TIMEOUT IS NOT A NEGATIVE LABEL LEARNING FROM BUDGET-CENSORED VERIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

A training timeout can leave a program's outcome unknown at the verification limit that defines task success. We study randomized verification allocation that preserves the policy gradient of this reference objective. Building on inverse weighting and randomized truncation, we derive the allocation criterion for the smooth improvement bound of an ordinary policy minibatch. Finite batching adds a squared-mean-gradient term to the usual cost–second-moment product, which can change the optimal allocation. Under an unknown trace law and a fixed observation kernel, universally unbiased conditional allocation feedback requires two independent records for the usual product and, in general, three for the finite-batch criterion. Distinct-record estimators attain these orders using only censored observations. In a high-signal control, the optimal verification probability changes from one to approximately 0.112, reducing the finite-batch objective by 47.6%. Censored feedback learns this allocation, although a plug-in estimator is more efficient in the stationary control. A separate executed-program study requires delayed verification for optimal program choice on half the held-out prompts. Negative timeout labels exclude those choices. Checkpoint analysis and exact update moments clarify when allocation improvements have an efficiency interpretation. These results characterize observable allocation feedback without implying a general training-speed advantage.

1 INTRODUCTION

A verifier may stop a generated program when training exhausts its allotted effort, even though the program would pass within the limit that defines task success. Labeling this interruption as failure changes the learning objective. Discarding the program can instead change the sample distribution. At the task's reference limit, an unfinished verification is a failure by definition.

Randomizing the training cutoff and recording its probability permits an unbiased correction. Choosing the randomization requires more information. The learner observes the executed prefix, its charge, and any endpoint reached. It does not observe the completion time or reward beyond an interruption. Allocation must therefore be learned from partial feedback, using a criterion appropriate for the policy update.

The basic ingredients are established. Inverse-inclusion weighting is classical (Horvitz & Thompson, 1952; Robins et al., 1994). The terminal correction is a sparse randomized telescope, and Beatson & Adams (2019) connect unbiased truncation, online tuning, and expected improvement per computation. We ask which allocation gradients admit exactly conditionally unbiased estimates from the current policy's censored records. Historical plug-in estimates can also guide allocation, but need not be unbiased current-batch gradients.

The answer depends on the update. Write C for expected charge, M for the corrected gradient's second moment, and γ for its mean. For an ordinary mean of n independent policy records, the smooth-improvement criterion is $C[M + (n-1)\|\gamma\|^2]$, rather than CM. The extra term can change the optimum even with one checkpoint and positive success probability for every action. Under the stipulated unknown-law observation model, universally unbiased allocation feedback requires two records for the usual product and can require three for the finite-batch criterion. Distinct-record estimators attain these orders. The polynomial-degree obstruction and independent-product

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.

construction are classical (Halmos, 1946; Hoeffding, 1948). Our contribution is the corresponding characterization for these computation-aware policy functionals under censored observations.

Two mechanism studies test these implications. A high-signal terminal-observation experiment separates the allocation optima and compares direct feedback with plug-in adaptation. A program-selection experiment requires slower, more reliable candidates for optimal choices on half the prompts. Every checkpoint remains informative within each observable stratum. Negative timeout labels exclude useful choices, rather than perturbing an objective with unchanged maximizers. Full verification and a corrected fixed schedule remain strong competitors. The evidence supports the feedback classification and the need to preserve the reference objective, not a universal training-speed advantage.

2 Reference rewards and randomized verification

A prompt x is drawn from a fixed distribution $\mathcal{D}$, and a complete program a is drawn from $\pi_\theta(\cdot \mid x)$. Conditional on (x, a), a potential verification trace has terminal time $T \in (0, \infty]$ and terminal pass indicator Y. The task fixes a reference budget $B_\star > 0$ and objective

$$R = Y\mathbf{1}\{T \le B_\star\}, \qquad \tau = \min\{T, B_\star\}, \qquad J(\theta) = \mathbb{E}_\theta R. \tag{1}$$

For $T = \infty$, the product defining R is zero. At time $B_\star$, an unfinished verification has the *observed* reference reward zero. Thus (τ, R) always has a finite endpoint, including a possible administrative endpoint at the cap.

Fix checkpoints $0 = b_0 < b_1 < \cdots < b_K = B_\star$ and a pre-verification stratum $h = h(x, a) \in [H]$. Before round t, the history $\mathcal{F}_{t-1}$ determines θ_t, a baseline $\beta_t(x) \in [0, 1]$, and log-survivals u_t in

$$\mathcal{U} = \{u \in \mathbb{R}^{H\times K} : 0 \ge u_{h1} \ge \cdots \ge u_{hK} \ge \log\epsilon\}, \quad 0 < \epsilon < 1. \tag{2}$$

Write $s_{hk} = e^{u_{hk}}$. After sampling (x, a), draw an independent $U \sim \mathrm{Unif}(0, 1)$ and set $L = \sum_{k=1}^{K} \mathbf{1}\{U \le s_{hk}\}$. Verification receives budget b_L, where $L = 0$ skips the check but still incurs generation and update overhead. Its endpoint is observed precisely when

$$j = \min\{k : \tau \le b_k\}, \qquad O = \mathbf{1}\{L \ge j\}. \tag{3}$$

When $O = 0$, the implementation does not evaluate the latent index j.

Assumption 1 (Randomization and policy regularity)**.** Conditional on the past and (x, a), U is independent of the potential trace. The conditional trace law has no further dependence on θ. The policy has a common support, is differentiable, and differentiation may pass under the expectation. Its score $\psi_\theta(x, a) = \nabla_\theta \log \pi_\theta(a \mid x)$ satisfies $\|\psi_\theta\| \le G$ on all iterates considered. A batch uses conditionally independent draws of the entire prompt–program–trace experiment. The allocation statements specify whether two or at least three draws are needed.

Define the latent centered score $Z = \psi_\theta(x, a)(R - \beta(x))$. The observable estimators are

$$\widehat{R} = \frac{OR}{s_{hj}}, \qquad g = \frac{OZ}{s_{hj}}, \tag{4}$$

with both expressions set to zero when $O = 0$. All weights and baselines are held fixed when forming the policy update.

Proposition 1 (Objective preservation)**.** *Under Assumption 1,* $\mathbb{E}[\widehat{R} \mid x, a, \tau, R, \mathcal{F}_{t-1}] = R$ *and* $\mathbb{E}[g \mid \mathcal{F}_{t-1}] = \nabla J(\theta_t)$*. These statements hold for history-adaptive schedules and arbitrary predictable prompt-only baselines.*

The proof conditions on the full trace, uses $\mathbb{E}[O \mid x, a, \tau, R, \mathcal{F}_{t-1}] = s_{hj}$, and applies the score identity (Williams, 1992). A successful endpoint reached with probability $1/4$, for example, contributes reward 4 when observed and zero otherwise. Its mean contribution is one; the zero branch does not label an unseen endpoint as failure. A program-dependent baseline generally needs an additional correction. Proposition 1 covers neither censoring based on an unlogged partial trace nor clipped inverse weights. Appendix D gives the proof and endpoint conventions.

Weighting the whole centered score in (4) yields the nonnegative second-moment expansion below. The alternative $\psi(OR/s_{hj} - \beta)$ is also unbiased and retains the baseline on censored draws, but need not admit that expansion. Weighting the baseline is a design choice, not a condition for unbiasedness.

Assumption 2 (Smooth reference objective)**.** The objective J is $\mathcal{L}$-smooth on all update segments and $0 \le J \le 1$. Set $\Delta = \sup_\theta J(\theta) - J(\theta_1)$.

3 WHAT SHOULD AN ALLOCATION OPTIMIZE?

We use a logical work model before discussing measured runtime. A draw has fixed overhead $c_0 > 0$ for non-verification operations and potential interval work $W_k = \min\{\tau, b_k\} - \min\{\tau, b_{k-1}\}$. The observed work and charge are

$$v_{hk} = \mathbf{1}\{h(x,a) = h, L \ge k\} W_k, \qquad D = C^{\mathrm{obs}} = c_0 + \sum_{h,k} v_{hk} \le C_{\max} := c_0 + B_\star. \tag{5}$$

The fixed overhead abstracts non-verification work; actual allocation costs can vary. For the current policy and baseline, define the unknown population coefficients

$$a_{hk} = \mathbb{E}[\|Z\|^2 \mathbf{1}\{h(x,a) = h, j = k\}], \qquad c_{hk} = \mathbb{E}[W_k \mathbf{1}\{h(x,a) = h\}]. \tag{6}$$

Conditioning on the potential trace yields

$$M(u) = \mathbb{E}\|g\|^2 = \sum_{h,k} a_{hk} e^{-u_{hk}}, \qquad C(u) = \mathbb{E}D = c_0 + \sum_{h,k} c_{hk} e^{u_{hk}}. \tag{7}$$

The familiar truncation objective is $F = CM$ (Rhee & Glynn, 2015; Beatson & Adams, 2019; Cui et al., 2021; Zheng et al., 2026). Flattening the coordinates gives the useful expansion

$$F(u) = c_0 \sum_j a_j e^{-u_j} + \sum_{i,j} c_i a_j e^{u_i - u_j}. \tag{8}$$

It is convex because every term has a nonnegative coefficient. It is not the exact variance–cost product $C(M - \|\nabla J\|^2)$.

Theorem 2 (Allocation for a finite minibatch)**.** *Let $\gamma = \nabla J(\theta)$ and let $\bar{g} = n^{-1}\sum_{i=1}^n g_i$ use independent records. Suppose J is $\mathcal{L}$-smooth, $\mathcal{L} > 0$, and the unconstrained maximizing step lies in the smooth domain. For $\gamma \ne 0$, maximizing the smooth quadratic lower bound on $\mathbb{E}[J(\theta + \eta\bar{g}) - J(\theta)]/(nC)$ over admissible $\eta \ge 0$ gives*

$$\mathcal{B}_n^*(u) = \frac{\|\gamma\|^4}{2\mathcal{L}\Phi_n(u)}, \qquad \Phi_n(u) = C(u)\{M(u) + (n-1)\|\gamma\|^2\}. \tag{9}$$

The allocation criterion Φ_n is convex on $\mathcal{U}$. A minimizer of F is within a factor n of the optimal value of Φ_n.

Theorem 2 applies the smooth-improvement calculation of Beatson & Adams (2019) to an averaged minibatch with second moment $[M + (n-1)\|\gamma\|^2]/n$. The calculation uses the standard descent lemma and does not assert that the lower bound is attained. The additional unknown squared mean raises the required order of unbiased allocation feedback. Appendix A.2 gives the optimizing step, degenerate cases, and sharper factor $1 + (n-1)\|\gamma\|^2/M(u_F)$.

3.1 WHEN FINITE BATCHING CHANGES THE OPTIMUM

Proposition 3 (One-checkpoint allocation regimes)**.** *Suppose every verification takes time $d > 0$, a common survival $q \in [\epsilon, 1]$ either runs it or skips it, and $a = \mathbb{E}\|Z\|^2 > 0$. Put $\kappa = c_0/d$ and $\xi = (n-1)\|\gamma\|^2/a$. The usual product F is minimized at $q_F = 1$. The finite-batch criterion is minimized at*

$$q_n^* = \begin{cases} 1, & \xi \le \kappa, \\ \sqrt{\kappa/\xi}, & \kappa < \xi < \kappa/\epsilon^2, \\ \epsilon, & \xi \ge \kappa/\epsilon^2. \end{cases} \tag{10}$$

In the interior regime, $\Phi_n(1) - \Phi_n(q_n^*) = ad(\sqrt{\xi} - \sqrt{\kappa})^2$.

For a scalar logistic policy with $\pi(A) = 0.9$, success probabilities $(0.99, 0.01)$, zero baseline, $d = 1$, $c_0 = 0.02$, and $n = 3$, we have $a = 0.00972$ and $\gamma = 0.0882$. Thus $q_F = 1$ but $q_3^* = 0.111780$. The finite-batch objective falls from 0.02578405 to 0.01350943, or 47.6%. Every action has positive reward mass, so the separation does not rely on an unproductive stratum. Appendix A.3 proves the formula and handles the boundaries. The experiment in Section 7 tests whether censored feedback learns the changed allocation.

4 LEARNING THE ALLOCATION FROM CENSORED RECORDS

An observed record is the tuple (v, D, e, g, m), where v is executed interval work, $D = c_0 + \sum v$, e is the reached-endpoint indicator vector, and $m = \|g\|^2$. For example, take checkpoints $(1, 4)$, survivals $(1, 1/4)$, and a potential successful endpoint at time three. A cutoff at one returns $v = (1, 0)$, $D = c_0 + 1$, and $e = g = m = 0$. A full cutoff returns $v = (1, 2)$, $D = c_0 + 3$, $e = (0, 1)$, $g = 4\psi$, and $m = 16\|\psi\|^2$, for zero baseline. The interrupted record contains no information about its eventual endpoint beyond the observed prefix.

The moment identities are

$$\mathbb{E}v = \nabla C, \quad \mathbb{E}D = C, \quad \mathbb{E}m = M, \quad \mathbb{E}(me) = -\nabla M. \tag{11}$$

A single-record product need not factor into these expectations. For example, $\mathbb{E}[mv] = M\nabla C + \mathrm{Cov}(m, v)$. Independent records remove the covariance terms and give

$$H_F^{(2)} = \tfrac{1}{2}\{m_2(v_1 - D_1 e_2) + m_1(v_2 - D_2 e_1)\}. \tag{12}$$

Theorem 4 (Paired feedback)**.** *Conditional on the past, $\mathbb{E}H_F^{(2)} = \nabla F$ and*

$$\mathbb{E}\|H_F^{(2)}\|^2 \le V = 4C_{\max}^2 G^4 \epsilon^{-3}. \tag{13}$$

The larger-batch version $H_F^{(n)}$ averages $m_j(v_i - D_i e_j)$ over ordered pairs $i \neq j$. Distinct records factor the unknown moments; reusing one record introduces covariance bias. Independence must hold across whole prompt experiments. Two completions of a shared random prompt generally target $\mathbb{E}_x[C_x M_x]$, not $(\mathbb{E}_x C_x)(\mathbb{E}_x M_x)$.

Theorem 5 (Batch-aware feedback and minimum record order)**.** *For $n \ge 3$, put*

$$H_\Phi^{(n)} = H_F^{(n)} + \frac{n-1}{n(n-1)(n-2)} \sum_{i,j,k \text{ distinct}} v_i \langle g_j, g_k \rangle. \tag{14}$$

Then $\mathbb{E}H_\Phi^{(n)} = \nabla\Phi_n$. It is computable in $O(n(d + HK))$ arithmetic after the gradients are available, where $d = \dim\theta$. With a fixed observation kernel and no supplied unknown-law coefficients, two records are necessary for universally unbiased feedback for ∇F, and three are necessary for $\nabla\Phi_n$ when $n > 1$.

The minimum-order statement concerns exact conditional unbiasedness, not the performance of all adaptive methods. The proof constructs trace-law mixtures whose allocation gradients are quadratic and cubic in the mixing probability, respectively. One- and two-record estimators cannot match the corresponding higher-degree polynomials. A distinct-index U-statistic (Halmos, 1946; Hoeffding, 1948) attains each required order. These lower bounds apply to the general unknown-law class. Known work moments or other restrictions can reduce the order. Our experiments also compare historical plug-in estimation, to which this lower bound does not apply.

Algorithm 1 Batch-aware allocation from terminal censored feedback

1: Choose $n \ge 3$, $u_1 \in \mathcal{U}$, θ_1, and a charged-work budget Q.
2: **while** remaining charge is at least $nC_{\max}$ **do**
3: Freeze the policy, baseline, and allocation. Draw n independent prompts and programs.
4: Randomize cutoffs, execute only assigned prefixes, and retain $(g_i, m_i, v_i, D_i, e_i)$.
5: Set $u_{t+1} = \Pi_{\mathcal{U}}(u_t - \alpha H_{\Phi,t}^{(n)})$.
6: Set $\theta_{t+1} = \theta_t + \eta n^{-1} \sum_i g_i$ and deduct $\sum_i D_i$.
7: Update any baseline using only the now-completed history.
8: **end while**

Projection applies pooled adjacent violators within each stratum and clips the result to $[\log\epsilon, 0]$, using $O(HK)$ operations. All n records contribute to the policy step, and no complete pilot trace is needed. Appendix A gives the linear-time feedback formula and proofs. Standard projected online gradient descent yields a changing-policy comparator bound for Φ_n, with an explicit path-length term. Reserving $nC_{\max}$ before each batch gives a hard logical-work cap and a cost-weighted stationarity bound for the mean update. The comparators are evaluated at the learner's policies, not along separate training trajectories. Appendix A.6 states the bounds and constants.

5 OPERATIONAL CONSEQUENCES AND LIMITS

Checkpoint structure is a separate source of value. With one stratum and two checkpoints, positive signal coefficients $(a_1, a_2) = (0.125, 0.015)$, work coefficients $(c_1, c_2) = (0.1, 0.45)$, and $c_0 = 0.05$, endpoint-only thinning is minimized by full verification with $F = 0.084$. A checkpoint schedule $(1, 0.2)$ attains $F = 0.048$. Both intervals contain useful signal. Theorem 8 derives the strict gap and supplies a realizing reward law. This comparison isolates checkpoint structure, whereas Proposition 3 isolates the effect of batching.

Changing the policy update changes the criterion. The cross-cost update $K = (D_1 g_2 + D_2 g_1)/2$ has mean $C\gamma$. Its second moment is $V_K = \frac{1}{2}\mathbb{E}[D^2]M + \frac{1}{2}\|\nu\|^2$, where $\nu = \mathbb{E}[(c_0 + \tau)Z]$ does not depend on the allocation. Its smooth improvement bound per expected charge is ordered by V_K/C, not CM. Proposition 9 gives the derivation and an instance where an 80.9% reduction in CM accompanies a 16.2% increase in V_K/C. The reversal concerns the ordering of bounds, not necessarily that of realized training outcomes. Algorithm 1 uses the ordinary mean update to match (9).

Unbiased rewards do not justify a nonlinear normalization. Unbiasedness need not survive a nonlinear transformation. With one prompt, two actions, success probabilities $(0.9, 0.6)$, and reveal probabilities $(0.1, 1)$, individually unbiased corrected rewards yield a two-record group-normalized update with mean $-0.456p(1-p)$, while the reference gradient is $0.3p(1-p)$. Normalization discards the rare correction's magnitude and retains its ordering. Propositions 10 and 11 characterize the finite-group direction, its stabilizer dependence, and the equal-propensity and large-group cases. This effect concerns fixed-prompt censoring propensities rather than every GRPO implementation. The unnormalized updates used here avoid this transformation.

6 RELATION TO PRIOR WORK

Inverse inclusion weighting (Horvitz & Thompson, 1952), augmentation (Robins et al., 1994), and score-function learning (Williams, 1992) provide the foundations. HyPeR studies off-policy learning with partially observed rewards and secondary signals (Takehi et al., 2025). Censoring-aware reinforcement learning also appears in treatment-regime estimation (Paul & Greiner, 2025). The contribution here concerns verification allocation, not the missing-reward formulation or its standard correction.

The closest optimization precedent is Beatson & Adams (2019). Their randomized-telescope estimator contains our endpoint correction as the sparse-increment case $\Delta_k = Z\mathbf{1}\{j = k\}$. They optimize expected improvement per computation using a cost–second-moment criterion and tune truncation distributions online. Their adaptive procedure estimates intermediate-gradient differences and periodically tunes the distribution. We instead characterize exactly conditionally unbiased allocation feedback from censored terminal records, including the two-versus-three record-order distinction induced by the policy minibatch. The terminal estimator remains a randomized telescope, and a plug-in estimator of the same moments is a valid competitor. Classical unbiased-function estimation and distinct-index statistics supply the record-order arguments (Halmos, 1946; Hoeffding, 1948).

Monte Carlo work studies optimal randomized truncation (Rhee & Glynn, 2015; Cui et al., 2021; Zheng et al., 2026). Online importance-sampling adaptation and dynamic comparison are also established (Borsos et al., 2019; El Hanchi & Stephens, 2020). Our online bounds apply standard projected-gradient arguments (Zinkevich, 2003). For Gaussian processes, Potapczynski et al. (2021) show that removing truncation bias need not improve optimization. Our exact cross-cost moments give a realizable reversal for one cost–second-moment proxy. Contextual rollout allocation (Lu et al., 2026) and imperfect process-verifier backtracking (Rohatgi et al., 2026) concern different decisions from interrupting a stipulated terminal verifier. GRPO (Shao et al., 2024), its optimization-bias analyses (Liu et al., 2025), and REINFORCE leave-one-out methods (Ahmadian et al., 2024) motivate the normalization comparison. The unequal-propensity characterization is proved here. The result-level comparison is summarized in Table 3.

7 EXPERIMENTS THAT DISTINGUISH THE TWO MECHANISMS

The first study fixes the policy and tests which allocation criterion is learned. The second updates a shared policy over executed programs and tests whether useful candidates require delayed verification. We distinguish a lower allocation objective from faster policy learning.

7.1 WHEN FINITE BATCHING CHANGES THE ALLOCATION

We use the positive-reward two-action law of Proposition 3, with one checkpoint, unit verification time, and batch size three. The oracle survival is 0.111780, rather than one for CM. We compare paired-CM feedback, triple-Φ_3 feedback, a running-moment plug-in estimator, and a known-cost pair specialization. Each receives only censored outcomes. Forty seeds are evaluated at both 12,000 records and charged-work budget 1200. The returned schedule averages log-survival over the latter half of the fitting budget. The charge is 0.02 per sampled record plus one when verification runs. No policy update or fitted completion-time model is used in this study.

Table 1: Frozen high-signal control. Returned survival is the exponential of the mean log-survival over the latter half of the fitting budget. Excess is $100(\Phi_3/\Phi_3^* - 1)$; $\Phi_3^* = 0.01350943$. Forty seeds, means $\pm$ 95% intervals for excess. Unit verification cost is known in this control, permitting the known-cost two-record specialization.

Feedback	q, 12k records	Excess (%)	q, work 1200	Excess (%)
Paired CM	1.0000	90.86 ± 0.00	1.0000	90.86 ± 0.00
Triple Φ_3	0.1345	0.57 ± 0.24	0.1463	1.12 ± 0.38
Plug-in Φ_3	0.1123	0.03 ± 0.02	0.1126	0.05 ± 0.03
Known-cost pair	0.1132	0.03 ± 0.01	0.1200	0.15 ± 0.08

The plug-in and known-cost estimators perform better than the generic triple construction in this stationary control. At matched records, the triple estimator nevertheless reaches a criterion 0.57% above the oracle; at matched charge, its excess is 1.12%. Paired-CM feedback stays at full verification, as its criterion requires. Changing the criterion therefore changes the learned allocation. Since verification time is known here, ∇C is known and two records suffice to estimate the added squared mean. This control tests the *difference between criteria*, not three-record necessity for this restricted law. Appendix L reports fitting costs, final-iterate variation, the signal/batch-size/overhead grid, and the known-cost formula.

7.2 WHEN A USEFUL PROGRAM NEEDS DELAYED VERIFICATION

The program study contains 60 modular-summation prompts, 12 candidates per prompt, and 24 input cases per prompt. All 17,280 reference cases are executed. Fast candidates use periodic decomposition; slow candidates sum directly. Input-dependent answer perturbations allow both types to succeed or fail. A task-mode bit reverses their reliability relation on half the prompts. The fixed stratum map $h(x, a) = (x + 2a) \bmod 3$ mixes modes, algorithms, and positive endpoints at every checkpoint. The corpus is constructed to test this mechanism, not sampled from natural code.

Training, development, and test use 36, 12, and 12 disjoint prompts within this grammar. A 48-parameter linear softmax uses action indicators, two numerical prompt descriptors, and the task-mode bit. Development and test each contain six slow-optimal prompts. Labeling all endpoints after fuel 64 as failure removes *every* reference-optimal candidate from the maximizing set on each slow-optimal prompt. The maximizing sets on the other half remain unchanged. This statement concerns the available per-prompt candidates, not the unrestricted parameter optimum of the shared policy. At initialization, the reference and timeout-label gradients on the slow-optimal test subset have cosine -0.614. The all-prompt cosine is positive, so the reversal does not extend to the aggregate gradient.

Seven allocation/label rules use ordinary mean updates with three independent prompt records. Each receives the same seven-rate, three-seed development search. Endpoint-only and checkpoint plug-in comparators fit censored moments of Φ_3, whereas paired feedback targets CM. The fixed corrected schedule is $(1, 0.75, 0.5, 0.25)$. The negative-label control stops at 64 and assigns zero to

interruptions. All executions share the reference cap of 8192. Logical charge is $0.02+$ executed fuel$/8192$, with a three-record worst-case reservation. Source construction, compilation, verification, and allocation run live rather than through trace replay. CPU measurements are diagnostic; the comparisons match logical work.

Table 2: Executed-program study with delayed optimal actions. Reference success (%) on 12 held-out prompts, including 6 slow-optimal prompts. Twelve training seeds and common development search per method. Intervals describe seed variability conditional on this fixed corpus, not variability across natural code tasks.

Allocation / labels	$Q = 180$	$Q = 540$	Slow prompts, $Q = 540$
Full verification	76.95 ± 3.11	77.61 ± 3.32	81.01 ± 4.64
Fixed + correction	79.89 ± 3.70	81.01 ± 3.96	83.48 ± 3.77
Endpoint-only	77.50 ± 3.65	77.54 ± 3.84	80.39 ± 4.63
Paired CM	78.00 ± 3.17	78.37 ± 3.26	82.04 ± 5.16
Batch-aware	77.74 ± 3.06	78.83 ± 3.61	82.15 ± 5.17
Plug-in Φ_3	77.34 ± 3.28	78.35 ± 3.57	77.87 ± 5.19
Short timeout = failure	64.22 ± 1.45	64.83 ± 1.35	48.82 ± 0.16

At charge 540, the fixed corrected schedule has the largest mean held-out reference success, 81.01%. Batch-aware allocation obtains 78.83%, versus 64.83% for negative timeout labels. The paired difference is 13.99 points, with 95% interval $[10.41, 17.57]$. On slow-optimal test prompts, batch-aware allocation places 99.82% mean policy mass on slow actions, compared with 0.50% under negative labels. These results support preserving the reference objective, but not an allocation advantage over simpler correct methods. The batch-aware versus paired-CM difference remains unresolved. Appendix L reports all paired comparisons, development choices, and final schedules.

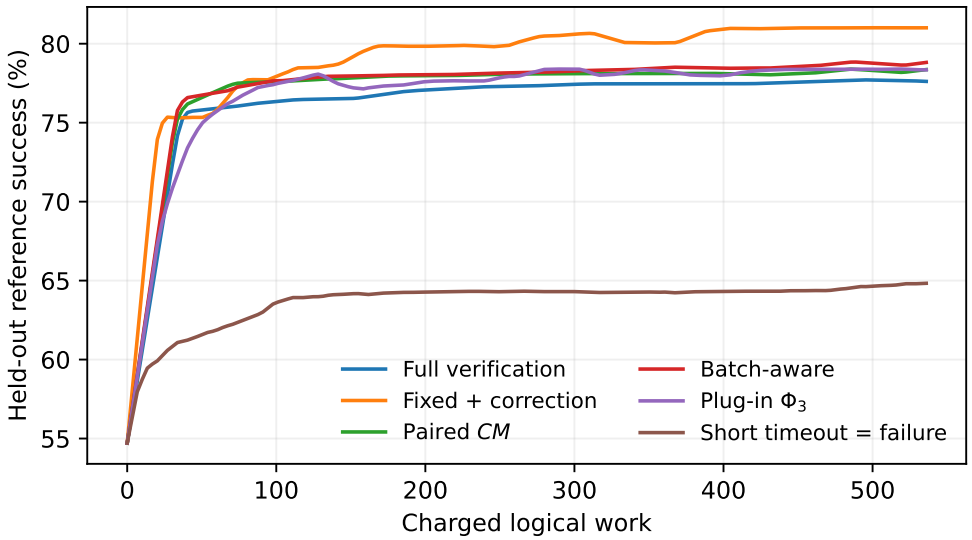

Figure 1: Reference success on held-out prompts in the slow-optimal program study. Curves average twelve training seeds and evaluate stored policy snapshots after training. Table 2 reports endpoint uncertainty. The corrected fixed schedule remains competitive with learned allocation. The horizontal axis is charged logical work, not CPU time.

Controls and numerical consistency. The fast-optimal corpus supports the value of checkpoints but scarcely separates the finite-batch term. Its numerical checkpoint optimum is 40.5% below endpoint-only allocation, whereas its CM optimum is only about 0.003% above the Φ_3 optimum. The stationary plug-in baseline again performs better than direct feedback. An equal-reward diagnostic compares *expected charge times gradient variance*, rather than variance divided by charge.

Appendices M and N retain these controls and the CPU-negative four-prompt experiment. Two plug-in trajectories reveal sensitivity to BLAS threading. Single-thread runs recover the primary values; five-thread runs recover the observer values. The reported trajectories use a single BLAS thread. Appendix L.4 gives the controlled comparison.

8 DISCUSSION

The observations available to an allocator determine which computation-aware gradients it can estimate without bias. Finite batching changes the target functional and, for a general unknown trace law, raises the minimum fresh-record order from two to three. The high-signal control shows that this change can alter the preferred allocation. The program study shows why delayed positive outcomes can be necessary for choosing the best reference-budget candidates.

Exact conditional unbiasedness does not make direct feedback the most efficient estimator or order the final accuracies of different training trajectories. Known work moments can reduce the required record order. Plug-in estimation can exploit stationary history, and a corrected fixed schedule performs well in the program study. Truncation therefore need not replace full verification. The evidence concerns constructed finite programs and fixed features, not natural-code prevalence, learned representations, or large-model training efficiency. The contribution is the allocation functional, its observable-feedback characterization, and the resulting distinctions between objective preservation, allocation, and learning.

AI USE STATEMENT

Generative AI was used for grammatical checks and proofreading. It was also used for reference/literature search and verification. The authors assume full responsibility for the content.

ETHICS STATEMENT

The experiments use constructed arithmetic programs and do not involve human subjects, personal data, or deployment decisions. A reference-budget timeout is always counted as failure. The execution harness is designed only for the included trusted templates and is not a security sandbox for arbitrary model-generated code. Reduced training verification must not be interpreted as permission to weaken deployment checks.

REPRODUCIBILITY STATEMENT

Code and data are provided in the supplement. The appendices contain complete proofs and the experimental methods for the mechanism studies and complementary controls.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, 2024. doi: 10.18653/v1/2024.acl-long.662. URL https://aclanthology.org/2024.acl-long.662/.

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 of *Proceedings of Machine Learning Research*, pp. 534–543, 2019. URL https://proceedings.mlr.press/v97/beatson19a.html.

Zalán Borsos, Sebastian Curi, Kfir Yehuda Levy, and Andreas Krause. Online variance reduction with mixtures. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 705–714, 2019. URL https://proceedings.mlr.press/v97/borsos19a.html.

Zhenyu Cui, Chihoon Lee, Lingjiong Zhu, and Yunfan Zhu. On the optimal design of the randomized unbiased Monte Carlo estimators. *Operations Research Letters*, 49(4):477–484, 2021. doi: 10.1016/j.orl.2021.05.004.

Ayoub El Hanchi and David Stephens. Adaptive importance sampling for finite-sum optimization and sampling with decreasing step-sizes. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://papers.nips.cc/paper/2020/hash/b58f7d184743106a8a66028b7a28937c-Abstract.html.

Paul R. Halmos. The theory of unbiased estimation. *The Annals of Mathematical Statistics*, 17(1):34–43, 1946. doi: 10.1214/aoms/1177731020. URL https://www.jstor.org/stable/2235902.

Wassily Hoeffding. A class of statistics with asymptotically normal distribution. *The Annals of Mathematical Statistics*, 19(3):293–325, 1948. doi: 10.1214/aoms/1177730196.

D. G. Horvitz and D. 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.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-like training: A critical perspective. arXiv:2503.20783, 2025. URL https://arxiv.org/abs/2503.20783.

Xiaodong Lu, Xiaohan Wang, Jiajun Chai, Guojun Yin, Wei Lin, Zhijun Chen, Yu Luo, Fuzhen Zhuang, Yikun Ban, and Deqing Wang. Contextual rollout bandits for reinforcement learning with verifiable rewards. arXiv:2602.08499, version 2, 2026. URL https://arxiv.org/abs/2602.08499.

Animesh Kumar Paul and Russell Greiner. Censoring-aware tree-based reinforcement learning for estimating dynamic treatment regimes with censored outcomes. arXiv preprint arXiv:2503.06690, 2025. URL https://arxiv.org/abs/2503.06690.

Andres Potapczynski, Luhuan Wu, Dan Biderman, Geoff Pleiss, and John P. Cunningham. Bias-free scalable Gaussian processes via randomized truncations. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8609–8619, 2021. URL https://proceedings.mlr.press/v139/potapczynski21a.html.

Chang-Han Rhee and Peter W. Glynn. Unbiased estimation with square root convergence for SDE models. *Operations Research*, 63(5):1026–1043, 2015. doi: 10.1287/opre.2015.1404.

James M. Robins, Andrea Rotnitzky, and Lue Ping Zhao. Estimation of regression coefficients when some regressors are not always observed. *Journal of the American Statistical Association*, 89(427): 846–866, 1994. doi: 10.1080/01621459.1994.10476818.

Dhruv Rohatgi, Abhishek Shetty, Donya Saless, Yuchen Li, Ankur Moitra, Andrej Risteski, and Dylan J. Foster. Taming imperfect process verifiers: A sampling perspective on backtracking. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.03149.

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.

Rikiya Takehi, Masahiro Asami, Kosuke Kawakami, and Yuta Saito. A general framework for off-policy learning with partially-observed reward. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2506.14439.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992. doi: 10.1007/BF00992696.

Chao Zheng, Jiangtao Pan, and Qun Wang. Optimal distributions for randomized unbiased estimators with an infinite horizon and an adaptive algorithm. *IMA Journal of Numerical Analysis*, 46(2): 970–1001, 2026. doi: 10.1093/imanum/draf017.

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the Twentieth International Conference on Machine Learning*, pp. 928–935, 2003. URL https://aaai.org/papers/icml03-120-online-convex-programming-and-generalized-infinitesimal-gradient-ascent/.

Table 3: Foundations, additional results, and evidence. The comparisons concern the stated observation model and do not certify exhaustive novelty.

Component	Established foundation	Additional result in this paper
Endpoint reward correction	Inverse weighting and sparse randomized telescopes	Explicit reference-cap observation contract
Allocation objective	Cost–second-moment tuning for unbiased gradients	Finite-minibatch criterion and convexity
Adaptive feedback	Independent-product / U-statistic estimation	Two-versus-three minimum record order under censored terminal feedback
Online and budget bounds	Projected online gradients and smooth ascent	Predictably stopped specialization, with explicit limitations
Operational separations	Bias–variance tradeoffs and group-relative bias	Strict checkpoint gap, exact cross-cost reversal, propensity-specific direction criteria

A Batch-aware allocation and the order of observable feedback

This section conditions throughout on a fixed pre-batch history. The policy, baseline, strata, and cutoff law are frozen while the independent records are drawn. Write $\gamma = \mathbb{E}g = \nabla J(\theta)$.

A.1 Randomized-telescope reduction

For a complete potential trace define $G_k = Z\mathbf{1}\{j \le k\}$ and $G_0 = 0$. Then the increments $\Delta_k = G_k - G_{k-1} = Z\mathbf{1}\{j = k\}$ vanish except at one endpoint. The Russian-roulette randomized telescope of Beatson & Adams (2019) becomes

$$\sum_{k=1}^{L} \frac{\Delta_k}{s_{hk}} = \frac{OZ}{s_{hj}} = g. \tag{15}$$

No unobserved increment is required. A zero observed prefix G_k means that the endpoint increment has not occurred, rather than that the final reward is zero. Administrative failure at the cap is included in Z, which matters when the baseline is nonzero. Endpoint correction and the general cost–second-moment principle thus follow from randomized truncation. We ask how many censored records are needed for unbiased feedback to adapt this criterion and its finite-batch counterpart without complete pilot traces.

A.2 Proof of the finite-batch allocation criterion

Proof of Theorem 2. For n independent complete prompt experiments, let $\bar{g} = n^{-1}\sum_{i=1}^{n} g_i$. Expanding the square and using independence gives

$$\mathbb{E}\|\bar{g}\|^2 = \frac{M}{n} + \frac{n-1}{n}\|\gamma\|^2. \tag{16}$$

Smoothness on every update segment implies

$$\mathbb{E}[J(\theta + \eta\bar{g}) - J(\theta)] \ge \eta\|\gamma\|^2 - \frac{\mathcal{L}\eta^2}{2n}\{M + (n-1)\|\gamma\|^2\}. \tag{17}$$

Expected batch charge is nC. Dividing the right side by nC and maximizing this concave quadratic in $\eta \ge 0$ gives

$$\eta^* = \frac{n\|\gamma\|^2}{\mathcal{L}\{M + (n-1)\|\gamma\|^2\}}, \qquad \mathcal{B}_n^*(u) = \frac{\|\gamma\|^4}{2\mathcal{L}\Phi_n(u)}, \qquad \Phi_n(u) = C(u)\{M(u) + (n-1)\|\gamma\|^2\}. \tag{18}$$

Here $\mathcal{L} > 0$, and the step is admissible only when the assumed smooth domain contains that segment. At $\gamma = 0$ the maximal quadratic lower bound is zero, attained by $\eta = 0$ whenever $M > 0$. At $M = 0$, unbiasedness forces $\gamma = 0$ and all sampled gradients vanish almost surely. These degenerate cases need no division by zero.

The term CM is convex by (8). Also,

$$(n-1)\|\gamma\|^2 C(u) = (n-1)\|\gamma\|^2 c_0 + (n-1)\|\gamma\|^2 \sum_i c_i e^{u_i} \tag{19}$$

is convex. Compactness of $\mathcal{U}$ gives attainment. The criterion optimizes the *smooth quadratic lower bound*, not the unknown actual improvement. Its population-optimal learning rate need not be observable.

For comparison with the uncorrected surrogate, let u_F minimize $F = CM$. Since $M \geq \|\gamma\|^2$,

$$\Phi_n(u_F) \leq nF(u_F) \leq nF(u) \leq n\Phi_n(u) \quad \text{for every } u \in \mathcal{U}. \tag{20}$$

Thus u_F achieves at least $1/n$ of the largest lower bound $\mathcal{B}_n^*$ when $\gamma \neq 0$. The sharper factor is $1 + (n-1)\|\gamma\|^2/M(u_F)$ in place of n. It approaches one when the squared mean gradient is negligible relative to its second moment. This comparison concerns a frozen policy and optimized steps, not alternative learning trajectories. □

A.3 One-checkpoint phase boundaries and known-cost feedback

Proof of Proposition 3. For a complete potential gradient Z and an independent $O \sim \text{Bernoulli}(q)$, the observed gradient is $g = OZ/q$. Thus $M = a/q$ and $C = c_0 + dq$. The usual product is $F = da + c_0 a/q$, whose derivative in q is strictly negative because $c_0, a > 0$. It is minimized at $q = 1$.

For $\xi = (n-1)\|\gamma\|^2/a$ and $\kappa = c_0/d$,

$$\Phi_n(q) = ad\{1 + \kappa\xi + \kappa/q + \xi q\}. \tag{21}$$

When $\xi = 0$, the preceding monotonicity argument applies. For $\xi > 0$, the derivative is $ad(-\kappa/q^2 + \xi)$, with positive second derivative $2ad\kappa/q^3$. Clipping its unique zero $\sqrt{\kappa/\xi}$ to $[\epsilon, 1]$ gives the three regimes in (10). At either equality boundary the adjacent expressions coincide. In the interior, substitution gives

$$\Phi_n(q_n^*) = ad(1 + \sqrt{\kappa\xi})^2, \qquad \Phi_n(1) = ad(1+\kappa)(1+\xi). \tag{22}$$

Subtracting yields $ad(\sqrt{\xi} - \sqrt{\kappa})^2$. This is positive precisely in the interior under discussion. At a clipped lower boundary, (21) directly supplies the value. No division by ξ is needed in the zero-gradient or $n = 1$ case. □

For the example in Section 3, the logistic score is 0.1 on A and -0.9 on B. Therefore

$$a = 0.9(0.99)(0.1)^2 + 0.1(0.01)(0.9)^2 = 0.00972, \tag{23}$$

$$\gamma = 0.9(0.1)(0.99 - 0.01) = 0.0882. \tag{24}$$

With $n = 3$, $\gamma^2 = 0.00777924$, $\kappa = 0.02$, and $\epsilon = 0.1$, the optimum is 0.1117801138. The two objective values are 0.0257840496 and 0.0135094269. These values concern a frozen policy's allocation bound. They do not specify an optimal finite-horizon training policy.

Known cost changes the record-order requirement in this control. In this deterministic-runtime model, $\nabla_{\log q} C = dq$ is known. Since

$$\nabla_{\log q} \Phi_n = -c_0 M + (n-1)dq\|\gamma\|^2, \tag{25}$$

any $r \geq 2$ independent records give the unbiased estimator

$$H_{\text{known}}^{(r)} = -\frac{c_0}{r}\sum_i m_i + \frac{(n-1)dq}{r(r-1)}\sum_{i \neq j}\langle g_i, g_j\rangle. \tag{26}$$

Its expectation follows from $\mathbb{E}m_i = M$ and $\mathbb{E}\langle g_i, g_j\rangle = \|\gamma\|^2$. In contrast, the general unknown-cost construction replaces the known dq by an independent observed work record. In the same special model, the paired-CM formula itself reduces to $-c_0 r^{-1}\sum_i m_i$. A one-record specialization is therefore possible for F. Both specializations fall outside the unrestricted unknown-law lower bounds. The high-signal experiment compares target criteria and includes the stronger known-cost estimator.

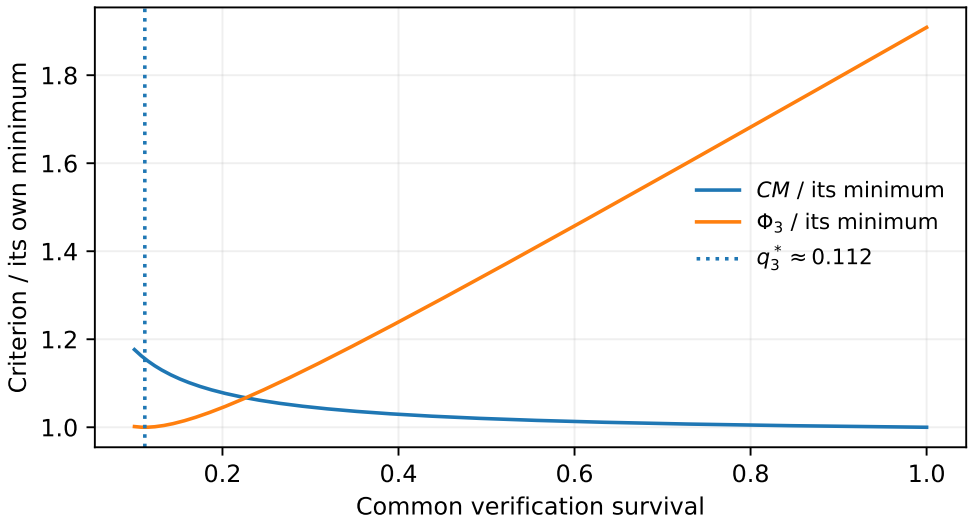

Figure 2: Different frozen-policy allocation targets in the high-signal control. Each curve is divided by its own minimum; the denominators are not the same. The CM criterion prefers full verification, whereas the ordinary three-record mean update's criterion prefers survival near 0.112.

A.4 Observable distinct-triple feedback

For $n \geq 3$, define ordered distinct-index sums

$$H_F^{(n)} = \frac{1}{n(n-1)} \sum_{i \neq j} m_j (v_i - D_i e_j), \tag{27}$$

$$T^{(n)} = \frac{1}{n(n-1)(n-2)} \sum_{i,j,k \text{ distinct}} v_i \langle g_j, g_k \rangle, \tag{28}$$

$$H_\Phi^{(n)} = H_F^{(n)} + (n-1) T^{(n)}. \tag{29}$$

Here $D_i = C_i^{\text{obs}}$ is the realized charge. All quantities in these expressions are observed records, with zero contributions on unrevealed endpoints.

Proof of the feedback statement in Theorem 5. The pair identity already proves $\mathbb{E} H_F^{(n)} = \nabla(CM)$. For three distinct records,

$$\mathbb{E}[v_i \langle g_j, g_k \rangle] = \mathbb{E} v_i \, \langle \mathbb{E} g_j, \mathbb{E} g_k \rangle = \nabla C \, \|\gamma\|^2. \tag{30}$$

The gradient mean has no dependence on the cutoff law, so differentiating Φ_n gives exactly the expectation of (29). No reward or completion-time predictor is needed. The inner product $\langle g_j, g_k \rangle$ can be negative on an individual batch; clipping it to zero would generally destroy unbiasedness.

Let $G_\Sigma = \sum_i g_i$ and $M_\Sigma = \sum_i m_i$. For each fixed i,

$$\sum_{\substack{j,k \neq i \\ j \neq k}} \langle g_j, g_k \rangle = \|G_\Sigma - g_i\|^2 - (M_\Sigma - m_i). \tag{31}$$

Therefore

$$(n-1) T^{(n)} = \frac{1}{n(n-2)} \sum_i v_i \{ \|G_\Sigma - g_i\|^2 - M_\Sigma + m_i \}. \tag{32}$$

Together with the pair reduction in (100), this requires $O(n(d + HK))$ arithmetic after the policy gradients are available, where d is the policy parameter dimension. Projection adds $O(HK)$. This applies the classical distinct-index U-statistic construction (Hoeffding, 1948) to censored records; independent-product estimation itself is established.

For a moment bound, Jensen's inequality gives the pair bound V from Theorem 4. For one ordered triple, Cauchy–Schwarz and record independence give

$$\mathbb{E}\|v_i\langle g_j, g_k\rangle\|^2 \le B_\star^2 \mathbb{E}[\|g_j\|^2\|g_k\|^2] \le B_\star^2 G^4 \epsilon^{-2}. \tag{33}$$

Thus $\mathbb{E}\|H_\Phi^{(n)}\|^2 \le V_n$ with

$$V_n = 8C_{\max}^2 G^4 \epsilon^{-3} + 2(n-1)^2 B_\star^2 G^4 \epsilon^{-2}. \tag{34}$$

This bound does not exploit any variance reduction from increasing n. □

A.5 Why two and three are the minimum universal record orders

Proof of the record-order statement in Theorem 5. Fix a history, policy, and schedule, and let the unknown complete trace law vary over a mixture $P_\lambda = (1-\lambda)P_0 + \lambda P_1$. Assume the observation and internal-randomization kernels are fixed and have no direct dependence on λ. The distribution of one censored record is then $Q_\lambda = (1-\lambda)Q_0 + \lambda Q_1$. For any integrable estimator based on at most r independent records, its expectation is a polynomial of degree at most r in λ. To see this directly, expand the product measure

$$Q_\lambda^{\otimes r} = \sum_{S\subseteq[r]} \lambda^{|S|}(1-\lambda)^{r-|S|} \bigotimes_{i=1}^{r} Q_{\mathbf{1}\{i\in S\}}. \tag{35}$$

The integrals of the fixed estimator under each product distribution are constants. The conclusion also holds with estimator-internal randomness independent of λ. The examples below have finite support, so integrability is immediate.

For the pair lower bound, use a uniform two-action scalar logistic policy with scores $+1/2, -1/2$, one stratum, and reward one on both actions. Independently of the action, the endpoint is b_1 with probability p, and b_2 otherwise. Put $s_1 = 1$, $s_2 = q \in (\epsilon, 1)$, $z^2 = 1/4$, $A = c_0 + b_1$, and $d = b_2 - b_1 > 0$. Then

$$\partial_{\log q} F = dz^2 qp(1-p) - Az^2(1-p)/q. \tag{36}$$

Its coefficient of p^2 is nonzero. No one-record estimator can have this expectation for every p in an interval. Two records suffice by the pair construction.

For the triple lower bound, keep the same fixed uniform policy and checkpoint design. Under P_0, both actions fail at b_1. Under P_1, both finish at b_2, action A passes, and action B fails. Under their mixture P_λ, the action marginal is still the prescribed uniform policy. Now

$$C = A + dq\lambda, \qquad M = \lambda/(8q), \qquad \gamma = \lambda/4. \tag{37}$$

The derivative of CM is at most quadratic in λ, whereas

$$\partial_{\log q}\{(n-1)C\gamma^2\} = (n-1)dq\lambda^3/16 \tag{38}$$

has a nonzero cubic coefficient for every $n > 1$. Consequently no estimator based on at most two records is universally unbiased for $\nabla\Phi_n$. Three records suffice. For a target batch size $n = 2$, use three independent records in $H_F^{(3)} + (n-1)T^{(3)}$; for $n \ge 3$, the existing policy minibatch supplies the needed triples.

These lower bounds concern conditional record order under *exact universal unbiasedness*. They allow consistent plug-in designs, useful biased feedback, known parametric trace models, information accumulated under an unchanged law, and alternative criteria. They are not minimax regret or runtime bounds. Parameter independence of the observation kernel is essential, as in Proposition 17. □

A.6 Online comparison and a hard-budget bound for mean updates

Theorem 6 (Online comparison for batch-aware allocation)**.** *Under Assumption 1, let $n \ge 3$, $u_{t+1} = \Pi_{\mathcal{U}}(u_t - \alpha H_{\Phi,t}^{(n)})$, and let $z_t \in \mathcal{U}$ be a predictable comparator evaluated at the learner's policy. For $D_{\mathcal{U}} = \sqrt{HK}|\log\epsilon|$ and $P_N = \sum_{t=2}^{N}\|z_t - z_{t-1}\|$,*

$$\mathbb{E}\sum_{t=1}^{N}[\Phi_{n,t}(u_t) - \Phi_{n,t}(z_t)] \le \frac{D_{\mathcal{U}}^2 + 2D_{\mathcal{U}}\mathbb{E}P_N}{2\alpha} + \frac{\alpha N V_n}{2}. \tag{39}$$

For a bounded predictable-start stopping rule, N in the final term is replaced by its expectation and the path length is stopped at the same final batch.

Proof. The projection inequality, comparator telescoping, and predictable-start argument in Appendix G apply with F replaced by the convex function Φ_n and V by (34). In particular,

$$\Phi_{n,t}(u_t) - \Phi_{n,t}(z_t) \le \langle \nabla \Phi_{n,t}(u_t), u_t - z_t \rangle. \tag{40}$$

Conditioning on the pre-batch history replaces the gradient by the feedback expectation. Nonexpansiveness yields the telescoping squared-distance term, the sum of comparator movements is bounded by $2D_{\mathcal{U}}P_N$, and the squared-feedback term is at most $\alpha^2 V_n$ per started batch. Every term is multiplied by a pre-batch measurable start indicator in the stopped case. The argument is the standard online-gradient proof. □

Theorem 7 (Mean-gradient learning with a charged-work cap)**.** *Under Assumptions 1 and 2, let $\mathcal{L} > 0$. Use $n \ge 3$ independent records per batch, the mean update $\theta_{t+1} = \theta_t + \eta \bar{g}_t$, and the allocation update of Theorem 6. Start a batch only if at least $nC_{\max}$ charge remains. Let $Q > nC_{\max}$, let N be the number started, and put $q = \mathbb{E}[\text{total charge}]$. For $0 < \eta \le n/[\mathcal{L}(n-1)]$, define $A_n = \mathbb{E}\sum_{t\le \mathsf{N}} \Phi_{n,t}(z_t)$ and let $\mathcal{R}_n$ denote the stopped right side of* (39)*. Then $Q - nC_{\max} < q \le Q$ and*

$$\frac{\mathbb{E}\sum_{t\le \mathsf{N}} C_t \|\nabla J(\theta_t)\|^2}{\mathbb{E}\sum_{t\le \mathsf{N}} C_t} \le \frac{2nC_{\max}\Delta}{\eta q} + \frac{\mathcal{L}\eta C_{\max}}{c_0 q}(A_n + \mathcal{R}_n). \tag{41}$$

Proof. Each batch costs between nc_0 and $nC_{\max}$, so the number started is bounded by $\lfloor Q/(nc_0) \rfloor$, no overshoot is possible, and unused charge is less than $nC_{\max}$. Predictability gives $q = n\mathbb{E}\sum_{t\le \mathsf{N}} C_t$. Smooth ascent and (16) yield

$$\mathbb{E}_t[J(\theta_{t+1}) - J(\theta_t)] \ge \frac{\eta}{2}\|\gamma_t\|^2 - \frac{\mathcal{L}\eta^2}{2n} M_t. \tag{42}$$

The step restriction justifies the coefficient $\eta/2$. Sum against the predictable start indicators and telescope the bounded objective to obtain

$$\mathbb{E}\sum_{t\le \mathsf{N}} \|\gamma_t\|^2 \le 2\Delta/\eta + \frac{\mathcal{L}\eta}{n}\mathbb{E}\sum_{t\le \mathsf{N}} M_t. \tag{43}$$

Since $C_t \le C_{\max}$, $C_t \ge c_0$, and $\Phi_{n,t} \ge C_t M_t$, the weighted numerator is at most $C_{\max}$ times the left side and $M_t \le \Phi_{n,t}/c_0$. Divide by q/n and apply the allocation regret bound. This proves (41). □

For fixed $n, c_0, \epsilon, C_{\max}, \mathcal{L}$, suppose $A_n = O(Q)$ and let P_{N} denote the stopped comparator path length. Choose constant steps $\alpha \asymp Q^{-1/2}$ and $\eta \asymp Q^{-1/2}$, with the latter satisfying the theorem's restriction. Substitution in (39) and (41) gives

$$O\left(Q^{-1/2} + \frac{\mathbb{E}P_{\mathsf{N}}}{Q}\right). \tag{44}$$

A static comparator, or more generally $\mathbb{E}P_{\mathsf{N}} = O(\sqrt{Q})$, therefore gives the $O(Q^{-1/2})$ rate. Bounded average comparator loss alone is insufficient. A linear path length can leave a nonvanishing tracking contribution. The factor $C_{\max}/c_0$ can make the bound less informative than the exact local moments. The guarantee concerns logical charge, not CPU time. It neither certifies the experimental step choices nor orders final task accuracies.

B When checkpoints matter and when the surrogate can mislead

B.1 A strict advantage over endpoint thinning

Theorem 8 (Strict checkpoint advantage)**.** *Consider one stratum and two checkpoints with $a_1, a_2, c_1, c_2, c_0 > 0$. Write $A = c_0 + c_1$. If $q_* := \sqrt{Aa_2/(c_2 a_1)} \in (\epsilon, 1)$, the best schedule for F is $s = (1, q_*)$. Its advantage over the best endpoint-thinning schedule $s = (r, r)$ is*

$$F^{\star}_{\text{end}} - F_{\star} = (\sqrt{c_2 a_1} - \sqrt{A a_2})^2 > 0. \tag{45}$$

Proof of Theorem 8. With one stratum and two checkpoints, write $s_1 = r$ and $s_2 = rq$, where $r \le 1$, $q \le 1$, and $rq \ge \epsilon$. The cost–moment product is

$$F(r, rq) = \left(\frac{c_0}{r} + c_1 + c_2 q\right)\left(a_1 + \frac{a_2}{q}\right). \tag{46}$$

For fixed feasible q, increasing r to one preserves feasibility and strictly decreases F because $c_0, a_1, a_2 > 0$. Thus an optimum has $s_1 = 1$. With $A = c_0 + c_1$,

$$F(1, q) = Aa_1 + c_2 a_2 + c_2 a_1 q + Aa_2/q. \tag{47}$$

Its derivative vanishes at $q_0 = \sqrt{Aa_2/(c_2 a_1)}$, and its second derivative is positive. Clipping q_0 to $[\epsilon, 1]$ gives the constrained optimum. If $\epsilon < q_0 < 1$, its value is

$$F_\star = (\sqrt{Aa_1} + \sqrt{c_2 a_2})^2. \tag{48}$$

An endpoint-thinning schedule has $s_1 = s_2 = r$, hence

$$F_{\mathrm{end}}(r) = (c_0/r + c_1 + c_2)(a_1 + a_2). \tag{49}$$

It is minimized at $r = 1$. Subtracting the checkpoint optimum gives

$$F^\star_{\mathrm{end}} - F_\star = (\sqrt{c_2 a_1} - \sqrt{Aa_2})^2 > 0. \tag{50}$$

All signal and work coefficients are strictly positive. The separation does not rely on omitting a zero-signal stratum. □

A realizable example has one uniform two-action logistic policy, scores $+1/2, -1/2$, $c_0 = 0.05$, and checkpoints $0.1, 1$. Action A always passes at 0.1. Action B finishes at 1 and passes with probability 0.12. With zero baseline,

$$(a_1, a_2) = (0.125, 0.015), \qquad (c_1, c_2) = (0.1, 0.45). \tag{51}$$

The best endpoint-thinning value is 0.084, while $s = (1, 0.2)$ gives 0.048, a $3/7$ relative reduction. This population allocation example does not establish a runtime advantage. A late check can succeed and retains positive continuation probability.

B.2 Exact cross-cost moments

Proposition 9 (Exact cross-cost penalty)**.** *For two independent records, with $\nu = \mathbb{E}[(c_0 + \tau)Z]$,*

$$V_K := \mathbb{E}\|K\|^2 = \tfrac{1}{2}\mathbb{E}[D^2]M + \tfrac{1}{2}\|\nu\|^2, \qquad \operatorname{tr}\operatorname{Cov}(K) = V_K - C^2\|\gamma\|^2. \tag{52}$$

At fixed policy and baseline, ν does not depend on the cutoff law. The smooth improvement bound per expected charge is ordered by V_K/C, not by CM, at both a common step and its quadratic optimum.

Proof of Proposition 9. Use two independent records with realized charges D_1, D_2, and set $K = (D_1 g_2 + D_2 g_1)/2$. Expanding the square,

$$\mathbb{E}\|K\|^2 = \tfrac{1}{4}\mathbb{E}[D_1^2\|g_2\|^2 + D_2^2\|g_1\|^2 + 2D_1 D_2\langle g_1, g_2\rangle] \tag{53}$$

$$= \tfrac{1}{2}\mathbb{E}[D^2]M + \tfrac{1}{2}\|\mathbb{E}[Dg]\|^2. \tag{54}$$

On an observed endpoint, actual execution stops at τ and costs $c_0 + \tau$. On an unobserved endpoint, $g = 0$. Conditioning on the potential trace cancels the endpoint inclusion probability and therefore

$$\nu := \mathbb{E}[Dg] = \mathbb{E}[(c_0 + \tau)Z]. \tag{55}$$

It has no dependence on u at fixed policy and baseline. Subtracting $\|\mathbb{E}K\|^2 = C^2\|\gamma\|^2$ gives the trace of the covariance.

The expected smooth lower bound per expected pair charge is

$$\frac{\eta}{2}\|\gamma\|^2 - \frac{\mathcal{L}\eta^2}{4C}V_K, \qquad V_K = \mathbb{E}\|K\|^2. \tag{56}$$

Thus V_K/C, not CM, orders these lower bounds at a common positive step. Maximizing over the step gives $C\|\gamma\|^4/(4\mathcal{L}V_K)$ when $\gamma \neq 0$. The same ordering applies to this optimized lower bound. Actual task improvement remains outside the claim. □

The following fully specified instance reverses the surrogate ordering. Let $p(A) = 0.1$, $\mu_A = 0.1$, $\mu_B = 0.05$, $\tau_A = 0.02$, $\tau_B = 1$, $c_0 = 0.01$, and use the scalar logistic score 0.9 on A and -0.1 on B. Both actions share one stratum. With checkpoints $0.02, 1$, compare $s = (1, 1)$ and $s = (1, 0.1)$.

Quantity	Full	Shortened tail
C	0.912	0.1182
M	0.00855	0.0126
CM	0.0077976	0.00148932
$\mathbb{E}D^2$	0.91818	0.092628
V_K/C	0.0043139883	0.0050143334

Here $\gamma = 0.0045$ and $\nu = -0.004275$ under both schedules. The shortened tail lowers CM by about 80.9% but increases the cross-cost curvature penalty by about 16.2%. A smaller surrogate therefore need not give a better smooth cross-cost step bound. The realized learning trajectory need not follow this ordering.

B.3 Exact cross-cost moments for larger batches

For the off-diagonal batch update $K^{(n)}$ in (100), put $d_2 = \mathbb{E}D^2$. Direct classification of repeated indices gives

$$\begin{aligned} n(n-1)\mathbb{E}\|K^{(n)}\|^2 = d_2 M + \|\nu\|^2 \\ + (n-2)\{d_2\|\gamma\|^2 + C^2 M + 2C\langle\nu,\gamma\rangle\} \\ + (n-2)(n-3)C^2\|\gamma\|^2. \end{aligned} \tag{57}$$

To verify the count, expand the ordered pairs (i, j) and (k, ℓ) with $i \neq j$, $k \neq \ell$. Identical ordered pairs contribute $d_2 M$ and reversed pairs contribute $\|\nu\|^2$, each with $n(n-1)$ occurrences. Exactly three distinct indices contribute $d_2\|\gamma\|^2$, $C^2 M$, or one of two copies of $C\langle\nu,\gamma\rangle$, each with $n(n-1)(n-2)$ occurrences. Four distinct indices contribute $C^2\|\gamma\|^2$ with $n(n-1)(n-2)(n-3)$ occurrences. Divide by the squared denominator $n^2(n-1)^2$. The mean remains $C\gamma$.

C Direction preservation beyond the two-action counterexample

C.1 Normalization statements and objective-preserving alternatives

Fix one prompt, a two-action logistic policy with $p = \pi(A)$, success probabilities μ_A, μ_B, and independent reveal probabilities $s_A, s_B > 0$. Let $W_a = O_a Y_a / s_a$ be the corrected reward. For a group of two, centering and dividing by its population standard deviation, with tied groups set to zero, gives

$$G_{\mathrm{norm}} = \frac{\psi_1 - \psi_2}{2}\,\mathrm{sgn}(W_1 - W_2). \tag{58}$$

Proposition 10 (Direction reversal)**.** *With $(\mu_A, \mu_B) = (0.9, 0.6)$ and $(s_A, s_B) = (0.1, 1)$, each corrected reward is unbiased, but*

$$\nabla J = 0.3p(1-p), \qquad \mathbb{E}G_{\mathrm{norm}} = -0.456p(1-p). \tag{59}$$

Normalization replaces the correction's rare magnitude ten by its sign. This mechanism persists with a positive stabilizer and differs from the GRPO biases studied by Liu et al. (2025).

Proposition 11 (A direction criterion)**.** *Put $\alpha = s_A\mu_A$ and $\beta = s_B\mu_B$. Then $\mathbb{E}G_{\mathrm{norm}} = p(1-p)\kappa$, where*

$$\kappa = \begin{cases} \alpha - \beta + \alpha\beta, & s_A < s_B, \\ s_A(\mu_A - \mu_B), & s_A = s_B, \\ \alpha - \beta - \alpha\beta, & s_A > s_B. \end{cases} \tag{60}$$

Thus strict direction preservation is equivalent to $\kappa(\mu_A - \mu_B) > 0$ when the true gradient is nonzero.

The subsections below give the stabilizer formula, prove direction preservation at equal propensities for every group size, and derive the positively rescaled fixed-prompt large-group limit. Unequal propensities can cause finite-group reversal without reversing that limit. Prompt-specific rescaling need not preserve the aggregate multi-prompt objective.

Without random standard-deviation scaling, the conventional leave-one-out update

$$G_{\mathrm{loo}} = \frac{1}{n}\sum_i \psi_i \left(\widehat{R}_i - \frac{1}{n-1}\sum_{j\neq i} \widehat{R}_j \right) \tag{61}$$

is unbiased for every $n \geq 2$ under independent within-prompt records (Ahmadian et al., 2024). Our live program study uses unnormalized updates. A predictable action-dependent model can alternatively use the classical augmentation

$$\widetilde{R} = m(x,a) + \frac{O}{s_{hj}}\{R - m(x,a)\}. \tag{62}$$

It remains unbiased even if m is wrong, but its cost–moment objective need not be convex. A reference-cap failure must contribute the observed residual $-m$, not remain an imputed success. The full arguments are in Appendices I and J.

C.2 Exact characterization for groups of two

Proof of Proposition 11. Condition on a fixed prompt and history. Put $p = \pi(A)$, $\alpha = s_A\mu_A$, and $\beta = s_B\mu_B$. The corrected reward for A equals $1/s_A$ with probability α and zero otherwise, and similarly for B. Same-action groups have equal policy scores and zero centered update. The two mixed-action orderings give

$$\mathbb{E}G_{2,\delta} = p(1-p)\mathbb{E}\left[\frac{\widehat{R}_A - \widehat{R}_B}{|\widehat{R}_A - \widehat{R}_B| + 2\delta}\right], \tag{63}$$

where a zero-over-zero term is assigned zero. Let $w_\delta(d) = d/(|d| + 2\delta)$ with $w_\delta(0) = 0$. Partitioning the mixed pair into exactly one or both positive corrected rewards gives

$$\begin{aligned}\kappa_\delta = {} & \alpha(1-\beta)w_\delta(1/s_A) - \beta(1-\alpha)w_\delta(1/s_B) \\ & + \alpha\beta w_\delta(1/s_A - 1/s_B).\end{aligned} \tag{64}$$

This is the exact coefficient of $p(1-p)$. At $\delta = 0$, it reduces to

$$\kappa_0 = \begin{cases} \alpha - \beta + \alpha\beta, & s_A < s_B, \\ s_A(\mu_A - \mu_B), & s_A = s_B, \\ \alpha - \beta - \alpha\beta, & s_A > s_B. \end{cases} \tag{65}$$

The desired coefficient is $\mu_A - \mu_B$. Comparing signs gives the necessary and sufficient direction criterion, including the zero-update boundary. The reversal example uses the first branch with $(\alpha, \beta) = (0.09, 0.6)$. □

C.3 Equal reveal probabilities preserve direction at every finite group size

Proposition 12 (Common-propensity protection)**.** *For the fixed-prompt two-action experiment, let $0 < p < 1$, $s_A = s_B = s > 0$, and $\mu_A \neq \mu_B$. For every group size $n \geq 2$ and additive standard-deviation stabilizer $\delta \geq 0$, the expected normalized update has the same sign as $\mu_A - \mu_B$. Tied zero-variance groups contribute zero.*

Proof. Let $V_i = \mathbf{1}\{\widehat{R}_i = 1/s\}$ and $r = ps\mu_A + (1-p)s\mu_B$. The assumptions imply $0 < r < 1$. Conditional on $K = \sum_i V_i = k$, with $0 < k < n$, the expected fraction of action A among positive and zero-reward samples is respectively

$$v_1 = \frac{ps\mu_A}{r}, \qquad v_0 = \frac{p(1 - s\mu_A)}{1-r}. \tag{66}$$

Their difference is

$$v_1 - v_0 = \frac{p(1-p)s(\mu_A - \mu_B)}{r(1-r)}. \tag{67}$$

Write $z = k/n$. The sample covariance of the scalar score $\mathbf{1}\{A\} - p$ with V/s has conditional expectation $z(1-z)(v_1 - v_0)/s$. The sample population standard deviation of V/s is $\sqrt{z(1-z)}/s$, which is constant conditional on K. Dividing by that standard deviation plus δ multiplies $v_1 - v_0$ by a strictly positive factor. The events $0 < K < n$ have positive probability, and all other groups contribute zero. Averaging preserves the stated strict sign. □

C.4 Large groups and what they do not repair

Proposition 13 (Fixed-prompt large-group limit)**.** *Fix positive reveal probabilities, a two-action policy with $0 < p < 1$, and a nondegenerate corrected reward W with variance $\sigma_W^2 > 0$. With independent records, the normalized update converges almost surely and in expectation as $n \to \infty$ to*

$$\frac{p(1-p)(\mu_A - \mu_B)}{\sigma_W + \delta}. \tag{68}$$

Proof. The positive propensity floor bounds W. The strong law gives convergence of the empirical covariance between the score and W to $\mathrm{Cov}(\psi, W) = \mathbb{E}[\psi W] = \gamma$, and convergence of the empirical standard deviation to σ_W. The ratio therefore converges almost surely. At every finite n, Cauchy–Schwarz bounds the absolute normalized update by the sample standard deviation of the Bernoulli action indicator, hence by $1/2$, including at $\delta = 0$ with the tied-group convention. Dominated convergence gives the expectation limit. □

Unequal propensities can therefore reverse a small-group update without reversing its fixed-prompt infinite-group limit. The limit is generally not the original gradient's magnitude. Across prompts, even positive prompt-specific factors can change the aggregate direction and objective weighting. This proposition does not justify population GRPO under arbitrary prompt mixtures, multiple epochs, clipping, or token-level credit assignment.

The leave-one-out estimator in (61) is exactly unbiased for every group size, not merely asymptotically direction-correct. Appendix I proves this identity. The baseline construction is conventional REINFORCE, with language-model precedents including Ahmadian et al. (2024). Our analysis isolates unequal censoring propensities from previously studied GRPO biases (Liu et al., 2025). The accompanying enumeration evaluates all four-category count vectors rather than relying on Monte Carlo signs.

D Observation model and objective preservation

D.1 Notation and conditioning

Let $\mathcal{F}_{t-1}$ contain all observations and algorithmic random variables available before pair t. The policy parameter, baseline, allocation, and comparator at round t are $\mathcal{F}_{t-1}$-measurable. When discussing one round, suppress its index. Conditional on this history, sample the prompt, program, and potential trace, and then the independent cutoff random variable U. This ordering constructs the probability space; the implementation neither generates nor reads an unexecuted trace suffix.

For a fixed stratum, the cutoff distribution is

$$\Pr(L = 0) = 1 - s_{h1}, \qquad \Pr(L = k) = s_{hk} - s_{h,k+1}\ (1 \le k < K), \qquad \Pr(L = K) = s_{hK}. \tag{69}$$

Its masses are nonnegative and sum to one. Conversely, every distribution on these $K + 1$ cutoffs induces a nonincreasing survival sequence. The floor $s_{hK} \ge \epsilon$ assigns at least ϵ probability to a reference-budget check in every stratum.

If verification has not terminated by $B_\star$, its reference reward is $R = 0$ and its endpoint is $\tau = B_\star$. Running to the cap reveals this reward. An earlier cutoff reveals only the executed prefix unless verification has already terminated. Thus $O = \mathbf{1}\{L \ge j\}$ holds for successful, failed, and administrative endpoints. Equality $\tau = b_k$ belongs to interval k, so an endpoint at a checkpoint is observed.

Table 4: Principal notation. Costs, scores, and allocation coefficients are conditional on the history preceding a round unless otherwise stated.

Symbol	Meaning
x, a, θ_t	Prompt, sampled program, and current policy parameter
T, Y	Potential terminal verification time and terminal pass indicator
$B_\star, \tau, R$	Reference limit, capped endpoint, and reference reward
$b_0, \dots, b_K$	Ordered verification checkpoints, with $b_K = B_\star$
$h \in [H]$	Stratum computed before verification from (x, a)
u_{hk}, s_{hk}	Log-survival and survival probability at checkpoint k
L, j, O	Sampled cutoff index, latent endpoint interval, and observation indicator
$\beta_t(x), \psi_t, Z$	Predictable prompt baseline, policy score, and centered score
g, m, e	Corrected score estimator, its squared norm, and observed endpoint vector
W_k, v, C^{obs}	Potential interval work, observed work vector, and realized charge
a_{hk}, c_{hk}	Population second-moment mass and potential interval work mass
M, C, F	Expected squared score, expected charge, and product CM
H_t, K_t	Paired allocation gradient and cross-cost policy update
ϵ, D, V	Survival floor, feasible-set diameter, and allocation second-moment bound
z_t, P_N	Predictable allocation comparator and its path length
Q, N, q	Hard charged-work budget, number of completed pairs, and expected charge

D.2 Proof of Proposition 1

Proof. Let $\mathcal{A} = \sigma(\mathcal{F}_{t-1}, x, a, \tau, R)$. Conditional randomization and the cutoff construction imply

$$\mathbb{E}[O \mid \mathcal{A}] = s_{hj}. \tag{70}$$

All factors other than O in the expressions OR/s_{hj} and OZ/s_{hj} are $\mathcal{A}$-measurable. Positivity therefore gives

$$\mathbb{E}[\widehat{R} \mid \mathcal{A}] = R, \qquad \mathbb{E}[g \mid \mathcal{A}] = Z. \tag{71}$$

Although j is latent when $O = 0$, these are valid random-variable identities. The observable implementation returns zero in that branch without evaluating j.

For each prompt, common policy support and differentiation under the integral give

$$\mathbb{E}[\psi_\theta(x, a) \mid x, \mathcal{F}_{t-1}] = \int \nabla_\theta \pi_\theta(a \mid x)\, da = \nabla_\theta 1 = 0. \tag{72}$$

The trace law conditional on (x, a) is independent of θ. Applying the same identity to the reward integral yields

$$\nabla J(\theta) = \mathbb{E}[\psi_\theta(x, a) R \mid \mathcal{F}_{t-1}], \tag{73}$$

$$\mathbb{E}[Z \mid \mathcal{F}_{t-1}] = \mathbb{E}[\psi_\theta R \mid \mathcal{F}_{t-1}] - \mathbb{E}_x[\beta(x)\mathbb{E}[\psi_\theta \mid x, \mathcal{F}_{t-1}]] = \nabla J(\theta). \tag{74}$$

The tower property and (71) finish the proof. No independence between terminal runtime and reward is assumed. Both can depend arbitrarily on the sampled program and on one another. □

A baseline fitted on previous observations is admissible. Fitting it on the current pair and reusing it for that pair violates predictability. An action-dependent baseline is also not covered by this argument because $\mathbb{E}[\psi_\theta \beta(x, a) \mid x]$ need not vanish. Appendix J treats an explicitly corrected action-dependent model.

D.3 Observed costs and exact moments

For a fixed potential trace, $W_k = 0$ after its endpoint interval. Entering interval k incurs work W_k; otherwise the work is zero. Therefore

$$\sum_{h,k} v_{hk} = \min\{\tau, b_L\}. \tag{75}$$

This includes early terminations inside an interval. It is not the assigned cutoff b_L unless the check runs to that cutoff. Conditional randomization gives

$$\mathbb{E}[v_{hk} \mid x, a, \tau, R, \mathcal{F}_{t-1}] = \mathbf{1}\{h(x,a) = h\}\, s_{hk} W_k. \tag{76}$$

After averaging over the draw, $\mathbb{E}v_{hk} = c_{hk} s_{hk}$ and $\mathbb{E}C^{\mathrm{obs}} = c_0 + \sum_{hk} c_{hk} s_{hk}$. Likewise,

$$\mathbb{E}[\|g\|^2 \mid \mathcal{A}] = \frac{\|Z\|^2}{s_{hj}}, \tag{77}$$

$$\mathbb{E}[\|g\|^4 \mid \mathcal{A}] = \frac{\|Z\|^4}{s_{hj}^3}. \tag{78}$$

These identities prove (7). Since $\|Z\| \le G$, they also give

$$\mathbb{E}\|g\|^2 \le G^2 \epsilon^{-1}, \qquad \mathbb{E}\|g\|^4 \le G^4 \epsilon^{-3}, \qquad \|g\| \le G\epsilon^{-1} \quad \text{almost surely.} \tag{79}$$

All three bounds continue to hold conditional on every preceding history.

The fixed charge c_0 is incurred even if $L = 0$. It accounts for generation, score computation, allocation, and update work through a fixed per-draw charge, not measured CPU time. Increasing this overhead changes the allocation problem; it is not free preprocessing.

E Convex allocation, its boundary, and projection

Theorem 14 (Full-verification boundary for the usual product)**.** *Full verification minimizes F if and only if every stratum and checkpoint suffix satisfies*

$$C(0) \sum_{\ell=k}^{K} a_{h\ell} \ge M(0) \sum_{\ell=k}^{K} c_{h\ell}. \tag{80}$$

The proof uses the negative suffix directions of the ordered feasible set. It also covers zero coefficients. The same argument for Φ_n replaces $M(0)$ on the right by $M(0) + (n-1)\|\gamma\|^2$. Thus batching can change whether full verification minimizes the bound, even at a fixed policy.

E.1 Proof of Theorem 14

Proof. Flatten the HK coordinates. Expanding $C(u)M(u)$ gives (8). For a vector $d \in \mathbb{R}^{HK}$, its second directional derivative is

$$d^\top \nabla^2 F(u) d = c_0 \sum_j a_j e^{-u_j} d_j^2 + \sum_{i,j} c_i a_j e^{u_i - u_j} (d_i - d_j)^2 \ge 0. \tag{81}$$

Thus F is convex on the entire log-coordinate space, including when some coefficients vanish. The set $\mathcal{U}$ is nonempty, compact, and convex. Since F is continuous, it attains a minimum on $\mathcal{U}$.

Differentiating the product gives

$$\partial_{hk} F(u) = M(u) c_{hk} e^{u_{hk}} - C(u) a_{hk} e^{-u_{hk}}. \tag{82}$$

Let r_{hk} be the vector that equals one in the coordinates (h, ℓ) with $\ell \ge k$ and zero elsewhere. Every feasible u can be written as

$$u = -\sum_{h=1}^{H} \sum_{k=1}^{K} \delta_{hk} r_{hk}, \qquad \delta_{h1} = -u_{h1}, \quad \delta_{hk} = u_{h,k-1} - u_{hk}\ (k > 1), \tag{83}$$

where $\delta_{hk} \ge 0$. The negative suffix indicators therefore generate the feasible directions at zero. For each such direction,

$$\langle \nabla F(0), -r_{hk} \rangle = C(0) \sum_{\ell=k}^{K} a_{h\ell} - M(0) \sum_{\ell=k}^{K} c_{h\ell}. \tag{84}$$

If zero minimizes F, this quantity cannot be negative, because $-\delta r_{hk} \in \mathcal{U}$ for every sufficiently small $\delta > 0$. This proves necessity.

Conversely, suppose all quantities in (84) are nonnegative. Equation (83) then gives $\langle \nabla F(0), u \rangle \geq 0$ for every $u \in \mathcal{U}$. Convexity implies

$$F(u) \geq F(0) + \langle \nabla F(0), u \rangle \geq F(0), \tag{85}$$

which proves sufficiency. The proof does not divide by $M(0)$, so it includes the zero-signal case $M(0) = 0$, in which F vanishes everywhere. □

The criterion concerns the stated second-moment surrogate. Subtracting $C(u)\|\nabla J\|^2$ to obtain a variance–cost product adds a negative multiple of $\sum_i c_i e^{u_i}$, and the convexity argument no longer applies without additional structure. Optimality for F_t also does not rank the final policies produced by different training algorithms.

E.2 Uniform thinning and overhead

When every survival equals $q \in [\epsilon, 1]$, the cutoff is zero with probability $1 - q$ and the reference cap with probability q. Substituting $u = (\log q)\mathbf{1}$ gives

$$F((\log q)\mathbf{1}) = \left(\frac{c_0}{q} + \sum_i c_i \right) \sum_i a_i. \tag{86}$$

Its derivative with respect to q is $-c_0 \sum_i a_i / q^2$. When $c_0 > 0$ and the signal is nonzero, uniform pre-verification thinning is therefore strictly worse for this surrogate than full verification. This statement does not apply to a checkpoint-dependent uniform schedule such as $(1, 3/4, 1/2, 1/4)$, or to stratum-dependent thinning.

E.3 Ordered projection

We prove the projection formula used in Algorithm 1. Rows of $\mathcal{U}$ do not interact, so it suffices to project a row $z \in \mathbb{R}^K$. First project onto the unbounded nonincreasing cone. The pooled-adjacent-violators algorithm starts with singleton blocks and merges adjacent blocks whenever the left block's mean is smaller than the right block's mean. The final fitted vector r assigns each index its block mean.

Each resulting block $[p, q]$, with mean $\bar{z}$, satisfies

$$\sum_{i=p}^{k} (z_i - \bar{z}) \leq 0 \quad (p \leq k < q), \qquad \sum_{i=p}^{q} (z_i - \bar{z}) = 0. \tag{87}$$

The property is trivial for a singleton. Upon merging means $a < b$ to mean m, a prefix contained in the left block has nonpositive residual because $m \geq a$. A prefix ending r entries into the right block has residual at most $n_L(a - m) + r(b - m)$, which is nonpositive because $r \leq n_R$ and $n_L(a - m) + n_R(b - m) = 0$. This proves the invariant inductively.

To verify optimality, let $\lambda_i = -\sum_{j=1}^{i} (z_j - r_j)$ for $1 \leq i < K$, and set $\lambda_0 = \lambda_K = 0$. Equation (87) gives $\lambda_i \geq 0$. Block sums vanish, so $\lambda_i = 0$ at every strict descent of r. Moreover,

$$r_i - z_i + \lambda_{i-1} - \lambda_i = 0. \tag{88}$$

These are the Karush–Kuhn–Tucker conditions for minimizing $\frac{1}{2}\|r - z\|^2$ subject to $r_{i+1} - r_i \leq 0$. Strict convexity identifies the unique projection.

Now clip r coordinatewise to $[\log \epsilon, 0]$, obtaining p. Clipping preserves order. For any feasible w, the unbounded projection inequality gives $\langle z - r, w - r \rangle \leq 0$. Since $r - p$ is constant on each original block and each block has zero residual sum,

$$\langle z - r, r - p \rangle = 0. \tag{89}$$

Scalar interval projection also gives $\langle r - p, w - p \rangle \leq 0$. Adding these identities yields

$$\langle z - p, w - p \rangle = \langle z - r, w - r \rangle + \langle z - r, r - p \rangle + \langle r - p, w - p \rangle \leq 0. \tag{90}$$

This characterizes the Euclidean projection onto $\mathcal{U}$. Each block is pushed once and removed at most once. Pooling, expanding the final blocks, and clipping therefore require $O(K)$ operations per stratum and $O(HK)$ overall.

F THE PAIRED ALLOCATION GRADIENT

F.1 PROOF OF THEOREM 4

Proof. All expectations in this proof condition on $\mathcal{F}_{t-1}$. For one draw, Appendix D gives

$$\mathbb{E}v = \nabla C(u), \qquad \mathbb{E}C^{\mathrm{obs}} = C(u), \qquad \mathbb{E}m = M(u). \tag{91}$$

The endpoint vector is zero unless the endpoint is observed. Coordinatewise,

$$\mathbb{E}[me_{hk}] = \mathbb{E}\left[\frac{O\|Z\|^2}{s_{hj}^2}\mathbf{1}\{h(x,a)=h, j=k\}\right] = \frac{a_{hk}}{s_{hk}} = -\partial_{hk}M(u). \tag{92}$$

Independence across complete draws gives the following identities; independence within a draw is unnecessary.

$$\mathbb{E}[m_2 v_1] = M(u)\nabla C(u), \tag{93}$$

$$\mathbb{E}[C_1 m_2 e_2] = -C(u)\nabla M(u). \tag{94}$$

Both ordered cross-products have the same expectation. Their average is therefore $M\nabla C + C\nabla M = \nabla F$.

For the moment bound, $\|v_r\| \le \|v_r\|_1 \le B_\star$, $C_r \le C_{\max}$, and $\|e_r\| \le 1$. Hence

$$\|v_1 - C_1 e_2\| \le 2C_{\max}. \tag{95}$$

The squared norm of an average is at most the average squared norm, so

$$\mathbb{E}\|H_t\|^2 \le \frac{1}{2}\mathbb{E}\big[m_2^2\|v_1 - C_1 e_2\|^2 + m_1^2\|v_2 - C_2 e_1\|^2\big] \tag{96}$$

$$\le 4C_{\max}^2\mathbb{E}\|g\|^4 \le 4C_{\max}^2 G^4\epsilon^{-3}. \tag{97}$$

The last step is (78). The projection statement was proved in Appendix E.3. □

For $\beta = 0$, failed observed endpoints have $m = 0$. An implementation may omit their endpoint vector in expressions that only use me without changing H_t. The shortcut fails for a nonzero baseline because a failed endpoint can then have nonzero centered score. The baseline implementation records those endpoints explicitly.

F.2 WHY A SAME-SAMPLE SUBSTITUTE FAILS

Consider one prompt with two equally probable actions. Parameterize their probabilities as $\sigma(2\theta)$ and $1-\sigma(2\theta)$ at $\theta = 0$, giving scores $+1$ and -1. Set the baseline to zero. The first action passes at time 0.2, and the second fails at time 1. Use two action-specific strata, one checkpoint at $B_\star = 1$, and $c_0 = 0.1$. Then

$$a = (0.5, 0), \qquad c = (0.1, 0.5). \tag{98}$$

At survival $s = (0.8, 0.7)$, the second coordinate of the correct gradient is

$$\partial_2 F = Mc_2 s_2 = (0.5/0.8)(0.5)(0.7) = 0.21875. \tag{99}$$

For a same-sample substitute $m(v - C^{\mathrm{obs}}e)$, that coordinate is zero on every draw. When the second stratum is sampled, its reward and m are zero. When the first is sampled, the work and endpoint vectors have no second coordinate. The substitute cannot reduce expenditure on the second stratum even though that expenditure contributes to F.

The same issue arises when sharing prompts. Two independent completions of the *same* random prompt generally estimate the gradient of $\mathbb{E}_x[C_x(u)M_x(u)]$, not the gradient of $(\mathbb{E}_x C_x(u))(\mathbb{E}_x M_x(u))$. They need not give the unconditional product in this paper. Similarly, cross-cost policy updates sharing a prompt have mean $\mathbb{E}_x[C_x(u)\nabla J_x(\theta)]$, which is not generally a scalar multiple of ∇J. Algorithm 1 therefore samples prompts independently within its minibatch.

F.3 Extension to larger independent batches

For $n \geq 2$ conditionally independent draws at one frozen policy and schedule, define

$$H^{(n)} = \frac{1}{n(n-1)} \sum_{i \neq j} m_j (v_i - C_i e_j), \qquad K^{(n)} = \frac{1}{n(n-1)} \sum_{i \neq j} C_i g_j. \tag{100}$$

Linearity and the preceding proof give $\mathbb{E}H^{(n)} = \nabla F$ and $\mathbb{E}K^{(n)} = C\nabla J$. Jensen's inequality gives the same upper moment bounds as for a pair, although these bounds do not exploit variance reduction from a larger batch.

The sums do not require quadratic arithmetic. With $V_\Sigma = \sum_i v_i$, $M_\Sigma = \sum_i m_i$, $E_\Sigma = \sum_i m_i e_i$, $C_\Sigma = \sum_i C_i$, and $G_\Sigma = \sum_i g_i$,

$$n(n-1)H^{(n)} = M_\Sigma V_\Sigma - \sum_i m_i v_i - C_\Sigma E_\Sigma + \sum_i C_i m_i e_i, \tag{101}$$

$$n(n-1)K^{(n)} = C_\Sigma G_\Sigma - \sum_i C_i g_i. \tag{102}$$

The estimator therefore extends to independent-prompt minibatches. The two-record cross-cost theorem and four-prompt controls use $n = 2$. The held-out-prompt studies use $n = 3$, with the corresponding finite-batch criterion and mean-update guarantee in Appendix A.

G Allocation regret under changing policies and stopping

This section records the two-record CM variant, which uses $H_t = H_F^{(2)}$ and $u_{t+1} = \Pi_{\mathcal{U}}(u_t - \alpha H_t)$. The batch-aware extension is stated separately in Appendix A.6. Both guarantees use the classical projected-online-gradient argument of Zinkevich (2003).

Theorem 15 (Changing-policy comparison)**.** *Let $z_t \in \mathcal{U}$ be any predictable comparator and $P_N = \sum_{t=2}^{N} \|z_t - z_{t-1}\|$. With $D = \sqrt{HK}|\log\epsilon|$, the allocation iterates satisfy*

$$\mathbb{E}\sum_{t=1}^{N}\{F_t(u_t) - F_t(z_t)\} \leq \frac{D^2 + 2D\mathbb{E}P_N}{2\alpha} + \frac{\alpha NV}{2} =: \mathcal{R}_N. \tag{103}$$

G.1 Proof of Theorem 15

Proof. Let $d = HK$ and $\ell = |\log\epsilon|$. The feasible set is contained in $[-\ell, 0]^d$, so its diameter is at most $D = \sqrt{d}\,\ell$. The all-zero and all-$-\ell$ vectors belong to the set and attain this distance, hence D is its diameter.

Euclidean projection and $z_t \in \mathcal{U}$ give the pathwise inequality

$$\|u_{t+1} - z_t\|^2 \leq \|u_t - \alpha H_t - z_t\|^2 = \|u_t - z_t\|^2 - 2\alpha\langle H_t, u_t - z_t\rangle + \alpha^2\|H_t\|^2. \tag{104}$$

Rearranging and summing to any integer $n \geq 1$ yields

$$\sum_{t=1}^{n}\langle H_t, u_t - z_t\rangle \leq \frac{S_n}{2\alpha} + \frac{\alpha}{2}\sum_{t=1}^{n}\|H_t\|^2, \quad S_n = \sum_{t=1}^{n}\big(\|u_t - z_t\|^2 - \|u_{t+1} - z_t\|^2\big). \tag{105}$$

The moving comparator telescopes as

$$\begin{aligned} S_n = {} & \|u_1 - z_1\|^2 - \|u_{n+1} - z_n\|^2 \\ & + \sum_{t=2}^{n}\big(\|u_t - z_t\|^2 - \|u_t - z_{t-1}\|^2\big). \end{aligned} \tag{106}$$

All points in each difference belong to $\mathcal{U}$. The difference-of-squares identity and the diameter bound imply

$$\big|\|u_t - z_t\|^2 - \|u_t - z_{t-1}\|^2\big| \leq 2D\|z_t - z_{t-1}\|. \tag{107}$$

Discarding the final nonpositive term therefore gives

$$S_n \le D^2 + 2D \sum_{t=2}^{n} \|z_t - z_{t-1}\|. \tag{108}$$

At round t, the function F_t is convex even though its coefficients depend on the current policy and baseline. Conditional on $\mathcal{F}_{t-1}$, these coefficients and u_t, z_t are fixed. Theorem 4 implies

$$F_t(u_t) - F_t(z_t) \le \langle \nabla F_t(u_t), u_t - z_t \rangle \tag{109}$$

$$= \mathbb{E}[\langle H_t, u_t - z_t \rangle \mid \mathcal{F}_{t-1}]. \tag{110}$$

Take expectations in (105), use (108), and apply $\mathbb{E}[\|H_t\|^2 \mid \mathcal{F}_{t-1}] \le V$. The result is exactly (103). □

The proof allows adaptive losses because it uses only the feedback's conditional mean for the current loss, which is fixed before sampling. It requires no independence between F_{t+1} and H_t, but z_t must be chosen before the current draw. A population minimizer of F_t meets this requirement under a fixed measurable tie-breaking rule. A comparator chosen from the current realized pair does not.

For a deterministic upper bound $\bar{P}$ on expected path length, choosing

$$\alpha = \sqrt{\frac{D^2 + 2D\bar{P}}{NV}} \tag{111}$$

when $V > 0$ gives regret at most $\sqrt{(D^2 + 2D\bar{P})NV}$. For a fixed comparator, this gives the usual $D\sqrt{NV}$ bound. A dynamic oracle with linear path length does not generally have sublinear regret. If $G = 0$, all score estimators, objectives, and regret differences vanish, so no step-size formula involving division by V is needed.

G.2 The stopped regret inequality

For the budget rule, let N be the number of pairs completed. Set

$$N_{\max} = \left\lfloor \frac{Q}{2c_0} \right\rfloor, \qquad I_t = \mathbf{1}\{\mathsf{N} \ge t\}. \tag{112}$$

Each I_t is $\mathcal{F}_{t-1}$-measurable because the decision to start pair t uses only the remaining charge. Also $\mathsf{N} \le N_{\max}$ deterministically. After stopping, extend the sequences arbitrarily within their domains up to $N_{\max}$. The added values are multiplied by $I_t = 0$.

The pathwise inequality (105) holds simultaneously for every $n \le N_{\max}$. It can therefore be evaluated at $n = \mathsf{N}$ without invoking an optional-stopping theorem. In the expectation step,

$$\mathbb{E} \sum_{t=1}^{N_{\max}} I_t \langle \nabla F_t(u_t) - H_t, u_t - z_t \rangle = 0, \tag{113}$$

$$\mathbb{E} \sum_{t=1}^{N_{\max}} I_t \|H_t\|^2 \le V \sum_{t=1}^{N_{\max}} \mathbb{E} I_t = V \mathbb{E}\mathsf{N}. \tag{114}$$

Predictability and conditional expectation give the first identity. The same argument and Theorem 4 give the second. Hence

$$\mathbb{E} \sum_{t=1}^{\mathsf{N}} \big(F_t(u_t) - F_t(z_t)\big) \le \frac{D^2 + 2D\mathbb{E}P}{2\alpha} + \frac{\alpha V \mathbb{E}\mathsf{N}}{2}. \tag{115}$$

This establishes the stopped form used in Theorem 16. It charges every sampled pair, including pairs that provide no observed reward.

H POLICY LEARNING UNDER A HARD CHARGED-WORK BUDGET

This section analyzes the two-record cross-cost update, in contrast to the mean update in Algorithm 1. The conservative bound motivates the exact-moment calculation in Proposition 9.

$$K_t = \frac{C_1 g_2 + C_2 g_1}{2}, \qquad \theta_{t+1} = \theta_t + \eta K_t. \tag{116}$$

Its conditional moments obey

$$\mathbb{E}[K_t \mid \mathcal{F}_{t-1}] = C_t(u_t)\nabla J(\theta_t), \qquad \mathbb{E}[\|K_t\|^2 \mid \mathcal{F}_{t-1}] \le C_{\max} F_t(u_t). \tag{117}$$

The expected update divided by expected pair cost is therefore $\eta \nabla J/2$. This is a ratio of conditional expectations, not an unbiased ratio estimator. Equation (116) is not a gradient of $C_t J$ and does not differentiate the allocation distribution.

Run the two-record CM allocator and cross-cost policy update under budget $Q > 2C_{\max}$. Let N be the number of completed pairs. Since each pair costs between $2c_0$ and $2C_{\max}$, the stopping time is bounded and the total charge never exceeds Q. Its expected value is

$$q = \mathbb{E}\sum_{t=1}^{\mathsf{N}} (C_{t1} + C_{t2}) = 2\mathbb{E}\sum_{t=1}^{\mathsf{N}} C_t(u_t), \qquad Q - 2C_{\max} < q \le Q. \tag{118}$$

For a predictable comparator sequence define

$$A = \mathbb{E}\sum_{t=1}^{\mathsf{N}} F_t(z_t), \qquad P = \sum_{t=2}^{\mathsf{N}} \|z_t - z_{t-1}\|,$$
$$\mathcal{R} = \frac{D^2 + 2D\mathbb{E}P}{2\alpha} + \frac{\alpha V \mathbb{E}\mathsf{N}}{2}. \tag{119}$$

Theorem 16 (Computation–stationarity guarantee)**.** *Under Assumptions 1 and 2, for any $\eta, \alpha > 0$,*

$$\frac{\mathbb{E}\sum_{t=1}^{\mathsf{N}} C_t(u_t)\,\|\nabla J(\theta_t)\|^2}{\mathbb{E}\sum_{t=1}^{\mathsf{N}} C_t(u_t)} \le \frac{2\Delta}{\eta q} + \frac{\mathcal{L}\eta C_{\max}}{q}(A + \mathcal{R}). \tag{120}$$

The same statement holds at a fixed number of rounds, with the corresponding expected charge and comparator sums.

H.1 THE CROSS-COST IDENTITY

Write $\mu_t = \nabla J(\theta_t)$. Conditional on $\mathcal{F}_{t-1}$, Proposition 1 and independence of the two draws give

$$\mathbb{E}[C_1 g_2] = \mathbb{E}[C_1]\mathbb{E}[g_2] = C_t(u_t)\mu_t. \tag{121}$$

The same identity holds with the indices reversed. Thus $\mathbb{E}K_t = C_t(u_t)\mu_t$. This argument does not require independence between cost and reward within a draw.

For the second moment, convexity of squared norm gives

$$\mathbb{E}\|K_t\|^2 \le \tfrac{1}{2}\mathbb{E}\big[C_1^2\|g_2\|^2 + C_2^2\|g_1\|^2\big] \tag{122}$$
$$= \mathbb{E}[(C^{\mathrm{obs}})^2]\, M_t(u_t) \tag{123}$$
$$\le C_{\max}\, C_t(u_t) M_t(u_t) = C_{\max} F_t(u_t). \tag{124}$$

The last inequality follows pointwise from $(C^{\mathrm{obs}})^2 \le C_{\max} C^{\mathrm{obs}}$. These are the two claims in (117).

The update is implemented as a score estimator with the realized weights detached from policy differentiation. It is not obtained by differentiating an empirical cost–reward product. A same-draw product $C^{\mathrm{obs}} g$ would instead have mean $C\mu + \mathrm{Cov}(C^{\mathrm{obs}}, g)$ and does not give the stated identity in general.

H.2 CHARGE ACCOUNTING

Every completed pair costs between $2c_0$ and $2C_{\max}$. When a pair is started, at least $2C_{\max}$ charge remains, so its completion cannot exceed Q. Because every pair has positive minimum cost, the algorithm eventually stops. At that point its remaining charge is strictly below $2C_{\max}$, and hence the realized cumulative charge S satisfies

$$Q - 2C_{\max} < S \le Q. \tag{125}$$

As $Q > 2C_{\max}$, at least one pair is started. Using the predictable indicators from Appendix G.2,

$$q = \mathbb{E}S = \sum_{t=1}^{N_{\max}} \mathbb{E}[I_t(C_{t1} + C_{t2})] \tag{126}$$

$$= 2\sum_{t=1}^{N_{\max}} \mathbb{E}[I_t C_t(u_t)] = 2\mathbb{E}\sum_{t=1}^{\mathsf{N}} C_t(u_t). \tag{127}$$

Taking expectations in (125) proves (118). The calculation keeps the random observation count; it does not replace that count by Q divided by average cost.

H.3 PROOF OF THEOREM 16

Proof. Smoothness of J along the update segment implies the ascent inequality

$$J(\theta_t + \eta K_t) \ge J(\theta_t) + \eta\langle \mu_t, K_t\rangle - \frac{\mathcal{L}\eta^2}{2}\|K_t\|^2. \tag{128}$$

Taking the conditional expectation and using (117) yields

$$\mathbb{E}[J(\theta_{t+1}) - J(\theta_t) \mid \mathcal{F}_{t-1}] \ge \eta C_t(u_t)\|\mu_t\|^2 - \frac{\mathcal{L}\eta^2 C_{\max}}{2} F_t(u_t). \tag{129}$$

Multiply by I_t, sum to $N_{\max}$, and take expectations. Predictability preserves the conditional identities. The realized increments telescope to $J(\theta_{\mathsf{N}+1}) - J(\theta_1)$, which is at most Δ. Therefore

$$\eta\mathbb{E}\sum_{t=1}^{\mathsf{N}} C_t(u_t)\|\mu_t\|^2 \le \Delta + \frac{\mathcal{L}\eta^2 C_{\max}}{2}\mathbb{E}\sum_{t=1}^{\mathsf{N}} F_t(u_t). \tag{130}$$

The stopped allocation bound (115) gives

$$\mathbb{E}\sum_{t=1}^{\mathsf{N}} F_t(u_t) \le A + \mathcal{R}. \tag{131}$$

Divide (130) by $\eta\mathbb{E}\sum_{t=1}^{\mathsf{N}} C_t(u_t) = \eta q/2$, which is positive. This gives (120). At a deterministic number of rounds, use $I_t = 1$ up to that number and the identical calculation applies. □

The allocator and policy may use the same pair. Both updates are computed at the frozen u_t, θ_t, β_t and affect only the next history, so their dependence does not enter the argument.

H.4 AN EXPLICIT RATE AGAINST A FIXED SCHEDULE

Fix $z \in \mathcal{U}$ and suppose $F_t(z) \le \bar{F}$ for every history. For a full-verification comparator one may take $\bar{F} = C_{\max}G^2$. For any admissible fixed schedule one may take $\bar{F} = C_{\max}G^2/\epsilon$. Assuming $V > 0$, choose

$$\alpha = \frac{D}{\sqrt{VN_{\max}}}. \tag{132}$$

Then the stopped regret is at most $D\sqrt{VN_{\max}}$, and $A \le \bar{F}N_{\max}$. The theorem implies

$$\frac{\mathbb{E}\sum_{t\le\mathsf{N}} C_t\|\nabla J(\theta_t)\|^2}{\mathbb{E}\sum_{t\le\mathsf{N}} C_t} \le \frac{2\Delta}{\eta q} + \frac{\mathcal{L}\eta C_{\max}}{q}\left(\bar{F}N_{\max} + D\sqrt{VN_{\max}}\right). \tag{133}$$

For $Q \geq 4C_{\max}$, $q > Q/2$ and $N_{\max} \leq Q/(2c_0)$. Taking $\eta = \eta_0/\sqrt{Q}$, with fixed positive η_0, gives an $O(Q^{-1/2})$ stationarity term and an $O(Q^{-1})$ allocation-regret contribution when the other parameters are fixed. Constants deteriorate with small ϵ through V and the feasible-set diameter.

For an oracle that changes with the learner, replace the fixed-comparator bound by (119). This compares schedules at the same sequence of policies and baselines. It is not a comparison with the policy path a separate oracle-guided training algorithm would have generated. The stationarity expression is a ratio of expectations, not an expectation of a random normalized ratio, and says nothing by itself about global optimality or terminal task accuracy.

I Normalization and unbiased alternatives

I.1 Proof of Proposition 10

Proof. For two corrected rewards W_1, W_2, their within-group mean is $(W_1 + W_2)/2$ and their population standard deviation is $|W_1 - W_2|/2$. On a nontied group, the two normalized advantages are $\operatorname{sgn}(W_1 - W_2)$ and its negative. Averaging their score-weighted contributions gives (58). A tied group is assigned zero by definition.

If the two sampled actions agree, their scores agree, so the update is zero even when their random rewards differ. A mixed ordered pair has probability $p(1-p)$ in each ordering. For ordering (A, B), the score difference is $(1-p) - (-p) = 1$. For ordering (B, A), both score difference and reward difference change sign. Thus

$$\mathbb{E}G_{\text{norm}} = p(1-p)\mathbb{E}[\operatorname{sgn}(W_A - W_B)]. \tag{134}$$

Here W_A equals 10 with probability $s_A\mu_A = 0.09$ and zero otherwise. Independently, W_B equals 1 with probability 0.6 and zero otherwise. Therefore

$$\Pr(W_A > W_B) = 0.09, \tag{135}$$

$$\Pr(W_A < W_B) = (1 - 0.09)0.6 = 0.546, \tag{136}$$

$$\mathbb{E}\operatorname{sgn}(W_A - W_B) = 0.09 - 0.546 = -0.456. \tag{137}$$

Yet $\mathbb{E}W_A = 0.9$ and $\mathbb{E}W_B = 0.6$, so both action-level estimates are unbiased. Since $J(\theta) = p\mu_A + (1-p)\mu_B$ and $dp/d\theta = p(1-p)$,

$$\nabla J(\theta) = p(1-p)(\mu_A - \mu_B) = 0.3p(1-p). \tag{138}$$

Substitution into (134) proves the claimed reversal for every $p \in (0, 1)$. At $p = 1/2$, the two quantities are 0.075 and -0.114. □

This example embeds in the verification model with one terminal checkpoint and action-specific probabilities of entering the check. The terminal reward is Bernoulli, and every verification has the same potential terminal time. The reversal therefore requires neither within-action runtime–correctness correlation nor reward-model misspecification.

I.2 A positive normalization stabilizer

Suppose the normalized advantages divide by the within-group standard deviation plus $\delta > 0$. For a mixed pair the sign factor becomes $(W_A - W_B)/(|W_A - W_B| + 2\delta)$. The coefficient multiplying $p(1-p)$ is therefore

$$D(\delta) = 0.036\frac{10}{10 + 2\delta} + 0.054\frac{9}{9 + 2\delta} - 0.546\frac{1}{1 + 2\delta}. \tag{139}$$

For $0 \leq \delta \leq 1$, the sum of the first two terms is at most 0.09, and the magnitude of the final term is at least 0.182. Hence $D(\delta) \leq -0.092 < 0$ throughout this interval. The reversal is not an artifact of an undefined zero standard deviation. Using the sample rather than population standard deviation without a stabilizer merely multiplies all nontied updates by the same positive constant and also preserves the reversed direction.

The result concerns this normalization step. Policy clipping, multiple optimization epochs, token-level objectives, and other GRPO variants are outside the analysis and are not needed for the counterexample.

I.3 Leave-one-out centering

Fix a prompt x and the pre-group history. Draw $n \geq 2$ independent completions, verifier traces, and cutoff random variables from the current policy and predictable schedule. Write $J_x(\theta) = \mathbb{E}[R \mid x]$. For every i, objective preservation gives

$$\mathbb{E}[\psi_i \widehat{R}_i \mid x] = \nabla J_x(\theta), \qquad \mathbb{E}[\psi_i \mid x] = 0. \tag{140}$$

For $i \neq j$, independence conditional on x yields

$$\mathbb{E}[\psi_i \widehat{R}_j \mid x] = \mathbb{E}[\psi_i \mid x]\mathbb{E}[\widehat{R}_j \mid x] = 0. \tag{141}$$

Taking expectations term by term in (61) proves

$$\mathbb{E}[G_{\mathrm{loo}} \mid x] = \nabla J_x(\theta). \tag{142}$$

Averaging over x yields the reference policy gradient. Subtracting the ordinary group mean instead gives $(1 - 1/n)\nabla J_x$, because each score is paired with its own reward once in that mean. Neither calculation justifies division by a random within-group standard deviation.

A deterministic constant common to every prompt preserves the gradient direction. In contrast, multiplying the update for prompt x by a nonconstant positive weight $w(x)$ gives $\mathbb{E}_x[w(x)\nabla J_x]$. If w is fixed with respect to θ, this is the gradient of the reweighted objective $\mathbb{E}_x[w(x)J_x]$, not generally of $\mathbb{E}_x J_x$. A predictable history-dependent weight has the same conditional distinction.

J Model assistance and the administrative endpoint

Let $m(x, a)$ be a predictable bounded reward model, frozen before the current draw and held fixed in the policy estimator. It need not equal the conditional mean reward. Define

$$\widetilde{R} = m + \frac{O}{s_{hj}}(R - m), \tag{143}$$

with the residual term set to zero when $O = 0$. Conditional on the full potential draw,

$$\mathbb{E}[\widetilde{R} \mid \mathcal{A}] = m + \frac{s_{hj}}{s_{hj}}(R - m) = R. \tag{144}$$

Therefore $\mathbb{E}[\psi \widetilde{R} \mid \mathcal{F}_{t-1}] = \nabla J(\theta)$. This augmentation identity uses a known randomized propensity. It does not give double robustness for a misspecified estimated censoring probability.

J.1 Why capped failures must be observed

For a trace that reaches $B_\star$ unfinished, the reference reward is zero. If this endpoint is observed, its residual is $-m$, and its corrected reward is $m(1 - 1/s_{hK})$. If an earlier cutoff interrupts the trace, its corrected reward is m. Averaging over censoring gives

$$s_{hK} m(1 - 1/s_{hK}) + (1 - s_{hK})m = 0. \tag{145}$$

Treating the reference-cap timeout as still missing instead returns m in both cases. It leaves a spurious positive expectation whenever $m > 0$. Omitting the reference endpoint therefore changes the estimator's expectation.

J.2 The augmented cost product need not be convex

Conditional on the full draw, direct expansion gives

$$\mathbb{E}[\widetilde{R}^2 \mid \mathcal{A}] = R^2 + (R - m)^2(1/s_{hj} - 1). \tag{146}$$

Define

$$b_{hk} = \mathbb{E}[\|\psi\|^2 (R - m)^2 \mathbf{1}\{h(x, a) = h, j = k\}], \qquad d = \mathbb{E}[\|\psi\|^2 R^2] - \sum_{hk} b_{hk}. \tag{147}$$

Then

$$\mathbb{E}\|\psi\widetilde{R}\|^2 = d + \sum_{hk} b_{hk} e^{-u_{hk}}. \tag{148}$$

Each b_{hk} is nonnegative, but d can be negative. Multiplication by $C(u)$ introduces terms $dc_{hk}e^{u_{hk}}$ that defeat the nonnegative exponential expansion used in Theorem 14.

A concrete example establishes nonconvexity. At $\theta = 0$, take the policy with probabilities $\sigma(2\theta)$ and $1 - \sigma(2\theta)$, so both squared scores are one. Both actions always fail at time one. Put them in one stratum with one checkpoint, use $m = 1$ and $c_0 = 0.1$, and write $s = e^u$. Equation (148) becomes $M_{\mathrm{aug}}(u) = e^{-u} - 1$, while $C(u) = 0.1 + e^u$. Thus

$$C(u)M_{\mathrm{aug}}(u) = 0.1e^{-u} + 0.9 - e^u, \qquad \frac{d^2}{du^2}\{C(u)M_{\mathrm{aug}}(u)\} = 0.1e^{-u} - e^u. \tag{149}$$

The second derivative is negative on an interior neighborhood below $u = 0$. The bounded model and policy score satisfy the stated conditions, so convexity does not hold in general. The reported model-assisted baseline accordingly uses a fixed verification schedule. Predictable prompt-only centering of the entire score, as in the main allocation method, retains the nonnegative second-moment coefficients and is a different construction.

K Information from long checks

The positivity floor supports identification but incurs a variance cost. The following standard missing-data arguments delimit randomized verification without further assumptions on the reward mechanism.

Proposition 17 (Unobserved tails and a variance lower bound)**.** *Suppose all information distinguishing reward zero from reward one appears at the reference endpoint. If that endpoint is never observed, the reference mean is not identified from the training observations. More generally, let Y_i be independent Bernoulli(μ) rewards, with $0 < \mu < 1$, and let D_i indicate observing their endpoints, with predictable reveal probabilities s_i and conditional randomization independent of Y_i. Conditional on the observed past, the reveal and internal-randomization kernels have no direct dependence on the unknown μ, and the estimator is a fixed function of the observed transcript rather than of μ. For a fixed n, any estimator $\widehat{\mu}$ based on the observed transcript, unbiased for μ in a neighborhood and satisfying differentiation under expectation, obeys*

$$\mathrm{Var}_\mu(\widehat{\mu}) \geq \frac{\mu(1-\mu)}{\mathbb{E}_\mu \sum_{i=1}^n s_i}, \tag{150}$$

provided the denominator is positive and the estimator has finite variance.

Proof. For nonidentification, compare two verification environments with identical prefixes before the cap and with terminal reward respectively zero and one. If the cap is never reached, the observed laws coincide, while their reference means differ.

For the variance bound, the randomization policy has no direct dependence on the unknown μ conditional on the observed past. The log-likelihood score of the observed transcript is therefore

$$S_\mu = \sum_{i=1}^n \frac{D_i(Y_i - \mu)}{\mu(1-\mu)}. \tag{151}$$

Each summand has conditional mean zero. Conditional randomization and $\mathrm{Var}(Y_i) = \mu(1-\mu)$ give conditional variance $s_i/[\mu(1-\mu)]$. Martingale orthogonality then gives

$$\mathbb{E}S_\mu = 0, \qquad \mathbb{E}S_\mu^2 = \frac{\mathbb{E}\sum_i s_i}{\mu(1-\mu)}. \tag{152}$$

Differentiating $\mathbb{E}_\mu\widehat{\mu} = \mu$ yields

$$1 = \mathbb{E}_\mu[\widehat{\mu}S_\mu] = \mathbb{E}_\mu[(\widehat{\mu} - \mu)S_\mu]. \tag{153}$$

Cauchy–Schwarz gives $1 \leq \mathrm{Var}_\mu(\widehat{\mu})\mathbb{E}S_\mu^2$, proving (150). □

With constant reveal probability ϵ and $\mu = 1/2$, this is $1/(4n\epsilon)$. Thus a dependence on the number of full-budget observations cannot generally be removed by a different unbiased estimator. Separately, (78) attains $\|Z\|^4/\epsilon^3$ when a nonzero score is only revealed with probability ϵ. This accounts for the fourth-moment term in the allocation feedback bound. It does not establish minimax optimality of the constant or the full ϵ dependence.

Predictability alone is insufficient for this proposition. If one were allowed the parameter-dependent design $D_1 \sim \mathrm{Bernoulli}(\mu)$ independently of Y_1, then D_1 itself would be unbiased for μ, with variance $\mu(1-\mu)$, below $\mu(1-\mu)/\mathbb{E}D_1$ when $\mu < 1$. Such a reveal rule is excluded by the stated kernel assumption.

L Finite-batch and delayed-optimum mechanism studies

The terminal-observation control tests whether censored feedback learns the change induced by finite batching. The live program study tests whether retaining delayed outcomes changes which useful programs a policy learns. Neither estimates the prevalence of these regimes in natural code. Individual outcomes and supporting calculations are provided in the supplement.

L.1 High-signal terminal-observation control

One prompt has two actions. The scalar logistic policy is fixed at $p(A) = 0.9$. Rewards are Bernoulli with means 0.99 and 0.01, and the baseline is zero. An opened verification costs one unit and reveals its terminal reward. Every sampled record also costs $c_0 = 0.02$, including a skipped verification. The common survival is constrained to $[0.1, 1]$. Action, potential reward, and reveal randomization are independent across records, conditional on the current survival. A learner receives the action, its score, whether the check was opened, the observed charge, and the reward only when opened. It is not given either reward mean.

For the scalar logit score, the uncensored second moment and mean are

$$a = 0.9(0.99)(0.1)^2 + 0.1(0.01)(0.9)^2 = 0.00972, \tag{154}$$

$$\gamma = 0.9(0.1)(0.99 - 0.01) = 0.0882. \tag{155}$$

Thus $M = a/q$, $C = 0.02+q$, and $\Phi_3 = C(a/q+0.01555848)$. The numerical values follow directly from Proposition 3, not from a fitted oracle. They are $q_3^* = 0.1117801138$, $\Phi_3(1) = 0.0257840496$, and $\Phi_3(q_3^*) = 0.0135094269$. The reduction is 47.6055%. Both actions have positive success probability. Full verification, which minimizes CM, has finite-batch excess 90.8597% relative to $\Phi_3(q_3^*)$.

Fitting procedures. Every method starts at $u = 0$ and receives batches of three records. Let O_i indicate an opened check and $g_i = O_i\psi_i R_i/q$. The generic pair feedback specializes to $-c_0\sum_i g_i^2/3$. It is nonpositive on every batch, so projection leaves its survival at one. The generic triple feedback adds

$$\frac{2}{3}\{O_1 g_2 g_3 + O_2 g_1 g_3 + O_3 g_1 g_2\}. \tag{156}$$

Each direct rule projects a log-survival step with $\alpha_t = (1 + (t-1)/1000)^{-0.6}$. The allocation steps were fixed before the study, without a search over test outcomes.

The plug-in rule maintains running averages of qg_i^2 and g_i, estimating a and γ. Every 32 batches it sets q to the clipped value $\sqrt{c_0\widehat{a}/(2\widehat{\gamma}^2)}$, using one when the estimated mean is zero. It is a historical plug-in estimate, not a conditionally unbiased current-batch allocation gradient. The known-cost rule uses the same three-record batch to average the distinct-pair specialization in (26). It exploits the known derivative $\nabla C = q$ rather than estimating it with an independent work record. This restricted control consequently does *not* witness necessity of three records. Appendix A.5 gives the general lower-bound witness and unknown-law assumptions.

There are 40 seeds, numbered 600–639, per method and budget. At the record budget, each fit uses exactly 12,000 records. At the charged-work budget 1200, a batch starts only when its maximal charge 3.06 can be reserved. The returned log-survival is the arithmetic mean of the post-update log-survivals from batches whose accumulated record count or charge is at least half the respective

Table 5: Allocation-fitting accounting and final-iterate variability. The final column describes the last, unaveraged survival. The returned tail-averaged criterion is reported in Table 1. CPU measurements in the CSV are diagnostic only because fitting runs were parallelized.

Budget	Feedback	Records	Charge	Last q
records	Paired CM	12000.0	12240.00	1.0000 ± 0.0000
records	Triple Φ_3	12000.0	2183.25	0.1335 ± 0.0086
records	Plug-in Φ_3	12000.0	1665.95	0.1121 ± 0.0017
records	Known-cost pair	12000.0	1938.32	0.1122 ± 0.0029
work	Paired CM	1176.0	1199.52	1.0000 ± 0.0000
work	Triple Φ_3	5716.1	1197.52	0.1435 ± 0.0098
work	Plug-in Φ_3	8473.4	1197.62	0.1124 ± 0.0020
work	Known-cost pair	6443.8	1197.45	0.1152 ± 0.0037

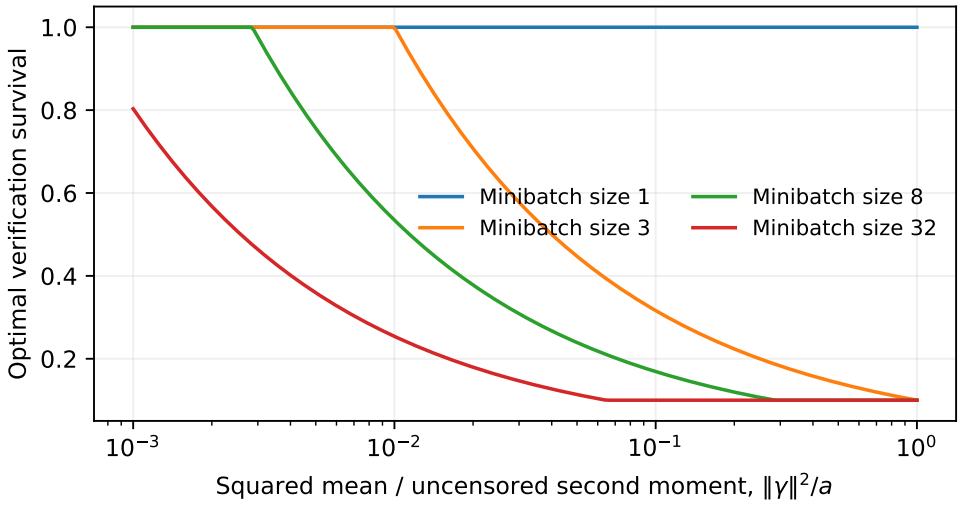

Figure 3: One-checkpoint allocation regimes with overhead-to-verification ratio $\kappa = 0.02$ and floor 0.1. The horizontal ratio is $\|\gamma\|^2/a$, not a success probability. Proposition 3 gives the full/interior/floor boundaries. The graph concerns a frozen-policy smooth-bound criterion, not an ordering of training trajectories.

budget. It is not a charge-weighted average. Table 1 evaluates the exponential of this average; Table 5 additionally reports the last iterate. The average uses every post-update iterate in the specified budget interval. Displayed trajectories retain every twentieth iterate and the endpoint.

The plug-in and known-cost rules have lower variability than the generic triple construction, which nevertheless learns the changed allocation. Equal record counts yield different charges across schedules; equal charges give full verification fewer records. We report both comparisons separately. CPU times come from parallel fitting jobs and are diagnostic, not CPU-matched evidence of a hardware speedup.

Allocation regimes. The stored grid crosses $p(A) \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$, reward means $(0.55, 0.45)$, $(0.7, 0.3)$, $(0.9, 0.1)$, or $(0.99, 0.01)$, batch size $1, 2, 3, 8$, or 32, and overhead $0.002, 0.02, 0.2$, or 2. All 400 cases retain both actions' positive reward probability. The grid is an evaluation of the formula, not 400 additional learning experiments. Figure 3 displays its governing dimensionless ratio. Increasing overhead favors full verification; increasing batch size or squared gradient relative to uncensored second moment can favor thinning.

Table 6: Initial-policy gradient comparison under reference outcomes versus labeling every endpoint after fuel 64 as failure. Cosine is between the two gradients on the stated prompt subset. The all-prompt gradient does not reverse; the slow-optimal subset does.

Prompt subset	Count	Cosine	$\|\gamma\|$	Bias norm
train / all	36	0.600	0.0730	0.0754
train / fast	18	0.989	0.1168	0.0551
train / slow	18	-0.556	0.0917	0.1378
test / all	12	0.469	0.0725	0.0884
test / fast	6	0.984	0.1153	0.0575
test / slow	6	-0.614	0.1086	0.1573

L.2 A PROGRAM GRAMMAR WITH NECESSARY DELAYED SUCCESSES

This study uses the target, reference answer, input-bank generator, fuel meter, checkpoints, and 36/12/12 prompt split specified in Appendix M. It changes candidate reliability and includes an observed task-mode bit. The 60 prompts specify modular summation. For action a, put $v = \lfloor a/2 \rfloor$, $d_v \in \{3, 4, 5, 6, 7, 9\}$, and $z = (n + o_x + v) \bmod d_v$. Even actions use periodic decomposition and odd actions sum directly. Each candidate adds either zero or one to its computed sum according to

Prompt type	Fast candidate adds one when	Slow candidate adds one when
x even	$z = 0$	$z \neq 0$
x odd	$z \geq \max(1, \lfloor d_v/2 \rfloor)$	$z = 0$

The fixed, input-dependent perturbations are constructed errors, not measured failure modes of a code generator. Runtime differences arise from the actual two summation algorithms, not inserted sleeps or padding loops. Only trusted templates are executed; the harness is not an isolation system for arbitrary code.

Input seed 6119 supplies 24 cases for each prompt as previously described. All 720 source strings and all 17,280 reference outcomes are retained, together with source hashes and input arrays. No reference case is discarded. The resulting finite distribution has exactly 30 slow-optimal prompts, all with odd index. On each, every unrestricted per-prompt reference maximizer is removed by labeling endpoints later than fuel 64 as failure. The other 30 prompts preserve their maximizing sets. Training, development, and test contain 18, 6, and 6 slow-optimal prompts, respectively. These are properties of the recorded candidate sets and case distribution, not a claim about the optimum of every shared-parameter policy class.

The fixed stratum is $h(x, a) = (x+2a) \bmod 3$. The 48 policy features are three action-one-hot blocks scaled by 1, $(m_x - 10)/7$, and $(o_x - 3.5)/3.5$, and a fourth action block scaled by $2(x \bmod 2) - 1$. The fourth block permits the shared policy to condition on task mode, which is observed before verification. Strata still mix both modes and both algorithms. At the initial policy, every one of the 12 stratum/checkpoint cells has positive reward-weighted score mass; the smallest is approximately 0.03786. The experiment therefore cannot be explained by zero-signal families or strata containing only successful candidates.

The exact gradient diagnostic evaluates each prompt subset under its own uniform prompt distribution. The negative-label gradient replaces R by $R\mathbf{1}\{\tau \leq 64\}$. On the six slow-optimal test prompts, the cosine with the reference gradient is -0.6135. On all twelve test prompts it is 0.4686. The reversal is confined to the slow-optimal subset and does not extend to the aggregate task. Among reference-maximizing candidates on slow-optimal test prompts, mean best success is 0.9444. Restricting choices to the short-label maximizing set lowers the best reference success within that set to 0.5417. This diagnostic is independent of whether a finite training run reaches either maximizing set.

L.3 LIVE POLICY FITTING AND MATCHED LOGICAL WORK

All seven methods use the ordinary mean of three fresh, independent prompt–action–input–cutoff records. They sample only training prompts. Initial action intercepts range linearly from 1.2 to -1.2; all other parameters are zero. The baseline is zero. The policy and allocation remain fixed while

Table 7: Policy-step selection on 12 development prompts. Each method receives the same seven rates and three seeds at charged-work budget 120. Selection uses development success, not held-out test success. No selected rate is at a grid boundary.

Method	Selected step	Development success (%)
Full verification	1.35	77.88 ± 8.16
Fixed + correction	1.35	76.17 ± 15.81
Endpoint-only	1.35	77.02 ± 8.75
Paired CM	1.35	74.81 ± 15.17
Batch-aware	1.35	76.83 ± 5.16
Plug-in Φ_3	0.45	77.40 ± 15.34
Short timeout = failure	0.15	65.09 ± 3.08

Table 8: Seed-paired differences in held-out reference success, in percentage points. Entries give mean difference and unadjusted 95% interval. All twelve seeds are retained in every contrast.

Contrast	$Q = 180$	$Q = 540$
Batch-aware minus Short timeout = failure	$13.52\ [10.56, 16.48]$	$13.99\ [10.41, 17.57]$
Fixed + correction minus Short timeout = failure	$15.67\ [11.68, 19.66]$	$16.17\ [11.96, 20.39]$
Full verification minus Short timeout = failure	$12.73\ [9.62, 15.83]$	$12.78\ [9.42, 16.13]$
Batch-aware minus Full verification	$0.79\ [-0.98, 2.57]$	$1.22\ [-0.50, 2.94]$
Batch-aware minus Paired CM	$-0.26\ [-0.96, 0.45]$	$0.45\ [-0.43, 1.33]$
Batch-aware minus Fixed + correction	$-2.15\ [-6.85, 2.55]$	$-2.18\ [-7.13, 2.77]$
Batch-aware minus Plug-in Φ_3	$0.40\ [-4.88, 5.68]$	$0.48\ [-4.78, 5.74]$

a batch is collected, and both are updated afterwards. Constructing each source, compiling it, and running its fuel-limited verification occur live. Reference arrays are first used after the measured training stop to evaluate the stored policy.

Full verification has all survivals one. Fixed randomized verification uses $(1, 0.75, 0.5, 0.25)$ with correction. Endpoint-only allocation uses a learned single survival per stratum. Paired feedback targets CM; batch-aware feedback targets Φ_3. The endpoint and checkpoint plug-in comparators both fit Φ_3 in this study, using exponential moving averages with coefficient 0.025 for a, c, and γ. Coefficients start at 0.001 and the mean at zero. They solve every 32 batches. All learned schedules have floor 0.1. The two direct allocation methods use step 0.25. The negative-label control always uses cutoff 64 and treats interruptions as zero. Its zero late survival is deliberate, so the positivity-based reference-gradient theorem does not apply to that baseline.

Reported schedules are those actually applied. The full, fixed, and negative-label controls use their prescribed schedules rather than the unused allocator parameters.

Every method receives the same policy-step grid $\{0.05, 0.15, 0.45, 1.35, 4.05, 12.15, 36.45\}$, development seeds 21–23, and charge 120. Selection maximizes mean success on the twelve development prompts. None of the seven selected steps is at a grid boundary. The common search covers policy steps; it does not exhaustively tune allocation steps, moving-average coefficients, or refresh frequencies. The 147 development fits are separate from the 168 evaluations using seeds 700–711 at charges 180 and 540. Job order is permuted with seed 83112. All comparisons in this study use logical work, and no test prompt or reference test outcome enters step selection.

Logical charge is $0.02+$ executed fuel$/8192$ per record. A new batch requires an unused allowance of at least 3.06. Every completed row satisfies $Q - 3.06 <$ spent $\leq Q$. The constant abstracts generation, policy, and allocation effort; it is not a claim of equal instruction counts or wall time. CPU component measurements are included for diagnosis and were collected during parallel jobs. Reference construction, development fitting, and final evaluation are accounted for separately. We make no CPU speed-ratio claim for this study.

All intervals are unadjusted Student-t intervals across independent training seeds, conditional on this grammar, feature map, fixed case bank, and prompt partition. A paired interval uses seed-matched differences. Reusing the same twelve held-out prompts does not create twelve additional independent training replicates per seed. The fixed corrected schedule performs strongly, and the

Table 9: Applied final schedules at charge 540, averaged over three strata and twelve seeds. The final column is slow-action policy mass (%) on the six slow-optimal test prompts. Fixed/negative schedule overrides are applied explicitly, not inferred from unused learned parameters.

Method	s_1	s_2	s_3	s_4	Slow mass (%)
Full verification	1.000	1.000	1.000	1.000	99.84
Fixed + correction	1.000	0.750	0.500	0.250	99.86
Endpoint-only	0.872	0.872	0.872	0.872	99.56
Paired CM	0.999	0.964	0.914	0.861	99.82
Batch-aware	0.999	0.951	0.902	0.843	99.82
Plug-in Φ_3	0.585	0.193	0.174	0.148	99.75
Short timeout = failure	1.000	0.000	0.000	0.000	0.50

Table 10: Effect of BLAS thread count on plug-in/cross-cost trajectories. One thread gives the primary measurements, while five threads give the previously discrepant observer values. All other inputs are held fixed.

Seed	BLAS threads	Reference success	Logical work
100	1	0.8965273124	356.9959765625
100	5	0.8965260852	356.9952441406
101	1	0.8943364670	356.9610302734
101	5	0.8982148912	356.9837548828

batch-aware/paired contrast remains unresolved. The study shows the cost of optimizing the wrong reward objective, not that three-record feedback is the best optimizer of the correct objective.

L.4 Sensitivity to numerical threading

Two plug-in/cross-cost trajectories in the fast-optimal study differed between the primary runs and a separate observer run. Evaluating the stored final policies gives the reported success in each case. The discrepancy arises in the training trajectory.

Changing only the BLAS thread count reproduces both trajectories. With Python 3.13.5, NumPy 2.3.5, SciPy 1.17.0, and OpenBLAS 0.3.30 on SkylakeX, one BLAS thread reproduces the primary paths at seeds 100 and 101. Five threads reproduce the formerly discrepant observer values. All code, random seeds, input arrays, and scientific parameters are otherwise held fixed. At the first SLSQP call, after 32 batches, both estimated coefficient arrays are bit-identical between the thread conditions. The returned log-survival vectors differ by 6.66×10^{-16} and 4.44×10^{-16}, respectively, while the returned objective values agree at floating-point precision. Later numerical optimization and adaptive sampling amplify these differences.

With single-thread BLAS, two repetitions of all 24 plug-in/cross-cost paths at seeds 100–111 and both logical budgets reproduce the primary policy trajectories. All 36 observer/primary comparisons agree within 10^{-10} for success and 10^{-9} for charge. The supplement includes the thread-control measurements and earlier observer results.

Single-thread agreement does not imply identical trajectories across numerical platforms. Scientific comparisons should be interpreted at the scale of their across-seed uncertainty, not these numerical perturbations.

L.5 Study sizes

The allocation study contains 320 fits, and the policy study contains 147 development fits and 168 evaluation runs, for a total of 635 fits and training runs. The scalar phase grid contains 400 analytic evaluations, not additional fits. The program corpus contains 17,280 reference cases. These counts exclude the complementary fast-optimal and four-prompt studies.

Code, data, and supporting numerical calculations are provided in the supplement.

M FAST-OPTIMAL PROGRAM CONTROLS AND DIAGNOSTICS

M.1 SCOPE OF THE CONTROLS

The controls serve four purposes. Finite-law calculations isolate checkpoint structure, gradient versus reward moments, and the cross-cost surrogate mismatch. Frozen-policy censored replay compares learned schedules with population optima under the same information restrictions. Live executions compare policy learning after independent development-stage step selection. An offline observer evaluates conditional moments along selected trajectories without changing their feedback. We report these measurements and their costs separately.

The program corpus is constructed, not sampled from a pretrained code model. It contains 60 numerical prompts and 720 distinct Python source strings, with 24 input cases per prompt, giving 17,280 executed reference endpoints. Its policy has 36 shared linear-softmax parameters. On all 60 prompts, the best fast candidate is strictly better than the best slow candidate. Labeling endpoints after fuel 64 as failure preserves the unrestricted per-prompt maximizing sets. Delayed optimal candidates are therefore unnecessary in this corpus. Appendix L studies a corpus that requires them. Training, development, and test prompts are disjoint, but all share the same modular-summation grammar. Consequently the held-out evaluation concerns new numerical prompt parameters, not unseen algorithm families, natural-code generalization, or language-model fine-tuning.

M.2 PROGRAMS, VERIFICATION, AND PRE-EXECUTION INFORMATION

Prompt $x \in \{0, \dots, 59\}$ specifies

$$f_x(n) = \sum_{i=1}^{n} ((i + o_x)^{p_x} \bmod m_x), \quad m_x \in \{5, 7, 11, 13, 17\}, \quad p_x \in \{1, 2\}. \tag{157}$$

The modulus cycles with $x \bmod 5$, the exponent is $1 + (\lfloor x/5 \rfloor \bmod 2)$, and $o_x = 1 + \lfloor x/10 \rfloor$. Even-numbered actions use periodic decomposition with a loop of length m_x. Odd-numbered actions directly sum n terms. Each pair of actions has one of six divisibility-based answer perturbations. For variant $v = \lfloor a/2 \rfloor$ and divisor $d_v \in \{3, 4, 5, 6, 7, 9\}$, the fast action adds one precisely when $(n + o_x + v) \bmod d_v = 0$, and the slow action adds one otherwise. Thus both algorithm types can pass and fail depending on the input. Runtime differences come from different summation algorithms, not from inserting a sleep or padding loop. The perturbations impose this reward/runtime relation by construction; they do not estimate its prevalence in natural programs.

Input seed 6119 generates six integers in each of the half-open ranges $[32, 64)$, $[65, 512)$, $[513, 2048)$, and $[2049, 8191)$ for every prompt and shuffles them. Repeated integers are allowed. A reference execution receives 8192 fuel units. Each candidate loop iteration consumes one unit, followed by one final answer-check unit. The reference answer is computed from the periodic formula. Checks stop on fuel exhaustion, and an exhausted reference cap is observed reward zero. The harness executes only the supplied trusted templates and is not a security mechanism for untrusted code. The corpus and prompt partitions are provided in the supplement.

The four checkpoints are 64, 512, 2048, and 8192 fuel units. Among all equally weighted reference cases, 6480, 4320, and 2160 exceed the first three checkpoints, respectively. Of those interrupted cases, 1273, 830, and 420 subsequently pass at the reference cap. These are finite-corpus counts under uniform case weighting, not under every learned policy. Every observable stratum has positive reward-weighted score mass in all four endpoint intervals at the initial policy. The corresponding coefficients are included in the supplement.

The main stratum map is $h(x, a) = (x + 2a) \bmod 3$. It mixes fast and slow algorithms in every stratum and does not reveal a correctness label. The no-feature ablation pools all actions into one stratum. A second static ablation hashes the source into three strata. This is a scrambled pre-execution partition, not an estimated representation or a within-trace continuation rule. No allocator uses an observed partial execution to redefine its current stratum.

The policy feature vector concatenates an action one-hot vector with two copies scaled by $(m_x - 10)/7$ and $(o_x - 3.5)/3.5$. These fixed descriptors are not learned features. Initial logits on the action intercepts are evenly spaced from 1.2 to -1.2 and all other weights are zero. Unlike the uniform-policy four-prompt study, this policy has unequal initial score norms. Their squared values range

from approximately 0.732 to 2.730 across training prompts and actions. The baseline is zero in this study. The four-prompt controls and exact tests cover nonzero baselines.

M.3 Allocation baselines and information matching

Every live method has access to the same prompt, source, static stratum, and reached-prefix information. All use a positivity floor of 0.1 and three independently sampled prompts per batch. The seven allocation rules are as follows.

Full and fixed checks. Full verification uses survival one. Fixed randomized verification uses $(1, 0.75, 0.5, 0.25)$ in every stratum, with inverse-probability reward correction.

Endpoint-only scheduling. One learned probability per stratum chooses either no verification or the full reference cap. It estimates the same signal and interval-work coefficients as the checkpoint plug-in method, sums them within strata, and solves the restricted convex problem. Thus the comparator has the same features and feedback as checkpoint scheduling but cannot truncate at an intermediate checkpoint. It is not given which source families succeed.

Paired, reward-only, and batch-aware scheduling. The paired rule uses the all-distinct-pair CM feedback in (100). The reward-only rule replaces the squared policy-gradient sample by the square of the corrected scalar reward in that feedback. The batch-aware rule adds the distinct-triple correction for Φ_3. Each takes a projected step of size 0.25 after every batch. The correction is not clipped or normalized. A common allocation step controls this choice but does not exhaustively tune each allocator's tracking behavior.

Plug-in scheduling. From an observed record, the arrays $s\,me$ and v/s are unbiased for a and c, respectively. The live plug-in methods maintain exponential moving averages with coefficient 0.025, initialized at 0.001 per coefficient. Every 32 batches they solve the ordered convex CM program, or its endpoint-only restriction, using analytic derivatives and the previous schedule as initialization when applicable. This is a censored-moment plug-in comparator inspired by adaptive randomized-truncation design, not a reproduction of all tuning heuristics in Beatson & Adams (2019). Its solver cost is included in the online CPU clock. The frozen-law experiment uses running averages instead, solves every 64 batches, and supplies the unrestricted plug-in method with the stronger batch criterion using its estimated squared gradient mean. The live and frozen protocols therefore adapt to changing and fixed laws, respectively.

No method accesses the reference reward table during a live training run. Reference outcomes are consulted only after its measured stopping point to compute exact success on the fixed training, development, and test case distributions. Source construction, compilation, verification, and both updates run in Python rather than through a reward-table simulator. Only the frozen-schedule studies and the separate offline diagnostic observer use censored replay.

M.4 Separating allocation and optimizer choices

Training prompts are 0–35, development prompts 36–47, and final test prompts 48–59. Each allocation method is crossed with the ordinary average of three corrected score vectors and with the off-diagonal three-record cross-cost update in (100). The latter is not multiplied by a hidden correction to match the mean rule's effective step.

Every method/update pair receives the same seven-step grid

$$\{0.05, 0.15, 0.45, 1.35, 4.05, 12.15, 36.45\}, \tag{158}$$

three development seeds $11, 12, 13$, and logical budget 60. Selection maximizes mean development-prompt success, never final test success. This gives 294 completed selection runs. The initial four-step development grid put every cross-cost choice at its upper boundary. Before completing the final evaluation, it was expanded for every method. The pilot records and partial evaluation are retained separately and excluded from the reported tables. The expansion criterion was the development boundary, not a favorable test comparison. One endpoint/cross-cost choice remains at the expanded upper boundary. The search is therefore not exhaustive optimizer tuning.

Table 11: Frozen-law learning from 9000 censored records per run. Ten seeds per method. Φ_3 intervals are 95% intervals; remaining columns are means. The record count, not charged work, is matched.

Allocator	Φ_3	C	M	Fit work
Endpoint-only	0.08231 ± 0.00000	0.11180	0.7225	1011.2
Paired CM	0.05092 ± 0.00083	0.04758	1.0574	548.3
Reward-only	0.05082 ± 0.00075	0.05009	1.0013	586.3
Plug-in	0.04926 ± 0.00009	0.04357	1.1173	402.2
Batch-aware	0.05079 ± 0.00080	0.04729	1.0618	544.5

After selection, each method/update pair is evaluated with seeds 100–111 at logical budgets 120 and 360, for 336 runs. The CPU comparison uses seeds 300–307 and a two-second process-CPU target, for 112 runs. A further 192 runs use six seeds per condition for the full, endpoint, paired, and batch-aware rules, both updates, and either pooled strata, source-hash strata, or overhead 0.1 or 0.5. These sensitivity conditions retain the previously selected policy steps rather than retuning after the shift. All job orders are randomized using seed 931.

Reported confidence intervals are Student-t intervals over the stated seeds and are not adjusted for multiple comparisons. Paired differences use the seed-matched differences, not subtraction of marginal confidence endpoints. They measure training randomness conditional on this fixed program grammar, feature map, and input bank. They do not describe arbitrary natural programs, and the small CPU sample does not establish performance across hardware. All hyperparameter choices and raw rows remain in the package, including unfavorable conditions.

M.5 Logical and measured accounting

The principal logical charge per sampled program is $0.02+$ executed fuel$/8192$. Three times the maximum charge is reserved before starting a batch, so logical runs cannot exceed their cap and may leave less than one maximum batch unused. Generation and optimization overhead are abstracted by the constant, not proven equal across methods. Sensitivity changes this declared constant, not the instruction-count trace itself.

The process-CPU clock starts after common descriptor and input-array initialization. It includes sampling, constructing source strings, compilation, executed verification, computing policy scores, policy updates, allocator updates, solver calls, and in-loop bookkeeping. It excludes the prior reference-corpus build, development search, and final reference-table evaluation. There is no pretrained-model generation cost in this controlled task. These excluded costs are separate recorded costs, not purported online savings. The CPU stop is checked between batches and may overshoot by the last batch. This CPU stopping rule differs from the theorem's hard logical-work guarantee. CPU decomposition labels "generation" and "verification" refer to source construction and the compiler/execution path, respectively; the residual between component sums and total includes clock checks and snapshot bookkeeping.

M.6 Frozen-law results and exact mechanism checks

At the stated initial policy, 50 independent allocation studies use ten seeds and five learned methods, each observing 9000 censored reference cases. This holds record count fixed, not computational cost. Charged work is recorded and differs by schedule. Replay CPU time is not a measurement of avoided live verification time. The final frozen schedules are evaluated using full reference coefficients only after learning. Population optima use the same ordered feasible set and positivity floor.

The best endpoint-only schedule is full verification in this corpus. Its Φ_3 value is approximately 0.0823055, compared with 0.0489930 for the numerical checkpoint optimum. The CM-optimal checkpoint schedule has $\Phi_3 = 0.0489945$, only about 0.003% above that optimum. Thus the corpus discriminates checkpoint structure from endpoint thinning but scarcely discriminates the population finite-batch correction. Reward-only checkpoint design is similarly close at the population optimum. The checkpoint gap does not show that the triple correction or gradient weighting is indispensable in this corpus.

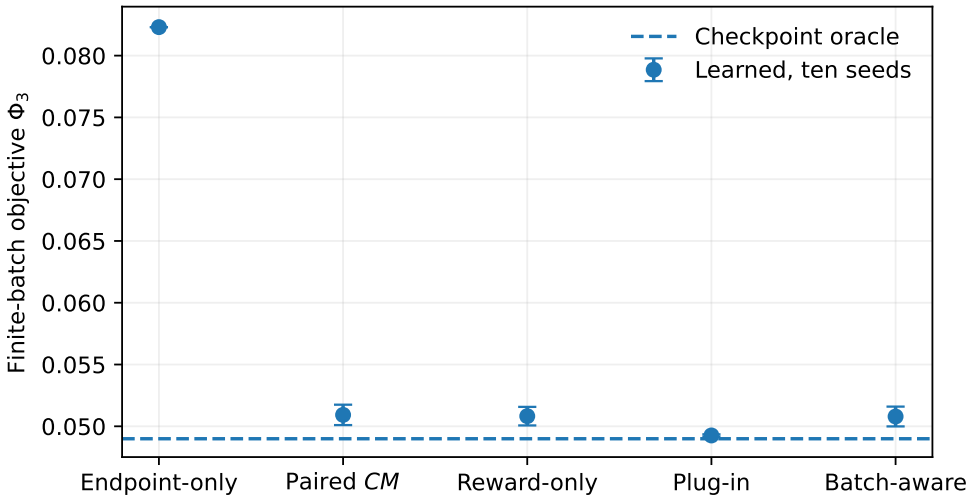

Figure 4: Frozen fast-optimal corpus. Checkpoint allocations improve on endpoint-only thinning, but the paired and finite-batch population optima nearly coincide. The plug-in learner remains a strong stationary baseline.

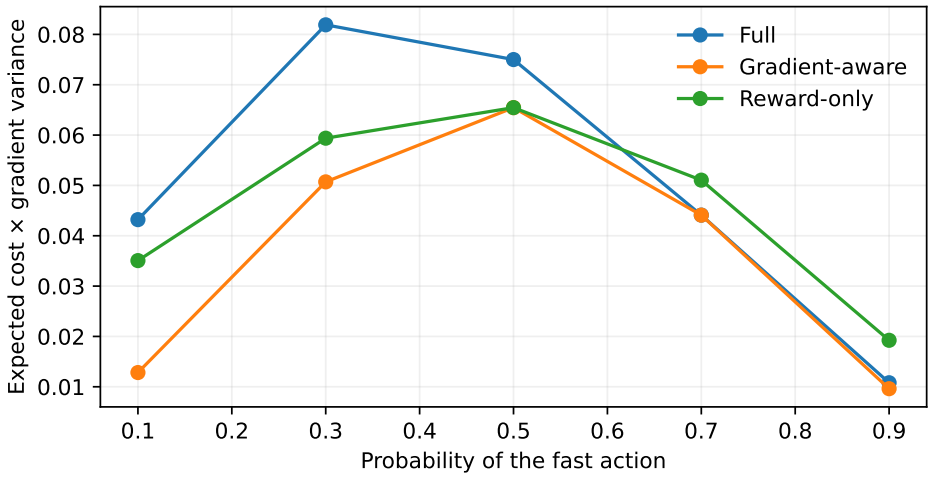

Figure 5: Exact cost-times-gradient-variance on the declared equal-reward two-action grid. Nonuniform policy scores can make reward-only sampling inefficient even when reward probabilities are equal. The common-probability point is retained as a control.

A separate exact two-action grid varies $p(A) \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ and a reward-mean gap of zero or 0.05, with times $0.1, 1$, overhead 0.05, and action-specific strata. At $p(A) = 0.9$ with equal success means 0.5, the gradient is zero and the variance–cost criterion equals CM. Gradient-aware allocation gives approximately 0.009617, while reward-only allocation gives 0.019234. The full grid, including cases where reward and gradient criteria agree, is provided. This calculation tests the gradient-MSE mechanism, not policy-training improvement.

M.7 Learning results, accounting, and sensitivity

Tables 12–16 report the complete method/update comparisons, development choices, CPU components, and sensitivity conditions. At logical charge 360 with mean updates, batch-aware minus full-verification test success is 0.22 percentage points, with paired 95% interval $[-4.10, 4.54]$. At the two-second CPU target the contrast is 1.42 points, with interval $[0.09, 2.74]$. The latter is evidence

Table 12: Live logical-work comparisons. Test success (%), with 95% intervals over 12 seeds. All values use the fixed development-selected steps.

Allocation	$Q = 120$, mean	$Q = 120$, cross	$Q = 360$, mean	$Q = 360$, cross
Full	80.95 ± 4.64	78.62 ± 5.54	81.19 ± 4.71	79.91 ± 5.98
Fixed + IPW	81.19 ± 2.63	78.21 ± 3.84	81.23 ± 2.64	80.31 ± 3.58
Endpoint-only	82.64 ± 3.70	76.16 ± 11.38	83.21 ± 3.79	77.09 ± 11.65
Paired CM	81.05 ± 4.74	75.49 ± 3.34	81.41 ± 4.87	78.75 ± 3.71
Reward-only	80.56 ± 4.57	76.61 ± 2.81	80.67 ± 4.66	80.06 ± 2.95
Plug-in	82.26 ± 2.84	79.25 ± 5.89	82.43 ± 2.93	80.53 ± 6.37
Batch-aware	81.05 ± 4.74	78.30 ± 5.49	81.41 ± 4.87	79.70 ± 5.95

Table 13: CPU-run accounting. Mean actual CPU seconds, logical charge, and batch count; the last column is the largest overshoot in milliseconds. The CPU budget is checked between batches and is not the hard logical-work theorem.

Allocation	Update	CPU s	Work	Batches	Max extra ms
Full	mean	2.0003	231.43	3468	0.51
Full	cross	2.0002	228.55	3248	0.56
Fixed + IPW	mean	2.0002	220.86	3353	0.43
Fixed + IPW	cross	2.0003	312.00	3241	0.48
Endpoint-only	mean	2.0002	207.49	3112	0.46
Endpoint-only	cross	2.0002	208.20	3135	0.53
Paired CM	mean	2.0004	200.70	2995	0.58
Paired CM	cross	2.0004	229.16	2844	0.68
Reward-only	mean	2.0004	196.86	2936	0.60
Reward-only	cross	2.0003	223.64	2712	0.70
Plug-in	mean	2.0004	223.07	3067	0.63
Plug-in	cross	2.0003	184.79	2722	0.86
Batch-aware	mean	2.0003	201.01	3000	0.49
Batch-aware	cross	2.0004	196.81	2787	0.60

about that particular method/update implementation, not about the incremental triple term. Comparisons with endpoint-only and paired-CM allocation do not establish a consistent advantage for that term in this corpus. All individual seed records and unfavorable comparisons are retained.

M.8 A TRAJECTORY OBSERVER FOR THEOREM-LINKED QUANTITIES

The diagnostic observer reruns selected work-budget trajectories with the same learner and random seeds, adding full-reference calculations that are not passed to either update. Its final success and charge are compared with the corresponding live run. The additional full-reference calculations are excluded from training-speed comparisons. All 36 observer reruns match the primary success within 10^{-10} and charge within 10^{-9} after pinning BLAS to one thread. The two prior plug-in/cross-cost discrepancies are reproduced by changing only the BLAS thread configuration to five. Appendix L.4 examines this numerical sensitivity. Table 17 reports the single-thread observer results.

At every batch of these trajectories, the observer computes the exact current γ, C, M, Φ_3, and both update second moments from the finite reference law. It accumulates $\sum C_t \|\gamma_t\|^2$ and $\sum C_t$. Their pooled ratio across seeds estimates the ratio of expectations appearing in the theorem, rather than treating a small final gradient as a guarantee of high success. Numerical oracle schedules are computed every 100 batches. Reported oracle path length and allocation gaps are therefore quantities on this sampled grid, not the full per-batch variation in a dynamic-regret theorem. Solver ties can also affect the path of a selected numerical minimizer.

The moment diagnostics distinguish the exact update penalty from conservative global bounds. They do not certify the chosen learning rates: the stationarity theorem's worst-case constants can be much larger. Small c_0 and a small positivity floor weaken the hard-budget bound even when empirical feedback moments are modest.

Table 14: Policy-step selection from 7 candidates and 3 development seeds per allocation/update pair at $Q = 60$. Values are selected steps and development-prompt success (%). Endpoint/cross selects the upper grid boundary.

Allocation	η, mean	Dev, mean	η, cross	Dev, cross
Full	1.35	76.98	12.15	73.32
Fixed + IPW	4.05	74.76	12.15	75.41
Endpoint-only	1.35	77.04	36.45	75.96
Paired CM	1.35	76.98	4.05	74.57
Reward-only	1.35	76.98	4.05	74.28
Plug-in	4.05	74.76	12.15	76.47
Batch-aware	1.35	76.98	12.15	75.79

Table 15: Mean CPU milliseconds for the mean-update CPU comparison. Total includes additional clock and snapshot bookkeeping. Source construction is not language-model generation.

Allocation	Source	Verify	Policy	Allocate	Total
Full	1013.9	908.6	61.5	1.7	2000.3
Fixed + IPW	1032.3	890.5	60.4	1.8	2000.2
Endpoint-only	946.4	841.7	56.4	142.4	2000.2
Paired CM	904.8	811.8	46.4	223.7	2000.4
Reward-only	880.8	795.4	45.0	265.7	2000.4
Plug-in	905.0	816.1	54.1	212.3	2000.4
Batch-aware	878.0	800.0	44.2	264.3	2000.3

For scale, the fixed policy descriptors satisfy $\|\psi\| \leq 2\sqrt{2 + (5/7)^2} < 3.17$ for every parameter vector. At $n = 3$, $\epsilon = 0.1$, $B_\star = 1$, and $c_0 = 0.02$, equation (34) therefore permits $V_3 \approx 919783$. In the observer runs, the realized mean squared batch-aware allocation feedback is approximately $5.9e - 05$ for mean updates and 0.000127 for cross-cost updates. These are empirical averages, not bounds or estimates of the worst-case constant. This gap explains why we do not use the global theorem to select experimental steps.

M.9 Normalization across group sizes

The finite-group study enumerates all multinomial count vectors for the four action/revealed-reward categories. It uses $p = 1/2$, $(\mu_A, \mu_B) = (0.9, 0.6)$, $s_B = 1$, $s_A \in \{0.1, 0.25, 0.5, 1\}$, and group sizes $2, 4, 8, 16, 32, 64$. It computes the expected update and mean-squared error relative to the true unnormalized gradient for group normalization, ordinary corrected REINFORCE, and leave-one-out centering. The normalized estimator has a different limiting scale, so its MSE against the original gradient is not purely a variance metric. The ordinary and leave-one-out means agree with the original gradient to numerical precision. The live program comparison itself uses unnormalized updates, not a complete GRPO or RLOO implementation.

Table 16: Sensitivity of test success (%) at $Q = 180$, with 6 seeds per cell. The columns change only the stratum map or the logical non-verification overhead; selected policy steps are not retuned under these shifts.

Allocation	Update	Pooled	Source hash	$c_0 = 0.1$	$c_0 = 0.5$
Full	mean	78.96 ± 8.19	78.96 ± 8.19	77.88 ± 7.57	74.92 ± 7.10
Full	cross	79.55 ± 9.45	79.55 ± 9.45	76.89 ± 7.52	80.52 ± 5.56
Endpoint-only	mean	78.96 ± 8.19	79.99 ± 9.04	78.06 ± 7.54	74.92 ± 7.10
Endpoint-only	cross	72.55 ± 19.74	72.72 ± 19.51	67.99 ± 24.19	70.26 ± 28.27
Paired CM	mean	78.21 ± 7.18	78.95 ± 8.18	77.88 ± 7.56	74.95 ± 7.06
Paired CM	cross	78.56 ± 5.58	74.98 ± 5.00	80.41 ± 4.43	77.81 ± 6.28
Batch-aware	mean	78.73 ± 7.84	78.95 ± 8.18	77.88 ± 7.56	74.95 ± 7.06
Batch-aware	cross	77.77 ± 11.36	78.07 ± 10.35	77.16 ± 8.07	80.52 ± 5.56

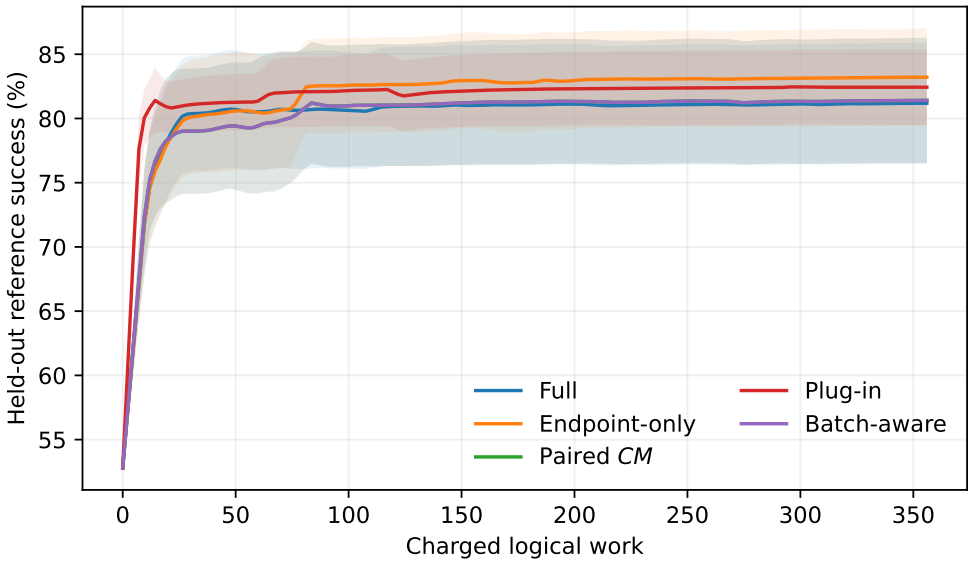

Figure 6: Held-out-prompt success during mean-update logical-work runs, using independently selected development steps. Curves average twelve seeds at budget 360. Intermediate evaluations use stored policy snapshots after the run, not additional feedback to the learner.

Table 17: Separate observer reruns on seeds 100–102 at $Q = 360$. $\widehat{S}_Q$ pools exact per-batch conditional numerator and denominator sums. The next column is mean exact update second moment. Oracle path length is sampled every 100 batches. All 36 observer/primary comparisons agree after pinning BLAS threading, as documented in the text. The diagnostics do not certify a speed advantage.

Allocation	Update	$\widehat{S}_Q$	Update moment	Oracle path
Full	mean	1.20×10^{-4}	1.32×10^{-2}	8.34
Full	cross	1.62×10^{-4}	3.94×10^{-5}	6.20
Endpoint-only	mean	1.17×10^{-4}	8.09×10^{-3}	11.57
Endpoint-only	cross	6.06×10^{-5}	1.16×10^{-5}	4.36
Paired CM	mean	1.11×10^{-4}	4.21×10^{-3}	4.42
Paired CM	cross	5.60×10^{-4}	2.53×10^{-4}	6.11
Reward-only	mean	1.11×10^{-4}	3.91×10^{-3}	4.42
Reward-only	cross	5.82×10^{-4}	1.91×10^{-4}	8.29
Plug-in	mean	4.92×10^{-5}	2.69×10^{-3}	5.31
Plug-in	cross	1.81×10^{-4}	3.15×10^{-5}	6.23
Batch-aware	mean	1.11×10^{-4}	4.21×10^{-3}	4.42
Batch-aware	cross	1.61×10^{-4}	3.26×10^{-5}	5.40

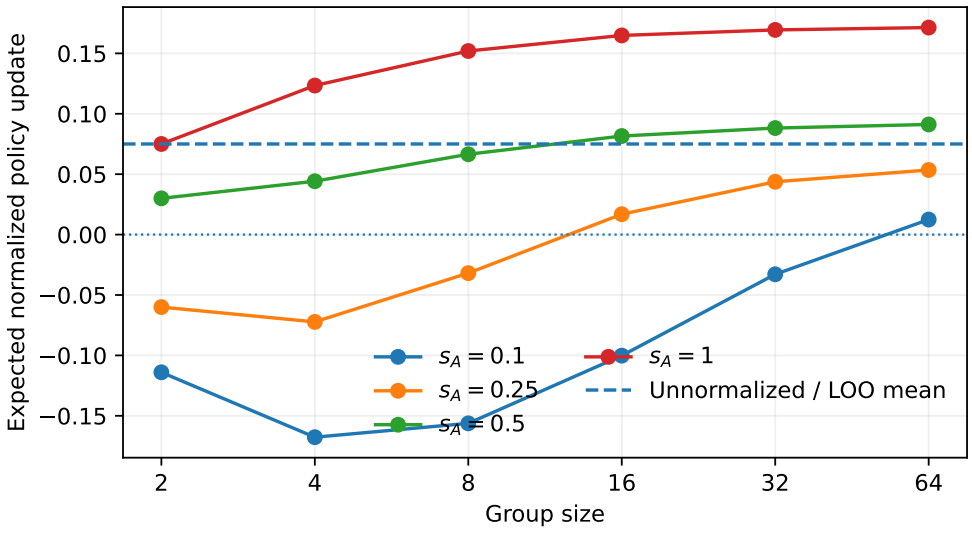

Figure 7: Exact expected normalized updates for unequal reveal probabilities across group sizes. The true unnormalized gradient is 0.075. Equal propensities preserve its sign at every finite group size, while highly unequal propensities can reverse small-group updates. Large-group convergence is to a positive fixed-prompt rescaling, not generally to the original magnitude.

Table 18: Reference-budget success (%). Entries are means ± 95% Student-t confidence intervals across 20 seeds. All methods share the same budget within each column.

Method	Logical work	CPU time
Full verification	92.16 ± 1.58	90.65 ± 2.02
Fixed schedule + IPW	92.72 ± 1.75	83.49 ± 3.19
Paired allocation	93.08 ± 1.00	86.41 ± 2.07
Short timeouts as failures	52.56 ± 0.60	38.76 ± 0.23
Discard interrupted checks	56.67 ± 0.79	46.04 ± 1.11
Increasing deterministic budgets	77.95 ± 3.10	78.97 ± 2.17
Fixed schedule + model assistance	92.58 ± 1.03	82.18 ± 1.84

N FOUR-PROMPT TEMPLATE EXPERIMENTS AND NEGATIVE CONTROLS

The four-prompt experiments provide sanity checks and negative controls rather than evidence for checkpoint-specific allocation. Endpoint thinning attains their frozen-policy oracle. Section 7 presents the main mechanism studies, and Appendix M gives the complementary held-out-prompt controls.

N.1 SCOPE AND DATA CONSTRUCTION

The experiments test the estimators and allocation mechanism in a finite, controlled program space. They do not measure timeout prevalence in an existing language model, and they do not fine-tune a language model. The execution-time heterogeneity is deliberately constructed. Its purpose is to expose cases in which a short training timeout and a reference-budget failure are different observations, together with controls in which they are not.

The four prompts ask for the functions

$$f_0(n) = \sum_{i=1}^{n} i, \quad f_1(n) = \sum_{i=1}^{n} i^2, \quad f_2(n) = \sum_{i=1}^{n} \mathbf{1}\{i \text{ odd}\}, \quad f_3(n) = \sum_{i=1}^{n} (i \bmod 3). \tag{159}$$

For each prompt there are twelve actions. An action selects one of four source-template families and one of three variants. The policy chooses the action, after which the source is constructed and compiled afresh. The family index, which is known before execution, supplies the allocation stratum.

Fast, imperfect formulas. The first family evaluates a closed-form expression for the target and adds one when n is divisible by a variant-specific modulus in $\{4, 5, 7\}$. One fuel instruction is charged inside the candidate. This family therefore contains fast programs that pass some test suites and fail others. For the fourth task the closed form is $3\lfloor n/3 \rfloor + d(n \bmod 3)$ with $d(0) = 0, d(1) = 1, d(2) = 3$.

Slow, correct loops. The second family accumulates the appropriate summand over $i = 1, \ldots, n$. Before each addition, it executes a redundant loop containing 32, 64, or 96 fuel instructions. These variants compute the correct answer but can be interrupted before the suite finishes. Their redundant work is intentional and is not evidence that realistic code has the same timing distribution.

Slow, incorrect loops and nontermination. The third family performs the analogous computation with 96, 128, or 160 fuel instructions per summand and adds one to the result. The fourth family loops indefinitely while consuming fuel. Its three action indices produce the same source, retaining the same action-space size across families. A run stopped at the reference cap has reward zero even when nontermination is the cause.

Inputs and verifier. A NumPy generator with seed 1701 draws a fixed array of shape $4 \times 24 \times 4$, with entries uniformly sampled from $\{2, \ldots, 32\}$. Each of the 24 suites for a prompt has four inputs. The verifier runs the candidate on those inputs in order, charges one additional fuel instruction per completed answer, and stops on the first incorrect answer or after all four correct answers. A fuel object increments its counter until the requested limit and then raises an exception. Thus fuel is an

explicit abstract instruction count, not a measurement of native instructions, Python bytecodes, wall time, or CPU cycles.

The reference cap is 12,288 fuel instructions and the checkpoints are $192, 768, 3072, 12{,}288$. Exhaustion at an earlier checkpoint is missing reward. Exhaustion at the reference cap is observed reward zero. The code compiles each generated candidate with a restricted built-in namespace, but this construction is *not* a security sandbox. Only the supplied trusted templates are executed.

Reference distribution and evaluation. Each regime has $4 \times 12 \times 24 = 1{,}152$ reference executions. The three regimes give 3,456 executions in total. Their rewards and capped endpoint times are computed independently of the training loop. Evaluation computes

$$J(\theta) = \frac{1}{4} \sum_{x=0}^{3} \sum_{a=0}^{11} \pi_\theta(a \mid x) \left(\frac{1}{24} \sum_{q=0}^{23} R(x, a, q) \right). \tag{160}$$

There is no evaluation Monte Carlo error relative to this finite distribution. The same suite bank defines the training distribution, so this is not a held-out-input or held-out-prompt generalization estimate. The learner samples suites and executes their candidates afresh. It does not use the reference reward array in its reward estimator, baseline, or allocator. The array is used only to compute diagnostic and final objective values.

N.2 Regimes and censoring diagnostics

The mixed regime uses the families above. Under its initially uniform policy, reference success is 36.20%, 25% of executions reach the reference cap without a terminal answer, and 71.96% are interrupted by a 768-instruction check. Of the latter, 34.62% pass at the reference cap. At 192 instructions, the corresponding interruption rate and delayed-pass fraction are 75.00% and 33.33%. At 3,072 instructions they are 45.49% and 34.35%.

The all-fast control changes every finite loop's repetition count to one and replaces the nonterminating family by a quickly returned incorrect answer. No reference execution is censored at the first checkpoint. Initial reference success remains 36.20%. The no-delayed-success control adds one to the second family's output, so every slow program is incorrect. Initial reference success falls to 11.20%. Its interruption rate at 768 instructions is 61.37%, but no interrupted program ultimately passes. Counts for all three shorter checkpoints are provided in the supplement.

These proportions describe the constructed benchmark, not deployed verifiers. The mixed regime makes delayed successes common by design; it does not estimate their prevalence in a code-generation system.

N.3 Policy parameterization, updates, and baselines

The trainable parameters are a 4×12 matrix of logits, initialized to zero. Each prompt has a softmax action distribution. Prompts and suites are sampled uniformly, independently between the two members of a pair. The score for a selected action is its one-hot indicator minus its prompt's action probabilities, embedded in the corresponding row of the parameter matrix.

All main methods use $K_t = (C_1 g_2 + C_2 g_1)/2$ with policy step size $\eta = 1$. Paired allocation starts at full survival, uses $\alpha = 1$ and $\epsilon = 0.1$, and projects after every pair. The estimator baseline in the main comparison is zero. No entropy penalty, KL penalty, clipping, group normalization, optimizer momentum, learned timing model, or pretraining is used. The allocation loss is not evaluated using population coefficients in live training.

Full verification and fixed randomized schedules. Full verification always checks to the reference cap or earlier termination. The fixed schedule uses survival $(1, 0.75, 0.5, 0.25)$ in each stratum, equivalently choosing each of the four cutoffs with probability $1/4$. It uses the endpoint inverse weight whenever a reward is observed. It is uniform over cutoff *levels*, not a uniform survival probability over time.

Short timeouts as failures and discarding. The short-failure baseline always checks to 768 instructions and returns zero for an interrupted execution. The discard baseline uses the same cap but

Table 19: Parameters of the reported experiments. Allocation-only studies and live policy training use different stated step sizes.

Parameter	Value
Prompts / actions per prompt / strata / suites per prompt	$4/12/4/24$
Reference fuel cap	$12{,}288$
Checkpoint fractions of cap	$(1/64, 1/16, 1/4, 1)$
Live policy step η / allocation step α	$1/1$
Allocation-only step α / rounds	$0.25/4{,}000$
Positivity floor ϵ / overhead charge c_0	$0.1/0.02$
Main logical-work budget / process-CPU target	$300/0.5$ seconds
All-fast / no-delayed-success work budgets	$100/300$
Training seeds / allocation seeds	100–$119/200$–219

rejects interrupted draws and samples again until an observed endpoint is obtained or its budget guard stops the run. Every rejected draw's generation and execution are charged. When two accepted draws are available, the cost attached to each includes its preceding rejected attempts. These baselines do not generally preserve the reference objective and are not applications of the unbiasedness theorem. In particular, rejection changes the sampled program distribution.

Increasing deterministic budgets. At consumed-budget fraction f, the current cutoff index is

$$\ell(f) = \min\{4, 1 + \lfloor 8f \rfloor\}. \tag{161}$$

The index counts checkpoints from one. Full verification is reached at $f = 3/8$ and retained thereafter. Early interruptions are labeled zero. The fraction uses charged work in work-matched runs and elapsed process CPU in CPU-matched runs.

Model assistance. The fixed-schedule model-assisted baseline uses the augmentation in (62). Each prompt–action pair has a $\mathrm{Beta}(1, 1)$ reward predictor. It is updated only by draws assigned the full reference budget, including full-budget administrative failures, and only after both current rewards and updates have been formed. Therefore the predictor is predictable and does not read the current pair's labels before estimating its rewards. Early observed rewards are not used to update this simple predictor, avoiding a need to model their informative observation pattern.

Prompt-baseline ablations. Full, fixed-schedule, and paired methods are also tested with a separate $\mathrm{Beta}(1, 1)$ predictor for each prompt as the centered baseline $\beta_t(x)$. Updates again use only full-assignment draws and occur after the pair. The entire centered score is inverse weighted, including negative centered scores from observed failures. The paired baseline allocator uses the corresponding centered second moment rather than reusing zero-baseline coefficients.

The unscaled-update ablation retains paired allocation but replaces K_t by $(g_1 + g_2)/2$. This estimator is also unbiased for ∇J per iteration, but it uses a different effective step per unit cost at the same numerical η. It is not an equal-learning-rate comparison in computational time, and Theorem 16 is not applied to it.

N.4 Computation and timing protocol

A draw's logical charge is $0.02+$ fuel used$/12{,}288$. This includes a fixed model of non-verification overhead, not a claim that generation and update costs are constant on hardware. A new pair is started only when at least $2(1.02)$ charge remains. Every final draw is included in the cumulative charge. In the main logical-work runs, realized total work ranges from 297.9624 to 299.4352 under a target of 300. The small unused remainder follows the conservative reservation rule.

Process-CPU runs use a 0.5-second target checked between pairs. They include fresh source construction, sampling, compilation, metered execution, policy and allocation arithmetic, and bookkeeping. In-loop diagnostic evaluation is also inside this clock. The precomputed reference-corpus construction and final reference evaluation are outside the training clock. They are evaluation infrastructure,

Table 20: Mean CPU milliseconds in the CPU-matched experiment. Generation includes sampling and source construction. Verification includes compilation and execution. Updates include policy and allocation computations. Total also includes bookkeeping and in-loop diagnostics.

Method	Generate	Verify	Update	Total
Full verification	56.4	364.5	4.2	500.7
Fixed schedule + IPW	82.5	308.2	6.2	500.5
Paired allocation	66.3	313.9	41.7	500.6
Short timeouts as failures	110.4	237.6	8.2	500.3
Discard interrupted checks	120.6	244.6	4.6	500.3
Increasing deterministic budgets	75.0	316.9	5.7	500.6
Fixed schedule + model assistance	81.5	308.8	6.5	500.5

identical across methods, and are not available for training decisions. The maximum recorded CPU overshoot is 0.001943 seconds. No claim of a pathwise 0.5-second CPU cap is made.

Runs are sequential, with BLAS and OpenMP thread counts set to one. The recorded environment is Python 3.13.5, NumPy 2.3.5, and SciPy 1.17.0 on an Intel Xeon Platinum 8272CL CPU at 2.60 GHz in a virtualized Linux environment. No GPU is used. The 640 live training runs consume 650.83 process-CPU seconds in aggregate, excluding corpus construction, allocation-only experiments, report generation, and mathematical checks. Timing values are specific to this implementation and environment. Method order is fixed within each seed rather than randomized, another reason not to generalize small timing differences.

The measured generation, verification, and update subtotals in Table 20 do not exhaust total CPU time. The remainder includes unprofiled bookkeeping and diagnostics. These subtotals are not used to adjust the stopping rule or retrospectively remove allocation overhead.

N.5 Allocation-only experiments and the oracle

At a frozen uniform policy, population a, c are computed from all finite reference outcomes for *evaluation only*. The learner receives a simulated censored prefix from the already executed corpus. The simulator returns the used work and, only if reached, the endpoint and its reward. It never supplies an unobserved terminal time or label to the allocation update. This separates exact loss evaluation from the censored feedback used for learning.

There are 20 seeds and 4,000 rounds, with step size 0.25, floor 0.1, and overhead 0.02. The paired learner starts at full verification. A same-sample ablation starts at the same point and uses $m(v - C^{\mathrm{obs}}e)$ from a separate single draw. The fixed schedule and full-verification losses are evaluated without adapting their schedules. The random stream interleaves paired and same-sample draws in the specified code. Losses are logged before the update at rounds $0, 100, \ldots, 3900, 3999$. Thus the final plotted and tabulated loss is after 3,999 allocation updates, whereas the saved final survival, when generated, is after all 4,000 updates.

The numerical oracle minimizes the convex F over the same ordered, bounded log-survival set. It uses SLSQP with analytic gradients, objective tolerance 10^{-13}, at most 1,000 iterations, and up to three initial points. A solution is accepted only if

$$\|u - \Pi_{\mathcal{U}}(u - \nabla F(u))\|_\infty \le 2 \times 10^{-6}. \tag{162}$$

An initial point meeting the stricter 10^{-8} tolerance ends the multi-start search early. The oracle sees the full coefficients only to define the comparison, not to initialize or inform the learner. It is a numerical minimizer of the stated surrogate, not an oracle for final policy accuracy.

For the changing-policy study, 4,000 rounds are divided into 20 blocks of 200. Let $f = b/19$ for block $b = 0, \ldots, 19$. Each prompt's logits are f for actions 0–2, $2\sin(\pi f)$ for actions 3–5, zero for actions 6–8, and $1.5\cos(\pi f)$ for actions 9–11. The softmax of these logits specifies a deterministic policy path. At each block boundary the evaluation coefficients and oracle are recomputed. The learner observes only its censored draws from the new policy. This is a tracking diagnostic under prescribed drift, not a measured dynamic-regret upper bound for policy training.

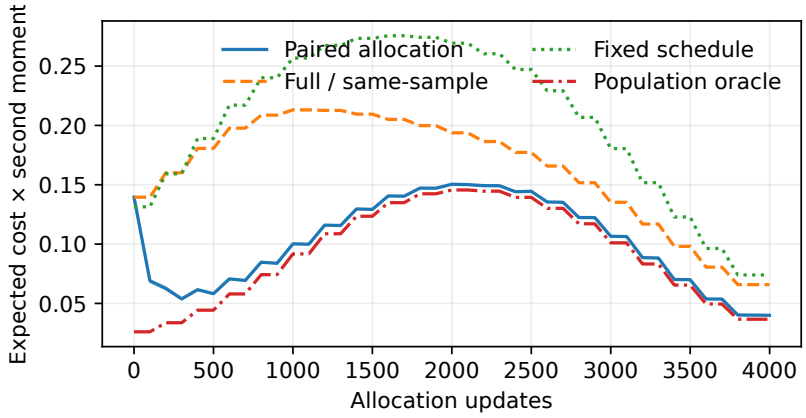

Figure 8: Allocation under a prescribed 20-block changing policy. Each curve is a 20-seed mean and the paired curve includes pointwise 95% intervals. Jumps reflect changes in the population objective, not changes in the plotting scale. The final paired value is 0.04000, versus 0.06584 for full verification and 0.03667 for the population oracle.

Table 21: Negative controls. Reference success (%) at matched logical work within each regime. All-fast uses $Q = 100$ and no-delayed-success uses $Q = 300$. Intervals describe 20-seed uncertainty.

Method	All-fast	No delayed success
Full verification	57.85 ± 0.29	43.21 ± 2.62
Fixed schedule + IPW	57.85 ± 0.29	43.39 ± 1.77
Paired allocation	57.85 ± 0.29	43.57 ± 2.13
Short timeouts as failures	57.85 ± 0.29	45.88 ± 0.97
Discard interrupted checks	57.85 ± 0.29	53.72 ± 0.49
Increasing deterministic budgets	57.85 ± 0.29	44.80 ± 1.78
Fixed schedule + model assistance	57.85 ± 0.29	42.79 ± 2.26

N.6 Negative controls, ablations, and learning curves

The all-fast control makes the seven methods essentially indistinguishable in final success. The no-delayed-success control removes the reason to reinterpret an interrupted candidate as a potentially positive label. In that regime, discarding achieves a higher finite-budget success rate than the corrected methods in these runs. This is not a contradiction of objective preservation. A biased estimator can have favorable finite-budget behavior on a particular distribution, and these experiments are not tuned to equalize the effective steps of rejected and unrejected sampling distributions. They also rule out a blanket claim that censoring correction is always the most efficient training choice.

The prompt-baseline runs do not establish a consistent advantage for paired allocation over full verification. The unscaled-update ablation reaches 99.27% mean success, above the cross-cost variants at the same nominal step size. Its different effective step shows that final accuracy depends on the update convention. The computation-weighted theorem therefore does not establish that its cross-cost update is the best empirical optimizer.

N.7 Uncertainty and tuning

Each live condition has 20 seeds, 100–119. The four seven-method comparisons contribute 560 runs, and four additional ablations contribute 80 runs. The allocation studies use seeds 200–219 separately for frozen and changing policies. A reported mean interval is $\bar{z} \pm t_{0.975,19} s_z / \sqrt{20}$. A paired method difference is formed within seed before computing the same interval. Common seeds do not imply identical sampled trajectories once policies or schedules differ. Intervals are not adjusted for multiple comparisons and should be read as descriptive, not as simultaneous guarantees. Plot shading is pointwise, not a simultaneous confidence band.

Table 22: Prompt-baseline and unscaled-update ablations at $Q = 300$. The unscaled update changes the effective policy step per unit work and is not a retuned comparison. CPU seconds are measured, not matched in this table.

Method	Success (%)	CPU seconds
Full verification	92.16 ± 1.58	0.602 ± 0.026
Paired allocation	93.08 ± 1.00	0.738 ± 0.030
Full verification + baseline	91.39 ± 0.71	0.545 ± 0.013
Fixed schedule + baseline	92.89 ± 0.80	0.776 ± 0.025
Paired allocation + baseline	91.95 ± 0.90	0.725 ± 0.019
Paired allocation, unscaled update	99.27 ± 0.15	0.683 ± 0.038

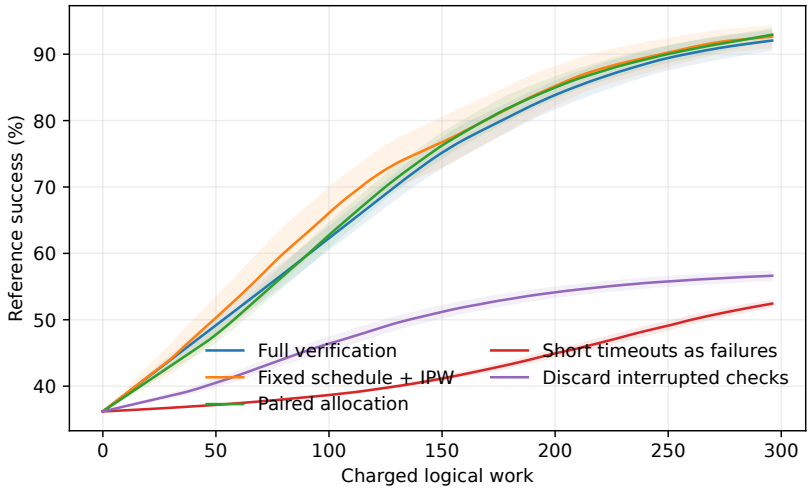

Figure 9: Live training under the main logical-work protocol. Curves show 20-seed means and pointwise 95% intervals for selected methods. Values between records logged every 100 pairs are linearly interpolated. The common displayed interval ends at 296 units, below every run's final charge. Final-budget comparisons use the un-interpolated endpoint records in Table 18.

The allocation step for live training is fixed at one and the allocation-only step at 0.25. The reported experiment does not use a held-out hyperparameter search or claim to exhaust tuning of any baseline. Seed-level results and policy trajectories are provided in the supplement. CPU-limited training can complete different numbers of updates across repeated runs, changing the final success probabilities even with the same random seed. Logical-work trajectories are deterministic for the specified seeds and numerical environment.

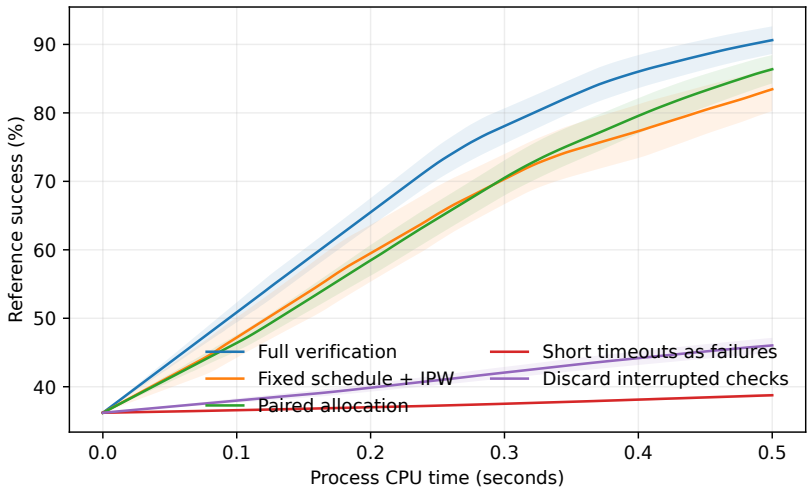

Figure 10: Live training under the 0.5-second process-CPU protocol, using the same plotting convention. All final records are at or beyond the displayed endpoint. Full verification is stronger than paired allocation in the final paired comparison.