THE VARIANCE COST OF PARTIAL REWARD VERIFICATION IN GROUP-BASED POLICY OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Partial reward verification can preserve a complete-label finite-batch surrogate gradient in expectation while increasing its error at a fixed verification cost. We study when the information purchased by an audit compensates for its cost, taking the fully verified finite-batch surrogate gradient as the target. Using established unbiased estimators, we express this comparison as a threshold on residual-gradient reduction and evaluate it with direct measurements of all 7.37 million trainable LoRA coordinates. A fixed two-stage rule has 2.20–2.77 times the cost-matched whole-group MSE on four BIRD development batches: its first audit creates gradient error in groups whose complete reward vectors are constant. A separate 128-task test shows that cost accounting can reverse the ranking of a frozen whole-group allocation, from an MSE ratio of 0.720 at SQL-body CPU to 1.076 at complete verifier-child CPU. Finally, arithmetic-risk predictors improve development comparisons but do not confirm that gain on 64 tasks from eight additional databases. Together, these fixed-gradient experiments identify residual reduction, the execution cost boundary, and budget transfer as distinct requirements for economical verification.

1 INTRODUCTION

In group-based policy optimization (Shao et al., 2024), correcting one reward changes both its learning signal and the normalization shared by the group. A cheap estimate of each reward can therefore produce a biased policy gradient even when the reward estimates themselves are unbiased. Reference verification resolves the labels, but consumes resources that could otherwise support more training.

Established randomized telescoping recovers the fully verified finite-batch gradient in expectation (Beatson & Adams, 2019; Zhang, 2026). The remaining question is economic: does buying some labels before deciding whether to complete a group reduce enough gradient uncertainty to repay those queries? Random whole-group auditing provides a direct comparator. It buys all labels together with positive probability and applies an inverse-inclusion correction, avoiding intermediate normalized reward vectors. For a two-stage audit, the comparison reduces to a concrete threshold: the fraction of residual-gradient error removed by the first stage must exceed the fraction of the expected budget spent on that stage.

We evaluate this criterion using saved language-model outputs, complete reference labels, directly measured parameter gradients, and measured verifier costs. Fixing the gradient target allows exact expectations over audit randomization, without conflating estimator variance with optimizer dynamics. Our empirical contributions are:

- **A mechanism for partial-audit variance.** On five BIRD development batches, a fixed first-label completion creates residual error, including in constant-reward groups with zero complete gradients. The cost threshold explains why the tested partial audits lose to cost-matched whole-group controls; an APPS ordering study supplies a second development setting (Figure 2).
- **A measured reversal across cost boundaries.** On 128 fresh BIRD tasks, the same whole-group allocation and gradient bank favor the candidate at SQL-body CPU and favor the

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.

control at complete child CPU. Executed audits and randomized-order cost repeats locate the reversal in the cost-matching calculation (Figure 3).

- **A prospective test of risk and budget transfer.** Two arithmetic-risk predictors improve development comparisons but fail a separately frozen, eight-database test. An outcome-blind budget-normalization amendment reveals how small predicted costs can produce near-full verification and undermine the expected savings (Figure 4).

The contribution is a measured account of when established unbiased verification rules cease to be economical. Complete LoRA updates provide timing context; the estimator comparisons themselves concern fixed-gradient error rather than downstream model quality.

2 Related Work

Reward correction and finite-group targets. Cai et al. (2025) develop noise-channel corrections and online appeals, including the cancellation of binary affine corrections under standardization and limitations from clipping. El Mansouri et al. (2025) discuss the same normalization invariance and propose noise-corrected normalization, with a clipping ablation. Our estimand conditions on realized outputs and reference labels and averages over audit randomization. The reward-IPW control diagnoses nonlinear plug-in bias; it does not reproduce either noise-channel method.

Randomized telescoping (Beatson & Adams, 2019) underlies the sequential estimator. The closely related audit-complexity study (Zhang, 2026) covers the complete-label group-relative target, Boolean query-degree hard caps, sequential completion, and cost-matched whole-group correction. We use these foundations. Its frozen software diagnostic measures a 32-dimensional added-logit-bias projection; our BIRD experiments measure every trainable LoRA coordinate, execute selective SQL audits, and test cost and risk allocation on additional databases.

Cost-aware sampling and label acquisition. Mohri et al. (2026) derive cost-aware gradient sampling and evaluate it in GRPO with reward-derived advantages available before selecting optimization work. In our audit setting, the allocation precedes those rewards. We test whether public SQL features can predict the quantities needed by the established square-root risk/cost rule. Chen et al. (2025) use auxiliary-feature balance and the cube method to reduce active-inference variance; our paired whole-group diagnostic examines a fixed development-set geometry within that broader sampling problem.

Recent RLVR work also allocates supervision or rollout compute. Wang et al. (2026) actively acquire ground-truth labels and combine them with pseudo-labels, while Nomand et al. (2026) correct joint inclusion probabilities for adaptively completed rollout pairs under an unclipped, unstandardized objective. We hold outputs fixed and evaluate reference verification for the normalized surrogate.

Reward execution and scheduling. Zhu et al. (2026) schedule batch-level critical requests in a disaggregated reward service; Cao et al. (2025) overlap bounded CPU agent and reward stages with GPU inference. Our timing diagnostics measure explicit verifier wait and whole-verification deletion under fixed local schedules. Their role is to distinguish acquisition cost from removable training time.

3 Problem and Estimator

Fix outputs Y, behavior probabilities, current parameters θ, masks, and reference labels $r^\star$. The target is $g^\star = \nabla_\theta \ell(\theta; r^\star)$ for the implemented finite-group surrogate. For a group of G responses, let the clipped ratio follow PPO (Schulman et al., 2017) and

$$A_i(r) = \frac{r_i - \bar{r}}{s(r) + \epsilon}, \tag{1}$$

$$\ell(\theta; r) = -\frac{1}{G} \sum_i \frac{1}{T_i} \sum_t \min\{\rho_{it}(\theta) A_i(r), \mathrm{clip}(\rho_{it}(\theta), 1-\delta, 1+\delta) A_i(r)\}. \tag{2}$$

Here s is the sample standard deviation and T_i counts valid response tokens. We also evaluate removing the division by $s + \epsilon$.

Known labels require no query. For m unknown labels, fix a query order from available information before sampling the stopping variable. Let ℓ_k denote the complete group loss after revealing the first k labels, with the remaining labels set to prespecified completion values. Choose $1 \geq p_1 \geq \cdots \geq p_m > 0$, draw $U \sim \mathrm{Uniform}(0, 1)$, and define $N = \sum_k \mathbf{1}\{U < p_k\}$. Then

$$\widehat{\ell} = \ell_0 + \sum_{k=1}^{N} \frac{\ell_k - \ell_{k-1}}{p_k}. \tag{3}$$

Since $P(N \geq k) = p_k$, the finite sum telescopes in expectation: $\mathbb{E}[\widehat{\ell}] = \ell_m$ and $\mathbb{E}[\nabla\widehat{\ell}] = \nabla\ell_m$ wherever the chosen surrogate is differentiable. Ordering, probabilities, and labels are treated as constants during differentiation. At clipping boundaries implementations must use a consistent subgradient convention.

The guarantee does not extend to the nonlinear Adam update, gradient norm clipping, or parameters produced after reusing one audit realization for multiple updates. Our initial implementation performs one optimizer update per rollout batch.

Strong full-group comparator. For $I \sim \mathrm{Bernoulli}(q)$, use $\widehat{\ell}_{\mathrm{group}} = \ell_0 + I(\ell_m - \ell_0)/q$. This is the classical inverse-inclusion correction (Horvitz & Thompson, 1952); it is also unbiased and requires qm expected queries when query costs are equal. Real verification times may differ, so count-matched comparisons alone do not establish cost-matched efficiency. When a two-stage rule always buys the first label and audits the rest with probability q, its estimate is $\ell_1 + I(\ell_m - \ell_1)/q$. A whole-group continuation control given that same first label and completion has exactly these branches, cost and conditional risk. Within this fixed completion family, a non-equivalent partial-audit rule must sometimes stop at another information state.

When does the first audit repay its cost? Consider independent group coins with a common continuation probability $0 < q < 1$. Let $c \geq 0$ be the total cost of the mandatory first audits and $C > c$ the total cost of full verification, under an additive cost model. The expected two-stage budget is $b = c + q(C - c)$; a matched whole-group rule uses $p = b/C$. Define $E_0 = \sum_j \|g_j^\star - g_{j,0}\|^2$ and $E_1 = \sum_j \|g_j^\star - g_{j,1}\|^2$, the residual sums before and after the first audits. For $E_0 > 0$, the standard variance formulas give

$$\mathrm{MSE}_{\mathrm{two}} < \mathrm{MSE}_{\mathrm{group}} \quad \Longleftrightarrow \quad \frac{E_0 - E_1}{E_0} > \frac{c}{b}. \tag{4}$$

Indeed, comparing $(q^{-1} - 1)E_1$ with $(p^{-1} - 1)E_0$ gives $E_1/E_0 < q(1 - p)/[p(1 - q)] = 1 - c/b$; the common batch-mean divisor cancels. This algebraic consequence of residual auditing makes the cost of information explicit. The experiments evaluate it with historical CPU replay. A zero baseline residual already gives zero whole-group risk, and full verification ($q = 1$) gives a zero-risk tie.

4 Experimental Protocol

We use Qwen2.5-Coder-3B-Instruct (Hui et al., 2024) with rank-16 LoRA (Hu et al., 2021), groups of eight responses, and fixed initial-policy rollouts for exact conditional gradient-risk comparisons. BIRD (Li et al., 2023) labels come from a pinned local SQL execution endpoint; APPS (Hendrycks et al., 2021) labels come from a frozen test-suite comparator. These labels need not equal semantic correctness or official benchmark scores. The three evidence levels are measured verification cost, fixed-snapshot gradient MSE, and online learning at matched resources. Only the first two have substantial evidence here.

The APPS development comparison has 60 problems, of which 15 contain unknown labels. At the shared verifier-CPU target, the coefficient-influence ordering has MSE 0.001998 versus 0.001646 for whole-group auditing; including ranking CPU widens the gap. Appendix A gives the calibration, ordering rule and complete tables. The BIRD two-stage diagnostic uses fixed response slot 0 and continuation probability one-half; it does not test that learned ordering.

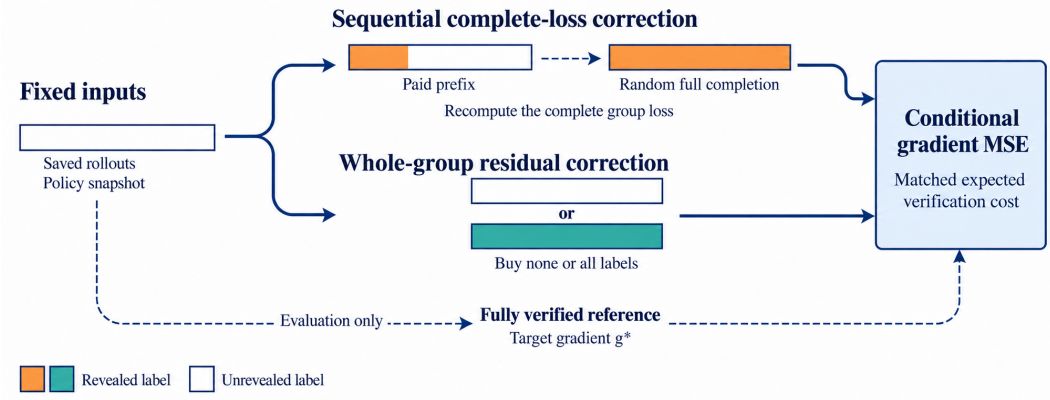

Figure 1: Fixed-gradient verification framework. Reference labels define the common target; each audit reveals only the labels it buys. The top branch illustrates a two-stage audit, which may stop after the paid prefix. Each ℓ_k evaluates the complete group-normalized surrogate with k revealed labels and prescribed completions elsewhere. Positive prescribed audit probabilities preserve the target in expectation before gradient clipping and the optimizer update. The comparison measures conditional gradient MSE at matched expected verification cost.

Finite enumeration. A float64 enumeration over 256 pairs of four-response binary label vectors, including non-unit ratios, matches the target gradient to a maximum absolute error of 5.55×10^{-17}. This checks the estimator identity in these cases.

5 Partial-Audit Information and Gradient Risk

The B8 diagnostic fixes 32 G8 rollouts, reference labels and the initial rank-16 LoRA adapter from a complete BIRD update. The full reference executes 32 gold queries and 256 candidate queries. Historical SQL costs and direct gradients define the comparison; Appendix D gives scheduling and pairing diagnostics.

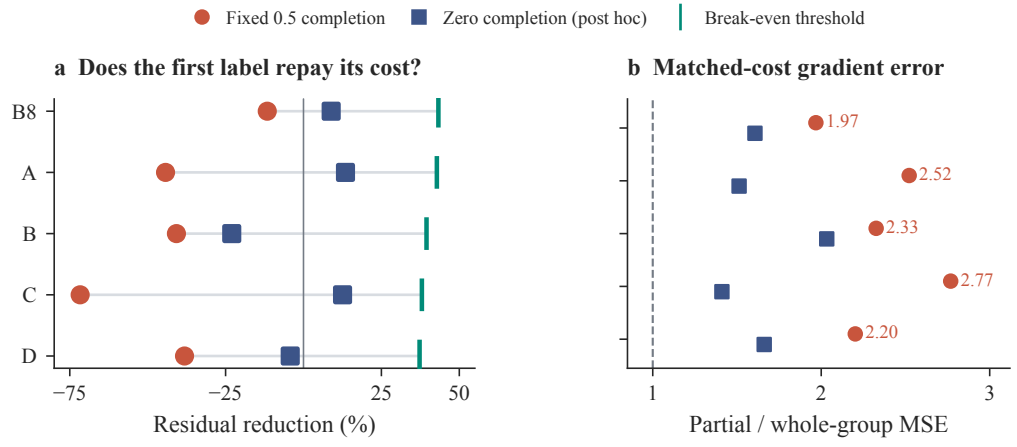

Figure 2: Why the tested partial audits lose. (a) Gradient-residual reduction must exceed the paid-prefix share of expected cost (green ticks). Both the fixed 0.5 completion and post-hoc zero completion miss this threshold in all five development batches. (b) Exact conditional MSE relative to cost-matched Bernoulli whole-group auditing. Costs replay historical SQL child CPU; these are fixed-gradient calculations, not repeated training estimates.

B8 fixed-gradient cost and variance. We directly differentiated three complete surrogate states per group: all unknown rewards set to 0.5, slot 0 revealed, and all eight labels revealed. A 96×96 FP64 cross-state Gram reconstructs both the fully verified and realized $q = 0.5$ update's recorded pre-clip gradient norms within 3.1×10^{-6} absolute error. The two-stage estimator always buys the gold query and slot 0, then uses an independent continuation coin per group to buy the other seven labels with probability $1/2$. If F_1 and F_y are that group's first-state and full-label gradients, its branches are F_1 and $2F_y - F_1$. With an equal per-job startup charge, historical full-run SQL child CPU gives $C_{\text{first}} = 189.348$ and $C_{\text{rest}} = 496.107$ seconds across the batch, so $C_{\text{RT}} = C_{\text{first}} + C_{\text{rest}}/2 = 437.401$ seconds. Whole-group Bernoulli uses independent coins with $q = C_{\text{RT}}/C_{\text{full}} = 0.6381185$; a same-marginal fixed-size whole-group design has the same expected replay cost. The fixed-size policy samples 20 or 21 complete groups with the same marginal probability (Appendix Table 11).

For $B = 32$ independent group coins, put $D_b = F_b(y) - F_b(c^0)$ for the first-revealed state c^0, $H_b = F_b(y) - F_b(c)$ for the all-0.5 state c, and $K_{bb'} = \langle H_b, H_{b'} \rangle$. Conditional on the fixed batch, the $q = 1/2$ two-stage and cost-matched whole-group risks are

$$R_{\text{two}} = \frac{\sum_b \|D_b\|^2}{B^2}, \qquad R_{\text{Bern}} = \frac{(q^{-1} - 1)\operatorname{tr} K}{B^2}. \tag{5}$$

For the fixed-size design, let $L = \lfloor Bq \rfloor$, $h = Bq - L$, and $\pi_2 = [(1-h)L(L-1) + hL(L+1)]/[B(B-1)]$. Its risk is

$$R_{\text{fixed}} = \frac{(q^{-1} - 1)\operatorname{tr} K + (\pi_2/q^2 - 1)(\mathbf{1}^\top K \mathbf{1} - \operatorname{tr} K)}{B^2}. \tag{6}$$

The second term records cross-group covariance; omitting it would give the wrong fixed-size comparator. Every value is an analytic expectation over audit coins using directly computed fixed-parameter gradients. The cost match uses retrospective full-run SQL child CPU and is an expected-cost calculation, not a timed randomized-policy run.

Equation 4 explains the B8 comparison. The first stage uses 43.29% of the expected budget, so it must remove more than 43.29% of the initial squared-gradient residual to beat whole-group auditing. With unknown rewards completed by 0.5, that residual instead increases by 11.63%. The post-hoc zero-completion variant removes 8.88%, still below the threshold. These values come from the same direct-gradient bank and cost replay; the zero-completion result is a diagnostic of the completion rule.

This tests fixed reward-vector completion, not the conditional-expectation completion of Zhang (2026); the observed failure does not establish a limit for posterior completion.

The two-stage rule has 96.84% higher MSE than cost-matched Bernoulli whole-group auditing and 100.07% higher MSE than fixed-size whole-group auditing. Nine all-zero-label groups contribute 19.46% of its total MSE: their fully verified and all-0.5 gradients vanish, while revealing a zero in slot 0 creates a nonconstant intermediate reward vector and often a nonzero gradient. The worker saved all 32 groups; a separate numerical audit validates the raw gradient comparison while preserving a failed post-run metadata check (Appendix I).

Four further development batches. Before new gradient work, we fixed the same slot-0/0.5 completion rule and controls on four other saved full-label BIRD cohorts. Their 128 training task IDs span 16 databases, are pairwise disjoint, and do not overlap B8. Each cohort supplies a separate 32-group conditional target at its own saved initial adapter; we directly backpropagated $F_b(c^0)$ and $F_b(y)$ over all 7.37 million LoRA coordinates. We replayed each full run's historical gold-plus-candidate SQL child CPU, assigned the recorded startup overhead equally per SQL job, and set its own whole-group q by expected-cost equality. Figure 2 compares all five batches; Appendix Table 12 gives absolute risks and costs.

The two-stage MSE is 2.20–2.77 times the matched independent whole-group MSE across A–D, and the fixed-size control is lower still in each batch. Eight, eight, fourteen, and nine all-zero groups contribute 13.7%, 13.7%, 54.5%, and 32.4% of two-stage risk, respectively. A post-hoc zero-completion sensitivity still yields 1.41–2.03 times the matched Bernoulli risk. The mandatory audits consume 37.25–42.80% of the expected budget, whereas the primary first-stage residual increases by 38.18–71.67%. Thus the same information-cost mechanism holds across these development batches. Appendix I records the numerical validation and original run statuses.

Test batch	Child CPU (s)	Candidate MSE	Bern. MSE	Ratio	Fixed ratio
A	27.43/37.31	1.092×10^{-4}	3.491×10^{-4}	0.313	0.314
B	55.00/81.98	1.007×10^{-3}	9.623×10^{-4}	1.046	1.045
C	28.98/33.58	6.429×10^{-4}	3.170×10^{-4}	2.028	2.051
D	28.54/28.73	2.072×10^{-6}	7.948×10^{-6}	0.261	0.262
Four batch estimators, averaged	—	1.101×10^{-4}	1.023×10^{-4}	1.076	1.079

Table 1: Fresh-source direct-LoRA-gradient conditional MSE at each batch's matched measured complete verifier-child CPU; child CPU is expected/full. The uniform and fixed-count probabilities are retrospectively matched to the candidate's expected measured cost; those matched policies were not rerun as new randomized trials. The last row averages the four independently randomized batch-gradient estimators, so its MSE is the sum of per-batch MSEs divided by 16. D nearly fully verifies.

Complete-update timing context. A separate B8 whole-group audit completed one AdamW update, selecting 15 of 32 groups. Its observed cold time was 278.235 seconds versus 395.390 seconds for the earlier full reference. Both SQL and backward work were skipped on unselected groups. Appendix H reports the execution counts, parameter differences, schedule controls and failed lookahead trial; the single observed contrast does not isolate a verification speedup.

6 Cost-Aware Allocation Across Execution Boundaries

To check whether a simpler allocation transfers beyond the previously inspected databases, we froze a five-feature, cost-only whole-group inclusion rule before generating responses on 13 previously unused BIRD training databases. Four task-disjoint 32×G8 Qwen2.5-Coder-3B batches each completed a full-reference LoRA update and a separate direct-gradient bank. We then executed 286 actions from the frozen candidate, Bernoulli, and fixed-count masks, plus two actions needed to observe every group's verification cost. All 288 executed binary reward vectors agree with their full-reference counterparts. The selected policy uses positive inclusion probabilities and an inverse-probability correction; its conditional squared gradient error is computed from the full group gradients, not inferred from the one realized mask.

The five pre-verification features are log mean parsed-SQL length, log mean `JOIN` and `SELECT` counts, fraction of distinct candidate SQL strings, and log database size in MiB. Standardizing these features on five previously exposed batches (160 groups), we fit ridge regression with penalty 10 to predict $\log(1 + C)$ for gold-plus-eight-candidate internal child CPU. For predicted cost $\hat{C}_i$, the frozen rule is $q_i = \mathrm{clip}(0.885290/\sqrt{\max(0.01, \hat{C}_i)}, 0.05, 1)$. The multiplier was calibrated to spend half of development cost in expectation and was not retuned on the final test. The historical internal-timer target omits process CPU charged by the final measured child-CPU evaluation; we retain that mismatch as part of the tested rule.

The cost-only rule therefore fails the predeclared best-control superiority gate despite favorable A and D outcomes. A weaker common-probability comparison pooling all 128 groups gives ratio 0.973, but its 13-database bootstrap interval $[0.274, 2.371]$ is inconclusive and the pooling ignores the separate batch budgets. This is a fixed-initial-policy estimator result, not a model-quality comparison or a measured training speedup.

Cost boundary as a decision variable. We also recomputed the same saved probability vector and direct gradients at two cost boundaries recorded by the executed audits. Summing the gold and candidate SQL child-body timers within each action, then averaging actions by group, yields 83.404 seconds over the 128 full groups and a four-batch candidate/Bernoulli MSE ratio of 0.720 at body-CPU-matched budgets. Charging the full observed child CPU yields 181.610 seconds and ratio 1.076. At the body boundary the cost-matched Bernoulli inclusion probabilities for A–D are 0.524, 0.622, 0.852 and 0.980; at the full-child boundary they are 0.735, 0.671, 0.863 and 0.993. Candidate q_i, labels, gradients and MSE do not change; the matched control's probability and MSE change with the cost definition. The allocation ranking therefore reverses. The outside-body difference is observed process CPU, but these records do not separate interpreter startup, imports, serialization,

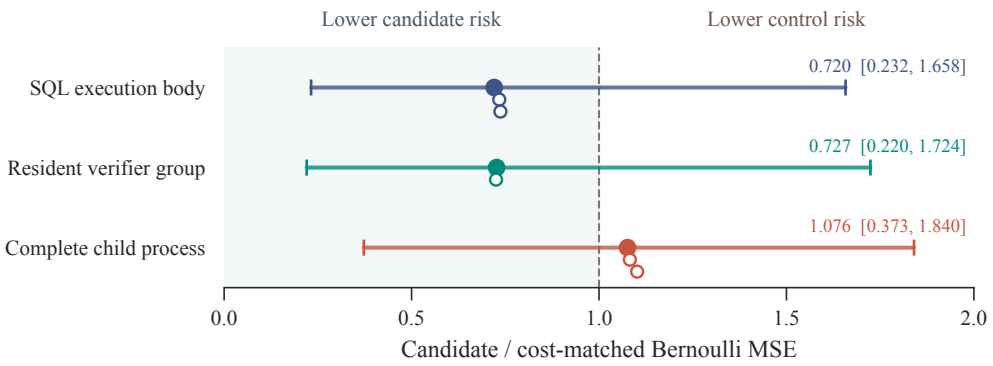

Figure 3: Changing the measured cost boundary reverses the point comparison for an unchanged allocation and gradient bank. Filled dots show the first evaluation, horizontal bars its 95% database-bootstrap interval, and vertically offset open dots repeated timings on the same 128 groups. Labels give the first ratio and interval. Every interval crosses one. The original isolated ledger includes completion audits; complete repeat passes use randomized order. Resident costs exclude staging, initialization, actor work and optimization.

and other components. For fixed q_i, write $R = \sum_i \|g_i\|^2$ for complete per-group gradients g_i, and let M be candidate batch MSE. The candidate beats a cost-matched independent Bernoulli rule exactly when $p < R/(R + 32^2 M)$. In batch B this threshold is 0.661: matched p is 0.622 at SQL-body CPU, 0.627 at resident group CPU and 0.671 at full-child CPU. B changes rank as overhead changes the matched control; C exceeds its 0.756 threshold at every measured boundary.

Two post-result randomized-order complete-audit passes on the same 128 groups give full-child ratios 1.082/1.102 and SQL-body ratios 0.734/0.737. Both original-pass 13-database bootstrap intervals include one; these repeats test cost stability, not method generalization. Appendix F gives the resampling protocol and intervals.

Resident-process cost ablation. Two more complete randomized-order passes in one resident verifier process per batch give marginal group-CPU ratios 0.727/0.725; batch C loses and omitting A raises the aggregate to 1.089/1.090. Database-cluster intervals cross one. Figure 3 summarizes the measured boundaries. Staging consumes another 218.827/222.499 CPU seconds; expected removable group CPU is only 9.11%/9.28% of measured post-validation verifier CPU under fixed staging. These post-result estimates do not show independent policy transfer or end-to-end savings. Appendix G details execution and cost.

Post-update sensitivity. At A's saved post-update adapter, three new B–D full-LoRA-gradient banks on fixed old-policy rollouts yield an aggregate resident-cost ratio of 1.049/1.049, compared with 1.089/1.090 at the initial adapter on the same B–D tasks. Both parameter states lose; only 1 of 23,861 eligible tokens has an active clipping branch. Appendix E gives the protocol and batch results.

7 Arithmetic Risk Prediction and Transfer

Whole-group allocation with $q_i \propto \sqrt{\hat{R}_i/\hat{C}_i}$ needs a pre-verification prediction of arithmetic squared-gradient risk $R_i = \|g_i\|^2$, where g_i is the complete per-group gradient and the all-equal cheap-reward baseline has zero gradient. Fitting log risk targets a different quantity; exponentiating that prediction need not estimate the arithmetic conditional mean. We compare predictors of arithmetic risk within the established cost-aware allocation rule.

Development screen. On the original 128 A–D groups, we leave each of 16 databases out in turn, refitting every transform and predictor on the other databases. Five variants compare five-feature

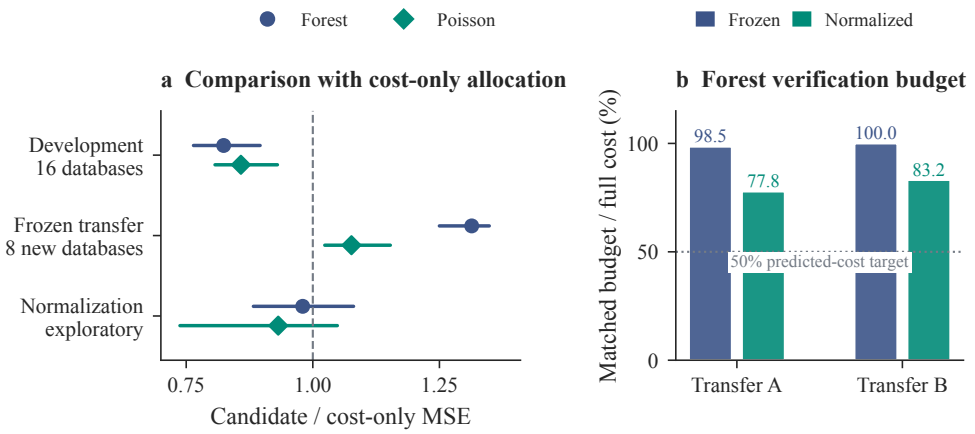

Figure 4: Risk allocation across development and prospective evaluation. (a) MSE ratios to a same-cost-predictor cost-only control with descriptive 95% database-bootstrap intervals. Development uses 16 databases and historical costs; transfer uses eight new databases and complete executed child CPU. Development intervals do not adjust for variant selection; frozen-transfer cost-only intervals exclude 1,730 zero/zero bootstrap ties. (b) Forest matched budget, including prediction CPU and capped at full verification cost; the exploratory 50% predicted-cost target does not deliver 50% measured cost. Exact comparisons with all controls appear in Appendix Table 5.

log-Ridge, five-feature Poisson mean regression, rich-feature Poisson, rich-feature forest, and rich Poisson with batch budget normalization. The rich features add SQL token diversity, pairwise token overlap, length and structure statistics, and prompt/response lengths to public database size. Repaired risk predictions shrink halfway toward training mean risk; the original log-Ridge is unshrunk. All variants use the same five-feature cost predictor. Rich forest and Poisson pass the declared development gate, with pooled MSE ratios 0.824 and 0.858 against cost-only allocation. Their development intervals do not adjust for variant selection. These gains do not isolate the target transformation: regression family, shrinkage and features also change across the variants.

Prospective new-database test. We freeze both models on all 128 development groups, selecting forest as primary and Poisson as secondary. Two new 32×G8 batches cover eight databases without prior saved candidate outputs. Each completes a reference LoRA update, a direct full-coordinate gradient bank, and a randomized-order complete isolated SQL-cost audit. All 64 audit reward vectors reproduce the reference. Candidate probabilities use only frozen predictors and public rollout features. Controls are retrospectively matched to expected measured complete-child CPU plus candidate prediction CPU; controls receive zero prediction overhead, conservatively favoring them. The two independent batch-gradient estimators are averaged, giving total MSE equal to the sum of batch MSEs divided by four.

The frozen forest fails the joint superiority gate. Its training-calibrated multiplier spends 98.54% of full measured cost in transfer batch A and fully verifies transfer batch B. After inspecting only these prelabel probabilities, before reading new gradient or cost outcomes, we freeze an exploratory amendment that calibrates each batch to 50% of *predicted* cost. Both original outcomes remain reported. This normalization spends 77.85%/83.17% of actual complete cost for forest, and does not recover strong-control superiority: cost-only ratios are 0.980 (forest) and 0.932 (Poisson), with descriptive database-cluster intervals $[0.883, 1.080]$ and $[0.738, 1.048]$.

Where prediction and cost accounting disagree. In transfer batches A/B, median cost predictions are 0.168/0.029 seconds per group, whereas measured complete-child medians are 0.776/0.726 seconds. Twenty-four of 64 predictions reach the 0.01-second floor. CPU outside the endpoint's internal per-query timers accounts for 94.95%/99.06% of measured complete child-process CPU. The verifier's per-query CPU timer differs from the SQL-body timer in Section 6. The remaining process CPU cannot be attributed to startup alone; Appendix B gives the recorded field definition.

A same-risk-model risk-only comparator has lower conditional MSE than the normalized hybrids, whose candidate/risk-only ratios are 1.318/1.341. That comparator's multiplier uses retrospective measured-cost matching; it does not establish a deployable risk-only improvement. Because the fitting and evaluation timers differ, this test does not isolate predictive transfer for a common cost target.

8 LIMITATIONS

BIRD and APPS verifier labels are execution outcomes, not a complete measure of semantic correctness. The five partial-audit BIRD batches are development-exposed; the separate final test has fresh tasks but one rollout seed and one initial adapter. Retrospectively matched controls and repeated costs on fixed gradients do not establish online learning gains. Resident CPU omits staging, the post-update check has little active clipping, and transfer uncertainty rests on eight database clusters. The tested fixed completions cannot rule out posterior or adaptive completion. Appendix J records the cohort and timing qualifications in detail.

9 CONCLUSION

The tested partial audits do not remove enough gradient residual to repay their first-query cost; intermediate normalization can create variance even when the complete gradient is zero. For whole-group allocation, charging complete verifier-process CPU reverses the favorable SQL-body point comparison on the same 128 saved groups. A prospective database test further exposes budget saturation and cost underprediction. These results make information value, the execution cost boundary, and risk/cost transfer separate tests of whether verification is economical.

AI USE STATEMENT

Generative AI assisted with three parts of this work: (1) drafting and editing the manuscript; (2) retrieving, identifying, and summarizing related literature; and (3) research ideation and execution, including hypothesis and experiment design, code implementation and review, mathematical derivation and proof exposition, and interpretation of recorded results. AI was also used to generate the schematic illustration. Numerical figures were produced by scripts from recorded results. The authors checked AI-assisted claims against available primary sources and experiment records and remain responsible for the text, mathematics, citations, code, and reported findings. AI-based reviews used models from the same family and are not independent validation.

REPRODUCIBILITY STATEMENT

The local artifact archive contains source revisions, split manifests, configurations, seeds, raw metrics, figure-generation scripts, estimator tests, and all guarded update receipts, including failed receipts. Appendix K names the principal evidence files. The four-batch read-only revalidation independently recomputes numerical risks from saved direct-gradient Grams and preserves original guard outcomes. The local paths are provenance identifiers, not public download links. An anonymous numerical supplement provides standalone code and compact numeric inputs for selected partial-audit, cost-boundary, and risk-transfer point estimates. It does not reproduce model training, gradient extraction, or confidence intervals.

REFERENCES

Alex Beatson and Ryan P. Adams. Efficient optimization of loops and limits with randomized telescoping sums. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 534–543. PMLR, 2019. URL https://proceedings.mlr.press/v97/beatson19a.html.

Xin-Qiang Cai, Wei Wang, Feng Liu, Tongliang Liu, Gang Niu, and Masashi Sugiyama. Reinforcement learning with verifiable yet noisy rewards under imperfect verifiers. *arXiv preprint arXiv:2510.00915*, 2025. URL https://arxiv.org/abs/2510.00915v4. Version 4.

Shiyi Cao, Dacheng Li, Fangzhou Zhao, Shuo Yuan, Sumanth R. Hegde, Connor Chen, Charlie Ruan, Tyler Griggs, Shu Liu, Eric Tang, Richard Liaw, Philipp Moritz, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. SkyRL-Agent: Efficient RL training for multi-turn LLM agent, 2025. URL https://arxiv.org/abs/2511.16108.

Boyu Chen, Zhixiang Zhou, Liuhua Peng, and Zhonglei Wang. Balanced active inference. In *Advances in Neural Information Processing Systems*, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/4a3dadd6fd404a113e190e07380f0761-Abstract-Conference.html.

Omar El Mansouri, Fathinah Asma Izzati, Mohamed El Amine Seddik, and Salem Lahlou. Noise-corrected grpo: From noisy rewards to unbiased gradients. *arXiv preprint arXiv:2510.18924*, 2025. URL https://arxiv.org/abs/2510.18924v3. Version 3.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. Measuring coding challenge competence with APPS. In *Advances in Neural Information Processing Systems*, 2021. URL https://arxiv.org/abs/2105.09938.

Daniel G. Horvitz and Donovan J. Thompson. A generalization of sampling without replacement from a finite universe. *Journal of the American Statistical Association*, 47(260):663–685, 1952. doi: 10.1080/01621459.1952.10483446.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. URL https://arxiv.org/abs/2106.09685.

Binyuan Hui, Jian Yang, Zeyu Cui, et al. Qwen2.5-Coder technical report. *arXiv preprint arXiv:2409.12186*, 2024. URL https://arxiv.org/abs/2409.12186.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C. C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. Can LLM already serve as a database interface? a BIg bench for large-scale database grounded text-to-SQLs. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2305.03111.

Clara Mohri, Amir Globerson, Haim Kaplan, Tomer Koren, and Yishay Mansour. Cost-aware learning. *arXiv preprint arXiv:2604.28020*, 2026. URL https://arxiv.org/abs/2604.28020.

Pixel Nomand, Elena Voss, Marcus Hale, and Sofia Reyes. Pair: Pairwise-aware inclusion reweighting for adaptive rollout allocation in rlvr. *arXiv preprint arXiv:2608.11368*, 2026. URL https://arxiv.org/abs/2608.11368.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

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

Li Wang, Xiaodong Lu, Xiaohan Wang, Yikun Ban, Jiajun Chai, Wei Lin, Tianhao Peng, and Guojun Yin. When self-belief misleads: Active label acquisition for reinforcement learning with verifiable rewards. *arXiv preprint arXiv:2605.25864*, 2026. URL https://arxiv.org/abs/2605.25864.

Shengwei Zhang. The audit complexity of group-relative policy updates, 2026. URL https://github.com/Bubu631/agent-audit-complexity. Research manuscript.

Split	Tasks	Programs	Cheap pass	Full pass	Disagree	Extra CPU (s)
Calibration	60	480	47	14	33	3.91
Dev	60	480	34	9	25	4.45

Table 2: Completed fixed-model feasibility pilot. Extra CPU is incremental full-verification process time, including interpreter/isolation startup. These are verifier labels, not semantic correctness or an online training result.

Ruidong Zhu, Mingcong Han, Yinmin Zhong, Wencong Xiao, Xuanzhe Liu, and Xin Jin. DistRS: Disaggregated reward service for RLVR with Batch-Level constraint. In *23rd USENIX Symposium on Networked Systems Design and Implementation (NSDI 26)*, pp. 1517–1531. USENIX Association, 2026. URL `https://www.usenix.org/conference/nsdi26/presentation/zhu-ruidong`.

A APPS Ordering and Development Protocol

From independent calibration data, predict label-error probability q_i and verification cost $\hat{c}_i$. Holding actor log probabilities fixed, calculate the change Δw_i in coefficients of token log-probability gradients after flipping the candidate label. Rank by

$$s_i = \frac{q_i \|\Delta w_i\|_2^2}{\hat{c}_i + \epsilon}. \tag{7}$$

This is a heuristic: coefficient distance is not parameter-gradient distance, and interactions among multiple errors may be missed. We evaluate against random ordering and $q_i/\hat{c}_i$ ordering. Neither hidden labels nor future realized verification times enter the ranking interface.

For the APPS diagnostic, the calibrator uses seven features of cheap-passing programs: log code length, log line count, AST validity, and counts of `for`, `while`, `input(`, and `print(`. Standardized logistic regression predicts cheap-to-full label disagreement, while standardized ridge regression predicts log incremental verifier CPU; both fit only the 60 calibration problems. For m unknown labels, the survival schedule is $p_k = \min(1, a\,0.8^{k-1})$, where bisection chooses a so $\sum_k p_k = mb$ for query fraction $b \in \{0.25, 0.5, 0.75\}$. The order is frozen before drawing the stopping variable. The BIRD two-stage diagnostic instead uses fixed response slot 0 and $p = (1, 0.5, \ldots, 0.5)$; it does not test this learned ordering.

Data and endpoint. The pilot model is Qwen2.5-Coder-3B-Instruct (Hui et al., 2024) with rank-16 LoRA (Hu et al., 2021). We restrict APPS (Hendrycks et al., 2021) to standard-input/output Python tasks with at least six distinct tests and prompts within 2,048 tokens. Two deterministic randomly selected tests form the cheap suite. A cheap failure implies full failure, and every method shares this short circuit. Full labels mean passing the specified suite under a frozen ordered-token comparator (exact integers; absolute/relative tolerance 10^{-6} for expected floating-point tokens), not semantic correctness. The SQL diagnostic uses BIRD (Li et al., 2023) training databases and a pinned local execution endpoint.

The frozen training-side filter retains 240 problems: 60 calibration, 60 development, and 120 policy-training problems. These sets are disjoint; the smaller-than-planned size is a feasibility limitation. APPS test is reserved for final evaluation. Three evidence levels are required: real verification cost profiling; fixed-snapshot gradient MSE; and online learning at matched resources. Evaluation labels are separate from the ranking inputs.

Initial fixed-snapshot comparison. On the 60 development problems, only 15 groups contain unknown labels and 12 contain cheap/full disagreements. At the shared CPU target in Table 3, group-influence ordering has higher descriptive MSE than both required comparators. Including measured prediction and ranking CPU increases the gap. The no-reward-standard-deviation control also shows no descriptive advantage over these comparators. These results do not support expanding the proposed method's online training.

Method	Verification CPU matched	Verification + ranking CPU matched
Random full group	0.001646	0.001646
Error / cost	0.001945	0.002406
Group influence / cost	0.001998	0.003202

Table 3: Initial normalized-surrogate gradient MSE at 0.03705 CPU seconds per problem. Values interpolate measured budget points on the corresponding cost axis. Several bootstrap resamples lack interpolation support; no confidence interval or statistical significance claim is reported.

At the directly evaluated middle design point on the total verification-plus-ranking CPU axis, full-group residual correction had expected CPU 0.037051 seconds per problem and MSE 0.001646; group-influence ordering had expected CPU 0.043016 seconds and MSE 0.002045, and error/cost ordering had expected CPU 0.039049 seconds and MSE 0.002003. The CPU values use measured candidate costs and analytically enumerated audit states; these randomized auditing policies were not executed as online training runs. At this evaluated point, the full-group method has both lower expected cost and lower MSE than the two ordering policies. The table's common-budget interpolation is descriptive: only 1,589 of 2,000 requested bootstrap resamples supported the comparison, so no confidence interval or significance conclusion is reported.

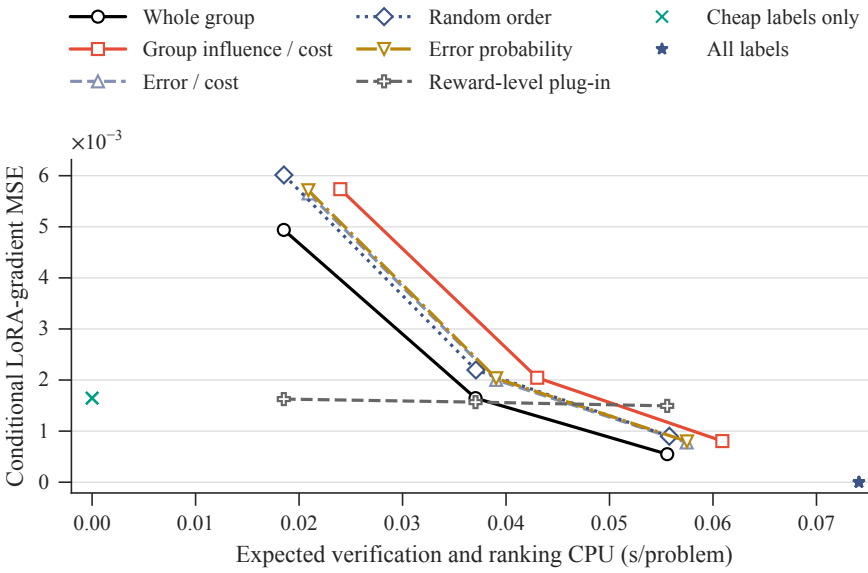

Figure 5: APPS development comparison on 60 fixed G8 groups, including measured verification and ranking CPU. Curves average conditional gradient MSE across groups. Cheap-only is biased; reward-IPW is a diagnostic plug-in control. Random order averages 32 permutations; uncertainty intervals are omitted because common-budget bootstrap support is incomplete.

B Risk-Transfer Protocol and Complete Screen

The endpoint-internal per-query CPU measure records verification work inside each query. Complete child-process CPU is measured around the whole verifier process, while SQL-body CPU isolates the executed SQL portion. Their difference also includes dispatch and bookkeeping and is not assigned solely to startup.

All candidate risk targets use the complete initial-policy group gradients. The original five-feature log-risk Ridge has penalty 10 and fits $\log(R + 10^{-4})$. Repairs fit risk divided by its training mean, using Poisson regression with penalty 1 or a forest with 200 trees, depth 3, minimum leaf size 8 and random seed 20260926. Each repaired prediction is $\bar{R}_{\text{train}}\,\text{clip}(0.5 + 0.5\hat{r}, 0.1, 10)$. The

cost Ridge has penalty 10 and predicts $\log(1 + C)$; inverse-transformed cost is floored at 0.01. Standardization fits only training folds. Inclusion probabilities are clipped to $[0.05, 1]$; the original multiplier spends half of training cost in expectation. The fifth development variant instead spends half of current-batch predicted cost, without accessing held-out outcomes. Every declared variant is reported below.

Development risk variant	Bernoulli	Fixed count	Cost-only	Risk-only
Log-Ridge, five features	0.901	0.923	0.993	0.929
Poisson, five features	0.817	0.844	0.903	0.913
Poisson, rich features	0.778	0.803	0.858	0.912
Forest, rich features	0.749	0.776	0.824	0.938
Poisson, rich, batch budget	0.847	0.877	0.903	0.945

Table 4: All five declared development variants. Exact values in `RISK_TARGET_REPAIR.json` supersede rounding. Development intervals are descriptive and do not account for selecting among these variants.

The subsequent source census found 11 databases without prior saved candidate generations. Nine supplied at least 12 source tasks. A frozen hash order selected eight; gold feasibility was checked on 12 hash-ranked tasks per database, and the first eight feasible tasks were retained. All 96 gold checks succeeded. Batch A contains `cs_semester`, `cookbook`, `mental_health_survey` and `human_resources`; B contains `computer_student`, `trains`, `retail_world` and `book_publishing_company`. These are source-conditioned training tasks, not an official BIRD evaluation. Their 64 direct gradients include 38 zero gradients, all retained.

The primary success gate requires at least 10% lower MSE than Bernoulli, randomized fixed count, and same-cost-predictor cost-only allocation, plus a cost-only interval upper bound below 1. Risk-only is a diagnostic ablation. The primary forest fails. Secondary Poisson is never substituted for the primary. Both fail again under the separately frozen exploratory amendment. Five thousand paired eight-database bootstrap draws retain original batch identities, with 41 empty-batch draws excluded. For the original cost-only comparison,1730 further draws are zero/zero ties; any nondegenerate ratio interval conditions on the remaining 3229 draws. For each amended cost-only comparison, one zero/zero tie leaves 4958 ratios. Positive-over-zero ratios would be retained as infinite, although none occurs here. These intervals condition on measured costs and frozen gradients, not repeated training.

Artifacts are under `reports/model-scale-natural-cost-20260924/`: `RISK_TARGET_REPAIR.json`, `RISK_CONFIRMATION_RESULTS.json`, `RISK_CONFIRMATION_NORMALIZED_RESULTS.json`, and `RISK_CONFIRMATION_RESULT_AUDIT.md`. The model and both probability freezes are under the corresponding `configs/` study directory; complete references, gradients and actual audits are in the `bird-risk-confirmation*` raw result directories. Shared database staging and historical full-gradient calibration are reported separately from marginal audit CPU. No selective wall-time trial or online learning gain is inferred from these exact conditional risks.

C Study Units and Exact Diagnostic Tables

D Scheduling and Secondary Diagnostics

Shared full-label scheduling controls. Each completed timing arm in this subsection performs one Qwen2.5-Coder-3B-Instruct rank-16 LoRA update with 32 groups of eight SQL responses, accumulating all groups before one AdamW step. The later fixed-gradient diagnostics perform no update. Full-verification control labels come from the pinned local read-only SQL endpoint; the zero-SQL arms reuse those labels only after exact rollout parity. On a separate frozen candidate set, the local endpoint agreed with the official execution core for 254 of 254 determinate labels; two timeouts remained unknown. These measurements concern the local endpoint rather than an official BIRD score.

Evidence	Risk model	Bernoulli	Fixed count	Cost-only
Development LODO	Forest	0.749	0.776	0.824
Development LODO	Poisson	0.778	0.803	0.858
New-database frozen	Forest	1.098	1.101	1.314
New-database frozen	Poisson	0.839	0.842	1.076
Exploratory normalization	Forest	0.916	0.919	0.980
Exploratory normalization	Poisson	0.862	0.864	0.932

Table 5: Candidate/control gradient-MSE ratios. Development uses historical cost replay; new-database rows use complete executed child CPU and prediction overhead. Development pools sixteen eight-group database comparisons; new-source rows average two 32-group batch estimators. Each row matches controls to its own budget. The exploratory amendment is separate from the failed primary confirmation.

Study	Gradient bank	Cost basis	Policy run
APPS dev, 60 G8	LoRA score Gram	verifier + planner CPU	analytic
BIRD B8, 32 G8	direct LoRA	SQL child CPU replay	one whole-group update
BIRD A–D, 4×32 G8	direct LoRA	SQL child CPU replay	none
BIRD final test, 4×32 G8	direct LoRA	executed audit child CPU	four full updates
BIRD risk transfer, 2×32 G8	direct LoRA	child + prediction CPU	two full updates

Table 6: What each comparison measures. Fixed-gradient MSE averages audit randomization conditional on saved outputs, labels and parameters. It is distinct from a timed selective policy and from online model quality.

The original TRACE3B cohort had 15 mixed-reward groups and 92 positive labels. A distinct cohort, HCJGF, was conditioned on gold-query feasibility within `sales_in_weather`; it had 21 mixed groups, 86 positive labels and nine candidate timeouts. Its two-group-worker execution exposed 226.21 seconds of scheduled wait. A stronger full-label control used one global pool of four SQL jobs, including gold jobs, under the same four-core/400% CPU quota. This control retained all 256 HCJGF generated responses and binary labels, used fresh read-only candidate connections with the original gold-plus-candidate cap, and reduced measured wait to 33.55 seconds (Table 7).

We also froze four development strata spanning four databases each. A complete census executed all 192 selected gold queries successfully; the first eight eligible tasks per database defined four disjoint 32-task cohorts, ordered by interleaving the four databases. Each cohort received the same 3B model, seed 11, G8 and global four-job full-label schedule. Strata A–D produced respectively 90, 144, 80 and 104 positive labels, with 20, 16, 15 and 18 mixed groups. These are conditional source checks across 16 training databases, not a held-out or representative BIRD sample.

All five global four-job controls fail the frozen operational screen requiring at least 15% explicit scheduled wait divided by cold time. This admission rule is distinct from a bound on possible savings: verification and actor work share CPU capacity, so deleting SQL can also change generation or backward time. A zero wait measurement therefore does not imply zero possible savings. We separately preregistered an exploratory 15% cold deletion-contrast screen for each oracle diagnostic below, with post-staging time as a secondary endpoint. These new diagnostics do not redefine or replace the original scheduled-wait gate.

Zero-SQL oracle timing diagnostics. We separately froze one additional run on each of A–D and HCJGF, executing no gold or candidate SQL and using the corresponding full-label run as reference. Each retained physical GPU8, the four-core/400% quota, initial adapter, 32×G8 generation, source staging and one optimizer update. Before labels entered backward, token IDs, masks, texts, parsed SQL and prompt identity had to match the reference exactly. Source loading, lookup and parity checks remain charged in cold time. Each run removes all 32 gold and 256 candidate jobs with child startup, queue and IPC work.

For A, cold time was 33.901 seconds lower with the oracle (Table 8), exceeding its 4.621-second explicit wait; staging itself differed by 0.957 seconds. For HCJGF, cold time was 49.613 seconds lower, exceeding its 33.552-second explicit wait. B and D oracle cold times were slower than their

Complete 3B update	Cold wall (s)	Explicit wait (s)	Wait/cold
Original TRACE3B cohort	418.40	0.00	0.00%
HCJGF, two group workers	582.07	226.21	38.86%
HCJGF, four global SQL jobs	405.24	33.55	8.28%
Four-database stratum A	426.97	4.62	1.08%
Four-database stratum B	321.34	0.00	0.00%
Four-database stratum C	329.00	0.00	0.00%
Four-database stratum D	306.61	0.00	0.00%

Table 7: Complete 32×G8 LoRA updates. Cold time runs from the guard's monotonic start through saving the updated adapter. Explicit wait counts time the actor waits for outstanding verifier jobs. It excludes contention that can slow actor computation and is not an upper bound on end-to-end savings. A–D all use four global SQL jobs.

Reference cohort	Full cold (s)	Oracle cold (s)	Cold reduction	Post-stage reduction
Four-database A	426.972	393.071	7.940%	10.933%
Four-database B	321.339	322.122	−0.244%	−0.475%
Four-database C	328.995	324.555	1.350%	1.510%
Four-database D	306.613	309.500	−0.942%	−0.967%
Single-database HCJGF	405.239	355.626	12.243%	13.591%

Table 8: One complete zero-SQL oracle run per reference cohort. Each reduction uses its own full-label reference interval as denominator; negative values mean the oracle was slower. Source staging and parity costs remain charged. These are single unrandomized, clairvoyant deletion contrasts, not measured selective-policy speedups.

references; C was 4.440 seconds faster. All five retained exactly 32 groups and 256 original binary labels and completed one optimizer update. No cold or post-staging reduction reached the separately frozen 15% exploratory threshold.

These are unrandomized deletion contrasts with a clairvoyant label source. Host load, cache temperature, run order and numerical variation can affect their differences. Although each oracle matched all input tokens, labels and recorded losses, its saved update vector differed from the corresponding full-SQL update by 6.7–7.2% in relative Euclidean norm; we do not infer a cause. Removing gold, candidate and scheduler work together does not isolate candidate CPU contention, and even the oracle contrast is not a statistical or mathematical upper bound on a deployable policy. Fresh paired controls are required for a causal timing claim.

Fixed-gradient falsifier on the conditional cohort. A shared prepare-only SQL arity screen certified 16 of 256 candidate labels as zero; all policies receive these same certificates. The earlier response-score reconstruction attempt failed its frozen 1% numerical gate at group 1 and contributed no admitted bank. We instead directly backpropagated three full-G8 reward states for every group at the saved initial LoRA adapter. Write $F_b(c)$ for the group-mean gradient with certified zeros and 0.5 on unknown labels, $F_b(c^j)$ after revealing the first unknown label j, and $F_b(y)$ for all original endpoint labels. Direct squared gradient differences, checked against a 32×3×3 FP64 Gram bank, give exact fixed-coin conditional MSE for this frozen batch, up to numerical arithmetic.

The plug-in two-stage rule always buys label j, then buys the remaining unknown labels with probability $1/2$. Its conditional batch MSE is $\sum_b \|F_b(y) - F_b(c^j)\|^2/32^2$. A whole-G8 residual correction uses the same $F_b(c)$ and a cohort-wide $q = 0.591505$ chosen to match expected *retrospective candidate-CPU replay* after the shared certificates; its MSE is $(1/q-1)\sum_b \|F_b(y) - F_b(c)\|^2/32^2$. Table 9 includes a same-prefix whole-G8 continuation policy that copies the first action and continuation coin and is branchwise identical to the two-stage rule.

Fixed-coin estimator	Candidate CPU replay (s)	Batch gradient MSE
Whole G8, matched $q = 0.591505$	218.693	0.000635752
Plug-in first label, then $q_1 = 0.5$	218.693	0.001079782
Same-prefix whole-G8 continuation	218.693	0.001079782

Table 9: Direct initial-adapter LoRA-gradient comparison on the conditional single-database cohort (32 groups, 256 responses). Candidate CPU is an expectation replayed from completed original SQL calls, not timed policy cost; equal shared gold/arity work and unmeasured planning overhead are excluded. The two-stage MSE is 69.84% higher than the matched whole-G8 value.

The adverse gap has a concrete plug-in mechanism. In nine groups, $y = (0, \ldots, 0)$ and $c = (0.5, \ldots, 0.5)$ both yield zero normalized advantages, hence $F_b(y) = F_b(c) = 0$. Revealing a first zero makes $c^j = (0, 0.5, \ldots, 0.5)$ nonconstant and generally gives $F_b(c^j) \neq 0$, so the two-stage conditional MSE is $\|F_b(c^j)\|^2$ while the matched whole-G8 contribution is zero. Those nine groups' summed two-stage error equals 84.76% of the net aggregate MSE gap. This diagnoses this fixed plug-in rule; it does not evaluate a posterior-expectation baseline, an actual timed policy, or learning after the update.

Fixed cross-group pairing diagnostic. On the same exposed cohort, we froze 16 pairs using only generated SQL features, then directly recomputed the two complete reward-state gradients for all 32 groups. Residual norms passed a prespecified repeatability check against the earlier direct bank. At the same 218.693-second expected historical candidate-CPU replay, independent whole-G8 inclusion has conditional batch MSE 0.000635904, the negatively correlated fixed pairs 0.000631059, and a same-marginal 18/19-group without-replacement control 0.000621126. Pairing is only 0.762% better than independent inclusion and 1.599% worse than the stronger control, failing the frozen 10% dual-control screen. The compact bank stores pair Grams and the squared norm of the residual sum but not all cross-group inner products, so the without-replacement risk is reproducible conditional on its recorded sum norm. This closes the fixed pairing on this development cohort, not vector-balanced sampling generally.

Full residual geometry and hindsight ceiling. A separately frozen diagnostic subsequently recomputed all 32 direct residuals and their full 32×32 FP64 Gram at the same initial adapter. Its independent source and numerical audit passed. On this repeat, independent and 18/19-group risks are 0.000635721 and 0.000620943; the original SQL-feature pairs have risk 0.000630869. If both the matching and each pair's Bernoulli coupling are selected *after observing the true residual Gram*, the exact best disjoint-pair design at the fixed marginal probability has risk 0.000487026, 21.57% below the without-replacement control. This passes the prespecified hindsight-geometry screen. Yet even an oracle choice of coupling for each *originally fixed* SQL-feature pair attains only 0.000606211, a 2.37% improvement over without-replacement sampling, below the 10% gate. The full optimum is an inaccessible ceiling for this one exposed cohort: it does not supply a pre-verification matching rule, a measured policy cost, independent confirmation, or an online improvement. The calculation covers independent two-member blocks at this probability; auxiliary-feature balanced sampling has prior coverage (Chen et al., 2025).

Exploratory APPS online pilot. An additional exploratory APPS pilot completed four online LoRA updates per arm, each with eight prompts and eight responses per prompt (32 prompts per arm in total). The full-verification arm bought nine full labels and the independent random whole-group $q = 0.5$ arm bought one; their walls were 612.513 and 607.467 seconds on different GPUs. On the same 32 development problems with one sampled answer per problem, base/full/group pass@1 counts were 0/1/2. There was only one discordant problem between trained arms (exact paired two-sided $p = 1$). This single-seed, sparse-label pilot does not establish a correctness gain. The prespecified gates did not admit multi-seed training or independent transfer evaluation.

Hindsight allocation bound on the BIRD final test. An inaccessible allocation bound quantifies the missing risk information. At each resident pass's observed expected budget, choosing group probabilities from the complete squared gradients and realized costs, with the same 0.05 floor, gives four-batch oracle/Bernoulli ratios 0.0120 and 0.0128. Only 12, 15, 22 and 15 groups in A–D have

nonzero complete gradients; the other groups have identical binary rewards and hence zero normalized advantages. On B–D the measured budgets can cover every nonzero-gradient group, leaving zero oracle variance within those batches. This hindsight bound is not an implementable verifier policy: its allocation inputs include the reference labels and actual group costs.

A development-only attempt to predict that risk from five pre-verification SQL features fit ridge models for log squared group-gradient norm and log verifier cost on the earlier A–D cohorts. In 16 leave-one-database-out folds, the resulting allocation had pooled MSE ratios 0.901 to batch-cost-matched Bernoulli and 0.993 to a cost-only rule given the same features. It beat the cost-only comparator on nine of 16 held-out databases; paired bootstrap intervals include one. The apparent 0.746 ratio against Bernoulli on the already examined final-test groups was inspected after method selection and cannot serve as independent confirmation. Together with the hindsight bound, this original log-risk screen does not establish an implementable improvement. Section 7 reports the subsequent arithmetic-risk repair and its separate new-database test.

E Fixed-Rollout Post-Update Gradient Sensitivity

The initial-gradient bank has ratio-one policy likelihoods and no clipping. To test parameter dependence, we reused the initial-policy B, C and D rollouts and their full labels, computed behavior log probabilities at the common initial adapter, then differentiated the clipped surrogate at the adapter saved after A's complete LoRA update. This produced three additional 32-group, 7.37-million-coordinate gradient banks without generation, SQL or another optimizer step. At the two resident cost passes, the respective candidate/Bernoulli ratios are 0.872/0.856 (B), 1.772/1.905 (C) and 0.100/0.106 (D). The equally weighted B–D estimator has ratios 1.049/1.049, versus 1.089/1.090 at the initial adapter on those same B–D groups. Both parameter states lose on B–D; batch A supplies the four-batch initial positive point estimate. Clipping suppressed the gradient of just one of 23,861 valid nonzero-advantage tokens. This is a sensitivity check on saved old-policy rollouts, not new-policy sampling or a test with substantial clipping activity.

F Complete-Audit Cost Replication

Every one of the 128 full groups was rerun twice in randomized order with the same SQL endpoint, giving 256 additional complete audits. All 256 label vectors matched saved full references. The two separate timing passes give four-batch candidate/Bernoulli ratios 1.082 and 1.102 at full-child CPU, versus 0.734 and 0.737 at SQL-body CPU. A paired 13-database cluster bootstrap on the original timing pass gives 95% percentile intervals [0.373,1.840] at full-child CPU and [0.232,1.658] at body CPU. Of 20,000 resamples, 19,292 contain at least one group from every fixed batch; a batch with matched control inclusion probability one contributes zero control variance and is retained. The bootstrap conditions on measured costs and gradients.

G Resident-Verifier Cost Accounting

The same 128 full groups were verified twice more in randomized order, keeping one Python verification process alive for each 32-group batch. All 256 reward vectors matched references. Measured group CPU includes gold and eight candidate SQL executions, dispatch and label checks and sums to 79.339/80.955 seconds. The unchanged frozen allocation's batch ratios are 0.121/0.120 (A), 0.863/0.846 (B), 1.949/2.095 (C), and 0.103/0.109 (D). Paired database-cluster bootstrap intervals are [0.220,1.724] and [0.214,1.779]. Separate database staging consumed 218.827/222.499 CPU seconds before startup and imports. Under this eager-staging schedule, the frozen probabilities would remove an expected 27.617/28.645 group CPU seconds, 9.11%/9.28% of measured post-validation verifier CPU if staging and other work remain fixed. These are accounting estimates, not executed selective-policy timings. The diagnostic reuses the same tasks and initial gradients after their outcomes were seen. It executes SQL calls in one resident Python process without the original per-query child-process isolation; label parity does not establish production timeout or fault isolation.

H B8 COMPLETE-UPDATE TIMING

Batched generation and a realized random audit. That B8 full-label reference generated all 32 G8 groups in four batched model calls, performed 288 SQL jobs (32 gold and 256 candidates), and completed one LoRA update. Its 48.240 seconds of generation and 176.585 seconds of explicit verifier wait show that faster model generation can expose the verifier under this batch-end schedule. Two separately frozen full/clairvoyant-zero timing pairs with exact rollout and label parity observed full-minus-zero cold differences of 175.151 and 177.986 seconds; the zero arms are not deployable. An attempted one-batch-lookahead full-label control failed the prespecified label-parity gate when one previously passing candidate hit the original 30-second timeout, so it provides no complete-update timing contrast.

For a separate realized update, we froze 32 independent Bernoulli$(1/2)$ whole-group decisions before generation. The realized mask bought 15 groups; all 32 token/mask/text/SQL rollouts exactly matched the B8 full reference, and every bought group performed fresh SQL whose binary labels agreed with that reference. Each bought response loss was multiplied by two before backward; unbought groups contributed zero, and one global clip and AdamW step followed. This is the established unbiased *pre-clipping gradient* whole-group baseline under fixed labels, not a new estimator. The complete run and controller census passed (Table 10). Relative to the earlier full reference, its one observed cold interval was 117.155 seconds (29.630%) shorter. The policy skipped 17 groups' old-policy scoring and backward as well as SQL, so this single-order difference cannot be assigned to verification alone. Both saved adapters changed all 288 LoRA tensors, but their update-vector cosine was 0.5604 and their difference was 93.84% of the full update norm. One mask does not estimate gradient MSE, the nonlinear update is not unbiased, and this comparison does not demonstrate learning gain.

B8 one-step arm	SQL jobs	Cold wall (s)	Explicit wait (s)	SQL child CPU (s)
Full reference	288	395.390	176.585	685.455
Whole-group $q = 0.5$ (15 selected)	135	278.235	116.468	277.657

Table 10: Observed B8 full versus one randomized whole-group LoRA update on the same 32 generated G8 rollouts. The full run preceded the randomized run; no randomized timing crossover or independent mask replication was performed. Unselected groups skip verifier and backward work.

Additional timing context. Five earlier full-label global-four-job controls exposed at most 8.28% explicit verifier wait per cold update, below the frozen 15% screen. Their five zero-SQL deletion contrasts changed cold time by -0.94% to 12.24%; none crossed a separately frozen 15% exploratory screen. Those controls and their cohort selection are detailed in Appendix D. The B8 batch schedule exposed 176.585 seconds of wait, but its one randomized whole-group update changes both verifier and backward work.

I GRADIENT-BANK NUMERICAL VALIDATION

All B8 groups were saved before the post-run recipe check failed: it compared a measured gradient-norm result with a configuration field of the same name. A read-only revalidator checks the source identity, raw Gram matrix, arithmetic and update-norm bridge. The original failed receipt is retained. For development A–D, reconstructed complete-gradient norms differ from the recorded pre-clip norms by at most 7.3×10^{-6}. One original receipt completed; three remain failed because a bit-exact risk-dictionary comparison rejected floating summation differences of at most 4.4×10^{-19}. A separate numerical audit checks the source identities, Gram geometry, costs and risk formulas with tolerance. These audited arrays support the fixed-gradient comparisons; the failed post-run checks remain part of the reproducibility record. The replayed information-cost fractions are recomputed from those same arrays and timers by `scripts/information_value_gate.py`.

Fixed-batch policy	Expected SQL CPU replay (s)	Gradient MSE
Two-stage telescoping, then $q = 0.5$	437.401	0.000918042
Whole group, Bernoulli $q = 0.6381185$	437.401	0.000466391
Whole group, fixed-size 20/21	437.401	0.000458863
Same-prefix whole-group continuation	437.401	0.000918042

Table 11: Conditional MSE from directly measured initial-adapter LoRA gradients on the fixed B8 development batch. Costs replay the full run's historical SQL child CPU; they are neither timed randomized-policy costs nor training wall times. The same-prefix control uses the same first label and continuation coin, hence exactly the same two estimator branches.

Batch	Expected CPU (s)	Matched q	Two-stage MSE	Bernoulli MSE	Fixed-size MSE
A	254.292	0.6361	0.001381	0.000547	0.000539
B	60.612	0.6230	0.001137	0.000489	0.000485
C	19.239	0.6172	0.001627	0.000588	0.000577
D	17.461	0.6145	0.001106	0.000502	0.000496

Table 12: Four additional fixed-gradient BIRD training batches, each 32×G8. Expected cost is a replay of each original full run's SQL child CPU; no randomized policy was timed. The same-prefix continuation exactly equals the two-stage risk in every row. Cohorts were previously used for timing diagnostics and are not a sealed benchmark test.

J Detailed Scope and Limitations

The reference suites and SQL endpoint can mislabel semantic correctness. Nested binary tests model cheap-verifier false positives, not arbitrary false-negative channels. Single-update training may not activate ratio clipping, and an unbiased pre-clip gradient can still have high variance or change after norm clipping and AdamW. The five partial-audit BIRD batches are training-side diagnostics: B8 is conditionally selected, and A–D were previously inspected for timing. Their task IDs are disjoint but they are not independent confirmation. The separate final test uses 128 fresh task IDs from 13 previously unused databases, yet one rollout seed and one initial adapter; it does not measure policy drift or online learning. Its cost-matched controls use retrospectively calibrated probabilities, while one frozen mask per arm was actually run. The B8 random whole-group comparison skips both verification and backward on unselected groups, so its wall difference does not isolate SQL. Single-run zero-SQL contrasts also confound host load, cache and run order.

The cost repeats reuse the same rollouts and initial gradients; resident group CPU excludes database staging, and cluster intervals cross one. The post-update diagnostic has little active clipping. The 64-task risk-transfer test has eight database clusters and one saved gradient bank; its amendment followed observed prelabel budget saturation. These diagnostics do not establish new-policy or downstream learning gains.

K Principal Artifact Paths

The APPS development comparison and expected-cost curves are recorded in `results/offline/comparison.json`; split summaries are `results/calibration/summary.json` and `results/dev/summary.json`. The original BIRD trace's census and primary cold denominator are in `results/model-scale-natural-cost-20260924/trace3b/attempt-001/controller-census.json`. The conditional single-database update and direct-gradient diagnostic are analyzed in `reports/model-scale-natural-cost-20260924/BIRD_HCJGF_STAGE1_ANALYSIS.json` and `reports/model-scale-natural-cost-20260924/BIRD_HCJGF_STATE_MSE_ANALYSIS.json`. The paired direct-gradient bank and audit are in `results/model-scale-natural-cost-20260924/bird-hcjgf-pair-gram-001/` and `reports/model-scale-natural-cost-20260924/BIRD_HCJGF_PAIR_GRAM_RESULT_AUDIT.md`; the full residual Gram and independent audit are in `results/`

model-scale-natural-cost-20260924/bird-hcjgf-full-gram-001/ and reports/model-scale-natural-cost-20260924/BIRD_HCJGF_FULL_GRAM_RESULT_AUDIT.md.

Under results/model-scale-natural-cost-20260924/, the full-label scheduling control is in bird-hcjgf-sched4/attempt-001/; the four additional controls are in bird-source-breadth-sched4/bin-A/attempt-001/ and the corresponding B–D directories. The oracle records are in bird-source-breadth-oracle0/bin-A/attempt-001/ and the corresponding B–D directories, plus bird-hcjgf-oracle0/attempt-001/. Each oracle directory contains receipt.json, controller-census.json, oracle-timing-comparison.json, oracle-cgroup.json, raw rollouts and pre/post-update artifacts. These paths identify the local research archive; they are not public download links.

The B8 full and realized random-audit updates are in results/model-scale-natural-cost-20260924/bird-hcjgf-b8-full/attempt-001/ and results/model-scale-natural-cost-20260924/bird-hcjgf-b8-q05/attempt-001/. The direct-gradient replay is in results/model-scale-natural-cost-20260924/bird-hcjgf-b8-rt-gram-001/; its analysis.json, grams.npz, audit.jsonl, and failed receipt.json must be read together. The read-only revalidator is configs/model-scale-natural-cost-20260924/audit_bird_hcjgf_b8_rt_raw.py, with output reports/model-scale-natural-cost-20260924/BIRD_HCJGF_B8_RT_RAW_REVALIDATION.json; it reports PASS_SCOPED_RAW_DIAGNOSTIC while preserving the original failed receipt. The four task-disjoint gradient banks are under results/model-scale-natural-cost-20260924/bird-breadth-rt-confirm/; their joint numerical audit is reports/model-scale-natural-cost-20260924/BIRD_BREADTH_RT_CONFIRM_RAW_REVALIDATION.json. The APPS exploratory online logs are in results/urgent-online-pilot-20260925/.

The final-test full-reference and direct-gradient banks are under results/model-scale-natural-cost-20260924/bird-final-test/ and results/model-scale-natural-cost-20260924/bird-final-test-gradient/. Executed selective audits and the A12/B20 cost completions are under results/model-scale-natural-cost-20260924/bird-final-test-actual-audit/. The exact conditional risks and group-level actual-cost ledger are reports/model-scale-natural-cost-20260924/BIRD_FINAL_TEST_RESULTS.json and reports/model-scale-natural-cost-20260924/BIRD_FINAL_TEST_COST_COMPLETION.json; the independent scoped integrity verdict is reports/model-scale-natural-cost-20260924/EXPERIMENT_AUDIT.md. The selected-after-test risk predictor is preserved separately in reports/model-scale-natural-cost-20260924/RISK_COST_DEVELOPMENT_GATE.json and is not submission-facing method evidence. The randomized-order cost-repeat ledger is reports/model-scale-natural-cost-20260924/BIRD_FINAL_TEST_COST_REPLICATION_R2.json.

L A Restricted Hard-Cap Counterexample

This short construction illustrates the query-degree obstruction already covered by Zhang (2026); it is included for self-containment, not claimed as a new theorem. Fix the first of four binary labels to zero and leave three unknown. Let f be the first normalized advantage, with $\epsilon = 0$ and the all-zero case defined as zero. If k unknown labels are one, then $f(0) = 0$, $f(1) = -1/2$, $f(2) = -\sqrt{3}/2$, and $f(3) = -3/2$. Its third mixed difference is $3\sqrt{3}/2 - 3 \neq 0$.

A deterministic decision tree that reads at most two labels has a leaf-indicator representation involving products of at most two queried binary variables or their complements. Thus its output is a multilinear polynomial of degree at most two. Under integrability, the expectation of a randomized tree retains this degree bound. It therefore cannot equal f for all eight assignments. This is a restricted impossibility example, not a general optimality theorem or a proof that every unbiased policy-gradient scheme must query a complete group.

M Conditional Variance and the Full-Group Comparator

Let $d_k = \nabla \ell_k - \nabla \ell_{k-1}$, conditioning on the query order as well as the group. For nested indicators $I_k = \mathbf{1}\{N \geq k\}$, $\mathbb{E}[I_k I_l] = p_{\max(k,l)}$. Therefore

$$\mathbb{E}\|\widehat{g} - g^\star\|_2^2 = \sum_{k,l=1}^{m} \left(\frac{1}{p_{\min(k,l)}} - 1 \right) \langle d_k, d_l \rangle. \tag{8}$$

This follows by expanding the squared norm of $\sum_k (I_k/p_k - 1) d_k$; it is a direct property of the established estimator, not a new variance theorem. In comparison, whole-group residual auditing has mean squared error $(1/q - 1)\|g^\star - g_0\|_2^2$. Cross terms can have either sign. Consequently, a ranking based on individual coefficient changes need not minimize parameter-gradient error or dominate whole-group auditing. We evaluate this possibility rather than assume a benefit.

N Scope of the Fixed-Snapshot Gradient Diagnostic

At the initial snapshot, old and current raw-model log probabilities agree, so ratios are one and clipping is inactive. Writing h_i for the parameter gradient of response i's mean token log probability gives $g(r) = \sum_i [-A_i(r)/G] h_i$. The stored Gram matrix $H_{ij} = \langle h_i, h_j \rangle$ thus gives exact squared parameter-gradient distances for any audited reward state, up to floating-point arithmetic. Audit stopping states are enumerated; random-order comparisons average 32 sampled permutations and retain that approximation. The diagnostic does not establish behavior after policy drift or with active clipping.

Groups with no unknown labels remain in all aggregate comparisons. Under our nested tests their rewards and gradients are all zero, so actor derivatives need not be computed for those groups. No synthetic zero Gram matrix is presented as a measurement.

O Post-hoc Diagnostic Checks

Only ten development groups have multiple unknown labels. The influence factor changes the error/cost query order in three groups. On the 34 cheap-passing development candidates, the error predictor's Brier score is 0.22776, versus 0.19574 for the calibration-set constant predictor. This diagnosis does not justify tuning on development outcomes.

At an expected query fraction of one half, we exhaustively enumerate query permutations under the fixed survival design using hidden reference labels and measured parameter-gradient Gram matrices. Even this inaccessible per-task oracle ordering has aggregate MSE 0.001917, compared with 0.001646 for full-group residual correction. These points match expected query counts, *not actual CPU*: oracle and full-group mean incremental CPU are 0.03601 and 0.03705 seconds respectively. This is a post-hoc diagnostic of ordering headroom, not a deployable policy or a general optimality result.

The calibration pilot was interrupted for directory migration after 36 complete groups. The completed prefix was preserved; because that legacy process lacked an RNG checkpoint, the remaining 24 groups use a separately recorded seed, 20260955. No task was selected or removed using its outcome. All subsequent checkpoint changes and the segment boundary are recorded in the code and run provenance.