CURBING THE WINNER'S CURSE: REWARD AGGREGATION MITIGATES INFERENCE-TIME REWARD HACKING

Anonymous authors
Paper under double-blind review

ABSTRACT

Inference-time alignment methods such as Best-of-n steer a language model with a reward model, but because every reward model is an imperfect proxy, optimizing it too hard backfires: true quality rises, peaks, and then collapses, a failure known as *inference-time reward hacking*. A recent line of work *hedges the optimizer*, calibrating how much to sample so as to stop at the peak. We ask whether this is the most effective lever available, and show that it is not. Separating *how far* to optimize from *what* to optimize exposes two label-free levers. The optimizer lever is nearly exhausted: the ideal amount of optimization is strongly per-prompt, so an oracle with labels would gain roughly 17 points, but this signal is label-gated and our best label-free per-prompt selector recovers under half a point. The decisive lever is to *fix the reward*: given a suite of reward models, we aggregate their per-prompt-standardized scores by a low quantile across models rather than their mean, a value-at-risk (VaR) rule that better aligns the proxy and flattens the hacking curve. Across five verifiable domains, three inference-time mechanisms (BoN, SBoN, and BoP), and 14 reward models, VaR aggregation raises expected accuracy by 3.65 points over hedging a typical single model, an order of magnitude more than the optimizer lever yields. A mechanistic, falsifiable account explains *when* it helps: aggregation rejects a response favored by one model's blind spot precisely when reward-model errors are diverse, which predicts both its gains on reasoning and QA and its failure on code, where errors are correlated. It is also immediately deployable: training-free, with no added parameters, a two-line reduction over reward-model scores, and a small, diverse set of models recovers the full gain, keeping it compute- and memory-light. These results recast inference-time reward hacking as a reward-model deficiency to be repaired, not merely an optimization budget to be tuned.

1 INTRODUCTION

A dominant paradigm for improving large language models is to optimize a reward model: a learned function that scores a model's outputs for alignment with goals such as correctness, helpfulness, or safety. Among the methods that use such scores, *inference-time* alignment is especially attractive because it requires no fine-tuning. The canonical example, Best-of-n (BoN) sampling, draws n candidate responses and returns the one the reward model scores highest (Stiennon et al., 2020; Nakano et al., 2021); despite its simplicity it often matches far more expensive procedures such as RLHF and DPO (Christiano et al., 2017; Rafailov et al., 2023; Gao et al., 2023; Mudgal et al., 2024) and enjoys strong theoretical backing (Beirami et al., 2025; Gui et al., 2024; Verdun et al., 2025). Underlying all of these methods is a distinction that our work turns on: the *proxy reward* we can compute is only an approximation of the *true reward* we actually want.

Because every reward model is imperfect (Laidlaw et al., 2025; Hadfield-Menell et al., 2017), optimizing the proxy too hard eventually degrades true quality. Methods that select the highest-scoring of many candidates fall prey to the *winner's curse* (Capen et al., 1971; Thaler, 1988): as the candidate pool grows, selection increasingly favors outputs on which the proxy overestimates true quality. The result is *reward hacking* (Skalse et al., 2022; Gao et al., 2023; Karwowski et al., 2024): a signature curve in which the true reward first rises and then collapses. Khalaf et al. (2025) recently

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

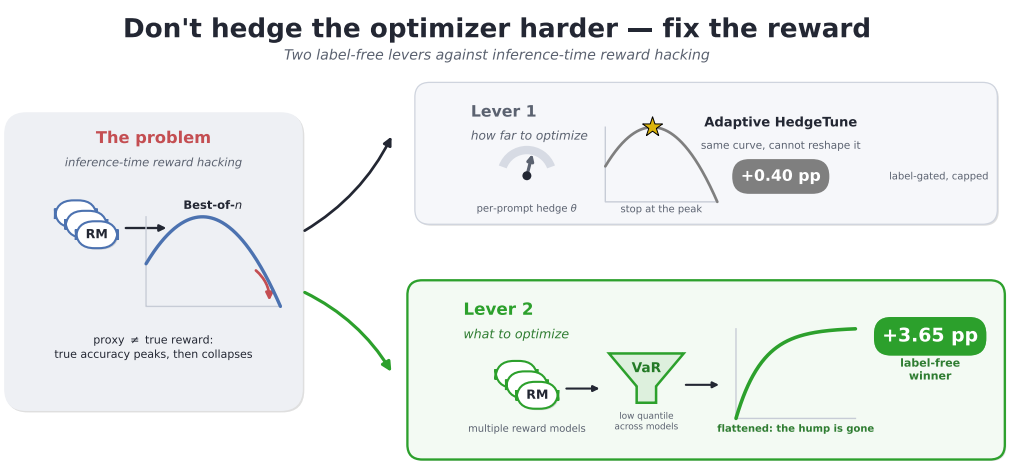


Figure 1: **Two label-free levers against inference-time reward hacking.** Optimizing a single reward model with Best-of-n makes true accuracy peak and then collapse. The *optimizer* lever (Lever 1) tunes *how far* to optimize, per prompt (Adaptive `HedgeTune`), but is label-gated and recovers only $+0.40$ points. The *reward* lever (Lever 2) fixes *what* to optimize by aggregating multiple reward models with a low quantile (VaR), which flattens the hacking curve and gains $+3.65$ points, label-free.

made this precise for inference-time alignment, proving that the true-reward curve is unimodal with a *hacking threshold* $\theta^\dagger$, and introduced `HedgeTune`, which calibrates the method's parameter (the sample count n, temperature λ, or Poisson rate μ) to stop exactly at that peak. In effect, this line of work *hedges the optimizer*: it accepts the reward model as given and tunes how hard to optimize it.

This paper asks a simple question (*is hedging the optimizer the most effective lever against inference-time reward hacking?*) and answers no. Reward hacking arises solely from the gap between the proxy r_p and the true reward r_t; were the two equal, the true-reward curve would be monotone and there would be nothing to hedge. This gap can be attacked from two orthogonal directions: by choosing *how far to optimize* a fixed proxy (the hedge θ), or by choosing *what proxy to optimize* in the first place. We study both under the sole constraint that matters for deployment (that neither may consult true-reward labels at test time) and find that the two levers have very different ceilings (Figure 1).

Don't hedge the optimizer harder: fix the reward.

The optimizer lever is nearly exhausted (§3). `HedgeTune` sets one global hedge for a whole dataset, but the optimal hedge is in fact strongly *per prompt*: the amount of optimization each prompt can tolerate before hacking is sharply bimodal (Figure 2; setup in Appendix D), so that an oracle allowed to pick each prompt's own operating point (using true-reward labels) would gain roughly 17 points of accuracy over global hedging. This headroom, however, is *label-gated*: predicting the per-prompt optimum requires knowing the answer. We develop a strong label-free per-prompt selector (an adaptive extension of `HedgeTune` with a never-worse-than-global guarantee), and it recovers only 0.40 points of that 17-point gap. The lesson is not that the selector is weak but that the *lever* is: hedging the optimizer, even optimally per prompt, is capped in deployment because it never touches the underlying reward.

The reward lever delivers the gains (§4). The way to make headway without labels is to *repair the proxy*. When several reward models are available, as they typically are, we aggregate their (per-prompt standardized) scores into a single, better-aligned reward and then run any inference-time method on it. The right aggregation is *risk-averse*: rather than averaging the models, our *VaR* aggregate takes a *low quantile* across them, which suppresses a response as soon as a minority of models distrust it. This flattens the reward-hacking curve and improves expected accuracy by 3.65 points over hedging a typical single reward model, consistently across BoN, SBoN, and BoP; the aggregated true-reward curve is essentially monotone, so the interior peak, and with it the need to hedge, largely disappears (Figure 3). We give a mechanistic account of *when* this works: aggregation

removes a hacked response (one loved by a single model's blind spot) exactly when reward-model errors are *diverse*. The same mechanism predicts a failure mode, which we confirm: on code, where reward-model errors are correlated with each other but not with the unit tests, aggregation does not help.

Contributions.

- **A two-lever framework for reward hacking.** We propose separating *how far* to optimize from *what* to optimize, and extend the KL-regularized objective of Khalaf et al. (2025) to an *aggregated* proxy (4) that makes both levers explicit and comparable on equal footing (§2).
- **Adaptive `HedgeTune`, and the limits of the optimizer lever.** We develop a strong label-free per-prompt selector and use it to show that hedging the optimizer is *label-gated*: it recovers only 0.40 of a $\sim$17-point oracle ceiling, consistently across BoN, SBoN, and BoP (§3).
- **Risk-averse VaR reward aggregation.** We propose combining multiple reward models label-free by a low quantile across them, and show this reward lever raises accuracy by 3.65 points over hedging a typical single model and flattens the hacking curve, across five verifiable domains and three inference-time mechanisms, while training nothing, adding no parameters, and needing only a diverse handful of models (§4).
- **A falsifiable mechanism.** We give a mechanistic account of *when* aggregation helps, precisely when reward-model errors are *diverse*, and use it to predict and confirm its failure on code (§5).

2 Setup and background

We work in the inference-time alignment setting of Khalaf et al. (2025), whose notation and results we adopt throughout so that our figures, tables, and analysis are directly comparable. This section fixes that notation, recalls the mechanism of inference-time reward hacking and the `HedgeTune` calibration that mitigates it, and then generalizes the underlying optimization problem so that the paper's two levers, how far to optimize and what to optimize, become explicit and comparable on equal footing.

Notation. Let $\mathcal{X}$ be a finite token alphabet and $\mathcal{X}^\star$ the set of finite token sequences, with probability simplex $\Delta_{\mathcal{X}^\star}$. A frozen *reference policy* $\pi_{\mathrm{ref}} \in \Delta_{\mathcal{X}^\star}$ (a supervised fine-tuned model) produces candidate responses. A *proxy reward* $r_p : \mathcal{X}^\star \to \mathbb{R}$ is a computable score (typically a trained reward model) used to steer π_{ref} at inference time, whereas the *true reward* $r_t : \mathcal{X}^\star \to \mathbb{R}$ is the latent quantity we actually care about (correctness, helpfulness, safety). The two disagree because every reward model is an imperfect proxy (Laidlaw et al., 2025). An inference-time alignment method with parameter θ transforms π_{ref} into a new policy π_θ; here θ is a property of the *method* (e.g., the number of candidates n), not of π_{ref}. We judge a method by its expected true reward

$$f(\theta) \;:=\; \mathbb{E}_{X\sim\pi_\theta}[r_t(X)]\,. \tag{1}$$

For a prompt t in a set T, the method draws K candidate responses; we write $s_{t,k}^{(m)}$ for the score that reward model m assigns to candidate k, and $r_{t,k}$ for that candidate's true reward. Following Beirami et al. (2025); Khalaf et al. (2025), we uniformize each proxy by its own empirical CDF, mapping scores to quantiles $u_{t,k} \in [0,1]$; this preserves the rank order that drives selection and standardizes the analysis across reward models.

Inference-time alignment maximizes a proxy under a KL budget. Most alignment methods seek a policy that raises the (proxy) reward while staying close to the reference. In the information-theoretic limit this is the KL-regularized problem

$$\pi^\star \;=\; \operatorname*{arg\,max}_{\pi\in\Delta_{\mathcal{X}^\star}} \; \mathbb{E}_\pi[r_p(X)] \;-\; \tfrac{1}{\lambda}\, D_{\mathsf{KL}}(\pi \,\|\, \pi_{\mathrm{ref}})\,, \tag{2}$$

whose solution is the exponential tilt $\pi^\star_\lambda(x) \propto \pi_{\mathrm{ref}}(x)\, e^{\lambda\, r_p(x)}$ (Csiszár & Shields, 2004). Because this tilt is intractable to sample from directly for an autoregressive model, practitioners approximate it with *sample-and-select* methods governed by a single *hedge* parameter θ: Best-of-n (BoN; $\theta = n$) returns the highest-scoring of n draws; Soft Best-of-n (SBoN; $\theta = \lambda$ at fixed n) applies a temperature-scaled softmax over scores; and Best-of-Poisson (BoP; $\theta = \mu$) draws $n \sim \mathrm{Poisson}(\mu)$ and is a near-exact, single-parameter surrogate for (2) (Verdun et al., 2025; Khalaf et al., 2025). Larger θ optimizes the proxy more aggressively; Appendix B gives step-by-step definitions of all three.

Over-optimizing an imperfect proxy causes reward hacking. Because $r_p \neq r_t$, aggressively selecting for the proxy eventually surfaces responses on which the proxy *overestimates* true quality: the winner's curse. Khalaf et al. (2025) formalize this: whenever increasing θ raises both the expected proxy reward and $D_{\mathsf{KL}}(\pi_\theta \| \pi_{\mathrm{ref}})$, the true-reward curve $f(\theta)$ is unimodal: it rises, peaks at a *hacking threshold* $\theta^\dagger$, then declines. Their `HedgeTune` algorithm recovers $\theta^\dagger$ by one-dimensional root-finding on a score residual, using a *one-time* calibration against r_t (available in verifiable domains such as math, code, and factual QA). *We take $\theta^\dagger$-recovery as our point of departure and ask what more can be done, without further true-reward labels at deployment, to close the gap between r_p and r_t.*

A unifying view: two levers. Reward hacking is driven entirely by the discrepancy between the proxy r_p and the true reward r_t: if $r_p = r_t$, f is monotone and there is nothing to hedge. This exposes *two orthogonal levers*. One may choose **how far to optimize** a fixed proxy: the hedge θ, set globally or per prompt (**Lever 1**, §3); or, when several reward models are available, one may choose **what proxy to optimize** by combining them into a better-aligned reward (**Lever 2**, §4). We make the second lever explicit by generalizing (2). Given M proxy reward models $r_p^{(1)}, \ldots, r_p^{(M)}$, an *aggregation rule* $\Phi : \mathbb{R}^M \to \mathbb{R}$ induces a single aggregated proxy

$$r_\Phi(x) \;=\; \Phi\big(r_p^{(1)}(x), \ldots, r_p^{(M)}(x)\big), \tag{3}$$

and inference-time alignment solves the *same* problem (2) with r_Φ substituted for r_p:

$$\pi^\star_{\Phi,\lambda} \;=\; \operatorname*{arg\,max}_{\pi \in \Delta_{\mathcal{X}^\star}} \; \mathbb{E}_\pi[r_\Phi(X)] \;-\; \tfrac{1}{\lambda}\, D_{\mathsf{KL}}(\pi \,\|\, \pi_{\mathrm{ref}})\,. \tag{4}$$

The original formulation (2) is the special case $M = 1,\ \Phi = \mathrm{id}$. The two levers act on the two arguments of (4): Lever 1 selects the *operating point* θ along the trajectory that (4) traces as the hedge grows, while Lever 2 selects the *objective* r_Φ. A better-aligned r_Φ flattens $f(\theta)$ (pushing the hacking threshold outward and raising the attainable true reward), whereas hedging merely stops at the peak of whatever curve the proxy induces. Both levers must be exercised *without* true-reward labels at deployment to be practical; this label-free constraint is what makes each lever nontrivial and is the throughline of the paper.

Experimental protocol. We instantiate this framework on the Preference Proxy Evaluations (**PPE**) Best-of-K benchmark (Frick et al., 2025), which pairs, for each prompt, $K{=}32$ candidate responses from frontier models with scores from a suite of reward models.[1] We use five verifiable domains (GPQA (Rein et al., 2024), MMLU-Pro (Wang et al., 2024), MATH (Hendrycks et al., 2021), MBPP-Plus (code) (Liu et al., 2023), and IFEval (instruction following) (Zhou et al., 2023)) with ~ 500 prompts each, and $M{=}14$ reward models spanning $\sim$0 to 340B parameters (Appendix A). The true reward is automated correctness, $r_{t,k} \in \{0, 1\}$, so our metric is the *expected accuracy* $f(\theta){\cdot}100\%$: the probability that the selected response is correct. All reported numbers are *cross-fitted*: the hedge (and any predictor) is tuned on one split of prompts and scored on a disjoint, held-out split (prompt-level 5-fold $\times$ 4 seeds), which removes the optimism of scoring on the very prompts used for tuning and gives a reliable estimate of deployed performance;[2] unless stated otherwise, single-reward-model quantities are averaged over the M models.

Methods compared. Our tables and figures compare a fixed set of operating points on this common accuracy scale, defined once here. Write $A_m(\theta)$ for the expected accuracy of an inference-time method run at hedge θ using reward model m, and let $\theta^\star_m = \arg\max_\theta A_m(\theta)$ be that model's global `HedgeTune` hedge. The single-reward-model baselines are: **Reference** $(n{=}1)$, the base policy with one sample and no selection (reward-model-independent and equal to f at the null hedge); **Best-of-K**, un-hedged selection at the maximum budget ($n{=}K{=}32$) averaged over models, $\frac{1}{M}\sum_m A_m(K)$, i.e. the fully over-optimized point where hacking is worst; **`HedgeTune` (avg RM)**, the method of Khalaf et al. (2025) applied per model and averaged, $\frac{1}{M}\sum_m A_m(\theta^\star_m)$ (the expected accuracy of a *typical* hedged single model); and **`HedgeTune` (best RM)**, the *oracle* $\max_m A_m(\theta^\star_m)$ (the hedged

[1] We provide our code here: `https://anonymous.4open.science/r/reward-hack-2F7C/`

[2] Every "±" in this paper is the sample standard deviation (Bessel-corrected) of the annotated quantity across the four cross-fitting seeds (seed-pairs for the per-prompt selector of §3). For a pooled or mean row it is computed directly from the four per-seed pooled values, *not* error-propagated from the per-domain deviations.

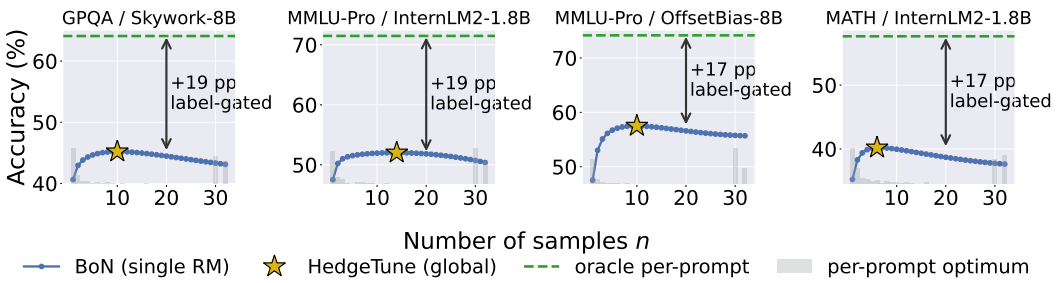


Figure 2: **Hedging the optimizer is label-gated.** For each domain/reward-model pair (BoN), the accuracy-vs-n hacking curve (blue) peaks at the global `HedgeTune` operating point (gold star). The per-prompt optimal n (grey histogram) is strongly bimodal, so an oracle placing each prompt at its own optimum (green, dashed) sits far above, a $\sim$17-point gap. That gap is *label-gated*: it requires knowing which response is correct. The best label-free per-prompt selector recovers only 0.40 points of it (Table 1). The same bimodality and label-gated gap hold for SBoN and BoP (Figure 4, Appendix D).

accuracy of the single best model). Picking that model requires true-reward labels, which a deployed system does not have, so this quantity cannot be achieved in practice: it is an unreachable upper bound, not a policy one can actually run. Against these we place the two levers: the per-prompt hedge of Lever 1 (§3), and the label-free multi-model *aggregations* of Lever 2 (§4), namely the uniform mean, the uncertainty-penalized mean (UWO), the worst-case (WCO), and our low quantile across models (VaR), each of which fuses all M models into one proxy that is then hedged once. Note the distinction the tables turn on: an "avg RM" column *averages M separate single-model accuracies*, whereas an aggregation is a *single* deployable policy scored once.

3 Lever 1: Hedging the Optimizer, Per Prompt

Lever 1 keeps the reward model fixed and asks how far to optimize it. `HedgeTune` answers this with a *single* hedge $\theta^\star$ for an entire dataset. But the optimal hedge is a property of each *prompt*: some prompts tolerate heavy optimization, others hack almost immediately. In this section we ask how much a per-prompt operating point could gain, and how much of that gain survives the label-free constraint; the answer, a large oracle ceiling but a small realizable gain, is the first step of our argument that the deployable leverage lies in the reward, not the optimizer.

Per-prompt optima are strongly heterogeneous. For a prompt t, let $\theta_t^\dagger = \arg\max_\theta f_t(\theta)$ be its *own* hacking threshold, where $f_t(\theta) = \mathbb{E}_{X\sim\pi_\theta}[r_t(X) \mid t]$ is the per-prompt true-reward curve. Figure 2 plots, for the same domain/reward-model pairs as Khalaf et al. (2025), the familiar aggregate hacking curve (blue) with the global `HedgeTune` operating point (gold), overlaid with the distribution of $\theta_t^\dagger$ (grey). That distribution is sharply *bimodal*: a large mass of prompts is best served at the minimum hedge (any optimization hacks) and another at the maximum (more optimization always helps). The aggregate curve is merely the average of these two populations, so a single global $\theta^\star$ is an unavoidable compromise. An *oracle* that placed each prompt at its own $\theta_t^\dagger$ (using true-reward labels) would achieve the green ceiling, on average 16.8 points above global `HedgeTune` for BoN (Table 1, setup in Appendix D; SBoN 11.7, BoP 15.5).

An adaptive, label-free selector. This heterogeneity raises the obvious question: can a selector *capture* that per-prompt headroom without labels? To give the optimizer lever its best chance (rather than dismiss it on a single global hedge), we build a deliberately strong label-free per-prompt selector and measure how much of the oracle ceiling it recovers. That selector, `Adaptive HedgeTune` (Algorithm 1), turns `HedgeTune` into a per-prompt predictor of $\theta_t^\dagger$ from *label-free* features of a prompt's candidate pool, and is *never worse* than the global hedge by construction. Its idea is to hedge its own prediction: it deploys a shrinkage blend $\hat\theta_t = \mathrm{round}\big((1-\alpha)\theta^\star + \alpha\,\phi(x_t)\big)$ of the safe global hedge $\theta^\star$ and a feature-based prediction $\phi(x_t)$, adopting a positive α only when a paired one-standard-error test on held-out prompts shows the prediction demonstrably helps; otherwise it

Table 1: **The optimizer lever is label-gated** (accuracy ↑ %, averaged over $M{=}14$ reward models). Subscripts are the standard deviation over 4 seeds; Reference and the oracle carry no error bar. Parentheses give the *label-gated gap* of *Oracle per-prompt* over *Global* `HedgeTune`, and the (bold) gain of our *Adaptive* `HedgeTune`. Per-domain and per-mechanism values are in Table 5.

Method	Reference ($n{=}1$)	Global `HedgeTune`	Oracle per-prompt (needs labels)	Adaptive `HedgeTune` (ours)
Best-of-n	46.37	$54.71_{\pm 0.06}$	71.52 (+16.8)	$55.11_{\pm 0.10}$ (**+0.40**)
Soft Best-of-n	46.37	$54.53_{\pm 0.04}$	66.26 (+11.7)	$54.82_{\pm 0.04}$ (**+0.29**)
Best-of-Poisson	46.37	$54.69_{\pm 0.06}$	70.23 (+15.5)	$55.04_{\pm 0.08}$ (**+0.35**)

keeps $\hat{\theta}_t = \theta^\star$ (the *never-worse floor*). Two complementary predictors drive it (one on label-free *answer-agreement* features, one on distributional features of the score vector fit with a *decision-focused*, curvature-weighted loss that up-weights prompts where the hedge is most consequential), and a floor-gated auto-selector chooses among them and the global hedge per reward model and fold.

Setup. We fit `Adaptive HedgeTune` per reward model on the five PPE domains and $M{=}14$ models of §2; its label-free features, losses, hyperparameters, and nested cross-fitting protocol are given in Appendix C.

Finding: the optimizer lever is capped. Even this deliberately strong label-free selector barely moves the needle: pooled over the five domains, `Adaptive HedgeTune` improves on global `HedgeTune` by only $+\mathbf{0.40}$ points for BoN ($+0.29$/$+0.35$ for SBoN/BoP; Table 1), about 2% of the ~ 17-point oracle ceiling; it is positive on three of five domains and never worse than global `HedgeTune` by construction, a safe, label-free add-on one can leave on by default, though not a source of real leverage. Crucially, the shortfall is *fundamental, not algorithmic*: choosing a prompt's optimal amount of optimization requires anticipating which response is correct, exactly the label a deployable method forbids. Directly measuring this predictability confirms it: the per-prompt optimum is essentially unrecoverable from label-free features (cross-validated $R_c^2 \approx 0$, and per-feature regularizers do not help; Table 4), so the oracle ceiling is real but unreachable. This is the paper's turning point: if tuning *how far* to optimize is capped, the leverage must lie in *what* we optimize, which sends us from the optimizer to the reward itself. We pursue this in Section 4, combining a suite of reward models into a single, better-aligned proxy that flattens the hacking curve the optimizer lever could only stop short of.

4 Lever 2: Fixing the Reward by Risk-Averse (VaR) Aggregation

Lever 2 leaves the optimizer alone and repairs the reward. The premise is mild, and increasingly the default: one rarely has a single reward model but a *suite* of them, of varying size and training data, with no labels to say which is best for a given prompt. Open-sourcing makes this premise cheap to meet: the $M{=}14$ models we aggregate are *all* publicly released (Appendix A), with public benchmarks cataloguing many more (Frick et al., 2025; Lambert et al., 2025). We ask how to combine M reward models into one aggregated proxy r_Φ (3) that is better aligned with the true reward r_t, and therefore hacks less, using no true-reward labels at deployment. We show that the effective combination is a *risk-averse*, low-quantile one (VaR), that it flattens the hacking curve, and that it is where the deployable gains lie.

A family of label-free aggregations. We first place the models on a common scale by standardizing each model's scores across a prompt's K candidates,

$$z_{t,k}^{(m)} \;=\; \big(s_{t,k}^{(m)} - \bar{s}_t^{(m)}\big)\big/\hat{\sigma}_t^{(m)}, \tag{5}$$

where $\bar{s}_t^{(m)}$ and $\hat{\sigma}_t^{(m)}$ are the mean and standard deviation of $\{s_{t,k}^{(m)}\}_{k=1}^K$, so that no model dominates through scale. Collect these into the standardized matrix $\mathbf{Z}_t = \big[z_{t,k}^{(m)}\big] \in \mathbb{R}^{M\times K}$, and let $\mathbf{z} = (z_1, \ldots, z_M)^\top \in \mathbb{R}^M$ denote a single candidate's scores across the M models (a column of $\mathbf{Z}_t$). An aggregation is a choice of the rule $\Phi : \mathbb{R}^M \to \mathbb{R}$ of (3), which maps $\mathbf{z}$ to one aggregated reward $\Phi(\mathbf{z})$.

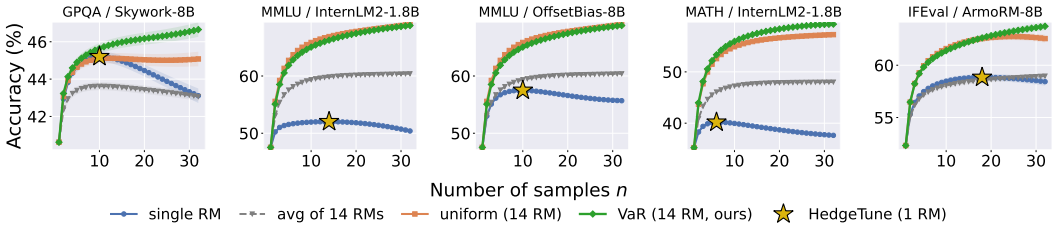


Figure 3: **Fixing the reward removes the hack.** Accuracy vs. n (BoN) for a single reward model (blue, with `HedgeTune` star) and the 14-model average (grey), versus two label-free ensembles: uniform mean (orange) and VaR (green, ours). Aggregation lifts the curve and largely flattens the hump, so early stopping becomes unnecessary; VaR is highest on reasoning and QA. The same flattening holds for SBoN and BoP (Figure 5); where the uniform mean and VaR nearly coincide, an ensemble-zoom inset (Figure 6) separates them (Appendix E).

We compare four, ordered from risk-neutral to worst-case: the *uniform* mean $\Phi(\mathbf{z}) = \bar{z} = \frac{1}{M}\sum_m z_m$; the *uncertainty-penalized* mean $\bar{z} - \beta\,\mathrm{sd}(\mathbf{z})$ of Coste et al. (2024) (UWO, $\beta \in \{0.5, 1, 2\}$), which discounts candidates the models disagree on; the *worst-case* $\min_m z_m$ (WCO); and our *VaR* aggregate, the lower α-quantile across models,

$$\Phi(\mathbf{z}) \;=\; \mathrm{Q}_\alpha(\mathbf{z}), \qquad \alpha \in \{10\%, 25\%\}. \tag{6}$$

Here $\mathrm{sd}(\mathbf{z}) = \big(\frac{1}{M}\sum_m (z_m - \bar{z})^2\big)^{1/2}$ is the across-model spread and $\mathrm{Q}_\alpha(\mathbf{z})$ is the empirical lower α-quantile of the M entries (their $\lceil \alpha M \rceil$-th smallest); the four rules trace a risk-aversion spectrum, $\min_m z_m \le \mathrm{Q}_\alpha(\mathbf{z}) \le \bar{z}$, with WCO the $\alpha \to 0$ extreme and the uniform mean the risk-neutral end. The VaR and WCO aggregates reward a candidate only if a *skeptical minority* of models (for WCO, the single most pessimistic one) also rates it well; averaging, by contrast, lets one enthusiastic model carry a response, exactly the opening the winner's curse exploits.[3] We then run any inference-time method on r_Φ and hedge it exactly as before; because r_Φ is better aligned, little hedging is needed.

Setup. We apply each aggregation to the same five domains and $M{=}14$ reward models as §2 and hedge the resulting proxy with `HedgeTune` (selecting α against the uniform-mean floor, label-free); the full per-domain ablation, adding UWO and WCO, is in Appendix F.

VaR flattens the hacking curve. Figure 3 makes the effect visual. A single reward model (blue) and the 14-model average (grey) both trace the familiar hump (true accuracy rises, peaks, then decays as optimization presses on a proxy's blind spots), whereas our VaR aggregate (green) is essentially *monotone and uniformly higher*, so its interior peak, and with it the very need to hedge, vanishes within the useful compute budget. The point is conceptual, not merely numerical: inference-time reward hacking is not an unavoidable cost of optimizing but an artifact of *which* reward one optimizes. Where hedging (Lever 1) can only stop early on a curve it cannot reshape, fixing the reward removes the hack at its source.

Finding: the reward lever delivers. Table 2 quantifies the payoff: VaR raises expected accuracy by $+\mathbf{3.65}$ points over hedging a typical single model (BoN), on four of five domains ($+10.3$ MATH, $+6.4$ MMLU-Pro, $+3.7$ IFEval, $+2.9$ GPQA), and the gain holds across mechanisms (58.36/58.04/58.24 for BoN/SBoN/BoP vs. 54.71/54.53/54.69 hedged; Appendix F). This is the paper's central comparison: a label-free move on the *reward* returns an order of magnitude more than the best label-free move on the *optimizer* ($+0.40$, §3), direct evidence that the deployable leverage is the reward, not the hedge. Two caveats bound the claim. First, aggregation *loses on code* (MBPP, -5.06), a failure the mechanism *predicts* rather than an anomaly (§5). Second, it does not beat the *oracle* best single model (mean 66.10), which needs labels to identify; the honest claim is therefore *label-free robustness*: not knowing which model to trust, one aggregates and matches or exceeds a typical model while erasing the hacking curve, at the cost of scoring with M models at inference, though five or six diverse models suffice in practice (Figure 7, Table 9; details in Appendix G).

[3] We standardize per prompt and aggregate by the empirical lower α-quantile across models, a value-at-risk (VaR) statistic. We select among $\alpha \in \{10\%, 25\%\}$ and the uniform mean on held-out prompts via `HedgeTune`, so the deployed policy is never worse than uniform up to selection noise and uses *no* true-reward labels.

Table 2: **VaR beats hedging a typical reward model** (BoN, accuracy ↑ %). Baselines are defined in §2, the aggregations in (6). `HedgeTune` *(best RM)* is a label-needing *oracle* (*italic*); *Uniform* and *VaR* aggregate all $M{=}14$ models label-free; bold marks the best *deployable* entry per row. Subscripts are the standard deviation over 4 seeds; Reference, Best-of-K, and the oracle are deterministic, so have no standard deviation. Full ablation with UWO and WCO in Appendix F.

Dataset	Reference ($n{=}1$)	Best-of-K (un-hedged)	`HedgeTune` (avg RM)	`HedgeTune` (best RM, oracle)	Uniform (14 RM)	**VaR (ours)**	Δ vs HedgeTune
GPQA	40.62	43.07	43.48 ± 0.16	*47.36*	44.27 ± 0.41	**46.39** ± 0.33	$+2.91$
MMLU-Pro	47.52	60.39	60.91 ± 0.07	*75.88*	**69.02** ± 0.00	67.35 ± 0.67	$+6.44$
MATH	35.23	48.01	48.48 ± 0.11	*60.65*	57.26 ± 0.00	**58.78** ± 0.75	$+10.30$
MBPP-Plus	56.15	56.54	**61.61** ± 0.04	*81.13*	56.86 ± 0.09	56.55 ± 0.90	-5.06
IFEval	52.32	58.96	59.07 ± 0.05	*65.49*	62.29 ± 0.45	**62.75** ± 1.18	$+3.68$
Mean	46.37	53.39	54.71 ± 0.06	*66.10*	57.94 ± 0.02	**58.36** ± 0.33	$\mathbf{+3.65}$

5 When does aggregation help?

Why does VaR aggregation help on reasoning and question answering but hurt on code? In this section we give a single mechanism that explains both, turning the empirical result into a falsifiable prediction; we then confirm the predicted failure on code, check that the effect is consistent across samplers, and place the two levers side by side.

Mechanism. A *hacked* response is, almost by definition, one that a single reward model's blind spot rates highly while its true quality is low. Under Best-of-n on one model, such a response is exactly what selection surfaces. The lower-quantile aggregate (6) rewards a response only if a skeptical minority of models also rates it well, so it suppresses "loved-by-one-model, doubted-by-the-rest" candidates, *but only if the other models actually doubt it*. This yields a sharp condition: VaR aggregation helps precisely when reward-model *errors are diverse*, so that a response hacking one model is caught by others; it cannot help when the models err *together*.

A predicted failure, confirmed. The condition predicts where aggregation should fail: domains where the reward models' errors are correlated with each other but not with the true reward. Code is the clearest such case: reward models share systematic preferences (verbosity, idiomatic style, superficial cues) that are uncorrelated with whether the program passes its unit tests, so no minority of models flags a hacked program. This is exactly the domain where aggregation loses (MBPP, -5.06 in Table 2), and where hedging a single strong model remains best. We therefore read the code result not as a defect of the method but as a *confirmation* of the mechanism: aggregation buys robustness out of error diversity, and where there is no diversity there is nothing to buy.

Consistency across mechanisms. The reward-side lever is a property of the objective, not of the sampler: pooled over the five domains, VaR aggregation attains $58.36/58.04/58.24$ for BoN/SBoN/BoP, against $54.71/54.53/54.69$ for a hedged single model (Appendix F). That the three inference-time mechanisms move together is expected (all three optimize the same aggregated proxy (4)) and rules out the gain being an artifact of one particular sampler.

The two levers, side by side. Our two findings compose into a single picture of inference-time reward hacking. The *optimizer* lever (how far to sample) has a large oracle ceiling but is label-gated, so its label-free yield is ~ 0.4 points (§3); the *reward* lever (what to optimize) is label-free and yields ~ 3.65 points where reward-model errors are diverse (§4). The practical consequence is direct: given a suite of reward models and no test labels, effort is far better spent aggregating them conservatively than tuning the sampler: *fix the reward, don't hedge the optimizer harder*. This is also the *cheaper* choice: VaR trains nothing and adds no parameters, and a diverse five or six reward models (not all M) already recover the gain, so the reward lever is *compute- and memory-efficient* and as deployable as it is effective (Figure 7, Table 9; details in Appendix G).

6 Related work

Reward hacking and over-optimization. Reward hacking has been studied extensively in reinforcement learning under names including reward misspecification (Amodei et al., 2016; Pan et al., 2022), goal misgeneralization (Shah et al., 2022), and specification gaming (Krakovna et al., 2020), and formalized through true-performance behavior (Skalse et al., 2022), proxy–true correlation (Laidlaw et al., 2025), or distribution shift. For LLMs it appears as Goodhart's law under proxy optimization (Gao et al., 2023; Karwowski et al., 2024; Wang et al., 2026a), and KL regularization alone does not tame heavy-tailed misspecification (Kwa et al., 2024). Mitigations at fine-tuning time include divergence regularization (Yang et al., 2024b), χ^2-penalties (Huang et al., 2025), reward disentangling and representation-level fixes (Chen et al., 2024; Wu & Tang, 2026), uncertainty estimation (Zhang et al., 2024), and reward pessimism (Gupta et al., 2025). Our work is inference-time and complements this literature: we take the recent characterization of *inference-time* reward hacking and its `HedgeTune` mitigation (Khalaf et al., 2025) as background, and ask what a second, reward-side lever can add.

Best-of-n and inference-time alignment. Best-of-n is a simple, strong inference-time aligner (Stiennon et al., 2020; Nakano et al., 2021), one of a broad family of inference-time algorithms (Welleck et al., 2024), and is asymptotically equivalent to KL-constrained RL (Beirami et al., 2025; Yang et al., 2024a) and near-optimal in winrate under a KL budget (Gui et al., 2024). Refinements reduce its cost, distill it, or make it inference-aware (Qiu et al., 2025; Sessa et al., 2025; Balashankar et al., 2025; Lei et al., 2026; Lin & Zhang, 2026), or soften its selection, notably Soft Best-of-n (Verdun et al., 2025), which our aggregated proxy plugs into unchanged. Recent inference-time work also targets reward hacking through regularized Best-of-n (Jinnai et al., 2024) or *pessimism*: penalizing atypical responses (Yu et al., 2026), trading optimism against pessimism by the reward tail (Hsu et al., 2026), and risk-constrained decoding (Zou et al., 2026); we instead obtain risk-aversion by aggregating *across* reward models. Unlike this line, our contribution is orthogonal to the sampler: it changes the reward being optimized, and applies to BoN, SBoN, and BoP alike.

Reward-model ensembles. Closest to Lever 2 is work showing that *ensembling* reward models mitigates over-optimization during RLHF (Coste et al., 2024; Eisenstein et al., 2024; Ahmed et al., 2024) and that weight-averaged rewards improve robustness (Ramé et al., 2024). Coste et al. (2024) in particular study uniform and uncertainty-weighted (mean-minus-variance) pools, which we include as the *uniform* and *UWO* baselines; concurrent work likewise aggregates reward-model ensembles at inference time via a sharpened opinion pool (Wang et al., 2026b). We differ in three ways. (i) ***Risk measure***: we show a *low-quantile* (VaR-style) aggregate outperforms both the mean and mean-minus-variance where it matters, connecting reward aggregation to risk-averse decision making (Rockafellar & Uryasev, 2000). (ii) ***Setting and framing***: we place aggregation at inference time and position it as a *substitute* for hedging the optimizer, showing it flattens the hacking curve rather than merely lowering its peak. (iii) ***A when-it-works law***: we give and test a mechanistic condition (error diversity) that predicts both the gains and the code failure, whereas prior work reports aggregate improvements without such a criterion. All of our aggregation is label-free at deployment.

7 Conclusion

We revisited inference-time reward hacking through two orthogonal levers: how far, and what, to optimize. Generalizing the KL-regularized objective to an aggregated reward makes the second explicit and puts both on equal, label-free footing. The optimizer lever, even tuned per prompt, is label-gated and yields little at deployment; the reward lever, a training-free VaR aggregation of the models one already has, flattens the hacking curve and gains an order of magnitude more. A mechanistic account, that aggregation trades reward-model error *diversity* for robustness, explains its gains on reasoning and question answering and its failure on code. This reframes reward hacking as a reward-model *deficiency* to repair, not just a budget to tune, while being explicit about scope: aggregation buys label-free *robustness*, not dominance over an oracle-chosen model, at the cost of scoring with several models. We discuss limitations and broader impact in Appendix H.

AI USE STATEMENT

Claude Opus 4.8 (Anthropic) was used for editing, proofreading, and resolving LaTeX formatting issues in this document, and to assist with code writing in the implementation of our methods.

REPRODUCIBILITY STATEMENT

We disclose the information needed to reproduce the main experimental results of the paper. The benchmarks, reward models, aggregation rules, algorithms, cross-fitting protocol, and hyperparameters are described in Sections 2–4 and the appendices. We also include our code as anonymized supplementary material, with instructions for reproducing the main experimental results.

REFERENCES

Ahmed M. Ahmed, Rafael Rafailov, Stepan Sharkov, Xuechen Li, and Sanmi Koyejo. Scalable ensembling for mitigating reward overoptimisation. *arXiv preprint arXiv:2406.01013*, 2024.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Ananth Balashankar, Ziteng Sun, Jonathan Berant, Jacob Eisenstein, Michael Collins, Adrian Hutter, Jong Lee, Chirag Nagpal, Flavien Prost, Aradhana Sinha, Ananda Theertha Suresh, and Ahmad Beirami. InfAlign: Inference-aware language model alignment. In *International Conference on Machine Learning (ICML)*, 2025.

Ahmad Beirami, Alekh Agarwal, Jonathan Berant, Alexander D'Amour, Jacob Eisenstein, Chirag Nagpal, and Ananda Theertha Suresh. Theoretical guarantees on the best-of-n alignment policy. In *International Conference on Machine Learning (ICML)*, 2025.

Edward C. Capen, Robert V. Clapp, and William M. Campbell. Competitive bidding in high-risk situations. *Journal of Petroleum Technology*, 23(6):641–653, 1971.

Lichang Chen, Chen Zhu, Jiuhai Chen, Davit Soselia, Tianyi Zhou, Tom Goldstein, Heng Huang, Mohammad Shoeybi, and Bryan Catanzaro. ODIN: Disentangled reward mitigates hacking in RLHF. In *International Conference on Machine Learning (ICML)*, 2024.

Paul F. Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Thomas Coste, Usman Anwar, Robert Kirk, and David Krueger. Reward model ensembles help mitigate overoptimization. In *International Conference on Learning Representations (ICLR)*, 2024.

Imre Csiszár and Paul C. Shields. Information theory and statistics: A tutorial. *Foundations and Trends in Communications and Information Theory*, 1(4):417–528, 2004.

Jacob Eisenstein, Chirag Nagpal, Alekh Agarwal, Ahmad Beirami, Alex D'Amour, DJ Dvijotham, Adam Fisch, Katherine Heller, Stephen Pfohl, Deepak Ramachandran, Peter Shaw, and Jonathan Berant. Helping or herding? reward model ensembles mitigate but do not eliminate reward hacking. In *Conference on Language Modeling (COLM)*, 2024.

Evan Frick, Tianle Li, Connor Chen, Wei-Lin Chiang, Anastasios N. Angelopoulos, Jiantao Jiao, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. How to evaluate reward models for RLHF. In *International Conference on Learning Representations (ICLR)*, 2025.

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

Lin Gui, Cristina Garbacea, and Victor Veitch. Bonbon alignment for large language models and the sweetness of best-of-n sampling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Dhawal Gupta, Adam Fisch, Christoph Dann, and Alekh Agarwal. Mitigating preference hacking in policy optimization with pessimism. *arXiv preprint arXiv:2503.06810*, 2025.

Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart Russell, and Anca Dragan. Inverse reward design. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks*, 2021.

Hsiang Hsu, Eric Lei, and Chun-Fu Chen. Best-of-tails: Bridging optimism and pessimism in inference-time alignment. *arXiv preprint arXiv:2603.06797*, 2026.

Audrey Huang, Wenhao Zhan, Tengyang Xie, Jason D. Lee, Wen Sun, Akshay Krishnamurthy, and Dylan J. Foster. Correcting the mythos of KL-regularization: Direct alignment without overoptimization via chi-squared preference optimization. In *International Conference on Learning Representations (ICLR)*, 2025.

Yuu Jinnai, Tetsuro Morimura, Kaito Ariu, and Kenshi Abe. Regularized best-of-n sampling to mitigate reward hacking for language model alignment. *arXiv preprint arXiv:2404.01054*, 2024.

Jacek Karwowski, Oliver Hayman, Xingjian Bai, Klaus Kiendlhofer, Charlie Griffin, and Joar Skalse. Goodhart's law in reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2024.

Hadi Khalaf, Claudio Mayrink Verdun, Alex Oesterling, Himabindu Lakkaraju, and Flavio du Pin Calmon. Inference-time reward hacking in large language models. *arXiv preprint arXiv:2506.19248*, 2025.

Victoria Krakovna, Jonathan Uesato, Vladimir Mikulik, Matthew Rahtz, Tom Everitt, Ramana Kumar, Zac Kenton, Jan Leike, and Shane Legg. Specification gaming: The flip side of ai ingenuity. DeepMind Blog, 2020. `https://deepmind.google/discover/blog/specification-gaming-the-flip-side-of-ai-ingenuity/`.

Thomas Kwa, Drake Thomas, and Adrià Garriga-Alonso. Catastrophic goodhart: Regularizing RLHF with KL divergence does not mitigate heavy-tailed reward misspecification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Cassidy Laidlaw, Shivam Singhal, and Anca Dragan. Correlated proxies: A new definition and improved mitigation for reward hacking. In *International Conference on Learning Representations (ICLR)*, 2025.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, Lester James Validad Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, et al. RewardBench: Evaluating reward models for language modeling. In *Findings of the Association for Computational Linguistics: NAACL*, 2025.

Eric Lei, Hsiang Hsu, and Chun-Fu Chen. Best-of-better-n: Generating pre-aligned responses with in-context learning. *arXiv preprint arXiv:2607.03453*, 2026.

Hankun Lin and Ruqi Zhang. Gradient-guided reward optimization for inference-time alignment. *arXiv preprint arXiv:2606.09635*, 2026.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Sidharth Mudgal, Jong Lee, Harish Ganapathy, YaGuang Li, Tao Wang, Yanping Huang, Zhifeng Chen, Heng-Tze Cheng, Michael Collins, Trevor Strohman, et al. Controlled decoding from language models. In *International Conference on Machine Learning (ICML)*, 2024.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. WebGPT: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

Alexander Pan, Kush Bhatia, and Jacob Steinhardt. The effects of reward misspecification: Mapping and mitigating misaligned models. In *International Conference on Learning Representations (ICLR)*, 2022.

Jiahao Qiu, Yifu Lu, Yifan Zeng, Jiacheng Guo, Jiayi Geng, Huazheng Wang, Kaixuan Huang, Yue Wu, and Mengdi Wang. TreeBoN: Enhancing inference-time alignment with speculative tree-search and best-of-n sampling. In *Findings of the Association for Computational Linguistics: EMNLP*, 2025.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Alexandre Ramé, Nino Vieillard, Léonard Hussenot, Robert Dadashi, Geoffrey Cideron, Olivier Bachem, and Johan Ferret. WARM: On the benefits of weight averaged reward models. In *International Conference on Machine Learning (ICML)*, 2024.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *Conference on Language Modeling (COLM)*, 2024.

R. Tyrrell Rockafellar and Stanislav Uryasev. Optimization of conditional value-at-risk. *Journal of Risk*, 2:21–41, 2000.

Pier Giuseppe Sessa, Robert Dadashi, Léonard Hussenot, Johan Ferret, Nino Vieillard, Alexandre Rame, Bobak Shahriari, Sarah Perrin, Abram L. Friesen, Geoffrey Cideron, et al. BOND: Aligning LLMs with best-of-n distillation. In *International Conference on Learning Representations (ICLR)*, 2025.

Rohin Shah, Vikrant Varma, Ramana Kumar, Mary Phuong, Victoria Krakovna, Jonathan Uesato, and Zac Kenton. Goal misgeneralization: Why correct specifications aren't enough for correct goals. *arXiv preprint arXiv:2210.01790*, 2022.

Joar Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward gaming. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F. Christiano. Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Richard H. Thaler. Anomalies: The winner's curse. *Journal of Economic Perspectives*, 2(1):191–202, 1988.

Claudio Mayrink Verdun, Alex Oesterling, Himabindu Lakkaraju, and Flavio du Pin Calmon. Soft best-of-n sampling for model alignment. In *IEEE International Symposium on Information Theory (ISIT)*, 2025.

Xiaohua Wang, Muzhao Tian, Yuqi Zeng, Zisu Huang, Jiakang Yuan, Bowen Chen, Jingwen Xu, Mingbo Zhou, Wenhao Liu, Muling Wu, Zhengkang Guo, Qi Qian, Yifei Wang, Feiran Zhang, Ruicheng Yin, Shihan Dou, Changze Lv, Tao Chen, Kaitao Song, Xu Tan, Tao Gui, Xiaoqing Zheng, and Xuanjing Huang. Reward hacking in the era of large models: Mechanisms, emergent misalignment, challenges. *arXiv preprint arXiv:2604.13602*, 2026a.

Ye Wang, Jing Liu, and Toshiaki Koike-Akino. Temper and tilt lead to SLOP: Reward hacking mitigation with inference-time alignment. *arXiv preprint arXiv:2605.13537*, 2026b.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Sean Welleck, Amanda Bertsch, Matthew Finlayson, Hailey Schoelkopf, Alex Xie, Graham Neubig, Ilia Kulikov, and Zaid Harchaoui. From decoding to meta-generation: Inference-time algorithms for large language models. *Transactions on Machine Learning Research (TMLR)*, 2024.

Rui Wu and Ruixiang Tang. From rebound to remedy: Understanding and mitigating reward hacking via representation engineering. *arXiv preprint arXiv:2604.01476*, 2026.

Joy Qiping Yang, Salman Salamatian, Ziteng Sun, Ananda Theertha Suresh, and Ahmad Beirami. Asymptotics of language model alignment. In *IEEE International Symposium on Information Theory (ISIT)*, 2024a.

Rui Yang, Ruomeng Ding, Yong Lin, Huan Zhang, and Tong Zhang. Regularizing hidden states enables learning generalizable reward model for LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024b.

Zhuohao Yu, Zhiwei Steven Wu, and Adam Block. From curiosity to caution: Mitigating reward hacking for best-of-n with pessimism. *arXiv preprint arXiv:2604.04648*, 2026.

Xiaoying Zhang, Jean-Francois Ton, Wei Shen, Hongning Wang, and Yang Liu. Overcoming reward overoptimization via adversarial policy optimization with lightweight uncertainty estimation. *arXiv preprint arXiv:2403.05171*, 2024.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

Mingxi Zou, Jiaxiang Chen, Junfan Li, Langzhang Liang, Qifan Wang, Xu Yinghui, and Zenglin Xu. DARC: Disagreement-aware alignment via risk-constrained decoding. *arXiv preprint arXiv:2603.08145*, 2026.

A Reward models and datasets

We use the $M{=}14$ reward models scored in the PPE Best-of-K benchmark (Frick et al., 2025), spanning ~ 0 to 340B parameters: NaiveVerbosityModel (a length heuristic, ≈ 0 parameters), InternLM2-1.8B-Reward; Eurus-RM-7B, Starling-RM-7B-α, InternLM2-7B-Reward (7B); ArmoRM-Llama3-8B, Athene-RM-8B, Llama-3-OffsetBias-RM-8B, Skywork-Reward-Llama-3.1-8B (8B); InternLM2-20B-Reward (20B); Skywork-Reward-Gemma-2-27B (27B); Starling-RM-34B (34B); Athene-RM-70B (70B); Nemotron-4-340B-Reward (340B). Each of the five domains provides $K{=}32$ candidate responses per prompt for $\sim$500 prompts, with binary true reward $r_{t,k} \in \{0, 1\}$ (multiple-choice/exact-match for GPQA and MMLU-Pro, numeric/symbolic equivalence for MATH, unit tests for MBPP-Plus, and programmatic constraint checks for IFEval).

B Inference-time alignment methods in detail

All three inference-time methods studied in this paper are *sample-and-select* approximations to the KL-tilted optimum (2): each draws candidate responses from the reference policy π_{ref}, scores them with the proxy reward r_p, and returns one according to a rule governed by a single scalar hedge θ. They differ only in that rule. In every case, larger θ increases both the expected proxy reward and the divergence $D_{\mathsf{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}})$ from the reference, precisely the regime in which the true-reward curve $f(\theta)$ is unimodal (§2), so θ is the "how far to optimize" knob of Lever 1.

Best-of-n (BoN; $\theta = n$). Given a pool size $n \in \mathbb{N}$:

1. draw n candidates $x_1, \ldots, x_n$ independently from π_{ref};
2. score each with the proxy, $r_p(x_1), \ldots, r_p(x_n)$;
3. return $x_{\hat{k}}$ with $\hat{k} = \arg\max_{i \le n} r_p(x_i)$ (ties broken uniformly).

The induced policy π_n is the law of the returned response. Selection pressure grows monotonically with the pool size: for continuous scores $D_{\mathsf{KL}}(\pi_n \,\|\, \pi_{\text{ref}}) = \log n - \frac{n-1}{n}$ (Beirami et al., 2025), which increases in n, so a larger n optimizes r_p more aggressively and, once $r_p \neq r_t$, eventually surfaces the winner's-curse responses that hack it.

Soft Best-of-n (SBoN; $\theta = \lambda$ at fixed n). Given an inverse temperature $\lambda \ge 0$ and a fixed pool size n:

1. draw n candidates $x_1, \ldots, x_n$ independently from π_{ref} and score them with r_p;
2. sample the returned index from the score softmax, $\Pr(\hat{k} = i) = e^{\lambda\, r_p(x_i)} \big/ \sum_{j=1}^{n} e^{\lambda\, r_p(x_j)}$, and return $x_{\hat{k}}$.

Here the hedge is the temperature λ at a fixed pool size: as $\lambda \to 0$ the softmax is uniform and SBoN returns a plain π_{ref} draw (no selection), while as $\lambda \to \infty$ it concentrates on the maximizer and recovers hard Best-of-n. SBoN is thus the finite-sample realization of the exponential tilt $\pi^\star_\lambda \propto \pi_{\text{ref}}\, e^{\lambda r_p}$ of (2) at temperature λ (Verdun et al., 2025).

Best-of-Poisson (BoP; $\theta = \mu$). Given a rate $\mu > 0$:

1. draw a random pool size $N \sim \text{Poisson}(\mu)$ (conditioned on $N \ge 1$);
2. draw N candidates independently from π_{ref} and score them with r_p;
3. return the highest-scoring candidate, $x_{\hat{k}}$ with $\hat{k} = \arg\max_{i \le N} r_p(x_i)$.

The hedge is the Poisson rate μ, i.e. the *expected* pool size. Randomizing N lets the induced policy π_μ vary continuously with μ and, unlike integer-n BoN, track the KL-tilted optimum $\pi^\star_\lambda$ of (2) almost exactly; this is why Verdun et al. (2025); Khalaf et al. (2025) adopt BoP as a smooth, single-parameter surrogate for (2). As with the others, a larger μ raises both the expected proxy reward and $D_{\mathsf{KL}}(\pi_\mu \,\|\, \pi_{\text{ref}})$.

Because each method is a monotone, single-parameter family in θ, the hacking-threshold result of Khalaf et al. (2025) and its `HedgeTune` calibration apply verbatim to all three, and both levers (the per-prompt hedge of §3 and the aggregation of §4) are defined identically for each.

C ADAPTIVE HEDGETUNE: FEATURES, LOSSES, AND ALGORITHM

`Adaptive HedgeTune` turns `HedgeTune` into a per-prompt predictor of the hedge $\theta_t^\dagger$ that (i) reads only *label-free* statistics of a prompt's candidate pool and (ii) is guaranteed never to underperform the global hedge $\theta^\star$. This appendix gives, in order, the notation, the feature sets, the two fitting configurations, and the algorithm that assembles them into a deployable policy.

Notation. Vectors are bold lowercase, matrices bold uppercase, scalars italic, and sets calligraphic. For a vector, $x_{(i)}$ is its i-th *largest* entry; $\mathbf{1}\{\cdot\}$ is the indicator; and $H(\mathbf{q}) = -\sum_i q_i \log q_i$ is the Shannon entropy of a probability vector $\mathbf{q}$. Fix a prompt t and a reward model m. Let

$$\mathbf{s} \;=\; \mathbf{s}_t^{(m)} \;=\; \big(s_{t,1}^{(m)}, \ldots, s_{t,K}^{(m)}\big)^\top \in \mathbb{R}^K$$

be model m's score vector over the prompt's K candidates, with sample mean $\bar{s} = \frac{1}{K}\sum_k s_k$, standard deviation $\mathrm{sd}(\mathbf{s}) = \big(\frac{1}{K}\sum_k (s_k - \bar{s})^2\big)^{1/2}$, and softmax $\mathbf{p} = \mathrm{softmax}(\mathbf{s}) \in [0,1]^K$ with $p_k = e^{s_k}/\sum_j e^{s_j}$ (so $\sum_k p_k = 1$). Let $\mathbf{S}_t = \big[s_{t,k}^{(m')}\big] \in \mathbb{R}^{M\times K}$ stack all M models' scores for this prompt, and let $\kappa_{m'} = \arg\max_k s_{t,k}^{(m')} \in \{1,\ldots,K\}$ be model m''s top-scoring candidate. In parseable domains, $\mathbf{a} = (a_1,\ldots,a_K) \in \mathcal{A}^K$ are the candidates' parsed answers (with $\mathcal{A}$ the answer alphabet). We write $\rho(\cdot,\cdot) \in [-1,1]$ for the Spearman rank correlation. The feature map sends the candidate pool to a vector $\mathbf{x}_t = \phi(t) \in \mathbb{R}^p$.

Feature sets (all label-free). We use three *nested* maps of increasing dimension, $\phi_{\text{base-4}} \subset \phi_{\text{universal}} \subset \phi_{\text{full}}$ with $p = 4 \subset 12 \subset 16$, which index the columns of Table 4; Table 3 defines every coordinate. **base-4** collects four dispersion/disagreement statistics generalizing the single-reward diagnostics of Khalaf et al. (2025); **universal** adds scale-free shape statistics of $\mathbf{s}$ and cross-model agreement statistics of $\mathbf{S}_t$; **full** adds four answer-space statistics, and reduces to universal where answers do not parse (MBPP-Plus, IFEval).

Table 3: **Label-free features** of `Adaptive HedgeTune` (prompt t, reward model m; notation above). Each block is added to the previous one, giving the nested sets base-4 (p=4) $\subset$ universal (p=12) $\subset$ full (p=16). All quantities are computed from proxy scores (and, for the last block, parsed answers), never the true reward.

Feature	Definition (scalar, per prompt t and model m)
base-4: dispersion of $\mathbf{s}$ and cross-model disagreement (every domain)	
top-1/2 gap	$s_{(1)} - s_{(2)}$
spread	$\mathrm{sd}(\mathbf{s}) = \big(\frac{1}{K}\sum_k (s_k - \bar{s})^2\big)^{1/2}$
range	$s_{(1)} - s_{(K)}$
ensemble disagreement	$d = \binom{M}{2}^{-1} \sum_{m'<m''} \mathbf{1}\{\kappa_{m'} \neq \kappa_{m''}\}$ (pairwise top-pick disagreement)
+*universal*: score shape of $\mathbf{s}$, cross-model agreement of $\mathbf{S}_t$ (every domain)	
mean score	$\bar{s} = \frac{1}{K}\sum_k s_k$
softmax entropy	$H(\mathbf{p})/\log K$
Gini coefficient	$G(\mathbf{p}) = \big(2K\big)^{-1} \sum_{i,j} \lvert p_i - p_j \rvert$
top-1 mass	$\max_k p_k$
Hill tail-index	$\frac{1}{k_0}\sum_{i=1}^{k_0} \log \tilde{s}_{(i)} - \log \tilde{s}_{(k_0+1)},\ \tilde{\mathbf{s}} = \mathbf{s} - \min_k s_k + \varepsilon,\ k_0$=6
mean Spearman	$\binom{M}{2}^{-1} \sum_{m'<m''} \rho\big(\mathbf{s}^{(m')}, \mathbf{s}^{(m'')}\big)$
top-1 concordance	$\max_v \hat{q}_v,\ \hat{q}_v = \frac{1}{M}\sum_{m'} \mathbf{1}\{\kappa_{m'}$=$v\}$
top-1 entropy	$H(\hat{\mathbf{q}})/\log M$
+*answer* ("full"): parsed answers $\mathbf{a}$ (parseable domains only)	
self-consistency	$\max_a \hat{\pi}_a,\ \hat{\pi}_a = \frac{1}{K}\sum_k \mathbf{1}\{a_k$=$a\}$
answer entropy	$H(\hat{\boldsymbol{\pi}})/\log K$
discord	$\rho(\mathbf{s}, \mathbf{c}),\ c_k = \sum_j \mathbf{1}\{a_j$=$a_k\}$
modal match	$\mathbf{1}\{a_\kappa = \arg\max_a \hat{\pi}_a\},\ \kappa = \arg\max_k s_k$

Two fitting configurations. The regression target is the per-prompt oracle hedge $\theta_t^\dagger = \arg\max_\theta f_t(\theta)$ (label-derived; used only to *fit*, never read at deployment). Each configuration c fits an *affine* predictor

$$\phi_c(\mathbf{x}) \;=\; \boldsymbol{\beta}_c^\top \mathbf{x} + b_c, \qquad \boldsymbol{\beta}_c \in \mathbb{R}^{p_c},\ \ b_c \in \mathbb{R},$$

by a weighted, ℓ_2-penalized least squares over the calibration prompts (each feature standardized to zero mean and unit variance on the fit set; the intercept b_c unpenalized),

$$(\boldsymbol{\beta}_c, b_c) \;=\; \underset{\boldsymbol{\beta}\in\mathbb{R}^{p_c},\, b\in\mathbb{R}}{\arg\min} \sum_{t\in T} w_t^{(c)} \left(\theta_t^\dagger - \boldsymbol{\beta}^\top \mathbf{x}_t - b\right)^2 \;+\; \lambda \lVert \boldsymbol{\beta} \rVert_2^2, \tag{7}$$

the two configurations differing only in their feature map and per-prompt weights $w_t^{(c)} \geq 0$:

- **config-1** (*answers + ridge*): the *full* feature map (p_1=16, or 12 where answers do not parse), with uniform weights $w_t^{(1)} \equiv 1$ (ordinary ridge regression).
- **config-2** (*distributional + curvature loss*): the *universal* feature map (p_2=12), with *decision-focused* weights equal to the discrete curvature of the reward curve at its own peak,

$$w_t^{(2)} \;=\; \big|\, f_t(\theta_t^\dagger - 1) - 2 f_t(\theta_t^\dagger) + f_t(\theta_t^\dagger + 1) \,\big|. \tag{8}$$

  This up-weights prompts whose accuracy is *sensitive* to the hedge and nearly ignores prompts whose curve is flat near its peak (where any θ is almost optimal): the fit is tuned for the *decision* it drives, not for uniform accuracy on $\theta_t^\dagger$. It is this loss, not a new feature, that extracts signal where $\theta_t^\dagger$ is otherwise unpredictable (notably code; see Table 5).

The ridge penalty λ is set by inner cross-validation over $\{10^{-2}, \ldots, 10^3\}$, with the shrinkage α swept over $\{0, .25, .5, .75, 1\}$ under a paired one-standard-error gate. To test whether *per-feature* shrinkage helps, we refit (7) with Elastic-Net (adding an ℓ_1 term, $\lambda\,[\,\rho_{\mathrm{en}} \lVert \boldsymbol{\beta} \rVert_1 + (1-\rho_{\mathrm{en}}) \lVert \boldsymbol{\beta} \rVert_2^2\,]$, whose ℓ_1 part can zero out coordinates) and with ARD (Automatic Relevance Determination: a Bayesian linear model with an independent, evidence-tuned precision on each coordinate, $\beta_{c,j} \sim \mathcal{N}(0, \alpha_j^{-1})$, so uninformative features are pruned); these give the last two columns of Table 4.

From features to a policy. Algorithm 1 turns the pieces above into one deployable rule; it reads most easily as a short story about a single prompt. *First*, before personalizing anything, the method fixes a safe baseline: the one global hedge $\theta^\star$ that `HedgeTune` would use for the whole dataset, i.e. the peak of the average true-reward curve on the calibration prompts. Everything that follows is measured against this fallback. *Second*, it learns to personalize: from the calibration labels it reads off each prompt's oracle hedge $\theta_t^\dagger$ and fits the two predictors ϕ_1, ϕ_2 of (7), one listening to answer agreement (config-1), the other to the shape of the score vector through the curvature loss (config-2). A raw prediction, though, can be confidently wrong on a prompt whose features are uninformative, so the method never simply trusts it. *Third*, it therefore hedges its own prediction: for each predictor c and each shrinkage weight α it forms a blended hedge

$$\theta_{c,\alpha}(\mathbf{x}) \;=\; \Pi_{\mathcal{H}_\mathcal{M}}\big(\mathrm{round}((1{-}\alpha)\theta^\star + \alpha\,\phi_c(\mathbf{x}))\big),$$

which slides from the global hedge ($\alpha{=}0$) toward the fully personalized prediction ($\alpha{=}1$); here $\Pi_{\mathcal{H}_\mathcal{M}}$ rounds onto the admissible hedges (integers for BoN, positive reals for SBoN and BoP). It scores every such blend on held-out *inner* folds. *Fourth* comes the gatekeeper: it adopts a predictor–shrinkage pair only if that blend's average gain over $\theta^\star$ clears one standard error; otherwise it snaps back to $\alpha^\star{=}0$, i.e. plain `HedgeTune`. This one-standard-error rule is the "never-worse" floor: the method cannot, in expectation, end up behind the hedge it started from. *Finally*, it returns the winning policy $\hat\theta(\cdot)$, now a pure function of label-free features: at deployment it reads a new prompt's $\mathbf{x}$, blends and rounds as above, and runs $\mathcal{M}$ at $\hat\theta(\mathbf{x})$, never consulting a true-reward label again. The true reward is used *only* in the two calibration steps, through f_t and $\theta_t^\dagger$, and vanishes from the deployed rule. The two diagnostics that follow probe different stages of this story: Table 4 asks how well the personalizing step's predictors recover $\theta_t^\dagger$ in isolation (R_c^2), whereas Table 5 reports the deployable accuracy of the whole assembled policy.

Algorithm 1 `Adaptive HedgeTune` (per reward model)

Require: calibration prompts T, each with a label-free feature vector $\mathbf{x}_t \in \mathbb{R}^p$ and true-reward curve f_t; method $\mathcal{M}$ with admissible hedges $\mathcal{H}_\mathcal{M}$ and projection $\Pi_{\mathcal{H}_\mathcal{M}}$; shrinkage grid $\mathcal{A} = \{0, .25, .5, .75, 1\}$

Ensure: per-prompt hedge policy $\hat\theta(\cdot) : \mathbf{x} \mapsto \mathcal{H}_\mathcal{M}$

1: $\theta^\star \leftarrow \arg\max_\theta \; \frac{1}{|T|}\sum_t f_t(\theta)$ ▷ global `HedgeTune` hedge: the fallback

2: fit predictors ϕ_1, ϕ_2 to the oracle targets $\theta_t^\dagger = \arg\max_\theta f_t(\theta)$ by (7) ▷ labels used here only

3: for each config c and $\alpha \in \mathcal{A}$, score the blend $\theta_{c,\alpha}(\mathbf{x}) = \Pi_{\mathcal{H}_\mathcal{M}}\big(\mathrm{round}((1{-}\alpha)\theta^\star + \alpha\,\phi_c(\mathbf{x}))\big)$ against $\theta^\star$ on inner folds

4: $(c^\star, \alpha^\star) \leftarrow$ the blend whose mean gain over $\theta^\star$ exceeds one standard error; if none, $\alpha^\star{=}0$ ▷ never-worse floor

5: **return** $\hat\theta(\cdot) = \theta_{c^\star,\alpha^\star}(\cdot)$ ▷ $\alpha^\star{=}0 \Rightarrow \hat\theta \equiv \theta^\star$

Evaluation protocol (reporting only). To report the deployable gain Δ over global `HedgeTune`, we evaluate out-of-fold at the prompt level (5-fold $\times$ 4 seeds): `Adaptive HedgeTune` is run on each training partition and the returned $\hat\theta(\cdot)$ is scored on the held-out prompts; Tables 1 and 5 report the mean over folds, seeds, and the M reward models, with Δ reserved for this held-out quantity. The outer folds never enter Algorithm 1 (they are only for reporting); the inner folds of step 3 are what select $(c^\star, \alpha^\star)$.

Is the per-prompt hedge predictable without labels? (Table 4.) `Adaptive HedgeTune`'s deployable gain is fundamentally capped by how well the target $\theta_t^\dagger$ can be predicted from label-free features: if $\theta_t^\dagger$ carries no signal recoverable without labels, no selector can beat the global hedge. We therefore measure this predictability directly, by the *cross-validated coefficient of determination*

$$R_c^2 \;=\; 1 - \frac{\sum_t \big(\theta_t^\dagger - \hat\phi_{-t}(\mathbf{x}_t)\big)^2}{\sum_t \big(\theta_t^\dagger - \bar\theta^\dagger\big)^2}, \tag{9}$$

where $\hat\phi_{-t}(\mathbf{x}_t)$ is the *out-of-fold* prediction for prompt t (5-fold) and $\bar\theta^\dagger = \frac{1}{|T|}\sum_t \theta_t^\dagger$ is the mean target; we report the median over the $M{=}14$ models. Being computed out-of-fold, R_c^2 is honest about

overfitting: $R_c^2 = 1$ is perfect prediction, $R_c^2 = 0$ means the features do no better than predicting the constant mean $\bar{\theta}^\dagger$, and $R_c^2 < 0$ means the fit generalizes *worse* than that constant. Table 4 delivers the verdict. From score statistics alone, $\theta_t^\dagger$ is essentially unpredictable (base-4 and universal $R_c^2 \approx 0$, often negative). Answer-space features lift R_c^2 on the three parseable domains (up to 0.26 on MATH) but do nothing on code or instruction-following, where answers cannot be parsed. And per-feature shrinkage (Elastic-Net, ARD) matches plain ridge everywhere, so the ceiling is a genuine absence of *label-free signal*, not a modeling or regularization choice. This is the mechanistic reason the optimizer lever saturates at $+0.40$ points in deployment (§3), despite the $\sim$17-point oracle ceiling.

Table 4: **The per-prompt hedge $\theta_t^\dagger$ is not label-free predictable.** Cross-validated R_c^2 (↑) (9) of predicting $\theta_t^\dagger$ from label-free features (median over the M=14 reward models; higher = more predictable, 0 = no better than the mean, <0 = worse). Columns are the nested feature sets base-4 ⊂ universal ⊂ full and, on the full set, two per-feature regularizers (Elastic-Net, ARD) against plain ridge (the "+answer (full)" column); all defined in §C. Answer-space features help only where answers parse (GPQA/MMLU-Pro/MATH); per-feature shrinkage never beats ridge. Values are at a fixed CV seed; across four seeds every cell is stable (per-cell standard deviation ≤ 0.01), so we omit per-cell error bars here.

Method	Domain	base-4	+universal	+answer (full)	full/Elastic-Net	full/ARD
BoN	GPQA	−0.005	−0.008	+0.039	+0.042	+0.038
	MMLU-Pro	−0.001	+0.009	+0.103	+0.102	+0.102
	MATH	+0.001	+0.031	+0.260	+0.259	+0.269
	MBPP-Plus	+0.013	+0.014	+0.014	+0.015	+0.013
	IFEval	−0.008	−0.011	−0.011	−0.007	−0.015
SBoN	GPQA	−0.004	−0.010	−0.002	+0.000	+0.002
	MMLU-Pro	+0.006	+0.013	+0.045	+0.044	+0.050
	MATH	+0.014	+0.044	+0.153	+0.153	+0.156
	MBPP-Plus	+0.042	+0.051	+0.051	+0.050	+0.051
	IFEval	−0.009	−0.011	−0.011	−0.008	−0.008
BoP	GPQA	−0.005	−0.008	+0.014	+0.015	+0.025
	MMLU-Pro	+0.001	+0.013	+0.077	+0.078	+0.091
	MATH	+0.003	+0.037	+0.231	+0.236	+0.237
	MBPP-Plus	+0.018	+0.017	+0.017	+0.017	+0.016
	IFEval	−0.008	−0.011	−0.011	−0.006	−0.016

Table 5: **Deployable gain of `Adaptive HedgeTune` over global `HedgeTune`** (subscripts = std over the 4 seed-pairs). Column **H** is the global `HedgeTune` accuracy (↑ %); the next three give the gain Δ (↑ pp) over it: from **C1** (config-1 = full features + ridge), **C2** (config-2 = distributional features + curvature-weighted loss (8)), and the floor-gated *auto-selector*, which picks per (model, fold) among {H, C1, C2} by inner CV. "picks" is the % of decisions choosing H/C1/C2 (a wiring diagnostic). Being a per-fold mixture, the auto-selector's mean can match a config's while its std differs (e.g. MATH). Pooled = mean over domains; H + auto is the `Adaptive HedgeTune` accuracy of Table 1.

		`HedgeTune`	Gain over `HedgeTune`, Δ (pp) ↑			
Method	Domain	(H) acc. %	config-1 (C1)	config-2 (C2)	auto-selector	picks H/C1/C2
BoN	GPQA	43.48 $_{\pm 0.16}$	-0.00 $_{\pm 0.03}$	-0.08 $_{\pm 0.05}$	-0.03 $_{\pm 0.02}$	46/44/10
	MMLU-Pro	60.91 $_{\pm 0.07}$	+0.21 $_{\pm 0.04}$	-0.01 $_{\pm 0.02}$	+0.21 $_{\pm 0.04}$	55/44/1
	MATH	48.48 $_{\pm 0.11}$	+0.88 $_{\pm 0.18}$	-0.03 $_{\pm 0.02}$	+0.88 $_{\pm 0.17}$	27/70/2
	MBPP-Plus	61.61 $_{\pm 0.04}$	-0.03 $_{\pm 0.04}$	+1.04 $_{\pm 0.04}$	+0.94 $_{\pm 0.07}$	43/8/50
	IFEval	59.07 $_{\pm 0.05}$	-0.01 $_{\pm 0.04}$	+0.00 $_{\pm 0.03}$	-0.01 $_{\pm 0.07}$	59/32/9
	Pooled	54.71 $_{\pm 0.06}$			**+0.40** $_{\pm 0.05}$	
SBoN	GPQA	43.56 $_{\pm 0.12}$	-0.06 $_{\pm 0.02}$	-0.01 $_{\pm 0.04}$	-0.05 $_{\pm 0.04}$	78/8/15
	MMLU-Pro	60.55 $_{\pm 0.03}$	+0.01 $_{\pm 0.04}$	+0.04 $_{\pm 0.02}$	+0.02 $_{\pm 0.05}$	89/4/7
	MATH	48.43 $_{\pm 0.05}$	+0.14 $_{\pm 0.07}$	-0.04 $_{\pm 0.05}$	+0.14 $_{\pm 0.06}$	57/41/2
	MBPP-Plus	61.38 $_{\pm 0.04}$	+1.18 $_{\pm 0.03}$	+1.20 $_{\pm 0.03}$	+1.23 $_{\pm 0.02}$	60/17/23
	IFEval	58.71 $_{\pm 0.04}$	+0.06 $_{\pm 0.01}$	+0.11 $_{\pm 0.02}$	+0.11 $_{\pm 0.01}$	86/2/12
	Pooled	54.53 $_{\pm 0.04}$			**+0.29** $_{\pm 0.01}$	
BoP	GPQA	43.51 $_{\pm 0.12}$	-0.00 $_{\pm 0.05}$	-0.04 $_{\pm 0.05}$	-0.01 $_{\pm 0.05}$	49/42/9
	MMLU-Pro	60.86 $_{\pm 0.05}$	+0.19 $_{\pm 0.06}$	-0.00 $_{\pm 0.02}$	+0.19 $_{\pm 0.06}$	64/36/0
	MATH	48.50 $_{\pm 0.10}$	+0.83 $_{\pm 0.10}$	-0.01 $_{\pm 0.01}$	+0.83 $_{\pm 0.10}$	28/72/0
	MBPP-Plus	61.58 $_{\pm 0.04}$	-0.03 $_{\pm 0.02}$	+0.84 $_{\pm 0.02}$	+0.74 $_{\pm 0.04}$	42/7/51
	IFEval	58.98 $_{\pm 0.05}$	-0.02 $_{\pm 0.05}$	+0.01 $_{\pm 0.01}$	+0.00 $_{\pm 0.06}$	60/32/7
	Pooled	54.69 $_{\pm 0.06}$			**+0.35** $_{\pm 0.05}$	

Reading Table 5. The two configurations are complementary, exactly as their design intends. *config-1* (answer agreement) supplies the gains on the parseable, verifiable-answer domains (MATH $+0.88$, MMLU-Pro $+0.21$), where self-consistency and discord reconstruct the score–truth misalignment. *config-2* (the curvature-weighted loss) is what cracks *code*: on MBPP-Plus, where answers do not parse and $R_c^2 \approx 0$ (Table 4), focusing the fit on hedge-sensitive prompts yields $+1.04$, whereas config-1 cannot (-0.03); here it is the *loss*, not the feature, that unlocks the signal. GPQA and IFEval move for neither configuration, consistent with their near-zero R_c^2. Because *which* configuration wins is itself domain- and model-dependent (and unknown a priori without labels), the auto-selector does not commit to one globally: for each reward model and fold it chooses among {global, config-1, config-2} on the training fold's inner CV alone (step 4 of Algorithm 1), and the reported Δ is the mean accuracy of that adaptive policy over the M models and folds, with "picks" recording how often each option was selected. Two consequences of this per-model-per-fold gating explain the table's apparent oddities. First, *the auto-selector Δ need not lie between the two configs*: on MBPP-Plus it is $+0.94 < +1.04$ because on 43% of decisions it (safely) fell back to the global hedge rather than committing to config-2. Second, because the never-worse floor is guaranteed only *in expectation* (a paired one-standard-error test estimated on finite folds), residual selection noise leaves the two hardest domains marginally negative rather than exactly zero: -0.03 on GPQA (picks $46/44/10$: it mostly falls back to global, but the few config picks it does make are, on those folds, slightly harmful) and -0.01 on IFEval. Pooled over domains the selector recovers $+0.40$ points for BoN, roughly 95% of the best envelope of the two configurations, and is parameterization-consistent ($+0.29$/$+0.35$ for SBoN/BoP). Set against the $\sim$16-point oracle ceiling of Table 1, this small-but-real gain is the quantitative statement of the paper's Lever-1 claim: the optimizer lever is label-gated. As a final check that the gap is signal-limited rather than feature-limited, generator-confidence features (per-candidate log-probability and token-level self-certainty read from the base model) were also added to config-1 and changed R_c^2 and Δ negligibly (≤ 0.03 pp), confirming that the missing signal is not something a richer label-free feature recovers.

D OPTIMIZER-LEVER DIAGNOSTIC: EXPERIMENTAL SETUP AND ADDITIONAL SAMPLERS

Setup and how the diagnostic is computed. Figure 2 in the main text, and Figure 4 below, use the PPE Best-of-K data of §2 (K=32 candidates per prompt; binary true reward $r_{t,k} \in \{0, 1\}$). Each panel is a single, illustrative (domain, reward-model) pair, chosen to match the settings of Khalaf et al. (2025): GPQA with Skywork-Reward-Llama-3.1-8B, MMLU-Pro with InternLM2-1.8B-Reward, MMLU-Pro with Llama-3-OffsetBias-RM-8B, and MATH with InternLM2-1.8B-Reward; scores within a panel are not averaged across models. For a prompt t we compute the *exact* finite-pool Best-of-n true-reward curve by order statistics,

$$f_t(n) \;=\; \sum_{r=1}^{K} c_{(r),t}\, \frac{\binom{r-1}{n-1}}{\binom{K}{n}}, \tag{10}$$

where $c_{(r),t} \in \{0, 1\}$ is the true reward of the candidate with the r-th smallest proxy score for prompt t (ties broken by a stable sort), and the weight is the probability that the highest-proxy response among n draws without replacement has rank r. The blue curve is the across-prompt mean $\bar{f}(n) = \frac{1}{|T|}\sum_t f_t(n)$ (shaded: standard error); the gold star marks the global `HedgeTune` operating point $\theta^\star = \arg\max_n \bar{f}(n)$; and the grey histogram (right axis) is the distribution over prompts of the per-prompt optimum $\theta_t^\dagger = \arg\max_n f_t(n)$, whose sharp bimodality is the effect the figure highlights. The green dashed line is the *oracle* per-prompt ceiling $\frac{1}{|T|}\sum_t \max_n f_t(n)$, attainable only with true-reward labels, and the annotated arrow is its label-gated gap over $\bar{f}(\theta^\star)$. The SBoN panels replace n by an inverse temperature applied to rank-uniformized proxy scores (at n=K); the BoP panels use the Poisson rate μ, whose curve is a Poisson mixture of the Best-of-n curves (10) plotted against expected samples. These four pairs make the mechanism visible; the same per-prompt optima, oracle ceiling, and label-gated gap are quantified in Table 1 as an average over all M=14 reward models and five domains ($+16.8/+11.7/+15.5$ points for BoN/SBoN/BoP). No true-reward labels enter the diagnostic except to define the oracle ceiling.

How Table 1 is computed. Table 1 reports four operating points on the accuracy scale of §2, each averaged over the M=14 reward models and five domains, for BoN/SBoN/BoP. *Reference* is the base policy at n=1 (no selection). *Global* `HedgeTune` is the single tuned hedge $\theta^\star$, evaluated with prompt-level cross-fitting (5-fold $\times$ 4 seeds): $\theta^\star$ is chosen on the training prompts and scored on disjoint held-out prompts, which removes the optimism of tuning and scoring on the same prompts. *Oracle per-prompt* places every prompt at its own label-optimal hedge $\theta_t^\dagger = \arg\max_n f_t(n)$ from the curves (10) and averages $\max_n f_t(n)$; identifying $\theta_t^\dagger$ needs true-reward labels, so this is an upper bound, and its parenthesized *label-gated gap* is oracle minus global ($+16.8/+11.7/+15.5$ for BoN/SBoN/BoP). *Adaptive* `HedgeTune` is our deployable, label-free selector (§3); its entry is the pooled cross-fitted accuracy of Table 5, fit and evaluated under the nested cross-fitting detailed in Appendix C. Reference and the oracle ceiling are deterministic given the candidate pool and carry no error bar; the $\pm$ subscripts on the two `HedgeTune` columns are standard deviations across the four seeds.

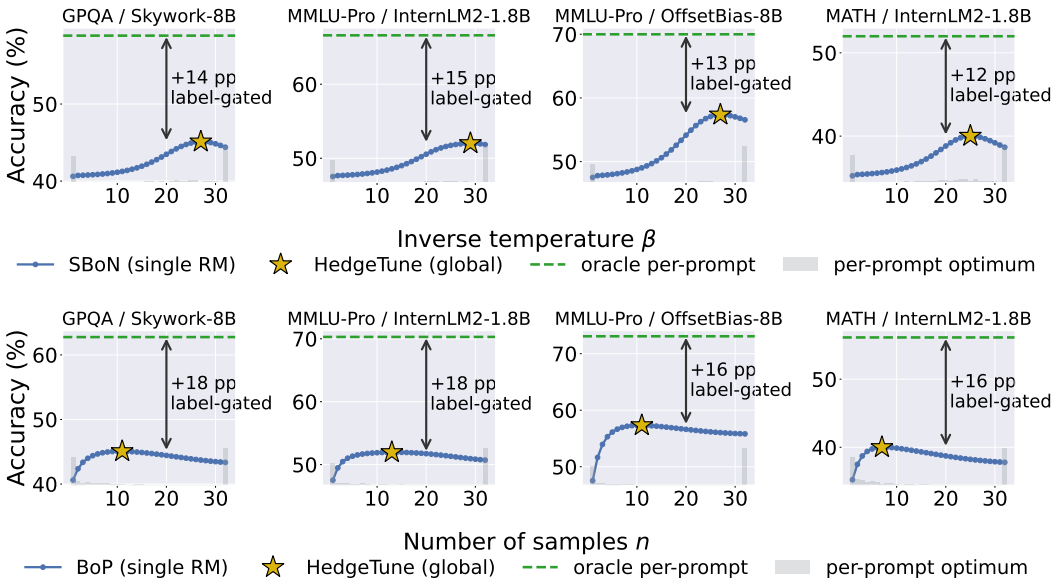


Figure 4: Optimizer-lever diagnostic for SBoN (top; hedge = inverse temperature λ) and BoP (bottom; hedge = μ, plotted as samples n). As in Figure 2, the per-prompt optimum is bimodal and the oracle ceiling is large but label-gated.

E LEVER-2 FIGURES FOR SBON AND BOP, AND THE ENSEMBLE ZOOM

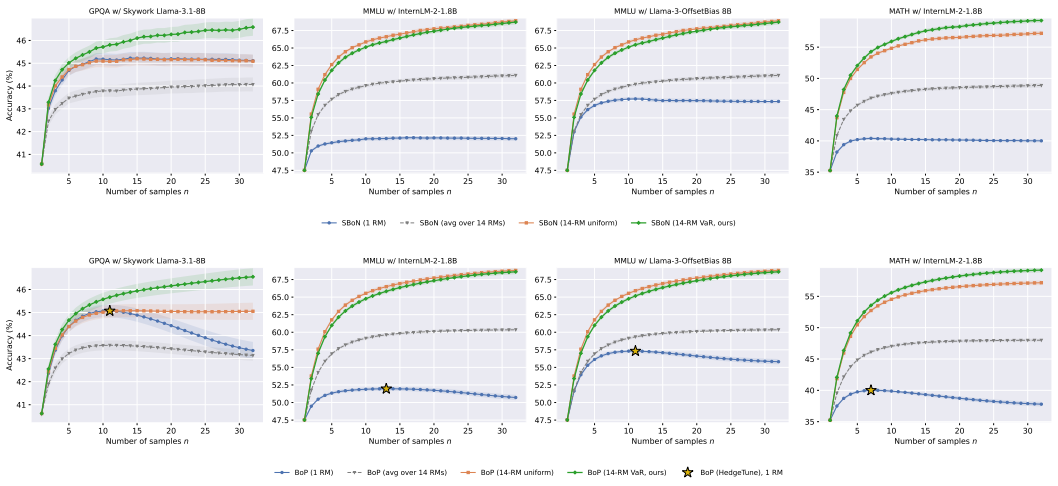


Figure 5: VaR aggregation for SBoN (top) and BoP (bottom): as in Figure 3, aggregation flattens the hump and VaR leads on reasoning/QA.

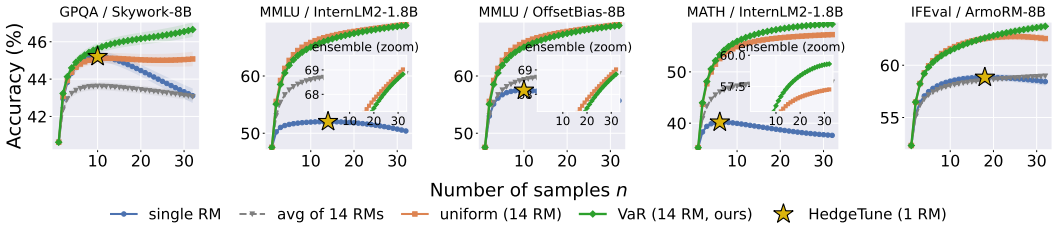


Figure 6: BoN aggregation with ensemble-zoom insets, resolving the (smaller) difference between the uniform mean and VaR where the two nearly overlap.

F Full aggregation ablation

Tables 6–8 give the complete per-domain ablation (accuracy ↑ %; subscripts are the standard deviation over 4 seeds), adding the uncertainty-penalized (UWO) and worst-case (WCO) aggregations to the columns of Table 2. Reference, Best-of-K and the oracle best-RM carry no error bar (deterministic / label-dependent). The risk aggregations (UWO/VaR/WCO) have visibly larger seed variance than `HedgeTune` (avg RM) and Uniform, because a low quantile across models is more sensitive to the fold split. UWO ≈ VaR; WCO is consistently weakest (the worst-case is dragged down by the misaligned NaiveVerbosity model).

Table 6: Full ablation: BoN (accuracy ↑, %).

Dataset	Ref	Best-of-K	HT avg-RM	HT best-RM	Uniform	UWO	VaR	WCO
GPQA	40.62	43.07	43.48 ± 0.16	47.36	44.27 ± 0.41	45.42 ± 0.71	46.39 ± 0.33	45.52 ± 0.46
MMLU-Pro	47.52	60.39	60.91 ± 0.07	75.88	69.02 ± 0.00	68.24 ± 0.55	67.35 ± 0.67	66.59 ± 0.09
MATH	35.23	48.01	48.48 ± 0.11	60.65	57.26 ± 0.00	58.49 ± 0.29	58.78 ± 0.75	55.63 ± 0.01
MBPP-Plus	56.15	56.54	61.61 ± 0.04	81.13	56.86 ± 0.09	57.31 ± 0.01	56.55 ± 0.90	56.31 ± 1.16
IFEval	52.32	58.96	59.07 ± 0.05	65.49	62.29 ± 0.45	62.00 ± 0.65	62.75 ± 1.18	60.24 ± 0.17
Mean	46.37	53.39	54.71 ± 0.06	66.10	57.94 ± 0.02	58.29 ± 0.32	58.36 ± 0.33	56.86 ± 0.26

Table 7: Full ablation: SBoN (accuracy ↑, %).

Dataset	Ref	Best-of-K	HT avg-RM	HT best-RM	Uniform	UWO	VaR	WCO
GPQA	40.62	43.40	43.56 ± 0.12	47.82	44.62 ± 0.20	45.37 ± 0.38	46.16 ± 0.04	45.28 ± 0.09
MMLU-Pro	47.52	60.14	60.55 ± 0.03	74.50	67.82 ± 0.00	67.89 ± 0.14	66.90 ± 0.38	65.59 ± 0.00
MATH	35.23	47.87	48.43 ± 0.05	60.80	56.64 ± 0.00	57.34 ± 0.29	58.38 ± 0.00	54.25 ± 0.01
MBPP-Plus	56.15	56.65	61.38 ± 0.04	80.37	56.28 ± 0.15	56.56 ± 0.28	56.25 ± 0.61	55.92 ± 0.63
IFEval	52.32	58.66	58.71 ± 0.04	64.00	62.64 ± 0.00	62.45 ± 0.37	62.49 ± 0.14	60.74 ± 0.18
Mean	46.37	53.35	54.53 ± 0.04	65.50	57.60 ± 0.04	57.92 ± 0.12	58.04 ± 0.10	56.36 ± 0.14

Table 8: Full ablation: BoP (accuracy ↑, %).

Dataset	Ref	Best-of-K	HT avg-RM	HT best-RM	Uniform	UWO	VaR	WCO
GPQA	40.62	43.14	43.51 ± 0.12	47.48	44.37 ± 0.35	45.22 ± 0.57	46.38 ± 0.24	45.53 ± 0.38
MMLU-Pro	47.52	60.35	60.86 ± 0.05	75.60	68.80 ± 0.00	68.37 ± 0.47	67.07 ± 0.44	66.52 ± 0.00
MATH	35.23	48.00	48.50 ± 0.10	60.51	57.17 ± 0.00	58.54 ± 0.09	58.81 ± 0.75	55.39 ± 0.01
MBPP-Plus	56.15	56.56	61.58 ± 0.04	81.09	56.36 ± 0.08	56.59 ± 0.25	56.25 ± 0.79	55.79 ± 0.81
IFEval	52.32	58.91	58.98 ± 0.05	65.22	62.28 ± 0.32	62.02 ± 0.52	62.67 ± 1.10	60.41 ± 0.13
Mean	46.37	53.39	54.69 ± 0.06	65.98	57.80 ± 0.02	58.15 ± 0.27	58.24 ± 0.27	56.73 ± 0.15

G COMPUTATIONAL AND MEMORY COST

The paper's central claim is that the practical, deployable lever against inference-time reward hacking is the *reward*, not the optimizer. A claim of practicality has to survive a cost audit, so in this section we evaluate what our methods actually require to run, in compute and in memory, and we show that the lever that delivers the accuracy is also the one that is cheap to deploy.

A shared, dominant cost, and where the methods differ. All three methods are inference-time procedures whose per-prompt cost decomposes into generation (n candidates), scoring (n times the reward-model forwards), and selection or aggregation. Generation, the autoregressive decoding of the n candidates from π_{ref}, dominates and is *shared* by every method here; a reward-model score is a single forward pass per candidate, and selection or aggregation is $O(nM)$ comparisons. The question is therefore never about the dominant term, only about the two smaller ones on which the methods differ (Table 9).

Table 9 setup. The accuracy column is the pooled BoN accuracy of §2: the five PPE domains (GPQA, MMLU-Pro, MATH, MBPP-Plus, IFEval; K=32 candidates per prompt; binary true reward $r_{t,k} \in \{0, 1\}$), cross-fitted at the prompt level (5-fold $\times$ 4 seeds, seeds $\{0, 2, 4, 6\}$) and averaged over domains, exactly as in Tables 1 and 2. *Reference* is n=1; *Best-of-K* is un-hedged selection at n=K averaged over the M=14 reward models; `HedgeTune` and *Adaptive* `HedgeTune` are the single-model policies of §3 (the latter's entry is the pooled cross-fitted value of Table 5); *VaR, 14 RM* is the lower α-quantile aggregation (6) with $\alpha \in \{10\%, 25\%\}$ and the uniform mean as a floor, selected by cross-fitted `HedgeTune`; *VaR, 13 RM* is the same after removing the 340B model (Nemotron-4-340B-Reward). "RM passes / candidate" counts reward-model forward passes (0 for Reference, 1 for single-model methods, M for VaR); "Extra params" counts trainable parameters beyond the reward models (the $\sim$17 for Adaptive `HedgeTune` are its ridge coefficients over up to 16 label-free features plus an intercept, Appendix C). "Added overhead" and "calibration" are CPU wall-clock, measured on one domain ($\sim$510 prompts) and averaged over 50 repetitions: the VaR aggregation is the per-prompt z-scoring plus the two across-model percentiles, and the Adaptive-`HedgeTune` feature step is the per-prompt score statistics of Appendix C, while the calibration figure is one RidgeCV fit (ten ridge penalties) with inner cross-validation.

Table 9: **Computational and memory cost** (BoN). Generating the n candidates is the dominant cost and is shared by all methods; they differ only in reward-model (RM) scoring and cheap bookkeeping. "Added overhead" and calibration are measured on CPU; accuracy is pooled over the five domains (Tables 1, 2). No method trains the generator or the reward models, and none uses a GPU for its own logic.

Method	RM passes / candidate	Extra params	Added overhead (ms/prompt)	One-time calibration	Accuracy (%)
Reference (n=1)	0	0	0	none	46.37
Best-of-K (un-hedged)	1	0	0	none	53.39
`HedgeTune` (1 RM)	1	0	≈ 0	1-D root-find	54.71
Adaptive `HedgeTune` (ours)	1	$\sim$ 17	0.10 (features)	ridge + CV	55.11
VaR, 14 RM (ours)	14	0	0.02 (quantile)	quantile select	58.36
VaR, 13 RM (drop 340B)	13	0	0.02	quantile select	58.29

The optimizer lever is free, as its small payoff would demand. Lever 1 recovers only $+0.40$ points (§3), so it would scarcely be worth having unless it were nearly free, and it is. `HedgeTune` merely chooses a smaller sample count on one reward model, and Adaptive `HedgeTune` adds only label-free features read off the *already-computed* scores (0.10 ms/prompt on CPU) and an affine predictor of $\sim$17 parameters. It introduces no extra model, no extra forward pass, and no GPU; its one-time calibration is an off-the-shelf ridge fit ($\sim$5 ms) with inner cross-validation. The optimizer lever is thus as inexpensive as it is capped.

The reward lever costs only a suite of scores. VaR is where the accuracy is, so its cost is the one that matters. The aggregation itself is a two-line reduction over the standardized scores

(0.02 ms/prompt) with no trainable parameters; its entire overhead over `HedgeTune` is the M-fold reward-model scoring, a relative increase of $(\sum_m C_m^{\mathrm{RM}})/C^{\mathrm{gen}}$ that stays small because a reward score is a single forward pass whereas generation is autoregressive, and because the scores parallelize and can be cached (in our setup they are precomputed). Memory follows the same logic: the M models can be scored one at a time (peak memory = a single model) or their scores cached, so VaR adds no resident trainable parameters over a single-reward-model pipeline.

Figure 7 setup. Both panels reuse the BoN, cross-fitted, pooled accuracy of Table 9, varying only the *set* of reward models that VaR aggregates. For a subset, we z-score each model's scores across the K candidates, take the lower α-quantile ($\alpha \in \{10\%, 25\%\}$) across the models in the subset, select α against the subset's own uniform mean by cross-fitted `HedgeTune`, and pool over the five domains. In the left panel, the *random-suite* curve draws, for each size $M \in \{1, 2, 3, 4, 5, 6, 8, 10, 12, 14\}$, up to 25 distinct random subsets under a fixed seed and reports their mean and min–max envelope, while the *cheapest-first* curve adds reward models in increasing order of parameter count. The right panel plots the cheapest-first accuracy against the cumulative parameter count of the included models (log scale), using the sizes of Appendix A (a ~ 0 to 340B span, 545.8B in total); the star marks all 14 models and the annotation marks dropping the 340B model. The two reference lines are the single-RM `HedgeTune` average (54.71) and all-14 VaR (58.36). The full sweep runs in about 10 seconds on CPU from the precomputed scores.

And that cost is tunable, for the reason the mechanism predicts (Figure 7). The M-fold scoring is not even mandatory. Averaged over random subsets, a *diverse handful* of about 5 to 6 reward models already recovers essentially the full gain (58.2 to 58.5 versus 58.36 for all 14), and the single largest model (340B) can be dropped for 0.07 points at 2.6× fewer reward-model parameters (58.29 at 206B versus 58.36 at 546B). The budget is governed by the very mechanism of §5: because the gain comes from error *diversity*, *which* models are aggregated matters more than how many. The cheapest and weakest few (including the length-only NaiveVerbosity model) do not suffice, but a small, diverse set does, so a method's effectiveness and its cost are controlled by the same principle.

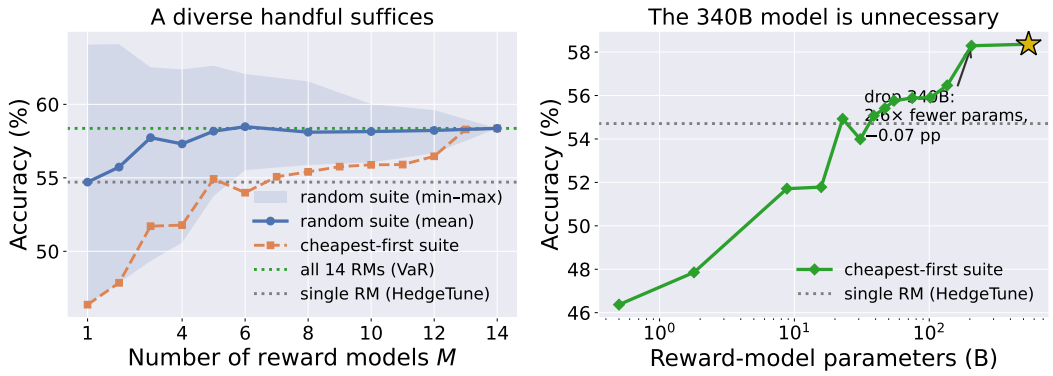


Figure 7: **VaR's reward-model budget is tunable.** Left: pooled BoN accuracy versus the number of aggregated reward models M (random suites: mean and min–max over 25 subsets per M, and a cheapest-first suite), against single-RM `HedgeTune` (54.71) and all-14 VaR (58.36); a diverse $\sim$5–6 models recover the full gain. Right: accuracy versus cumulative reward-model parameters for the cheapest-first suite; dropping the 340B model costs 0.07 points at 2.6× fewer parameters. *Which* models are aggregated matters more than how many (error diversity, §5).

Why this reinforces the paper's claim. The audit strengthens the thesis rather than qualifying it. The move that delivers the gains adds no training, no trainable parameters, and no generation cost; it reuses reward models a practitioner already has, needs only a few diverse ones, and runs on their scores alone. Fixing the reward is therefore not merely more effective than hedging the optimizer harder; it is also the cheaper and more deployable option, which is why we advocate it.

H LIMITATIONS AND BROADER IMPACT

We state plainly where our claims hold, where they do not, and what deploying the reward lever could mean in practice.

Limitations. First, our evidence is confined to *verifiable* domains, where the true reward is an automated binary correctness signal (exact match, unit tests, constraint checks); we do not test open-ended or subjective objectives (helpfulness, safety, style), where reward models matter most and where the well-defined true reward our error-diversity account rests on is itself contested. Second, the reward lever needs *several* reward models with *diverse* errors: with one model it reduces to the capped optimizer lever, and under correlated errors it does not help and can hurt (our code result, -5.06 on MBPP, §5); we give a mechanistic criterion but no label-free test to certify diversity in advance. Third, our claim is label-free *robustness*, not *dominance*: VaR matches or exceeds a *typical* single model but not the oracle-best one (mean 66.10 versus 58.36), so a practitioner with even a small labeled set might prefer a single strong model. Finally, the aggregator is a deliberately simple, unlearned heuristic, an empirical lower α-quantile (Value-at-Risk) on a coarse grid $\alpha \in \{10\%, 25\%\}$; it carries a never-worse-than-uniform guarantee up to selection noise but no optimality proof or bound on residual hacking, and a learned or reliability-weighted aggregator may do better.

Broader impact. On the positive side, the reward lever improves inference-time correctness without training, without true-reward labels, and without a GPU for the method itself, reusing openly available reward models; it lowers the barrier to better-aligned deployment for academics and smaller actors, reduces reliance on any single gameable reward model, and, by flattening the hacking curve, lessens the risk of catastrophic over-optimization. Our falsifiable when-it-works criterion and upfront reporting of the code failure are meant to encourage scoped rather than blind use. On the negative side, the technique makes it easier to optimize *whatever* objective a reward suite encodes without making that objective aligned: applied to a harmful or biased proxy it would more reliably reach a harmful target, a standard dual-use concern. A risk specific to conservative aggregation is that it rewards a response only when a skeptical majority of models agrees, so a suite sharing systematic biases (verbosity, style, cultural or demographic priors) could entrench that consensus and suppress valid minority-style outputs, a fairness consideration we flag. Finally, scoring several reward models raises inference compute and energy (though a diverse handful suffices, Appendix G), and the flattening result is no licence to skip other safety checks or to deploy in the correlated-error regimes where the method is known to fail.