,x^*})\}$; $\mathcal{U} \leftarrow \mathcal{U} \setminus \{x^*\}$
    Update current joint posterior using Equation (6)
end while
if $\mathcal{U} = \varnothing$ **then**
    Choose a^* from the fully observed $\widehat{V}_c$; preserve on a tie
end if
Apply the corresponding persistent-state transition
Append $\{(c, x, y) : (x, y) \in D\}$ to $\mathcal{H}_t$
Refit history between candidates and promote checkpoints under the calibration rule
return action and acquired-pair count $|D|$

B Experimental Protocol

B.1 Hosts, Scorers, and Validation Policies

BRACE replaces SkillOpt's acceptance gate (Yang et al., 2026). The EvoSkill adaptation (Alzubi et al., 2026) retains a capacity-three frontier, skill-only mutation, and round-robin parent selection following its main algorithm. A proposal is evaluated for admission against the weakest retained program, which need not be its mutation parent. Native retains the validation and admission procedure of each host. Each alternative replaces the evidence policy at the same decision boundary while keeping the host's proposal and update operations.

Both hosts use Qwen3.5-4B with common benchmark adapters: output-cell match on SpreadsheetBench and binary success at ANLS ≥ 0.5 on DocVQA. EvoSkill originally used Claude Code with Opus 4.5 on OfficeQA and SealQA. OfficeQA used 17 validation examples, training subsets of 12/24/36 examples, and 1.5 training epochs. Test results included exact-match accuracy. Each original configuration had one run. Our adaptation retains frontier-based selection with a common 40-task catalog, a fixed proposal count, and four repeated-use campaigns. It is a cross-benchmark adaptation, rather than a replication of those scores or epoch budgets.

Fixed panel draws a uniform task permutation once and reuses its budget prefix across candidates, following fixed-task evaluation (Lin et al., 2026). Random minibatch draws a new uniform permutation for each candidate and evaluates its prefix without replacement (Opsahl-Ong et al., 2024). DPP coreset combines difficulty and diversity from prior incumbent failure fingerprints (Pan et al., 2026), using normalized BGE-large-en-v1.5 embeddings, $\theta = 0.7$, score floor 0.1, and greedy threshold

10^{-10}. Its ordering is fixed before revealing current-candidate outcomes. Screening executions enter startup cost. Nested prefixes define the panel budget sweeps.

Panels select the candidate when the observed mean paired effect is positive and preserve the reference in ties or with empty panels. BRACE minimizes posterior expected deployment loss, with ties preserved. For EvoSkill, the action replaces or retains the frozen weakest frontier member. The evaluator returns a score on the common validation scale: the stored reference score plus the estimated paired-value change, using the panel mean or BRACE posterior mean. Programs are then reranked, and the highest-scoring retained program is the final endpoint. The test set never selects this endpoint.

Adaptive validation stops when independent numerical confirmation meets the posterior-risk tolerance η, or the catalog is exhausted. Fixed-budget replay disables early stopping and shares the posterior update and terminal Bayes rule across selectors.

In the current SkillOpt adapter, the Native gate evaluates the designated validation catalog. Proposal, execution, and audit random keys are matched independently of acquisition order. Exhaustive SkillOpt adapters therefore reproduce its actions under matched randomness. This is specific to the SkillOpt gate rather than the definition of Native. Partial validators expose their acquired evidence. End-to-end results measure the complete validation-and-evolution trajectory, while frozen-candidate replay isolates acquisition under a shared posterior and action rule.

B.2 Growing History and Temporal Isolation

Initial fitting, configuration selection, checkpoint checks, and evaluation use separate host streams. Workload-validation and audit tasks are separated from proposal tasks, grouping related workbook instances or document images before splitting. Each policy has four independent campaigns of 100 host runs. A run starts from the common initial skill and an empty rollout cache. BRACE carries its historical bank and response checkpoint between runs while host skills, frontier, and rollout caches reset. Campaigns have separate random streams and histories, with matched task schedules across policies.

After a candidate decision, BRACE appends all acquired paired outcomes, whether the candidate was committed or preserved. Unobserved outcomes remain missing. Between candidates, scheduled shared-model refreshes fit the accumulated bank. A new checkpoint becomes active after passing a fresh held-out check block. Failed refresh checks retain the existing checkpoint. Recording a purchased pair, refitting the model, and collecting check labels are separate operations.

Within one candidate, the historical reference posterior stays fixed while the current joint posterior over candidate and shared parameters incorporates new evidence. Replay restores the checkpoint and context available immediately before that candidate. All selectors receive this same starting information and maintain separate local replay states. Later events may start from larger histories and newer checkpoints. Future records, unqueried exhaustive references, check labels, and final audit labels are excluded from the fitting bank.

Let $s_{\pi,r} = 100k_{\pi,r,100}/100$ be the endpoint success rate of campaign r. The main table reports $\bar{s}_\pi \pm \mathrm{SD}(s_{\pi,1}, \ldots, s_{\pi,R})$. Comparisons with Native use matched campaign differences $d_r = s_{\mathrm{BRACE},r} - s_{\mathrm{Native},r}$. Any interval analysis resamples campaigns as paired clusters rather than treating the 400 task outcomes as independent runs. With four campaigns, mean $\pm$ standard deviation is descriptive. Table 3 records the individual campaign endpoint counts.

The main Qwen3.5-4B experiments disable thinking and use temperature 0.7, top-p 0.8, top-k 20, min-p 0, presence penalty 1.5, and repetition penalty 1.0, with at most 8,192 output tokens per model call. Validation policies share the same decoding configuration, tool environment, and execution limits within each executor-model setting. The four campaign root seeds are $\{0, 1, 2, 3\}$, and numerical replay indices are $0, \ldots, 9$. Computation-specific substreams separate proposal generation, execution, fitting, acquisition scoring, and independent stopping confirmation. Appendix A gives optimization and numerical-inference settings.

B.3 Denominators and Operating-Point Selection

Each system configuration contributes the run-100 endpoint from each of four independent campaigns, scored on 100 binary audit tasks per endpoint. Success is $100 \sum_{r=1}^{4} k_{r,100}/400$ with integer $k_{r,100}$.

Table 2: Experimental configuration.

Setting	Configuration
Validation catalog	40 tasks with one matched trial per task
Main Qwen3.5-4B opportunities	SkillOpt: 10 candidate decisions. EvoSkill: 2 warm-up + 10 competitive decisions
Cross-model end-to-end protocol	10 competitive proposal attempts after host initialization. V/A excludes warm-up proposals. Invalid or unchanged proposals are not retried.
Evaluation campaigns	4 independent campaigns $\times$ 100 host runs
Audit success	Run-100 endpoint of each campaign (4×100 binary tasks)
Calibration success	4 separate single-run endpoints $\times$ 100 tasks/configuration
Diagnostic cohort	4 controlled cohorts, 40 candidates each, 10 replays
Panel pair-budget grid	$\{1, 2, 4, 8, 12, 20, 24, 32, 40\}$
Workload and deployment loss	Uniform weights, $\lambda_{\mathrm{H}} = \lambda_{\mathrm{M}} = 1$
Adaptive replay	Common tolerance $\eta = .010$
History diagnostic default	40 source runs, one candidate/run, 25% row density, rank 8
Cost horizon and refreshes	$M = 100$ host runs with checks after runs 40 and 80

One endpoint has 1-point resolution, compared with 0.25-point resolution for the four-endpoint mean. This terminal score accompanies campaign-average cost.

Sample standard deviations need not lie on these binary-success grids, and displayed adaptive costs are rounded to one decimal place.

Table 3: Successes at four campaign endpoints, each evaluated on 100 audit tasks, underlying Table 1.

Setting	Policy	C1	C2	C3	C4	Mean $\pm$ SD
Opt/S	Native	16	17	18	19	17.50 ± 1.29
Opt/S	Fixed panel	16	18	19	20	18.25 ± 1.71
Opt/S	Random minibatch	17	18	18	21	18.50 ± 1.73
Opt/S	DPP coreset	17	18	19	21	18.75 ± 1.71
Opt/S	BRACE	18	18	19	21	19.00 ± 1.41
Opt/D	Native	85	87	88	90	87.50 ± 2.08
Opt/D	Fixed panel	86	86	89	90	87.75 ± 2.06
Opt/D	Random minibatch	86	88	89	90	88.25 ± 1.71
Opt/D	DPP coreset	86	88	89	89	88.00 ± 1.41
Opt/D	BRACE	87	88	89	90	88.50 ± 1.29
Evo/S	Native	16	18	19	21	18.50 ± 2.08
Evo/S	Fixed panel	17	19	20	21	19.25 ± 1.71
Evo/S	Random minibatch	18	18	20	21	19.25 ± 1.50
Evo/S	DPP coreset	17	19	20	22	19.50 ± 2.08
Evo/S	BRACE	18	20	20	22	20.00 ± 1.63
Evo/D	Native	85	87	89	89	87.50 ± 1.91
Evo/D	Fixed panel	86	87	88	90	87.75 ± 1.71
Evo/D	Random minibatch	85	88	89	89	87.75 ± 1.89
Evo/D	DPP coreset	86	88	88	90	88.00 ± 1.63
Evo/D	BRACE	86	88	89	90	88.25 ± 1.71

The mechanism comparisons use four controlled cohorts of 40 candidates each, distinct from host-level endpoints. Each candidate has ten numerical/acquisition replays with its initial checkpoint, q_0, prior-only action, and paired outcomes fixed. Seeds vary stochastic selection, fitting, predictive sampling, acquisition scoring, and stopping confirmation. Substreams keyed by candidate, replicate, evidence set, and computation role share random draws whenever selectors reach the same state. Acquisition ties follow catalog order.

Across settings, 1,600 candidate–replay decisions give frequency resolution of 0.0625 points and, with 40 uniform conditions and unit loss, regret resolution of 0.0015625 points. They are repeated evaluations of 160 candidates. DVI's ten-pair aggregate encodes 1,482 reference-action agreements, 50 harmful commits, and 198 catalog-loss units: 92.625%, 3.125%, and 0.309375 points before rounding. The fixed zero-evidence actions instead have pooled regret resolution of 0.015625 points.

Their regret is 1.796875, displayed as 1.80. Posterior losses and value MAE are continuous predictions and do not follow the binary count grid.

Calibration selects the least-cost configuration reaching the declared success target, breaking cost ties by higher calibration success and then grid order. Both hosts use eight non-exhaustive panel budgets and five BRACE tolerances. Figure 3 displays seven BRACE operating points per host–benchmark pair. The third through sixth points, ordered by cost, correspond to $\eta = 0.030, 0.020, 0.010, 0.005$. Three supplementary measurements extend the displayed cost range. Forced Prior-only decisions and the 0.002 setting are excluded from the four-tolerance sweep. The latter is below the numerical confirmation floor in Appendix A.3. Configuration-search expenditures are excluded from the reported cost metric for every policy. The four displayed tolerances are a subset of the search grid, not a claim that only four configurations were considered. Prior-only and exhaustive replacement policies are diagnostic references rather than Native definitions. A policy with no qualifying configuration reports its highest-calibration-score configuration, with cost breaking ties. Calibration and audit counts come from separate runs.

Table 4: Calibration success (%) and preset quality targets. Targets are selection thresholds. Achieved success follows the 0.25-point count grid.

Setting	Target	Native	Least-cost panel	Panel	BRACE
SkillOpt–Spreadsheet	18.4	17.50	DPP	18.75	18.50
SkillOpt–DocVQA	87.8	87.50	DPP	88.25	88.50
EvoSkill–Spreadsheet	19.25	18.25	DPP	19.25	19.50
EvoSkill–DocVQA	87.75	87.25	DPP	88.00	88.00

Differences and savings are calculated before display rounding. Mechanism diagnostics resample source streams, retaining the dependence among their candidates and numerical replays. Aggregate totals alone do not determine these intervals.

Full-catalog replay reference. Frozen-candidate replay evaluates every catalog condition to define the reference deployment value and action used by regret and decision-accuracy metrics. This reference is computed after candidate generation, is not an end-to-end validation policy, and is not labeled Native. SkillOpt's exhaustive replacement adapters can coincide with its Native gate under matched randomness, while EvoSkill Native remains its host-specific frontier evaluator.

B.4 Motivating Replay Protocol

The motivating study examines task selection on SkillOpt–SpreadsheetBench candidate events. Its replay protocol uses a separate bank of earlier host runs for historical priorities. The evaluated events remain grouped by originating evolution run, and the history available to an event excludes that event, later events, and their outcomes. Model settings and ranking hyperparameters are selected on separate calibration runs.

Locating changed tasks. For the fixed-ranking comparison, Random uses a uniform task permutation. Global historical priorities rank tasks by their historical fraction of nonzero paired outcomes, shrunk toward the pooled change frequency with four pseudo-observations. Candidate-related priorities use the same records but weight the eight nearest earlier candidates by exponentiated cosine similarity of their pre-validation context embeddings, with temperature 0.1. For each task, weights $\widetilde{w}_i$ are normalized over neighbors with observed outcomes. With effective neighbor count $n_{\text{eff}} = 1/\sum_i \widetilde{w}_i^2$, the weighted change frequency p_{local} is shrunk toward global priority p_{global} as $(n_{\text{eff}} p_{\text{local}} + 2p_{\text{global}})/(n_{\text{eff}} + 2)$. These ranking settings are fixed on calibration data. Missing task outcomes contribute no observations, and unsupported neighborhoods use global priorities. Rankings are frozen before revealing any current-event outcome. Changed-task recall divides discovered nonzero effects by all nonzero effects in the reference row. Events with no changed tasks are excluded from recall.

From change discovery to decision quality. The sequential comparison holds the contextual response predictor, historical checkpoint, target weights, and terminal Bayes action fixed. Random

acquires uniformly, change-first selects the largest posterior probability of a nonzero paired effect, and decision-directed selection uses DVI. All three update the same model family from their acquired outcomes and are evaluated at matched evidence counts with early stopping disabled. Main-study weights are uniform and losses symmetric. Regret is the loss relative to the full-catalog action, multiplied by 100 for percentage-point presentation. We first average results across task-selection seeds for each candidate event, then average across events from the same evolution run. The fixed-ranking recall comparison and sequential decision comparison use separately logged task orders.

C Validation Cost and Historical Reuse

C.1 Episode Accounting

A new branch execution costs one episode. A logical pair costs two if both branches are new, one if one is cached, and zero if both are cached. Cache eligibility requires identical agent state, executor configuration, task, matched trial, and environment initialization. All policies use this rule. Missing historical outcomes do not become cached observations.

For policy π, the reported validation cost over M subsequent host runs, excluding configuration search, is

$$C_{\pi,r}(M) = C_{\pi,r}^{\text{start}} + \sum_{j=1}^{M} \left(C_{\pi,r,j}^{\text{retained}} + C_{\pi,r,j}^{\text{gate}} + C_{\pi,r,j}^{\text{check}} \right). \tag{C.13}$$

Here r indexes an independent campaign. Table 1 reports $\sum_{r=1}^{4} C_{\pi,r}(100)/(4 \times 100)$. Ledger entries are campaign means, with startup divided by 100. Startup covers newly collected history, activation, and screening. Recurring terms count retained host assessment, fresh gate branches, and checkpoint checks. Configuration-search executions are excluded for all policies, so this metric does not measure the full expense of finding a configuration. Proposal-generation computation and research-only audit or exhaustive-reference episodes are also outside this validation metric.

The retained one-time startup term is

$$C_{\pi,r}^{\text{start}} = C_{\pi,r}^{\text{history}} + C_{\pi,r}^{\text{activation}} + C_{\pi,r}^{\text{screening}}. \tag{C.14}$$

The same exclusion applies to Fixed, Random, DPP, and BRACE. The comparison retains their selected operating points and measures rollout demand under those configurations.

SkillOpt's ten candidates on the 40-task catalog require 400 candidate executions plus 40 initial-incumbent executions under exhaustive validation. Each run starts without a prefilled incumbent cache. Fixed panel and DPP reuse their fixed supports, while Random and BRACE may query different tasks across candidates. Random uses 9, 17, and 28 fresh incumbent episodes at budgets 1, 2, and 4, respectively. SkillOpt retains another 160 episodes, split equally between slow-update comparisons and guidance validation.

EvoSkill's adapted Native procedure initializes a three-program frontier and then applies its retained evaluator to ten competitive proposals. The ledger records 120 episodes for frontier initialization and 400 for the ten proposal evaluations, for 520 episodes per run. Each run therefore contains twelve generated children. This accounting describes the common benchmark adapter rather than defining Native as an exhaustive commit/preserve rule.

$$C_{\text{Evo,Native}} = 3N + PN,$$

$$C_{\text{Evo},\pi} = 3N + C_{\pi}^{\text{start}}/M + C_{\pi}^{\text{check}} + \sum_{i=1}^{P} (n_{\pi i}^{\text{cand}} + n_{\pi i}^{\text{ref}}).$$

Here $N = 40$, $P = 10$, and $M = 100$. Fixed and DPP reuse the same support within a run, so admitted-program outcomes are available for later comparisons. Random and BRACE can query new conditions for the reference and pay for those cache misses. BRACE's selected candidate/reference counts are 40/18 on SpreadsheetBench and 60/32 on DocVQA. Including initialization, startup and checks gives 207.60 and 241.60 episodes. Training rollouts used to propose skills are outside this validation metric for every policy.

Figure 3 and Table 1 use the same amortized-validation-episodes-per-run metric, excluding configuration search. Common evaluation is 160 retained host episodes for SkillOpt and 120 frontier-initialization episodes for EvoSkill. All fresh comparison branches, including reference-side cache misses, enter the ledger.

Table 5: Validation episode ledger excluding configuration search for all policies. Cost/run = Startup$/100$ + Common + Cand. + Ref. + checkpoint checks (14.40 episodes/run for BRACE, zero otherwise). Common is retained assessment for SkillOpt or frontier initialization for EvoSkill. Recurring columns are per-run means.

Setting	Policy	Startup	Common	Cand.	Ref.	Cost/run
SkillOpt–Spreadsheet	Native	0	160	400	40	600.00
SkillOpt–Spreadsheet	DPP	160	160	200	20	381.60
SkillOpt–Spreadsheet	BRACE	1,520	160	30	9	228.60
SkillOpt–DocVQA	Native	0	160	400	40	600.00
SkillOpt–DocVQA	DPP	160	160	200	20	381.60
SkillOpt–DocVQA	BRACE	1,520	160	60	15	264.60
EvoSkill–Spreadsheet	Native	0	120	400	0	520.00
EvoSkill–Spreadsheet	DPP	160	120	200	0	321.60
EvoSkill–Spreadsheet	BRACE	1,520	120	40	18	207.60
EvoSkill–DocVQA	Native	0	120	400	0	520.00
EvoSkill–DocVQA	DPP	160	120	200	0	321.60
EvoSkill–DocVQA	BRACE	1,520	120	60	32	241.60

C.2 Startup, Refreshes, and Reuse

The initial sparse bank contains 400 paired records, charged as 800 fresh episodes for each host–benchmark setting. DPP screening purchases four 40-episode blocks. Each eligible configuration uses four isolated single-run calibration trials, and BRACE starts each from its initial bank. Calibration histories are discarded, distinct configurations use distinct trial keys, and the selected tolerance stays fixed during the campaign. These configuration-selection trials are outside the reported cost metric.

A checkpoint check uses predeclared terminal decision events from four independent held-out source runs. Each event receives 40 full-catalog pairs for regret, harmful-commit, predictive log-loss, and interval-coverage checks. A separate 100-task evaluation scores the resulting run endpoint: SkillOpt's final committed or preserved skill, or EvoSkill's highest-scoring retained program after terminal admission and reranking. The mean of these four endpoint success rates is compared with the same host–benchmark target used in configuration selection. Check-block labeling costs $4(2 \times 40 + 100) = 720$ episodes without cache discounts. Failed initial activation retains host validation, while a failed refresh retains the active checkpoint. All attempted check blocks are charged.

The ledger charges one initial activation block and two refresh blocks, after runs 40 and 80. Initial activation therefore contributes 720 startup episodes and refreshes contribute $2 \times 720/100 = 14.40$ episodes per run. Ordinary bank insertion never triggers an automatic exhaustive reevaluation. Reusing records, posterior computation, and model fitting add no agent episodes, but fresh check labels do.

Under the reported metric, Fixed and Random have no startup episodes, DPP has 160 screening episodes, and BRACE has 800 history plus 720 activation episodes. Configuration selection is excluded uniformly, while subsequent checkpoint checks remain charged.

Within each campaign, histories grow through runs 1–100 and shared checkpoints can refresh after runs 40 and 80. The terminal score therefore uses the checkpoint and history available in run 100. Recurring gate counts average over the entire campaign, allowing acquisition effort to change with history. Each campaign pays its own startup and every fresh execution. Its bank and checkpoint remain private. The four endpoint scores and four campaign cost totals define the joint reporting horizon of Table 1.

Cost interpretation. BRACE uses 24.9–40.1% fewer amortized validation episodes than DPP under this accounting. For SkillOpt on SpreadsheetBench, the costs are 228.60 versus 381.60 episodes per run, a 40.1% reduction. History acquisition, activation, screening, retained host evaluation, fresh comparison branches, and checkpoint checks remain charged. The comparison measures rollout execution demand at selected configurations, rather than configuration-search expense or verifier inference time.

D Evaluation Across Language Models

The main table fixes Qwen3.5-4B and compares host-native and replacement validation policies. We use two complementary protocols to evaluate BRACE across four executor-model configurations. The end-to-end protocol fixes ten competitive proposal attempts after host-specific initialization and does not retry an invalid or unchanged proposal. EvoSkill's two warm-up proposals initialize its three-program frontier and are outside this ten-attempt budget. Candidate yield and accepted updates count competitive proposals. The controlled replay protocol instead freezes the candidate–incumbent events, giving every selector the same evidence states and full-catalog references. The first protocol measures the realized evolution process, and the second isolates validation efficiency.

Table 6: Cross-model end-to-end comparison under ten competitive proposal attempts after initialization. "V/A" reports the campaign-mean numbers of valid and accepted competitive candidates, excluding warm-up proposals. Success is the terminal audit rate. The Qwen3.5-4B success entries reproduce Table 1, which reports campaign dispersion. This table reports quality and candidate yield, without a cross-model rollout-cost comparison.

			SpreadsheetBench		DocVQA	
Model	Host	Policy	V/A	Success (%)	V/A	Success (%)
Llama-4 Scout INT4	SkillOpt	Native	8/2	15.75	8.5/2.25	86.75
	SkillOpt	BRACE	8.25/2.75	16.75	8.5/3	87.25
	EvoSkill	Native	8/2.5	16.75	8.5/2.75	86.75
	EvoSkill	BRACE	8.25/3	17.75	8.5/3.25	87.50
Qwen3.5-4B	SkillOpt	Native	10/2.5	17.50	10/3	87.50
	SkillOpt	BRACE	10/3.5	19.00	10/3.5	88.50
	EvoSkill	Native	10/3	18.50	10/3.25	87.50
	EvoSkill	BRACE	10/4	20.00	10/3.75	88.25
Qwen3.5-9B	SkillOpt	Native	10/3	20.50	10/3.25	89.00
	SkillOpt	BRACE	10/4	21.75	10/4	89.75
	EvoSkill	Native	10/3.5	20.75	10/3.5	89.00
	EvoSkill	BRACE	10/4.25	22.00	10/4	90.00
Qwen3.5-27B	SkillOpt	Native	10/3.5	22.75	10/3.75	90.25
	SkillOpt	BRACE	10/4.25	23.50	10/4.25	91.00
	EvoSkill	Native	10/3.75	23.00	10/4	90.25
	EvoSkill	BRACE	10/4.25	24.00	10/4.5	91.00

BRACE improves mean terminal success relative to Native in every model–host–benchmark setting in Table 6. All ten competitive attempts produce valid candidates for the Qwen models. Scout produces fewer valid candidates, yet BRACE retains its mean-success advantage. These end-to-end results reflect both deployment decisions and the evolution trajectories they induce. Frozen-candidate replay below isolates evidence-allocation efficiency from differences in candidate yield and trajectory.

Calibration and test-time stopping. For each model and selector, separate calibration candidates select the configuration with the fewest mean pairs that meets a common 0.75-pp realized-regret target. The selected configuration is then frozen for held-out evaluation. Online stopping uses acquired evidence, posterior risk, and independent Monte Carlo confirmation. Full-catalog outcomes score realized regret after replay and do not determine test-time stopping. Unlike the shared-tolerance comparison in Appendix E.2, this protocol calibrates selectors separately to a common empirical target. The target need not be met on held-out candidates.

Table 7: Controlled cross-model replay at independently calibrated operating points using a 0.75-pp calibration regret target. Selectors share held-out candidate pools and action references within each model. Cells report test mean pairs and realized regret (pp), pooled across both hosts and benchmarks. Scout/Random reaches 0.76 pp. Lower pair counts, regret, and harmful-commit rates are better.

Model	Random	Value-var.	DVI	DVI harmful (%)
Llama-4 Scout INT4	10.9/0.76	7.4/0.72	5.9/0.69	4.2
Qwen3.5-4B	10.0/0.70	6.5/0.70	5.0/0.70	3.8
Qwen3.5-9B	9.8/0.67	6.3/0.65	4.7/0.61	3.2
Qwen3.5-27B	9.5/0.63	6.0/0.60	4.4/0.56	2.8

With candidate identity fixed, DVI requires fewer pairs than Random and Value-variance without higher realized regret in any of the four model settings (Table 7). Its savings therefore persist when differences in candidate yield are removed. On held-out candidates, DVI remains below the 0.75-pp calibration target throughout, whereas Scout/Random exceeds it. Together with the end-to-end results, this supports decision-directed evidence allocation across the evaluated executors. It does not test transferring a response checkpoint between models.

E Mechanism Diagnostics

E.1 Shared-Posterior Acquisition

All selectors start from the event's pre-validation checkpoint. Random samples uniformly. Uncertainty maximizes local predictive variance, while V-optimal targets workload-weighted predictive variance reduction. Value-variance targets aggregate deployment-value variance, and Action-entropy targets uncertainty about the full-catalog action. Behavior-aware is a selector-only adaptation of HarnessLens (Xu et al., 2026), using the edit, prior trajectories, task descriptions, and revealed evidence. DVI targets expected wrong-action loss reduction.

Table 8: Fixed-budget acquisition at ten pairs (25% catalog coverage), with early stopping disabled. Decision accuracy is agreement with the full-catalog action. Higher accuracy and lower regret and harmful-commit rates are better. Frequencies use all 1,600 candidate–replay decisions.

Acquisition	Regret (pp)	Decision acc. (%)	Harmful commits (%)
Random	0.74	86.5	6.6
Behavior-aware	0.68	87.2	6.1
Uncertainty	0.56	89.0	5.2
V-optimal	0.49	90.0	4.8
Value-variance	0.46	90.5	4.6
Action-entropy	0.48	91.2	4.1
DVI	0.31	92.6	3.1

DVI achieves lower regret, higher decision accuracy, and fewer harmful commits than both Value-variance and Action-entropy (Table 8). Value-variance seeks a more precise aggregate-value estimate, while Action-entropy seeks greater certainty about which action is correct. DVI instead weights evidence by its expected reduction in wrong-action loss. Its 32.7% regret reduction over Value-variance, computed before rounding, supports this objective under a shared predictor and fixed evidence budget. The comparison with Action-entropy further supports accounting for the cost of a wrong decision, beyond uncertainty about decision identity.

E.2 Adaptive Acquisition and System Configurations

Table 9 compares acquisition under a common posterior-risk tolerance $\eta = .010$ and independent stopping confirmation. Realized regret uses the full-catalog reference. Prior-only takes the initial Bayes action without checking this criterion. Counts are logical pairs before incumbent-cache adjustments.

Table 9: Adaptive acquisition at $\eta = .010$. Range spans the four cohort-level mean regrets, not a confidence interval. Prior-only performs no stopping check.

Acquisition	Mean pairs	Mean regret (pp)	Cohort range (pp)
Prior-only	0.0	1.80	1.69–1.88
Random	10.0	0.70	0.66–0.74
Uncertainty	7.5	0.69	0.65–0.73
V-optimal	7.0	0.71	0.67–0.75
Value-variance	6.5	0.70	0.66–0.74
DVI	5.0	0.70	0.64–0.76

DVI needs fewer pairs than Random and Value-variance at the same reported aggregate regret. With the risk tolerance and confirmation rule fixed, this comparison supports reaching the stopping criterion through more informative task selection. Cohort-level regrets still differ, and the common posterior-risk tolerance does not guarantee identical realized regret. Prior-only incurs higher regret, indicating that historical predictions alone are insufficient for these candidates. The pair counts isolate acquisition effort, while the system ledger additionally accounts for caching and other validation costs.

Table 10 links calibration-selected system configurations to the main ledger. Unlike common-tolerance replay, these pair counts reflect both host-specific candidate streams and separately selected tolerances.

Table 10: Calibration-selected BRACE configurations used in Table 1. Episode costs exclude configuration search.

Setting	η	Mean pairs	Regret (pp)	Ep./run
SkillOpt–Spreadsheet	0.020	3	0.94	228.60
SkillOpt–DocVQA	0.010	6	0.64	264.60
EvoSkill–Spreadsheet	0.020	4	0.86	207.60
EvoSkill–DocVQA	0.010	6	0.74	241.60

E.3 Response Transfer, Historical Evidence, and Calibration

Response-model ablations share histories and nested random task prefixes. Population prior removes candidate conditioning. Context-only response removes task-specific residuals. Independent effects factorizes initial predictive marginals, blocking cross-condition updates. Every variant clamps observed outcomes. MAE compares posterior-mean value with the realized weighted full-catalog value.

Figure 6(a) varies historical candidate coverage and observation density using nested source-run subsets with one candidate per run. Outcome-independent masks reveal 10%, 25%, or 50% of each 40-condition row. Defaults are 40 runs, 25% density, rank 8, and ten online pairs. These controlled snapshots do not measure chronological bank growth.

At a fixed budget of 400 historical pairs, covering 40 candidates at 25% density yields lower regret than covering 20 at 50% (0.31 versus 0.39 points). Thus, the difference reflects how historical evidence is distributed, rather than its total amount. This comparison supports learning reusable response structure from a broader range of updates instead of concentrating the same budget on fewer candidates.

Under the default history and DVI configuration, nominal 90% and 95% intervals attain 87% and 93% coverage. This undercoverage shows why low decision regret alone is insufficient to assess predictive calibration. Independent Monte Carlo confirmation controls sampling error relative to the fitted posterior, not model-calibration error (Appendix A.3). Coverage diagnostics therefore complement the decision metrics. Diagnostics weight candidates within source streams and settings equally, then average numerical replays.

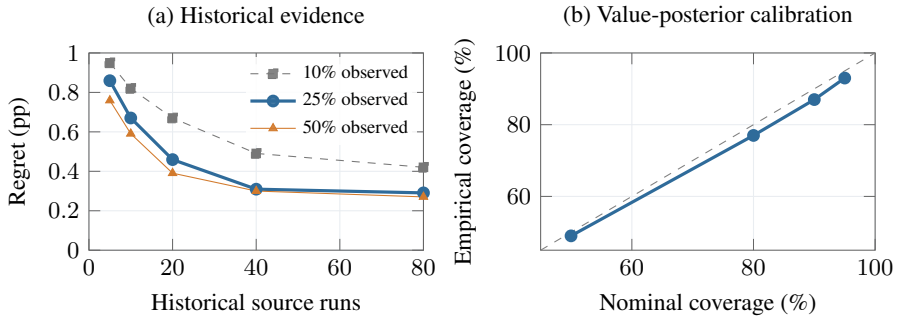

Figure 6: Diagnostics at ten online pairs. (a) Markers show history sizes, and legend percentages give observed row fractions. Lower regret is better. (b) Blue markers show coverage of equal-tailed predictive intervals for realized full-catalog value. Points below the dashed nominal-coverage diagonal indicate undercoverage.

E.4 Case Analysis

Table 11 follows a SkillOpt edit replacing blank cells with zeros. The edit improves numeric conversion but harms operations that depend on distinguishing blank cells from numeric values. The trace shows how evidence of these opposing effects changes the deployment decision.

Table 11: SkillOpt/SpreadsheetBench decision trace. C/P denote commit/preserve with unit loss coefficients. Rows show the action with lower posterior expected loss. Early stopping also requires independent numerical confirmation (Appendix A.3).

Step	Condition	y	Expected loss (C/P)	Preferred action
0	Candidate prior	–	0.035 / 0.049	Commit
1	x_{17}: Numeric conversion	$+1$	0.022 / 0.055	Commit
2	x_{24}: Blank-key lookup	-1	0.038 / 0.026	Preserve
3	x_{33}: Empty-cell filter	-1	0.055 / 0.008	Preserve

The first improvement strengthens the preference for commit, but does not resolve the update's effects on other tasks. The blank-key lookup regression then makes preserve the lower-loss action, and the empty-cell filter provides further support. Full-catalog evaluation contains three repairs, five regressions, and 32 unchanged outcomes, giving $\widehat{V}_c = -2/40 = -0.050$ and the same preserve action. The case illustrates why finding an improvement is insufficient when deployment depends on the aggregate effect. A change in the preferred action is also distinct from stopping, which still requires the risk criterion and independent numerical confirmation. The trace illustrates decision revision, while the cohort comparisons assess its average benefit.