Cost-Aware SGD: Theory and Applications to LLM Policy Optimization

Clara Mohri[*]
Kempner Institute
Harvard University

Amir Globerson
Google Research
Tel Aviv University

Haim Kaplan
Google Research
Tel Aviv University

Tomer Koren
Google Research
Tel Aviv University

Yishay Mansour
Google Research
Tel Aviv University

Abstract

We consider the problem of Cost-Aware Learning, where sampling different components of a finite-sum objective incurs different costs. The objective is to reach a target error while minimizing the total cost. We propose Cost-Aware SGD, which uses a distribution based on gradient norms and costs to sample components. We provide a thorough analysis of this algorithm, including cost-improvement bounds over baselines, a characterization of distribution proxy sub-optimality, and a lower bound. We apply our theoretical insights to reinforcement learning with language models, where the cost of including a sequence in a policy-gradient update scales with its length. We find that advantage magnitude is a high-fidelity proxy for sequence-level gradient norms, motivating Cost-Aware GRPO. Across 1.5B, 4B, and 8B LLMs, Cost-Aware GRPO matches or exceeds baseline accuracy while using up to 33% fewer policy-optimization FLOPs and 33% fewer total FLOPs.

1 Introduction

Stochastic gradient descent (SGD) [Nemirovsky et al., 1983] theory typically measures convergence in terms of the number of gradient steps required to reach a target error. This implicitly assumes that gradient evaluation on any data point incurs an identical computational cost. However, in modern language model training, the assumption of uniform cost is not always true. For example, in reinforcement learning for language models [Ouyang et al., 2022, Shao et al., 2024], the FLOPs required to compute a policy gradient scale linearly with sequence length, meaning a long reasoning trace can cost an order of magnitude more than a concise response.

Motivated by this computational heterogeneity, we study the problem of *Cost-Aware Learning*: given a finite-sum objective where each component has a known evaluation cost c_i, how should one sample gradients to reach ϵ-error at a minimum total cost? We first propose Cost-Aware SGD, which leverages importance sampling [Kloek and Van Dijk, 1978, Beygelzimer et al., 2009] to derive the optimal sampling distribution $p_i^* \propto G_i/\sqrt{c_i}$, where G_i and c_i are the Lipschitz constant of the gradient and the cost of component i respectively.

We analyze the cost-improvement over uniform and variance-reduction baselines, and validate this in a synthetic setting. Since exact gradient norms are often unavailable in practice, we characterize the sub-optimality of using a proxy distribution. We establish a near-matching lower bound for cost-aware learning, which motivates a subset selection algorithm. Finally, we show that the derived sampling strategy extends beyond convex objectives.

[*]Work done while Mohri was a Student Researcher at Google Research. Correspondence to: cmohri@g.harvard.edu.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

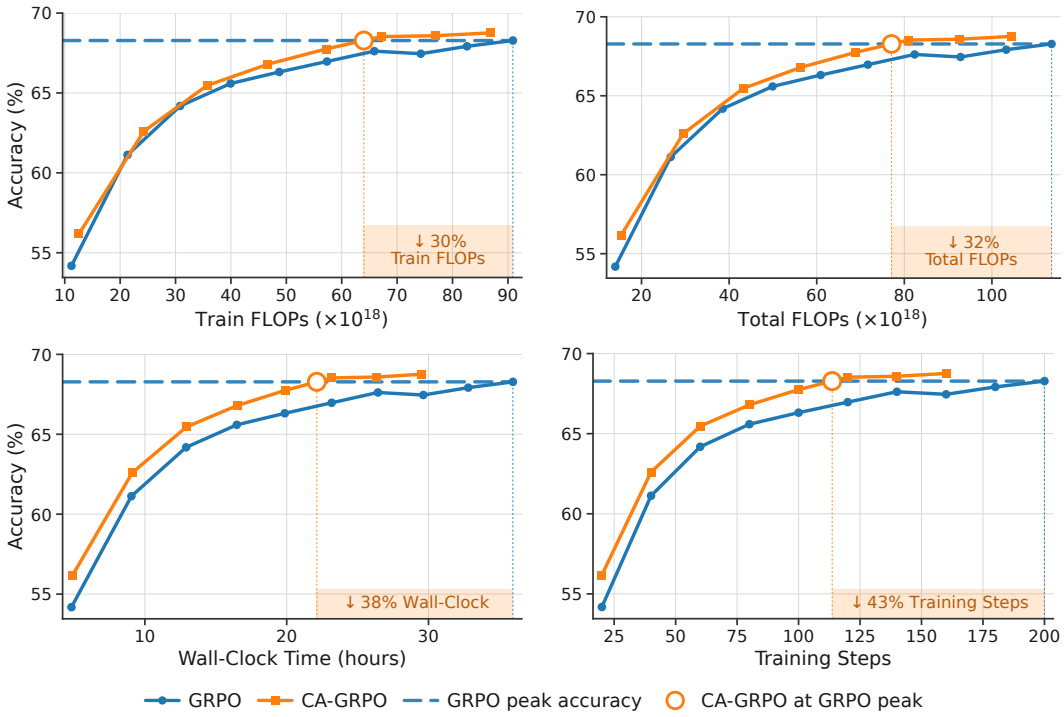


Figure 1: **Qwen3-8B training with GRPO vs. Cost-Aware GRPO (CA-GRPO).** We evaluate on AIME1983-2024 (`pass@1/mean@32`) and plot accuracy against four measures of training cost: training FLOPs used in policy optimization, total FLOPs including generation, wall-clock time, and training steps. We evaluate every 20 training steps until peak accuracy is attained. To match the peak accuracy of GRPO, CA-GRPO requires 30% fewer training FLOPs, 32% fewer total FLOPs, 38% less wall-clock time, and 43% fewer training steps.

We apply these insights to the policy optimization phase of GRPO [Shao et al., 2024], where sequence length provides a natural cost. A key empirical finding, motivated by the form of the policy gradient, is that the advantage magnitude $|A_i|$ serves as a high-fidelity proxy for G_i based on our sub-optimality metrics. Equipped with this proxy, we introduce Cost-Aware GRPO, which samples model-generated sequences according to p^* with $|A_i|$ in place of G_i during. Cost-Aware GRPO significantly reduces the FLOPs used in policy optimization while matching or exceeding baseline performance. While the direct application is to reduce training FLOPs, we find that Cost-Aware GRPO is effective in reducing cumulative FLOPs, wall-clock time, and training steps by similar margins. Through extensive ablations, we demonstrate robustness to proxy noise, hyperparameter choices, and show generalization beyond GRPO to the CISPO objective [Chen et al., 2025].

The main contributions of this paper are as follows:

- **Cost-aware SGD and theoretical analysis.** We derive the optimal cost-aware sampling distribution for convex and strongly convex finite-sum optimization, analyze its cost-improvement over baselines, characterize the sub-optimality of using a proxy distribution, and establish an information-theoretic lower bound.
- **Applications to GRPO.** We propose the Cost-Aware GRPO algorithm. We identify $|A_i|$ as a high-fidelity proxy for gradient norms and apply cost-aware sampling to GRPO. Experiments with Qwen2.5-Math-1.5B-Instruct [Yang et al., 2024], Qwen3-4B and Qwen3-8B [Yang et al., 2025] demonstrate consistent reductions in resource consumption without compromising on performance.

2 Related Work

We describe the primary related work, and defer the discussion of additional works to Appendix A.

Importance sampling for SGD. A significant body of work focuses on accelerating SGD convergence via importance sampling, primarily through variance reduction [Zhao and Zhang, 2015, Needell et al., 2014]. A foundational result by Alain et al. [2015] established that the optimal sampling distribution for variance reduction assigns probabilities proportional to the per-sample gradient norms. In the context of deep learning, where exact gradient norms are expensive to compute, methods have been proposed to approximate these norms using upper bounds or loss values [Katharopoulos and Fleuret, 2018, Johnson and Guestrin, 2018]. However, these approaches operate under a standard optimization framework that treats iterations as the unit of resource consumption. They aim to maximize the information gain *per step*, implicitly assuming uniform computational cost across samples. Our work generalizes this by considering the *cost-aware* setting where the aim is to minimize training cost, leading to a sampling distribution that balances gradient magnitude against sample cost.

In particular, the work of Chen et al. [2023b] shares similarities with our work by also studying a finite-sum setting with varying sample costs. However, our contributions differ significantly in analysis and scope: while their work focuses on asymptotic behavior, we provide finite-time guarantees complemented by a lower bound. Furthermore, we implement this cost-aware framework with language models via Cost-Aware GRPO.

RL for Language Models. While policy gradient methods [Sutton et al., 1999, Kakade, 2001, Schulman et al., 2015] have been used in many settings, the works of Ouyang et al. [2022], Dai et al. [2023] have garnered significant attention for their application to LLM post-training. Various methods such as DPO [Rafailov et al., 2023], PPO [Schulman et al., 2017, Zheng et al., 2023], and GRPO [Shao et al., 2024] have been shown to be extremely effective in improving accuracy on standard benchmarks. However, these methods incur substantial computational and memory costs. Several methods aim to improve data efficiency: [Wang et al., 2025, Fatemi et al., 2025, Li et al., 2025, Yu et al., 2025, Xu et al., 2025, Zheng et al., 2025]. Yu et al. [2025] introduces, among other contributions, a mechanism to continue resampling from a model until a sufficient amount of non-zero advantage responses have been generated, showing that this data helps the model achieve better performance. Xu et al. [2025] attempts to subsample the generations of a model in an effort to train on less data. Zheng et al. [2025] keeps track of the history of various prompts in an effort to remove low-signal prompts. Our work contrasts with these by using importance reweighting and taking into account the computational cost of sequences when making decisions.

3 Cost-Aware SGD

3.1 Problem Setup

We consider minimizing a convex function f in the finite-sum setting where $f(x) = n^{-1} \sum_{i=1}^{n} f_i(x)$. We assume that f is minimized at x^*, and that each component f_i is a convex and G_i-Lipschitz function. In the cost-aware setting, each component i has an evaluation cost $c_i \geq 0$ for querying its gradient. Namely, upon querying sample i at a point x, the algorithm obtains the exact gradient $\nabla f_i(x)$ and incurs cost c_i. We assume that the learning algorithm has access to G_i and c_i for all samples i. For a given $\epsilon > 0$, the objective of a (possibly randomized) cost-aware learning algorithm is to incur the minimum possible cost to obtain $\hat{x}$ such that $\mathbb{E}[f(\hat{x}) - f(x^*)] \leq \epsilon$.

3.2 Cost-Aware SGD

We propose Cost-Aware SGD, a simple variant of standard SGD, which differs only in the use of a non-uniform sampling distribution and importance weighting to account for bias. Pseudocode is provided in Appendix 1. While the analysis in this section assumes convex loss functions in order to provide meaningful insights, in Appendix F we show that the optimal sampling distribution derived does not strictly rely on convexity.

Consider a fixed sampling distribution $\mathsf{p} \in \Delta_n$. The total cost of the algorithm over T iterations is the random variable $\mathcal{K}_T$, whose expected total cost is:

$$\mathbb{E}[\mathcal{K}_T] = \mathop{\mathbb{E}}_{i_t \sim \mathsf{p}} \left[\sum_{t=1}^{T} c_{i_t} \right] = \sum_{t=1}^{T} \mathop{\mathbb{E}}_{i_t \sim \mathsf{p}} [c_{i_t}] = T \sum_{i=1}^{n} p_i c_i = T \cdot C(\mathsf{p}), \tag{1}$$

where $C(\mathsf{p}) = \sum_{i=1}^{n} p_i c_i$ represents the expected cost per iteration. Our goal is to minimize the total expected cost required to reach a target error ϵ. Let $T(\epsilon, \mathsf{p})$ denote the number of iterations required to achieve expected error ϵ using distribution p. Then, we seek to find p which minimizes:

$$\mathcal{K}(\epsilon, \mathsf{p}) = T(\epsilon, \mathsf{p}) \cdot C(\mathsf{p}). \tag{2}$$

Each component f_i is G_i-Lipschitz, so $\|\nabla f_i(x)\| \leq G_i$ for every x. For a sampling distribution p, let $S(\mathsf{p})$ denote the second moment of the importance-weighted stochastic gradient estimator. Under importance sampling, this is:

$$S(\mathsf{p}) = \sum_{i=1}^{n} p_i \left\| \frac{\nabla f_i(x)}{n p_i} \right\|^2 \leq \frac{1}{n^2} \sum_{i=1}^{n} \frac{G_i^2}{p_i}.$$

For general convex functions, standard convergence results for SGD [Nemirovsky et al., 1983, Bubeck et al., 2015] state that with step size $\eta \propto 1/\sqrt{T}$, the expected suboptimality satisfies:

$$\mathbb{E}[f(\overline{x_T}) - f(x^*)] \leq \frac{D\sqrt{S(\mathsf{p})}}{\sqrt{T}},$$

where D is the diameter of the parameter space and $\overline{x_T}$ is the averaged iterates over all time steps. Solving for T to attain expected error ε, and substituting this into the cost formula:

$$\mathcal{K}_c(\epsilon, \mathsf{p}) = \left(\frac{D^2 S(\mathsf{p})}{\epsilon^2} \right) C(\mathsf{p}) \propto S(\mathsf{p}) C(\mathsf{p}). \tag{3}$$

If f is μ-strongly convex, SGD with decaying step size $\eta_t = 1/(\mu t)$ achieves a rate of $O(S(\mathsf{p})/\mu T)$ [Rakhlin et al., 2011]. The expected total cost under distribution p to attain error ϵ is:

$$\mathcal{K}_\mu(\epsilon, \mathsf{p}) = \left(\frac{4 S(\mathsf{p})}{\mu \epsilon} \right) C(\mathsf{p}) \propto S(\mathsf{p}) C(\mathsf{p}). \tag{4}$$

In both cases, minimizing the total cost is equivalent to minimizing the product $J(\mathsf{p}) = S(\mathsf{p})C(\mathsf{p})$.

3.3 Cost-Aware Sampling Distribution

Theorem 3.1 (Optimal Cost-Weighted Distribution)**.** *The sampling distribution p^* that minimizes $\mathcal{K}(\epsilon, p)$ is:*

$$p_i^* \propto G_i / \sqrt{c_i}. \tag{5}$$

Under this distribution, the expected costs for general convex and μ-strongly convex functions are:

$$\mathcal{K}_c(\epsilon, \mathsf{p}^*) = \frac{D^2}{\epsilon^2} \left(\frac{1}{n} \sum_{i=1}^{n} G_i \sqrt{c_i} \right)^2, \qquad \mathcal{K}_\mu(\epsilon, \mathsf{p}^*) = \frac{4}{\mu \epsilon} \left(\frac{1}{n} \sum_{i=1}^{n} G_i \sqrt{c_i} \right)^2. \tag{6}$$

Proof. We minimize the product term $J(\mathsf{p}) = S(\mathsf{p})C(\mathsf{p})$. By the Cauchy-Schwarz inequality, for vectors $u, v \in \mathbb{R}^n$ defined by $u_i = \sqrt{p_i c_i}$ and $v_i = G_i / \sqrt{p_i}$:

$$\left(\sum_{i=1}^{n} p_i c_i \right) \left(\sum_{i=1}^{n} G_i^2 {p_i}^{-1} \right) = \|u\|^2 \|v\|^2 \geq \langle u, v \rangle^2 = \left(\sum_{i=1}^{n} G_i \sqrt{c_i} \right)^2. \tag{7}$$

Equality holds if and only if $u \propto v$, which implies $\sqrt{p_i c_i} \propto G_i / \sqrt{p_i}$, or $p_i \propto G_i / \sqrt{c_i}$. Substituting the minimum value of the product back into $J(\mathsf{p})$, we obtain $J(\mathsf{p}^*) = \frac{1}{n^2} (\sum G_i \sqrt{c_i})^2$. Substituting $J(\mathsf{p}^*)$ into Eq. 3 and Eq. 4 yields the stated costs. □

3.4 Cost-Improvement over Baselines

We compare sampling with p^* against two baselines. Uniform sampling sets $(p_{\text{unif}})_i = 1/n$. The variance strategy ignores costs and samples proportionally to gradient norms: $(p_{\text{var}})_i = G_i / \sum_j G_j$, which minimizes the variance of the gradient estimator [Alain et al., 2015, Zhao and Zhang, 2015,

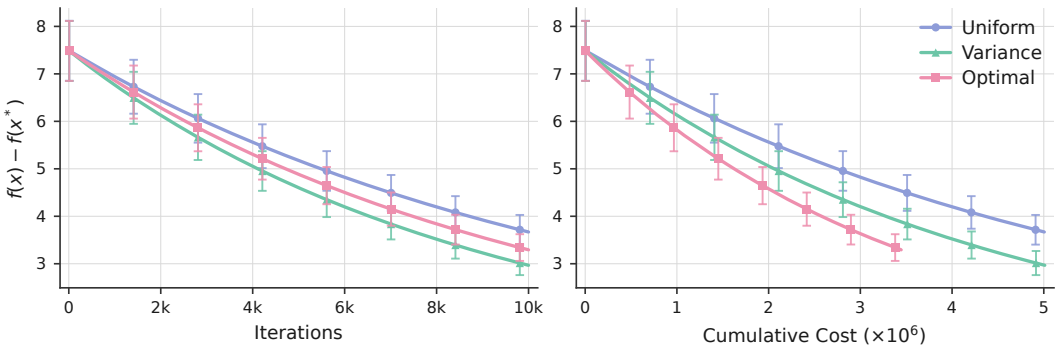


Figure 2: **Synthetic validation.** We compare the error with the total training steps and the total cost incurred by each sampling strategy. For a fixed error target, the cost-aware sampling distribution incurs the least total cost. On the other hand, the variance strategy converges the fastest in terms of iteration count. This is expected, as it minimizes the gradient estimator's variance without regard for computational budget.

Needell et al., 2014] (see Appendix D.1 for a derivation). The cost ratios relative to optimal sampling are:

$$\frac{\mathcal{K}(\epsilon, \mathsf{p}^*)}{\mathcal{K}(\epsilon, p_{\text{unif}})} = \frac{(\sum G_i\sqrt{c_i})^2}{(\sum G_i^2)(\sum c_i)} \leq 1, \qquad \frac{\mathcal{K}(\epsilon, \mathsf{p}^*)}{\mathcal{K}_{\text{var}}(\epsilon, p_{\text{var}})} = \frac{(\sum G_i\sqrt{c_i})^2}{(\sum G_i)(\sum_i G_i c_i)} \leq 1. \tag{8}$$

When all costs are equal, $\mathcal{K}_{\text{opt}} = \mathcal{K}_{\text{var}}$, but $\mathcal{K}_{\text{opt}}$ still improves over $\mathcal{K}_{\text{unif}}$ by minimizing variance (see Appendix D.1 for a detailed comparison). When costs and gradients are negatively correlated, the gains can be dramatic: setting $G_i = 2^{-i/4}$, $c_i = 2^i$ yields $\mathcal{K}_{\text{opt}}/\mathcal{K}_{\text{unif}} = \Theta(2^{-n/2})$ and $\mathcal{K}_{\text{opt}}/\mathcal{K}_{\text{var}} = \Theta(2^{-n/4})$.

In a synthetic setting, we evaluate the sampling strategies on a linear least squares task with $N = 3000$ points in dimension $d = 50$. The data vectors a_i are scaled such that their norms are bounded by $L = 10$. This geometry ensures that the Lipschitz constants G_i are bounded but heterogeneous across the dataset. Evaluation costs c_i are assigned randomly from a uniform distribution between 1 and 1000. We compare three sampling approaches: *Uniform*, *Variance* ($p_i \propto G_i$), and *Optimal* ($p_i \propto G_i/\sqrt{c_i}$). The results are shown in Figure 2. We average over 1000 trials and report standard deviation error bars.

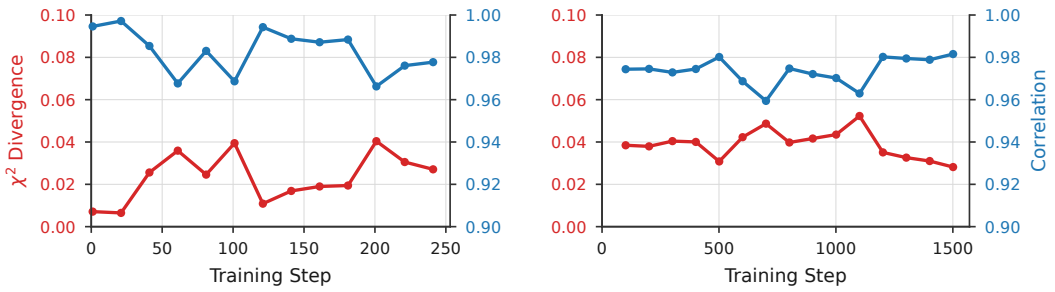
3.5 Proxy Sub-Optimality

In practice, it can be prohibitively costly to compute G_i. As such, one may not have access to the perfect sampling distribution p^*; in fact, this is the case for later experiments in this paper. In this subsection, we consider the sub-optimality of using a proxy for G_i. Recall that $J(\mathsf{p}) = S(\mathsf{p})C(\mathsf{p})$, and that $\mathcal{K}(\epsilon, \mathsf{p})$ is the expected cost to attain error ϵ under distribution p. Hence, the quantity $J(p')/J(p^*)$ represents the ratio between the cost incurred by p' and that of p^*.

For two distributions p and q, let $\chi^2(p\|q) = \sum_{i=1}^n p_i^2/q_i - 1$ denote their χ^2 divergence. In Theorem 3.2, we exactly characterize the sub-optimality due to using a different sampling distribution p' in place of p^*.

Theorem 3.2 (Sub-optimality gap for distribution proxy)**.** *Let p' be any empirical sampling distribution. We define the cost-biased distribution $\tilde{p}$ corresponding to p as: $\tilde{p}_i = (p_i c_i)/C(p)$. The sub-optimality gap is: $J(p')/J(p^*) = 1 + \chi^2(\tilde{p}^*||\tilde{p}')$.*

To provide further intuition as to when one can use some proxy G_i' in place of G_i for a proxy sampling distribution, we provide Theorem D.5 (Appendix D.5). It implies that, when G_i and G_i' have near-1 Pearson correlation, $\mathbb{E}[J(p')] \approx J(p^*)$. The Pearson correlation between random variables X and Y with expected values μ_X, μ_Y, and standard deviations σ_X, σ_Y, respectively, is defined as: $\rho_{X,Y} = \frac{\text{Cov}(X,Y)}{\sigma_X\sigma_Y}$. Both proofs are deferred to Appendix D.5.



(a) Qwen3-8B

(b) Qwen2.5-Math-1.5B-Instruct

Figure 3: **Sub-optimality metrics** for using $|A_i|$ in place of G_i for GRPO experiments across two model sizes. The Pearson correlations are near 1. The cost-biased χ^2-divergence between the true p^* and the distribution defined by this proxy is near 0.

3.6 Limits of Sampling

Having established the upper bounds for Cost-Aware SGD, a natural question is whether it is possible to design a sampling strategy that achieves an even lower total cost. In this section, we provide a lower bound for the class of convex functions.

Theorem 3.3. *Fix $\epsilon, G > 0$. For any number of components $n \geq G^2/\epsilon^2$, any set of nonnegative query costs $\{c_i\}_{i=1}^n$, and any (possibly randomized) algorithm with an expected error guarantee $\mathbb{E}[F(\hat{x}) - \min F(x)] \leq \epsilon$, there exists a convex, G-Lipschitz (i.e., with $G_i = G$ for all i) finite-sum problem instance over $\mathcal{X} = [-1, 1]$ such that the algorithm's expected total query cost is at least:*

$$\Omega\left(\frac{G^2}{\epsilon^2}\left(\frac{1}{n}\sum_{i\in S^*}\sqrt{c_i}\right)^2\right),$$

where $S^* \subseteq [n]$ is any set of components such that following cost-uniformity condition holds: $(\max_{i\in S^*}\sqrt{c_i})/(\frac{1}{n}\sum_{j\in S^*}\sqrt{c_j}) \leq \frac{G}{40\epsilon}$.

We defer the proof to Appendix E. The lower bound matches the upper bound from Theorem 3.1 in its dependence on $n^{-2}\epsilon^{-2}$, but differs in how it aggregates costs: the upper bound scales with $(\sum_{i=1}^n \sqrt{c_i})^2$, while the lower bound scales with $(\sum_{i\in S^*}\sqrt{c_i})^2$. For uniform costs, $S^* = [n]$ and the bounds match up to constants. When costs are highly non-uniform, the gap arises because expensive components are excluded from S^*. This motivates combining importance-weighted sampling with *subset selection*: first choosing which components to include, then optimizing sampling probabilities over the selected subset. We develop this approach in Appendix C, where we show that the subset selection problem reduces to a min-cost knapsack problem, for which a polynomial-time 2-approximation exists that simply selects the cheapest components.

4 Cost-Aware GRPO

We apply our theoretical analysis to GRPO [Shao et al., 2024]. Our direct objective is to reduce *training FLOPs*: computation spent in the policy-optimization stage. Since each optimizer step also requires generating rollouts, we additionally evaluate *total FLOPs*, which represents training FLOPs and rollout-generation FLOPs. For completeness, we also report wall-clock time and optimizer steps.

4.1 Experimental Setup

We perform experiments with the Qwen2.5-Math-1.5B-Instruct [Yang et al., 2024], Qwen3-4B and Qwen3-8B [Yang et al., 2025] models. We train with the DAPO dataset [Yu et al., 2025], designed for improving accuracy on AIME problems. Thus, we primarily evaluate on AIME1983-2024 [Veeraboina, 2023]. For further evaluation, we also use AMC [Art of Problem Solving, n.d.], MATH500 [Hendrycks et al., 2021], and GSM8K [Cobbe et al., 2021]. All experiments are performed using the Verl framework [Sheng et al., 2025], and we make only slight modifications for the purpose of our algorithm. Hyperparameters for all experiments are reported in Appendix G.

Training settings. We evaluate under two training settings: standard GRPO and GRPO with zero-variance filtering (ZVF) [Yu et al., 2025], which discards zero-advantage prompts and resamples to maintain a fixed batch size. We evaluate all combinations on the 1.5B model and only standard GRPO on the 4B and 8B models, as it outperforms GRPO+ZVF at the smaller scale.

Sampling methods. We compare training on all rollouts without resampling or importance weighting, importance-weighted sampling from p^* (Section 4.2, Algorithm 2), and its smoothed variant $p^*_{\text{smooth}}(\alpha) = (1-\alpha)p^* + \alpha\mathcal{U}$, where $\mathcal{U}$ is uniform.

Metrics. We evaluate all methods by benchmark accuracy and by four measures of cumulative training cost. Train FLOPs count only computation used to optimize the policy on selected prompt–response pairs (the policy-gradient forward/backward pass and the reference-policy log-probability pass). Total FLOPs additionally include the computation used to generate all rollouts, including rollouts that are generated but not selected for policy optimization. We also separately report wall-clock time and the number of optimizer steps. All reported accuracies denote `pass@1/mean@32`, the average correctness of querying the model 32 times. In all figures, we plot up to peak accuracy. Appendix G.6 details the full FLOP accounting.

4.2 Cost-Aware Learning applied to GRPO

Background on GRPO. In GRPO, training alternates between two stages. The first stage generates many responses for each prompt in order to build a dataset for the second stage. The second stage is where gradient updates are made. The policy gradient is computed on the current set of prompts and responses.

We assume the training dataset has "verifiable rewards," such as a math dataset where the answers are known. The reward is 1 if the response is correct and 0 otherwise. Given M responses for a prompt, advantages are normalized: $A_i = (r_i - \frac{1}{M}\sum_{k=1}^{M} r_k)/(\text{std}(\{r_k\}_{k=1}^{M}))$. One of the key benefits of GRPO-style algorithms is that the advantage is very cheap to obtain.

Defining c_i. The notion of cost fits well in this setting because different prompt and response pairs have different lengths. The cost corresponds to the incremental *training* computation of using a prompt–response pair in the policy-gradient update. Since policy-optimization FLOPs are proportional to the number of tokens processed, we define c_i as the prompt-plus-response token count. Thus, sampling according to p^* directly targets reducing the training FLOPs.

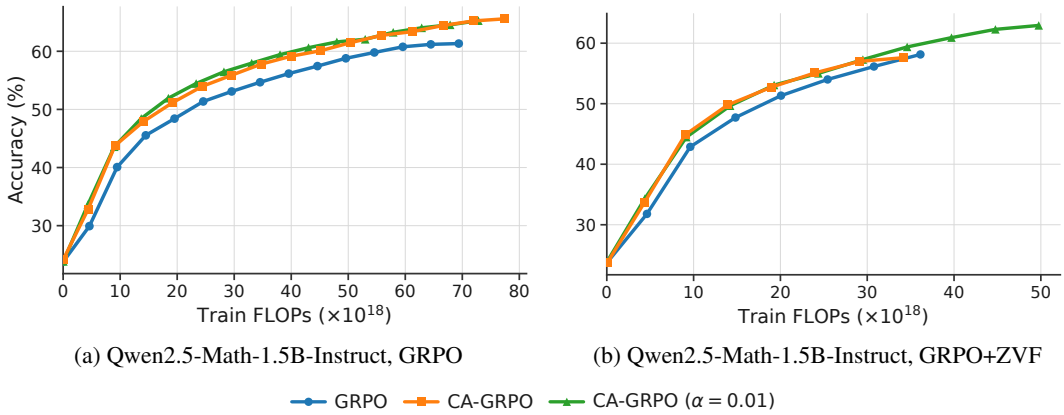
Proxy for G_i. It remains to find a strong proxy for G_i; recall that G_i represents the norm of the gradient of a sample. In language model training, it would not be logical to compute the norm of the gradient of each sample in order to obtain an optimal sampling distribution, as this has a very high computational burden and defeats the purpose. Crucially, *we estimate G_i with $|A_i|$*, the magnitude of the advantage of sample i. This is motivated by the expression for the norm of the policy gradient:

$$G_i = \|\nabla_\theta \ell(o_i;\theta)\| = |A_i| \cdot \left\| \sum_{t=1}^{|o_i|} c_{i,t}(\theta)\, \rho_{i,t}(\theta)\, \nabla_\theta \log \pi_\theta(o_{i,t} \mid q, o_{i,<t}) \right\|,$$

where $\rho_{i,t}(\theta) = \pi_\theta(o_{i,t} \mid q, o_{i,<t})/\pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})$ is the importance ratio and $c_{i,t}(\theta) = \mathbf{1}\{\text{sgn}(A_i)(\rho_{i,t}(\theta) - 1) < \varepsilon\}$ indicates whether clipping is inactive. Hence G_i is the product of $|A_i|$ and an importance-weighted, clipping-masked score-function norm. Once advantages have been computed, $|A_i|$ is available at no additional cost.

Empirically, we observe low cost-biased χ^2-divergence between the sampling proxy obtained this way and the true p^* distribution (invoking Theorem 3.2), as well as high Pearson correlation between G_i and $|A_i|$ (invoking Theorem D.5). In Figure 3, we report these metrics between $|A_i|$ and G_i across training for GRPO with the 8B and 1.5B models. In order to obtain G_i, we compute true sequence-level gradients. Upon computing the true gradient norms, we compute the cost-biased χ^2 divergence between the distributions obtained using $|A_i|$ and G_i respectively. We report sub-optimality metrics for all training settings in Appendix G.10.

Equipped with notions of cost and a gradient proxy, we combine cost-aware learning with GRPO in the policy-gradient stage of training. When we perform the policy gradient steps, we fill mini-batches according to the p^* distribution rather than simply using all samples once. The p^* distribution is obtained by letting $p_i^* \propto |A_i|/\sqrt{c_i}$. Thus, computing p^* only requires obtaining the advantage for all



(a) Qwen2.5-Math-1.5B-Instruct, GRPO

(b) Qwen2.5-Math-1.5B-Instruct, GRPO+ZVF

Figure 4: **Accuracy vs. Train FLOPs** on AIME throughout training for both GRPO and GRPO+ZVF settings for the 1.5B model. We plot the accuracy on the y-axis and the cumulative training FLOPs on the x-axis. We evaluate every 100 steps. CA-GRPO performs as intended, directly reducing the training FLOPs needed.

Model	Setting	Method	Total FLOPs	Train FLOPs	Wall-Clock	Steps
Math-1.5B-IT	GRPO	p^*	28.1%	28.2%	27.1%	29.4%
	GRPO	$p^*_{\text{smooth}}(\alpha = 0.01)$	**32.7%**	**32.8%**	**31.8%**	**30.6%**
	ZVF	p^*	—	—	—	—
	ZVF	$p^*_{\text{smooth}}(\alpha = 0.01)$	**10.8%**	**12.6%**	**9.4%**	**8.5%**
Qwen3-4B	GRPO	p^*	22.2%	19.3%	26.0%	33.5%
	GRPO	$p^*_{\text{smooth}}(\alpha = 0.01)$	**29.5%**	**26.9%**	**32.6%**	**41.1%**
Qwen3-8B	GRPO	p^*	**32.2%**	**29.6%**	**38.5%**	**43.2%**
	GRPO	$p^*_{\text{smooth}}(\alpha = 0.01)$	−0.2%	−4.1%	10.3%	16.7%

Table 1: **Cost-aware sampling reduces FLOPs, wall-clock time, and training steps to reach baseline peak accuracy.** We report the reduction in total FLOPs, training FLOPs, wall-clock time, and training steps to attain baseline peak accuracy.

samples. We use importance sampling to mitigate bias. Aside from this, we make no changes to the GRPO pipeline. The exact procedure is detailed in Algorithm 2.

4.3 Resource Reduction

In Figure 4, we report accuracy as a function of training FLOPs at the 1.5B scale. We compare the resources required to reach each baseline's peak AIME accuracy in Table 1. Cost-aware sampling reduces policy-optimization FLOPs across all evaluated model scales, with savings also reflected in total FLOPs, wall-clock time, and training steps. Smoothing improves resource savings for the 1.5B and 4B models, whereas unsmoothed p^* performs best for the 8B model. Under GRPO+ZVF, only the smoothed variant reaches the baseline's peak accuracy. FLOP accounting is detailed in Appendix G.

These efficiency gains generally preserve or improve benchmark accuracy (Table 2); under standard GRPO, both cost-aware variants improve average accuracy across all three model scales.

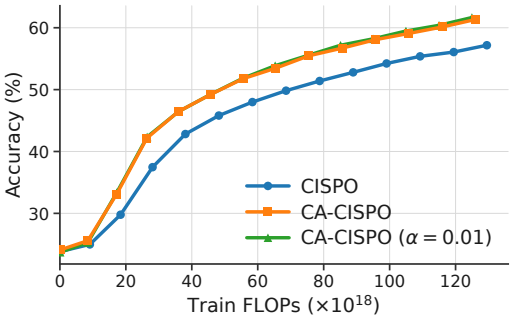
Lastly, Table 2 reports the best checkpoint accuracy for all models and methods. Crucially, these results demonstrate that cost-aware sampling preserves model performance with respect to standard GRPO training; in fact, it is even able to improve upon baseline performance.

4.4 Further Insights

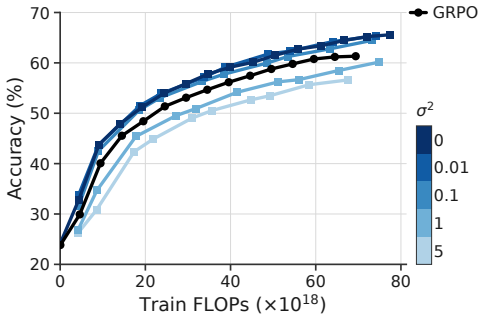
In order to assess the effects of Cost-Aware GRPO on model dynamics, we examine the average response length generated by the model throughout training. We find that, across all experiments,

Model	Setting	Method	AIME	AMC	MATH	GSM8K	Avg. Accuracy
Math-1.5B-IT	GRPO	No sampling	61.3	64.1	**73.2**	**86.2**	71.2
	GRPO	p^*	**65.6**	**71.2**	**73.2**	86.0	**74.0**
	GRPO	$p^*_{\text{smooth}}(\alpha = 0.01)$	65.3	68.1	72.3	86.0	72.9
	ZVF	No sampling	58.1	64.7	**73.2**	**86.1**	70.6
	ZVF	p^*	57.6	63.0	72.9	85.9	69.8
	ZVF	$p^*_{\text{smooth}}(\alpha = 0.01)$	**62.9**	**68.0**	**73.3**	85.8	**72.5**
Qwen3-4B	GRPO	No sampling	66.3	77.9	88.3	93.7	81.6
	GRPO	p^*	66.7	80.0	88.4	93.2	82.1
	GRPO	$p^*_{\text{smooth}}(\alpha = 0.01)$	**67.2**	**80.1**	**88.8**	**94.0**	**82.5**
Qwen3-8B	GRPO	No sampling	68.3	78.6	88.8	**95.1**	82.7
	GRPO	p^*	**68.8**	79.4	**89.0**	**95.1**	83.0
	GRPO	$p^*_{\text{smooth}}(\alpha = 0.01)$	**68.8**	**80.3**	88.6	94.9	**83.2**

Table 2: **Cost-aware sampling preserves model performance across benchmarks.** For each setting and method, we select the best AIME checkpoint and report benchmark accuracy on AMC, MATH, and GSM8k. p^* sampling and variants consistently preserve or improve overall model performance.



(a) **CISPO.** AIME accuracy (`pass@1/mean@32`) for training Qwen2.5-Math-1.5B-Instruct with CISPO. Cost-aware learning is robust to training objectives.



(b) **Noisy proxies.** Math-1.5B with GRPO when adding zero-mean, σ^2-variance noise to the proxy $|A_i|$ prior to computing the sampling distribution.

differences in average response length between the baseline and our method do not exceed 3.5%. The complete results are reported in Table 6.

We also measure the variance in the gradient as a function of the sampling method. We find that cost-aware sampling significantly reduces gradient variance over uniform sampling, and that oracle sampling with full gradient magnitude information would only further reduce gradient variance by at most a factor of 1.14.

4.5 Ablations

Training objective. We consider a variant of the GRPO objective, CISPO Chen et al. [2025]. CISPO applies a stop-gradient to the importance weight of the objective. We base our hyperparameters on findings from Khatri et al. [2025]. Results, reported in Figure 5a, show that our method is not specific to the GRPO objective; rather, it generalizes. p^* sampling and $p^*_{\text{smooth}}(0.01)$ sampling require 31% and 34% fewer tokens respectively to attain the baseline's peak accuracy, and ultimately outperform by about 4pp. This finding is further supported by strong sub-optimality metrics, performance on additional benchmarks, and smoothed variants in Appendix G.9.

Proxy sub-optimality. We ablate the noise level associated with our proxy $|A_i|$. In particular, we add zero-mean noise with variance σ^2 to $|A_i|$ prior to computing the sampling distribution. Results are shown in Figure 5b. We observe that Cost-Aware GRPO is robust to noise up until $\sigma^2 = 0.1$, but as noise approaches $\sigma^2 = 1.0$, it deteriorates with respect to the baseline.

Choice of α and proxy. In Appendix G.8, we ablate our choice of α and report results for p^*_{smooth} with $\alpha = 0.05$ and $\alpha = 0.1$ as well. We also ablate our choice of approximating G_i with $|A_i|$ and

consider using $p_i^* = 1/\sqrt{c_i}$, which corresponds to substituting G_i with 1 for all samples. We observe that it does not perform as well, which implies a need for correctly approximating the gradient norm of a sample. In Appendix G.13, we report smoothed variants for the 4B and 8B models as well.

5 Conclusion

In this paper, we formalize the problem of cost-aware learning and propose Cost-Aware SGD. Our analysis derives the optimal sampling distribution for convex and strongly convex objectives, providing theoretical guarantees on the expected cost to attain an error of ϵ. We extensively analyze the properties of this algorithm.

We apply our theoretical insights to GRPO and introduce Cost-Aware GRPO. Our use of $|A_i|$ as a proxy for G_i in the optimal sampling distribution is supported by strong sub-optimality metrics across scales. Our experiments demonstrate the broad effectiveness of this approach; it significantly reduces the tokens used in policy optimization, while maintaining or exceeding the peak performance of standard baselines.

Future Directions. In Cost-Aware GRPO, we primarily rely on the magnitude of the advantage as a proxy for the gradient norm; identifying superior proxies remains an interesting direction for future research. Furthermore, having demonstrated that cost-aware sampling generalizes successfully from standard GRPO to the CISPO objective, a natural next step is to explore its integration into a broader class of RLVR and RLHF algorithms. It would also be of interest to dynamically set α in smoothed variants of Cost-Aware GRPO in order to obtain superior performance. Lastly, closing the gap between the upper and lower bounds in Section 3.2 would be of theoretical significance.

References

Guillaume Alain, Alex Lamb, Chinnadhurai Sankar, Aaron Courville, and Yoshua Bengio. Variance reduction in sgd by distributed importance sampling. *arXiv preprint arXiv:1511.06481*, 2015.

Art of Problem Solving. Amc problems and solutions. https://artofproblemsolving.com/wiki/index.php/AMC_Problems_and_Solutions, n.d. Accessed: 2026-01-28.

Alina Beygelzimer, Sanjoy Dasgupta, and John Langford. Importance weighted active learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 49–56, 2009.

Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

Aili Chen, Aonian Li, Bangwei Gong, Binyang Jiang, Bo Fei, Bo Yang, Boji Shan, Changqing Yu, Chao Wang, Cheng Zhu, et al. Minimax-m1: Scaling test-time compute efficiently with lightning attention. *arXiv preprint arXiv:2506.13585*, 2025.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, et al. Alpagasus: Training a better alpaca with fewer data. *arXiv preprint arXiv:2307.08701*, 2023a.

Xiangyi Chen, Sijia Liu, Ruoyu Sun, and Mingyi Hong. On the convergence of a class of adam-type algorithms for non-convex optimization. *arXiv preprint arXiv:1808.02941*, 2018.

Ziang Chen, Jianfeng Lu, Huajie Qian, Xinshang Wang, and Wotao Yin. Hetersgd: Tackling heterogeneous sampling costs via optimal reweighted stochastic gradient descent. In Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent, editors, *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pages 10732–10781. PMLR, 25–27 Apr 2023b.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Corinna Cortes, Giulia DeSalvo, Claudio Gentile, Mehryar Mohri, and Ningshan Zhang. Region-based active learning. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2801–2809. PMLR, 2019.

Corinna Cortes, Giulia Desalvo, Claudio Gentile, Mehryar Mohri, and Ningshan Zhang. Adaptive region-based active learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 2144–2153. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/cortes20a.html.

János Csirik, Johannes Bartholomeus Gerardus Frenk, Martine Labbé, and Shuzhong Zhang. Heuristics for the 0-1 min-knapsack problem. *Acta Cybernetica*, 10(1-2):15–20, 1991.

Josef Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, and Yaodong Yang. Safe rlhf: Safe reinforcement learning from human feedback. *arXiv preprint arXiv:2310.12773*, 2023.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35: 16344–16359, 2022.

Sanjoy Dasgupta. Two faces of active learning. *Theoretical computer science*, 412(19):1767–1781, 2011.

Olivier Devolder, Francois Glineur, and Yurii Nesterov. First-order methods of smooth convex optimization with inexact oracle. *Mathematical Programming*, 146(1):37–75, 2014.

Mehdi Fatemi, Banafsheh Rafiee, Mingjie Tang, and Kartik Talamadupula. Concise reasoning via reinforcement learning. *arXiv preprint arXiv:2504.05185*, 2025.

Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM journal on optimization*, 23(4):2341–2368, 2013.

Nicholas J. A. Harvey, Christopher Liaw, Yaniv Plan, and Sikander Randhawa. Tight analyses for non-smooth stochastic gradient descent. In Alina Beygelzimer and Daniel Hsu, editors, *Proceedings of the Thirty-Second Conference on Learning Theory*, volume 99 of *Proceedings of Machine Learning Research*, pages 1579–1613. PMLR, 25–28 Jun 2019. URL https://proceedings.mlr.press/v99/harvey19a.html.

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. *arXiv preprint arXiv:2103.03874*, 2021.

Prateek Jain, Dheeraj Nagaraj, and Praneeth Netrapalli. Making the last iterate of sgd information theoretically optimal. In Alina Beygelzimer and Daniel Hsu, editors, *Proceedings of the Thirty-Second Conference on Learning Theory*, volume 99 of *Proceedings of Machine Learning Research*, pages 1752–1755. PMLR, 25–28 Jun 2019. URL https://proceedings.mlr.press/v99/jain19a.html.

Tyler B Johnson and Carlos Guestrin. Training deep models faster with robust, approximate importance sampling. *Advances in Neural Information Processing Systems*, 31, 2018.

Sham M Kakade. A natural policy gradient. *Advances in neural information processing systems*, 14, 2001.

Angelos Katharopoulos and Francois Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *International conference on machine learning*, pages 2525–2534. PMLR, 2018.

Devvrit Khatri, Lovish Madaan, Rishabh Tiwari, Rachit Bansal, Sai Surya Duvvuri, Manzil Zaheer, Inderjit S Dhillon, David Brandfonbrener, and Rishabh Agarwal. The art of scaling reinforcement learning compute for llms. *arXiv preprint arXiv:2510.13786*, 2025.

Teun Kloek and Herman K Van Dijk. Bayesian estimates of equation system parameters: an application of integration by monte carlo. *Econometrica: Journal of the Econometric Society*, pages 1–19, 1978.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. Limr: Less is more for rl scaling. *arXiv preprint arXiv:2502.11886*, 2025.

Deanna Needell, Nathan Srebro, and Rachel Ward. Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm. *Advances in neural information processing systems*, 27, 2014.

AS Nemirovsky, DB Yudin, and E Dawson. Wiley-interscience series in discrete mathematics, 1983.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. *Advances in neural information processing systems*, 34: 20596–20607, 2021.

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. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Alexander Rakhlin, Ohad Shamir, and Karthik Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. *arXiv preprint arXiv:1109.5647*, 2011.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

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

Ohad Shamir and Tong Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *International conference on machine learning*, pages 71–79. PMLR, 2013.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pages 1279–1297, 2025.

Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari Morcos. Beyond neural scaling laws: beating power law scaling via data pruning. *Advances in Neural Information Processing Systems*, 35:19523–19536, 2022.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999. URL https://proceedings.neurips.cc/paper_files/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf.

Hemish Veeraboina. Aime problem set 1983-2024, 2023. URL https://www.kaggle.com/datasets/hemishveeraboina/aime-problem-set-1983-2024.

Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Liyuan Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, et al. Reinforcement learning for reasoning in large language models with one training example. *arXiv preprint arXiv:2504.20571*, 2025.

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy S Liang. Data selection for language models via importance resampling. *Advances in Neural Information Processing Systems*, 36: 34201–34227, 2023.

Yixuan Even Xu, Yash Savani, Fei Fang, and Zico Kolter. Not all rollouts are useful: Down-sampling rollouts in llm reinforcement learning. *arXiv preprint arXiv:2504.13818*, 2025.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. Why gradient clipping accelerates training: A theoretical justification for adaptivity, 2020. URL https://arxiv.org/abs/1905.11881.

Peilin Zhao and Tong Zhang. Stochastic optimization with importance sampling for regularized loss minimization. In *international conference on machine learning*, pages 1–9. PMLR, 2015.

Haizhong Zheng, Yang Zhou, Brian R Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. Act only when it pays: Efficient reinforcement learning for llm reasoning via selective rollouts. *arXiv preprint arXiv:2506.02177*, 2025.

Rui Zheng, Shihan Dou, Songyang Gao, Yuan Hua, Wei Shen, Binghai Wang, Yan Liu, Senjie Jin, Qin Liu, Yuhao Zhou, et al. Secrets of rlhf in large language models part i: Ppo. *arXiv preprint arXiv:2307.04964*, 2023.

Contents

1 Introduction **1**

2 Related Work **2**

3 Cost-Aware SGD **3**
3.1 Problem Setup . . . 3
3.2 Cost-Aware SGD . . . 3
3.3 Cost-Aware Sampling Distribution . . . 4
3.4 Cost-Improvement over Baselines . . . 4
3.5 Proxy Sub-Optimality . . . 5
3.6 Limits of Sampling . . . 6

4 Cost-Aware GRPO **6**
4.1 Experimental Setup . . . 6
4.2 Cost-Aware Learning applied to GRPO . . . 7
4.3 Resource Reduction . . . 8
4.4 Further Insights . . . 8
4.5 Ablations . . . 9

5 Conclusion **10**

A Related Work **16**

B Algorithms **17**
B.1 Cost-Aware SGD . . . 17
B.2 Cost-Aware GRPO . . . 17

C Subset Selection **18**

D Proofs **19**
D.1 Variance reduction strategy derivation . . . 19
D.2 Cost-Improvement & Comparison Analysis . . . 19
D.3 Min Cost Knapsack . . . 20
D.4 Strongly Convex Subset Selection . . . 21
D.5 Sub-Optimality Proofs . . . 21

E Lower bound **24**

F Extensions and Generality of the Theoretical Analysis **27**
F.1 Differences between Theory and Practice . . . 27
F.2 Generality of the Cost Objective in Non-Convex Settings . . . 27
F.3 Average Iterate ($\bar{x}_T$) vs. Last Iterate (x_T) . . . 27

G Experimental Details and Ablations **28**
G.1 Experimental Details . . . 28
G.2 Implementation . . . 28
G.3 Importance Weighting with GRPO . . . 28
G.4 Hyperparameters . . . 29
G.5 Computational Requirements . . . 30
G.6 FLOPs accounting . . . 30
G.7 Response length analysis . . . 30
G.8 Ablation on variants of p^* . . . 31
G.9 Full CISPO results . . . 32
G.10 Full sub-optimality results . . . 33
G.11 Additional results for Qwen2.5-Math-1.5B-Instruct . . . 34
G.12 Additional results for Qwen3-4B . . . 35
G.13 Additional results for Qwen3-8B . . . 36

A Related Work

We describe further related work.

Training Data Pruning/ Selection. There is a large literature focused on pruning a training dataset. Typically, the objective is to obtain a better predictor or maintain performance with fewer samples. Related works include [Sorscher et al., 2022, Paul et al., 2021, Xie et al., 2023, Chen et al., 2023a]. Our work differs from these as it explicitly considers a variable cost per sample.

Active learning. There is a significant body of work studying the problem of active learning, which considers when to request more information about training samples. We refer the reader to [Dasgupta, 2011] for an in-depth overview of the literature. Most related to our work, Beygelzimer et al. [2009], Cortes et al. [2020, 2019] use importance weighting to probabilistically select samples for which to query a label. Generally, these algorithms fix a hypothesis set a priori, in which hypotheses are eliminated as more data is acquired. Our work shares the similarity of deciding when to request more information, however, in contrast, our work requests the *gradient* of a sample. Furthermore, we do not use a current hypothesis set in order to make these decisions.

B Algorithms

B.1 Cost-Aware SGD

Algorithm 1 Cost-Aware SGD

Input: Initial point x_1, iterations T, step size η, sampling distribution p^*.
for $t = 1$ **to** T **do**
  Sample index $i_t \sim \mathsf{p}^*$
  Query $\nabla f_{i_t}(x_t)$, incur cost c_{i_t}.
  Compute importance-weighted stochastic gradient:
    $\tilde{g}_t = \frac{1}{n p_{i_t}} \nabla f_{i_t}(x_t)$
  Update parameter: $x_{t+1} = \Pi_{\mathcal{X}}(x_t - \eta \tilde{g}_t)$
end for
Output: $\bar{x}_T = \frac{1}{T}\sum_{t=1}^{T} x_t$ (or suffix averaging solution for strongly convex case [Rakhlin et al., 2011])

B.2 Cost-Aware GRPO

Algorithm 2 Cost-Aware GRPO

Input: Initial policy π_θ, Dataset $\mathcal{D}$, Prompt batch size n, Rollouts per prompt M, Mini-batch size B, Iterations K
for iteration $k = 1$ **to** K **do**
  // Phase 1: Data Collection
  Sample prompt batch $X = \{x_1, \ldots, x_n\} \sim \mathcal{D}$
  Generate M outputs per prompt: $y_{i,j} \sim \pi_{\theta_{\text{old}}}(\cdot|x_i)$
  Form rollout dataset $\mathcal{D}_{\text{step}}$ of size $N = n \times M$
  Compute advantages for all samples in $\mathcal{D}_{\text{step}}$
  // Phase 2: Compute Sampling Distribution
  Estimate weights $s_u \leftarrow G_u/\sqrt{c_u}$ for all $u \in \mathcal{D}_{\text{step}}$
  Compute probabilities $p^*_u \leftarrow s_u / \sum_{v \in \mathcal{D}_{\text{step}}} s_v$
  // Phase 3: Importance Weighted Updates
  Set number of updates $T \leftarrow \lceil N/B \rceil$
  for update step $t = 1$ **to** T **do**
    // Sample mini-batch and optimize policy
    Sample mini-batch $\sim$ Multinomial(support $= \mathcal{D}_{\text{step}}$, probs $= p^*$, size $= B$)
    Update π_θ by maximizing $\hat{\mathcal{J}}^{(p^*)}_{\text{GRPO}}(\theta)$ over mini-batch
  end for
end for

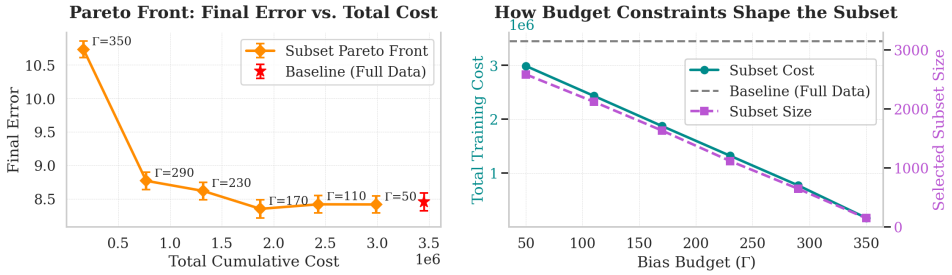


Figure 6: Synthetic validation of the greedy subset selection algorithm. Γ is a tunable parameter representing the allowable bias budget. As it is increased, the greedy algorithm chooses fewer (and cheaper) samples to train on.

C Subset Selection

The importance sampling strategy p^* achieves cost $(\sum_i G_i\sqrt{c_i})^2$, but may remain expensive when high-cost samples have large gradients. To reduce cost, we consider optimizing over a subset $\pi \subseteq \{1,\dots,n\}$, defining $f_\pi(x) = \frac{1}{n}\sum_{i\in\pi} f_i(x)$. This introduces bias, as optimization converges to x^*_π instead of x^*, but reduces computational cost.

Biased SGD on a subset. Let $\|\nabla f_i(x)\| \le G_i$. For a subset π and sampling distribution q supported on π, define the bias $\beta_\pi = \frac{1}{n}\sum_{i\notin\pi} G_i$ and note $\|\nabla f(x) - \nabla f_\pi(x)\| \le \beta_\pi$. Standard SGD results [Devolder et al., 2014] imply that with $\eta_t \propto 1/\sqrt{T}$, $\mathbb{E}[f(\bar{x}_T) - f(x^*)] \le \frac{D\sqrt{\sigma^2_\pi(\mathsf{q})}}{\sqrt{T}} + D\beta_\pi$, where $\sigma^2_\pi(\mathsf{q}) \le \frac{1}{n^2}\sum_{i\in\pi}\frac{G_i^2}{q_i}$. Thus, the cost to achieve error $\epsilon > D\beta_\pi$ is: $\mathcal{K}_\pi(\epsilon) = \frac{D^2}{n^2(\epsilon - D\beta_\pi)^2}\left(\sum_{i\in\pi} G_i\sqrt{c_i}\right)^2$, where we used the optimal sampling distribution restricted to π.

Optimal subset selection. We minimize $\mathcal{K}_\pi(\epsilon)$ subject to a bias budget $D\beta_\pi \le \Gamma$.

Theorem C.1 (Knapsack equivalence)**.** *This problem is equivalent to a min-cost knapsack:*

$$\min_{z_i\in\{0,1\}} \sum_i z_i G_i\sqrt{c_i} \quad s.t. \quad \sum_i z_i G_i \ge \sum_i G_i - \frac{n\Gamma}{D}.$$

We defer the proof to Appendix D.2. A similar result for strongly convex objectives is in Appendix D.4.

Greedy approximation. A standard greedy algorithm selects items by decreasing value-to-weight density $\rho_i = v_i/\omega_i$. In our setting:

$$\rho_i = \frac{v_i}{\omega_i} = \frac{G_i}{G_i\sqrt{c_i}} = \frac{1}{\sqrt{c_i}},$$

so the G_i terms cancel and the greedy order is determined entirely by cost c_i. The algorithm simply selects the cheapest components until the constraint is met, yielding a 2-approximation [Csirik et al., 1991]. Remarkably, the optimal greedy strategy does not depend on the Lipschitz constants G_i at all. Under a similar setting to Section 3.4, we implement this greedy approximation and report results in Figure 6. Within a smaller bias budget, subset selection is able to attain similar error at a significantly reduced cost. As the selected subset becomes much smaller, the error increases due to bias.

D Proofs

In this section we provide the proofs and derivations for statements made in the paper.

D.1 Variance reduction strategy derivation

In this case, we have:

$$J(\mathsf{p}) = S(\mathsf{p})C(\mathsf{p}) = \left(\frac{1}{n^2}\sum_{j=1}^{n}\frac{G_j^2}{p_j}\right)\left(\sum_{i=1}^{n} p_i\right),$$

where $\mathsf{p} \in \Delta_n$. From Cauchy-Schwarz, we can lower bound $J(p)$:

$$J(\mathsf{p}) \geq \frac{1}{n^2}\left(\sum_{i=1}^{n} G_i\right)^2.$$

This lower bound is attained when $G_i/\sqrt{p_i} \propto \sqrt{p_i}$, or equivalently, $p_i \propto G_i$. Hence, when costs are uniform, we focus on minimizing the second moment. This gives:

$$(p_{\text{var}})_i = \frac{G_i}{\sum_j G_j}. \tag{9}$$

When we apply p_{var} to the non-uniform cost setting, the total expected cost to attain error ϵ becomes:

$$\begin{aligned} K_{\text{var}}(\epsilon) &= \frac{D^2}{\epsilon^2 n^2}\left(\sum_{i=1}^{n} G_i\right)^2\left(\sum_{i=1}^{n}(p_{\text{var}})_i c_i\right) \\ &= \frac{D^2}{\epsilon^2 n^2}\left(\sum_{i=1}^{n} G_i\right)^2\left(\sum_{i=1}^{n}\frac{G_i}{\sum_j G_j}c_i\right) \\ &= \frac{D^2}{\epsilon^2 n^2}\left(\sum_{i=1}^{n} G_i\right)\left(\sum_{i=1}^{n} G_i c_i\right). \end{aligned}$$

D.2 Cost-Improvement & Comparison Analysis

Theorem D.1 (Comparison of Adaptive vs. Uniform)**.** *Unlike the optimal strategy, the adaptive strategy p_{var} is not guaranteed to outperform uniform sampling. The ratio of their costs is given by:*

$$\frac{K_{var}(\epsilon)}{K_{unif}(\epsilon)} = \frac{(\sum_{i=1}^{n} G_i)(\sum_{i=1}^{n} G_i c_i)}{(\sum_{i=1}^{n} G_i^2)(\sum_{i=1}^{n} c_i)}. \tag{10}$$

Let $\mathbb{E}[\cdot]$ denote the expectation taken over a uniform distribution of indices $i \sim Uniform(1, \ldots, n)$. Then $K_{var} \leq K_{unif}$ holds if and only if:

$$\mathbb{E}[G]\,\text{Cov}(G, c) \leq \mathbb{E}[c]\,\text{Var}(G). \tag{11}$$

Consequently, if gradients and costs are negatively correlated or uncorrelated, adaptive sampling is superior. However, if gradients and costs are strongly positively correlated (i.e., samples with large gradients are disproportionately expensive), uniform sampling may yield a lower total cost.

Proof. Recall that $K_{\text{unif}} \propto (\sum G_i^2)(\sum c_i)$ and $K_{\text{var}} \propto (\sum G_i)(\sum G_i c_i)$. We can rewrite these sums in terms of expectations over uniformly sampled indices. For any vector x, $\sum x_i = n\,\mathbb{E}[x]$. Thus:

$$\frac{K_{\text{var}}}{K_{\text{unif}}} = \frac{(n\,\mathbb{E}[G])(n\,\mathbb{E}[Gc])}{(n\,\mathbb{E}[G^2])(n\,\mathbb{E}[c])} = \frac{\mathbb{E}[G]\,\mathbb{E}[Gc]}{\mathbb{E}[G^2]\,\mathbb{E}[c]}.$$

We seek the condition under which this ratio is ≤ 1, which implies $\mathbb{E}[G]\,\mathbb{E}[Gc] \leq \mathbb{E}[G^2]\,\mathbb{E}[c]$. We apply the definitions $\mathbb{E}[Gc] = \text{Cov}(G, c) + \mathbb{E}[G]\,\mathbb{E}[c]$ and $\mathbb{E}[G^2] = \text{Var}(G) + \mathbb{E}[G]^2$. Substituting these into the inequality:

$$\mathbb{E}[G](\text{Cov}(G, c) + \mathbb{E}[G]\,\mathbb{E}[c]) \leq \left(\text{Var}(G) + \mathbb{E}[G]^2\right)\mathbb{E}[c].$$

Expanding both sides:

$$\mathbb{E}[G]\,\mathrm{Cov}(G,c) + \mathbb{E}[G]^2\,\mathbb{E}[c] \le \mathbb{E}[c]\,\mathrm{Var}(G) + \mathbb{E}[G]^2\,\mathbb{E}[c].$$

Subtracting the common term $\mathbb{E}[G]^2\,\mathbb{E}[c]$ from both sides yields the final condition:

$$\mathbb{E}[G]\,\mathrm{Cov}(G,c) \le \mathbb{E}[c]\,\mathrm{Var}(G).$$

Since $\mathbb{E}[G]$, $\mathbb{E}[c]$, and $\mathrm{Var}(G)$ are non-negative, this inequality always holds if $\mathrm{Cov}(G,c) \le 0$ (negative or zero correlation). It is violated only if $\mathrm{Cov}(G,c)$ is sufficiently large and positive. □

D.3 Min Cost Knapsack

Theorem D.2 (Equivalence to Min-Knapsack Problem)**.** *Let Γ be the maximum allowable bias budget such that convergence is possible. Finding the optimal subset π^* that minimizes the expected total cost subject to this budget is equivalent to solving the* ***Min-Cost Knapsack Problem****.*

Specifically, let $z_i \in \{0,1\}$ be the indicator variable where $z_i = 1$ if $i \in \pi$. The problem is:

$$\min_z \sum_{i=1}^{n} z_i \omega_i \quad \textit{subject to} \quad \sum_{i=1}^{n} z_i v_i \ge V_{req}, \tag{12}$$

where the "item cost" is $\omega_i = G_i\sqrt{c_i}$, the "item value" is $v_i = G_i$, and the required value coverage is $V_{req} = \sum_{j=1}^{n} G_j - \frac{n\Gamma}{D}$.

Proof. We aim to minimize the cost complexity factor derived for the optimal distribution restricted to π: $J(\pi) = (\sum_{i\in\pi} G_i\sqrt{c_i})^2$. The constraint is on the bias error contribution $D\beta_\pi \le \Gamma < \epsilon$. Recalling the definition of bias:

$$\beta_\pi \le \frac{1}{n}\sum_{j\notin\pi} G_j \implies D\left(\frac{1}{n}\sum_{j\notin\pi} G_j\right) \le \Gamma.$$

Minimizing the sum of gradients *excluded* from π is equivalent to maximizing the sum of gradients *included* in π. We rewrite the constraint:

$$\sum_{j\notin\pi} G_j \le \frac{n\Gamma}{D} \implies \sum_{j=1}^{n} G_j - \sum_{i\in\pi} G_i \le \frac{n\Gamma}{D} \implies \sum_{i\in\pi} G_i \ge \sum_{j=1}^{n} G_j - \frac{n\Gamma}{D}.$$

Therefore, if we let $V_{\text{req}} = \sum_{j=1}^{n} G_j - \frac{n\Gamma}{D}$, let $\omega_i = G_i\sqrt{c_i}$ as the penalty for including item i, and let $v_i = G_i$ as the gain toward satisfying the constraint, we recover the formulation in Equation (12). □

Using this algorithm, the worst case $\mathbb{E}[K(\epsilon)]$, where $\epsilon > \Gamma$, for $D\beta_\pi \le \Gamma$ is:

$$\min_{\{\pi:\beta_\pi\le\Gamma/D\}} \frac{D^2 J(\pi)}{n^2(\epsilon-\Gamma)^2}.$$

The bias constraint $D\beta_\pi \le \Gamma$ determines the required gradient coverage $V_{\text{req}} = \sum_i G_i - n\Gamma/D$, and thus defines a specific Min-Cost Knapsack instance as in Theorem C.1. By varying Γ over its feasible range and solving the associated knapsack problem for each choice, we obtain an optimal subset $\pi(\Gamma)$ minimizing the total cost subject to that bias budget. Since the total expected complexity

$$\mathbb{E}[K_\pi(\epsilon)] = \frac{D^2 J(\pi)}{n^2(\epsilon - D\beta_\pi)^2}$$

depends on both the numerator (the sampling cost) and the denominator (the bias floor), enumerating over Γ allows us to evaluate the achievable error–cost tradeoff for every subset. Consequently, we can compute the optimal target accuracy ϵ by selecting the bias level Γ that yields the smallest overall expected cost.

D.4 Strongly Convex Subset Selection

Theorem D.3 (Subset Selection for Strongly Convex Functions)**.** *Assume f is μ-strongly convex. Let $z_i \in \{0, 1\}$ be the indicator variable for including data point i in subset π. We seek to minimize the cost complexity $\sum z_i G_i \sqrt{c_i}$ subject to a budget Γ on the function value suboptimality $f(x^*_\pi) - f(x^*) \leq \Gamma$.*

This problem is structurally equivalent to the Min-Cost Knapsack formulation in Theorem C.1, with a modified requirement V_{req}. The optimal solution to the LP relaxation is obtained by selecting indices i strictly based on increasing cost c_i.

Proof. Using the Polyak-Lojasiewicz (PL) inequality and the fact that $\nabla f_S(x^*_S) = 0$, the gradient at the proxy minimizer is determined solely by the excluded gradients:

$$\nabla f(x^*_\pi) = \frac{1}{n} \sum_{j \notin \pi} \nabla f_j(x^*_\pi).$$

We bound the norm of this gradient using the triangle inequality ($\|\sum v\| \leq \sum \|v\|$) and the gradient upper bound definition ($\|\nabla f_j(x)\| \leq G_j$):

$$\begin{aligned} \|\nabla f(x^*_\pi)\| &= \left\| \frac{1}{n} \sum_{j \notin \pi} \nabla f_j(x^*_\pi) \right\| \\ &\leq \frac{1}{n} \sum_{j \notin \pi} \|\nabla f_j(x^*_\pi)\| \\ &\leq \frac{1}{n} \sum_{j \notin \pi} G_j. \end{aligned}$$

Substituting this back into the PL inequality ($f(x^*_\pi) - f(x^*) \leq \frac{1}{2\mu} \|\nabla f(x^*_\pi)\|^2$) yields:

$$f(x^*_\pi) - f(x^*) \leq \frac{1}{2\mu n^2} \left(\sum_{j \notin \pi} G_j \right)^2 = \frac{1}{2\mu n^2} \left(\sum_{i=1}^{n} (1 - z_i) G_i \right)^2.$$

We enforce the constraint $f(x^*_\pi) - f(x^*) \leq \Gamma$. Because the gradient norms G_i are non-negative, we can take the square root of the inequality to linearize the constraint without altering the optimization region:

$$\sum_{i=1}^{n} (1 - z_i) G_i \leq n \sqrt{2\mu\Gamma}.$$

This is now a linear constraint on the excluded mass. Let $B = n\sqrt{2\mu\Gamma}$ be the effective budget for the excluded gradients. We rewrite this in terms of the included variables z_i:

$$\sum_{j=1}^{n} G_j - \sum_{i=1}^{n} z_i G_i \leq B \implies \sum_{i=1}^{n} z_i G_i \geq \sum_{j=1}^{n} G_j - B.$$

The optimization problem is to minimize $\sum z_i (G_i \sqrt{c_i})$ subject to $\sum z_i G_i \geq V_{\text{req}}$. This is identical to Equation 12. This problem is structurally equivalent to the Min-Cost Knapsack formulation in Theorem C.1, with a modified requirement V_{req}. Its LP relaxation is a fractional knapsack problem whose optimal solution is obtained by ordering indices by increasing evaluation cost c_i (equivalently, decreasing density v_i/ω_i). □

D.5 Sub-Optimality Proofs

Theorem D.4 (Sub-optimality Gap via Cost-Biased Divergence)**.** *Let p^* be the optimal cost-aware sampling distribution and p' be any empirical sampling distribution. Let the expected cost per iteration for a distribution p be $C(p) = \sum_{i=1}^{n} p_i c_i$.We define the cost-biased distribution $\tilde{p}$ corresponding to p as:*

$$\tilde{p}_i = \frac{p_i c_i}{C(p)}.$$

The optimality gap is exactly determined by the Pearson χ^2-divergence between the cost-biased optimal distribution and the cost-biased empirical distribution:

$$\frac{J(p')}{J(p^*)} = 1 + D_{\chi^2}(\tilde{p}^* || \tilde{p}').$$

Proof. Recall that minimizing the total expected cost is equivalent to minimizing the product $J(p) = S(p)C(p)$. The optimal distribution is defined as $p_i^* = \frac{G_i/\sqrt{c_i}}{Z_{den}}$, where $Z_{den} = \sum_{j=1}^n \frac{G_j}{\sqrt{c_j}}$. We can rearrange this to express the true gradient norms in terms of p_i^*:

$$G_i = Z_{den}\, p_i^* \sqrt{c_i}.$$

Substituting this expression for G_i into the variance term $S(p') = \frac{1}{n^2}\sum_{i=1}^n \frac{G_i^2}{p_i'}$ yields the objective for the empirical distribution:

$$J(p') = \frac{Z_{den}^2}{n^2} C(p') \left(\sum_{i=1}^n \frac{(p_i^*)^2 c_i}{p_i'} \right).$$

Evaluating this same expression for the optimal distribution p^* (where $p_i' = p_i^*$) gives $J(p^*) = \frac{Z_{den}^2}{n^2} C(p^*)^2$. Taking the ratio of the two costs isolates the gap:

$$\frac{J(p')}{J(p^*)} = \frac{C(p') \sum_{i=1}^n c_i \frac{(p_i^*)^2}{p_i'}}{C(p^*)^2}$$

By the definition of the cost-biased distribution, we have $p_i = \tilde{p}_i \frac{C(p)}{c_i}$. Substituting this expression for both p^* and p' into the summation term:

$$\sum_{i=1}^n c_i \frac{(p_i^*)^2}{p_i'} = \sum_{i=1}^n c_i \frac{\left(\tilde{p}_i^* \frac{C(p^*)}{c_i}\right)^2}{\tilde{p}_i' \frac{C(p')}{c_i}}.$$

Simplifying the terms inside the sum:

$$= \sum_{i=1}^n c_i \frac{(\tilde{p}_i^*)^2 \frac{C(p^*)^2}{c_i^2}}{\tilde{p}_i' \frac{C(p')}{c_i}} = \frac{C(p^*)^2}{C(p')} \sum_{i=1}^n \frac{(\tilde{p}_i^*)^2}{\tilde{p}_i'}.$$

Substituting this simplified summation back into our original cost ratio:

$$\frac{J(p')}{J(p^*)} = \frac{C(p')}{C(p^*)^2} \left(\frac{C(p^*)^2}{C(p')} \sum_{i=1}^n \frac{(\tilde{p}_i^*)^2}{\tilde{p}_i'} \right) = \sum_{i=1}^n \frac{(\tilde{p}_i^*)^2}{\tilde{p}_i'}.$$

Using the definition of the Pearson χ^2-divergence, $D_{\chi^2}(P||Q) = \sum \frac{P^2}{Q} - 1$, we arrive at the identity.

Theorem D.5. *Let p^* be the optimal cost-aware sampling distribution using the true gradient norms G_i, and let p' be the empirical sampling distribution using estimated gradient norms G_i', such that $p_i' \propto G_i'/\sqrt{c_i}$.Assume the estimates follow an additive noise model $G_i' = G_i + \epsilon_i$, where ϵ_i are independent, zero-mean noise terms with variance σ_ϵ^2, independent of c_i. Let ρ be the Pearson correlation coefficient between the true norms G_i and the estimates G_i'. Assuming the noise ϵ_i is relatively small compared to G_i, the expected optimality gap is approximately bounded by:*

$$\frac{\mathbb{E}[J(p')]}{J(p^*)} \approx 1 + \left(\frac{1-\rho^2}{\rho^2} \right) \sigma_G^2 \frac{\sum_{i=1}^n \frac{\sqrt{c_i}}{G_i}}{\sum_{i=1}^n G_i \sqrt{c_i}}$$

where σ_G^2 is the variance of the true gradient norms.

Proof. The cost objective for a distribution p is $J(p) = S(p)C(p)$. For the estimated distribution p', we have $p'_i = \frac{G'_i/\sqrt{c_i}}{\sum_{j=1}^n G'_j/\sqrt{c_j}}$. Substituting this into the cost objective yields:

$$J(p') = \frac{1}{n^2}\left(\sum_{i=1}^n \frac{G_i^2\sqrt{c_i}}{G'_i}\right)\left(\sum_{i=1}^n G'_i\sqrt{c_i}\right).$$

Under the model $G'_i = G_i + \epsilon_i$, the covariance between G and G' is strictly the variance of G, namely $Cov(G, G') = \sigma_G^2$. The variance of G' is $Var(G') = \sigma_G^2 + \sigma_\epsilon^2$. The Pearson correlation coefficient ρ is defined as:

$$\rho = \frac{Cov(G, G')}{\sigma_G \sigma_{G'}} = \frac{\sigma_G^2}{\sigma_G\sqrt{\sigma_G^2 + \sigma_\epsilon^2}} = \frac{\sigma_G}{\sqrt{\sigma_G^2 + \sigma_\epsilon^2}}$$

Solving for the noise variance σ_ϵ^2 in terms of ρ:

$$\sigma_\epsilon^2 = \sigma_G^2\left(\frac{1-\rho^2}{\rho^2}\right).$$

Because G'_i appears in the denominator of the second moment, we apply a second-order Taylor expansion of $1/G'_i$ centered around G_i:

$$\frac{1}{G_i + \epsilon_i} \approx \frac{1}{G_i} - \frac{\epsilon_i}{G_i^2} + \frac{\epsilon_i^2}{G_i^3}$$

Substituting this into the first term of $J(p')$ and taking the expectation with respect to the noise ϵ:

$$\mathbb{E}\left[\frac{G_i^2\sqrt{c_i}}{G'_i}\right] \approx G_i^2\sqrt{c_i}\left(\frac{1}{G_i} - \frac{\mathbb{E}[\epsilon_i]}{G_i^2} + \frac{\mathbb{E}[\epsilon_i^2]}{G_i^3}\right) = G_i\sqrt{c_i} + \sigma_\epsilon^2\frac{\sqrt{c_i}}{G_i}.$$

Taking the expectation of the full cost $J(p')$, noting that the two sums are weakly dependent for large n and $\mathbb{E}[G'_i] = G_i$:

$$\mathbb{E}[J(p')] \approx \frac{1}{n^2}\left(\sum_{i=1}^n\left(G_i\sqrt{c_i} + \sigma_\epsilon^2\frac{\sqrt{c_i}}{G_i}\right)\right)\left(\sum_{i=1}^n G_i\sqrt{c_i}\right)$$

$$\mathbb{E}[J(p')] \approx \frac{1}{n^2}\left(\sum_{i=1}^n G_i\sqrt{c_i}\right)^2 + \frac{\sigma_\epsilon^2}{n^2}\left(\sum_{i=1}^n\frac{\sqrt{c_i}}{G_i}\right)\left(\sum_{i=1}^n G_i\sqrt{c_i}\right)$$

Recognizing that $J(p^*) = \frac{1}{n^2}\left(\sum_{i=1}^n G_i\sqrt{c_i}\right)^2$, we divide by $J(p^*)$:

$$\frac{\mathbb{E}[J(p')]}{J(p^*)} \approx 1 + \sigma_\epsilon^2\frac{\sum_{i=1}^n\frac{\sqrt{c_i}}{G_i}}{\sum_{i=1}^n G_i\sqrt{c_i}}.$$

Finally, substituting $\sigma_\epsilon^2 = \sigma_G^2\left(\frac{1-\rho^2}{\rho^2}\right)$ completes the proof.

E Lower bound

We consider the minimization of a finite-sum objective:

$$\min_{x\in\mathcal{X}} F(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x).$$

where each $f_i : \mathcal{X} \to \mathbb{R}$ is convex and G-Lipschitz continuous. We assume access to an exact first-order oracle: when queried at index i and point x, the oracle returns the exact gradient $\nabla f_i(x)$. Crucially, we assume there is a variable cost $c_i > 0$ to querying the i-th component (at any point x).

The goal is to minimize the total expected accumulated cost required to output a point $\hat{x}$ such that $\mathbb{E}[F(\hat{x}) - F(x^*)] \le \epsilon$, where the expectation is taken over any randomization in the algorithm producing $\hat{x}$.

Theorem E.1. *Fix $\epsilon, G > 0$. For any number of components $n \ge G^2/\epsilon^2$, any set of nonnegative query costs $\{c_i\}_{i=1}^n$, and any (possibly randomized) algorithm with an expected error guarantee $\mathbb{E}[F(\hat{x}) - \min F(x)] \le \epsilon$, there exists a convex, G-Lipschitz finite-sum problem instance over $\mathcal{X} = [-1, 1]$ such that the algorithm's expected total query cost is at least:*

$$\Omega\left(\frac{G^2}{\epsilon^2}\left(\frac{1}{n}\sum_{i\in S^*}\sqrt{c_i}\right)^2\right),$$

where $S^* \subseteq [n]$ is any set of components such that following cost-uniformity condition holds:

$$\frac{\max_{i\in S^*}\sqrt{c_i}}{\frac{1}{n}\sum_{j\in S^*}\sqrt{c_j}} \le \frac{G}{40\epsilon}. \tag{13}$$

Remark. If we let the costs be sorted $c_1 \le c_2 \le \ldots \le c_n$, then the set S^* that maximizes the lower bound (if it is nonempty) must be of the form $S = [k]$ for some $1 \le k \le n$, namely, the set of k cheapest costs. Indeed, if $j \in S^*$ then also $i \in S^*$ for any $i < j$, since otherwise including i into S^* increases the lower bound, while still satisfying Eq. (13): it can only increase the denominator on the LHS, while not affecting the numerator.

For reasonably uniform costs, say perfectly uniform up to a small (constant) fraction of arbitrarily large outliers, the condition in Eq. (13) is satisfied by fixing S^* to contain all components except for the outliers, as long as G/ϵ is larger than a constant. It could be, however, that the maximal S^* is empty (and the lower bound is zero) if the costs are highly non-uniform. For example, if the costs are exponentially increasing, say $c_i = 4^i$ for all i, then the ratio between the maximum and the sum of $\sqrt{c_i}$ over any set of the form $S^* = [k]$ is between $\frac{1}{2}$ and 1, and so the cost-uniformity condition implies that $\epsilon \le \frac{G}{20n}$. Thus, for any larger value of ϵ there is no set S^* for which the condition is satisfied.

Proof of Theorem 3.3. The proof proceeds by constructing a hard distribution of instances based on linear functions and applying information-theoretic lower bounds. We break the argument into four steps.

Step 1: Definition of hard instance. Consider the 1-dimensional domain $\mathcal{X} = [-1, 1]$. We define the component functions as linear functions:

$$\forall\, i \in [n]: \qquad f_i(x) = -b_i x,$$

where $b_i \in \mathbb{R}$ is a slope parameter. The global objective is $F(x) = -x \cdot \bar{b}$, where $\bar{b} = \frac{1}{n}\sum_{i=1}^n b_i$. The minimizer x^* depends entirely on the sign of the average slope $\bar{b}$; namely, $x^* = \mathrm{sign}(\bar{b})$. The value of the minimum is $F(x^*) = -|\bar{b}|$, and for any candidate $\hat{x} \in [-1, 1]$, the optimality gap is $F(\hat{x}) - F(x^*) = \bar{b}(x^* - \hat{x})$. In particular, for any algorithm producing $\hat{x}$,

$$\mathbb{E}[F(\hat{x}) - F(x^*)] \ge \bar{b} \cdot \mathbb{P}[\mathrm{sign}(\hat{x}) \ne \mathrm{sign}(\bar{b})]. \tag{14}$$

We define the hard distribution $\mathcal{D}$ as the uniform mixture of two priors, $\mathcal{D}_0$ and $\mathcal{D}_1$, parameterized by $\Delta_1, \ldots, \Delta_n \ge 0$:

- Under $\mathcal{D}_0$: $b_i = \text{clip}(z_i)$, where $z_i \sim \mathcal{N}(+\Delta_i, \sigma^2)$ are i.i.d., $\sigma = G/2$;
- Under $\mathcal{D}_1$: $b_i = \text{clip}(z_i)$, where $z_i \sim \mathcal{N}(-\Delta_i, \sigma^2)$ are i.i.d., $\sigma = G/2$.

Here, $\text{clip}(z_i) = \max(-G, \min(G, z))$ is a truncation to $[-G, G]$ that strictly enforces the Lipschitz condition (deterministically). The b_i are then distributed according to a truncated Gaussian distribution with variance σ^2 strictly smaller than G^2. For reasons that will become apparent in later steps, we will set the parameters Δ_i so that the following conditions are met:

$$\frac{1}{n}\sum_{i=1}^{n}\Delta_i = 20\epsilon, \qquad \text{and} \qquad \Delta_i \le \frac{G}{2} \quad \text{for all } i \in [n]. \tag{15}$$

Step 2: Reduction to hypothesis testing. Define the hypothesis testing error probability as

$$p_{err} = \tfrac{1}{2}p^0_{err} + \tfrac{1}{2}p^1_{err}, \quad \text{where} \quad p^0_{err} = \mathbb{P}_{\mathcal{D}_0}(\hat{x} < 0) \quad \text{and} \quad p^1_{err} = \mathbb{P}_{\mathcal{D}_1}(\hat{x} > 0).$$

We will next show that the expected error (Eq. (14)) is lower bounded in terms of p_{err}.

Conditioning on $\mathcal{D}_0$, a sufficient condition for the algorithm to fail is if $\hat{x} \le 0$ and $\bar{b} > 0$, in which case the optimization error is at least $|\bar{b}|$; thus

$$\mathbb{E}_{\mathcal{D}_0}[F(\hat{x}) - F(x^*)] \ge \mathbb{E}_{\mathcal{D}_0}[|\bar{b}| \cdot \mathbb{I}(\hat{x} \le 0) \cdot \mathbb{I}(\bar{b} > 0)]. \tag{16}$$

We next bound the expected value (under $\mathcal{D}_0$) of the mean of truncate variables. Let $g(\delta) = \mathbb{E}[\text{clip}(Z + \delta, -G, G)]$ where $Z \sim \mathcal{N}(0, \sigma^2)$. The mean of the clipped variable is $\mathbb{E}[b_i] = g(\Delta_i)$. By the mean value theorem, $g(\Delta_i) = g(0) + g'(\xi)\Delta_i = g'(\xi)\Delta_i$ for some $\xi \in (0, \Delta_i)$. The derivative is $g'(t) = \mathbb{P}(-G \le Z + t \le G)$. With $\sigma = G/2$ and $\Delta_i \le G/2$, the interval $[-G - \Delta_i, G - \Delta_i]$ always contains $[-G/2, G/2] = [-\sigma, \sigma]$. The Gaussian mass in this interval is > 0.5, thus $\mathbb{E}[b_i] \ge 0.5\Delta_i$. The aggregate mean then satisfies $\mathbb{E}[\bar{b}] \ge 0.5(\frac{1}{n}\sum_{i=1}^{n}\Delta_i) = 10\epsilon$.

The truncated variables b_i are independent and bounded in $[-G, G]$. By Hoeffding's inequality, $\bar{b}$ concentrates around its expectation: defining the event $A = \{\bar{b} \ge 5\epsilon\}$, the probability of the complement $\delta' = \mathbb{P}_{\mathcal{D}_0}(A^c)$ is bounded by

$$\delta' \le \mathbb{P}_{\mathcal{D}_0}\left(\bar{b} \le \mathbb{E}_{\mathcal{D}_0}[\bar{b}] - 5\epsilon\right) \le \exp\left(-\frac{2n^2(5\epsilon)^2}{n(2L)^2}\right) \le \exp\left(-\frac{3n\epsilon^2}{G^2}\right) \le 0.05,$$

where the final inequality is true since we assume $n \ge G^2/\epsilon^2$. Conditioning on the event A (where $\bar{b} \ge 5\epsilon > 0$) and recalling Eq. (16), we have

$$\begin{aligned}
\mathbb{E}_{\mathcal{D}_0}[F(\hat{x}) - F(x^*)] &\ge \mathbb{E}_{\mathcal{D}_0}[|\bar{b}| \cdot \mathbb{I}(\hat{x} \le 0) \cdot \mathbb{I}(A)] \\
&\ge 5\epsilon\, \mathbb{P}_{\mathcal{D}_0}(\{\hat{x} \le 0\} \cap A) \\
&\ge 5\epsilon(\mathbb{P}_{\mathcal{D}_0}(\hat{x} \le 0) - \mathbb{P}_{\mathcal{D}_0}(A^c)) \\
&= 5\epsilon(p^0_{err} - \delta').
\end{aligned}$$

Similarly, $\mathbb{E}_{\mathcal{D}_1}[F(\hat{x}) - F(x^*)] \ge 5\epsilon(p^1_{err} - \delta')$, thus $\mathbb{E}_{\mathcal{D}}[F(\hat{x}) - F(x^*)] \ge 5\epsilon(p_{err} - \delta')$. The guarantee $\mathbb{E}[F(\hat{x}) - F(x^*)] \le \epsilon$ implies $5\epsilon(p_{err} - 0.05) \le \epsilon$, which leads to $p_{err} \le 0.2 + 0.05 = 0.25$. Thus, the algorithm must identify the correct hypothesis with probability at least 0.75.

Step 3: Information-theoretic lower bound. Let P and Q denote the probability distributions of the entire sequence of oracle responses observed by the algorithm when the instance is drawn from $\mathcal{D}_0$ and $\mathcal{D}_1$, respectively. Let P' and Q' denote the distributions of the underlying unclipped Gaussian vectors $z = (z_1, \ldots, z_n)$ under $\mathcal{D}_0$ and $\mathcal{D}_1$. Since the observed gradients $-b_i$ are deterministic functions of z_i (specifically, $b_i = \text{clip}(z_i)$), the data processing inequality for KL divergences implies:

$$D_{KL}(P \,\|\, Q) \le D_{KL}(P' \,\|\, Q').$$

The variables z_i are independent, and $z_i \sim \mathcal{N}(\Delta_i, \sigma^2)$ under $\mathcal{D}_0$, while $z_i \sim \mathcal{N}(-\Delta_i, \sigma^2)$ under $\mathcal{D}_1$. The KL divergence between two univariate Gaussians with means μ_1, μ_2 and common variance σ^2 is $\frac{1}{2}(\mu_1 - \mu_2)^2/\sigma^2$. Here, the separation is $2\Delta_i$ and $\sigma^2 = G^2/4$, so the KL divergence for z_i is upper bounded by $8\Delta_i^2/G^2$. Since the gradient oracles are exact and the objectives are linear, repeated queries to the same component yield identical results. Thus, information is gained only from the set

of distinct components queried. Let q_i be the probability (under $\mathcal{D}_0$) that component i is queried at least once by the algorithm. By the chain rule for KL divergence (and linearity of expectation), the total divergence is bounded by:

$$D_{KL}(P \parallel Q) \leq D_{KL}(P' \parallel Q') \leq \sum_{i=1}^{n} q_i \frac{8\Delta_i^2}{G^2}. \tag{17}$$

To relate the divergence to the error probability p_{err}, we use Le Cam's argument, which states that for any hypothesis testing problem with uniform prior, $p_{err} \geq \frac{1}{2}(1 - TV(P, Q))$, where $TV(P, Q)$ is the total variation distance. Given our requirement that $p_{err} \leq \frac{1}{4}$, we must have:

$$\tfrac{1}{4} \geq \tfrac{1}{2}(1 - TV(P, Q)) \implies TV(P, Q) \geq \tfrac{1}{2}.$$

By Pinsker's inequality, $TV(P, Q) \leq \sqrt{\frac{1}{2} D_{KL}(P \parallel Q)}$, which yields the required lower bound on the KL divergence:

$$\tfrac{1}{2} \leq \sqrt{\tfrac{1}{2} D_{KL}(P \parallel Q)} \implies D_{KL}(P \parallel Q) \geq \tfrac{1}{2}.$$

Substituting the upper bound from Eq. (17) yields the condition:

$$\sum_{i=1}^{n} q_i \Delta_i^2 \geq \frac{G^2}{16}. \tag{18}$$

Step 4: Optimizing the lower bound. From Eq. (18), any algorithm solving the problem must satisfy the information constraint $\sum_{i=1}^{n} q_i \Delta_i^2 \geq G^2/16$. Since the total expected query cost C (under the mixture distribution $\mathcal{D}$) is lower bounded by half times the expected cost under $\mathcal{D}_0$, we also have $C \geq \frac{1}{2} \sum_{i=1}^{n} q_i c_i$. For a fixed set of parameters Δ_i, we can lower bound the algorithm's expected cost in terms of the "information density" $\rho_i = \Delta_i^2 / c_i$ of the components; indeed,

$$\frac{G^2}{16} \leq \sum_{i=1}^{n} q_i \Delta_i^2 = \sum_{i=1}^{n} (q_i c_i) \rho_i \leq \left(\sum_{i=1}^{n} q_i c_i \right) \max_i \rho_i \implies C \geq \frac{G^2}{32 \max_i \rho_i}.$$

To maximize this lower bound, the adversary minimizes $\max \rho_i$ subject to the constraints $\frac{1}{n} \sum_{i=1}^{n} \Delta_i = 20\epsilon$ and $\Delta_i \leq G/2$ for all i (recall Eq. (15)). The minimum is achieved when the ratios ρ_i are equalized across the active components. Thus, we set $\Delta_i = \alpha \sqrt{c_i}$ for components i in an "active" subset $S^* \subseteq [n]$, and $\Delta_i = 0$ otherwise. The scaling factor α is determined using the constraint $\frac{1}{n} \sum_{i=1}^{n} \Delta_i = 20\epsilon$:

$$\frac{1}{n} \sum_{i \in S^*} (\alpha \sqrt{c_i}) = 20\epsilon \implies \alpha = \frac{20n\epsilon}{\sum_{i \in S^*} \sqrt{c_i}}. \tag{19}$$

To certify the additional condition $\Delta_i \leq G/2$ for all i, we require for all $i \in S^*$:

$$\alpha \sqrt{c_i} \leq \frac{G}{2} \iff \frac{20n\epsilon \sqrt{c_i}}{\sum_{j \in S^*} \sqrt{c_j}} \leq \frac{G}{2} \iff \frac{\max_{i \in S^*} \sqrt{c_i}}{\frac{1}{n} \sum_{j \in S^*} \sqrt{c_j}} \leq \frac{G}{40\epsilon}.$$

This is the precisely the uniformity condition given in the theorem statement. The subset S^* is defined as the largest set of cheapest components satisfying this condition. Finally, substituting α back into the cost lower bound, we conclude:

$$C \geq \frac{G^2}{32} \left(\frac{\sum_{i \in S^*} \sqrt{c_i}}{20n\epsilon} \right)^2 = \frac{G^2}{12800\epsilon^2} \left(\frac{1}{n} \sum_{i \in S^*} \sqrt{c_i} \right)^2.$$

□

F Extensions and Generality of the Theoretical Analysis

In this section, we discuss the generality of the theory in Sections 3.2 and C. In particular, for the main paper we assume convexity in order to derive meaningful theoretical statements. Given that our empirical setting does not optimize a convex loss function, we address the differences here. Importantly, we emphasize that our theory does not strictly rely on convexity.

F.1 Differences between Theory and Practice

In the GRPO experiments, our training objective is not convex. Additionally, we do not use the averaged iterate but instead the last iterate for reporting final error. Furthermore, in GRPO, the objective changes every time a new set of rollouts is collected. As such, one can think of our implementation as many rounds of cost-aware learning. With every new set of rollouts, we define a new sampling distribution over the newly generated pool. The policy is then optimized by iterating over mini-batches sampled from this reweighted pool, ensuring the cost-aware distribution actively shapes the variance of the gradient updates at each step.

F.2 Generality of the Cost Objective in Non-Convex Settings

The mathematical optimality of p^* is not strictly limited to the convex setting. Its derivation depends solely on the relationship between iteration complexity (T) and estimator variance ($S(p)$). Across various smoothness conditions, the number of iterations T required to reach an ϵ-stationary point is fundamentally bottlenecked by the stochastic gradient variance. Specifically:

- **Non-Convex SGD:** Reaching an ϵ-stationary point requires an iteration complexity of $T \propto \frac{S(p)}{\epsilon^2}$ [Ghadimi and Lan, 2013].
- **Adaptive Optimizers (Adam):** Recent theoretical analyses of Adam-type algorithms demonstrate a convergence rate of $\tilde{O}\left(\frac{\sqrt{S(p)}}{\sqrt{T}}\right)$ [Chen et al., 2018], implying $T \propto \frac{S(p)}{\epsilon^2}$ (modulo sub-polynomial logarithmic factors).
- **Generalized Smoothness:** LLM loss landscapes exhibit generalized smoothness, where local curvature grows proportionally with the gradient norm. Optimization theory establishes that gradient clipping and adaptive methods guarantee convergence in this regime [Zhang et al., 2020]. Crucially, this theory establishes that the dominant stochastic term in the iteration complexity explicitly retains the $O\left(\frac{S(p)}{\epsilon^2}\right)$ dependence on the variance $S(p)$.

Because the total expected training cost $\mathcal{K}$ is the product of iterations and the expected per-step cost ($T \cdot C(p)$), the objective to minimize reduces to:

$$J(p) = S(p)C(p)$$

Since this objective remains structurally identical across these diverse optimization settings, minimizing the variance-cost product via p^* continues to be the precise optimal strategy. Thus, the optimal distribution $p_i^* \propto G_i/\sqrt{c_i}$ derived in Theorem 3.1 remains justified in non-convex regimes.

F.3 Average Iterate ($\bar{x}_T$) vs. Last Iterate (x_T)

While Algorithm 1 returns the average iterate $\bar{x}_T$ following standard conventions in the SGD literature, the properties of the last iterate x_T have also been extensively studied [Shamir and Zhang, 2013, Jain et al., 2019, Harvey et al., 2019]. As shown by Harvey et al. [2019], for a fixed step size, the last-iterate of SGD is a factor of $\log(T)$ worse than the average. While the average iterate generally achieves superior performance—sometimes strictly so—specific step-size modification schemes [Jain et al., 2019] allow the expected suboptimality of the last iterate to enjoy the same convergence guarantees. Consequently, sampling with the optimal distribution p^* derived in Theorem 3.1 provides similar theoretical guarantees for the last iterate as we establish for the averaged iterate.

G Experimental Details and Ablations

G.1 Experimental Details

To ensure the reproducibility of our experiments, we provide a comprehensive overview of the hyperparameter configurations and the computational framework employed. Our implementation is built upon the Verl framework, utilizing vLLM [Kwon et al., 2023] for efficient inference and Flash-Attention [Dao et al., 2022] for optimized memory throughput.

G.2 Implementation

The primary modifications to the Verl codebase involve the construction of mini-batches for the policy gradient update and the integration of importance sampling. To maintain training stability, we re-center the importance weights around 1. This normalization allows us to maintain a consistent gradient norm across experiments. Crucially, this simply multiplies the training objective by a constant.

G.3 Importance Weighting with GRPO

Let $P(Q)$ be the distribution over prompts and $\pi_{\theta_{\text{old}}}(q)$ be the distribution over responses from policy $\pi_{\theta_{\text{old}}}$ given prompt q. The objective contribution of a prompt q and response o is as follows:

$$\ell(o;\theta,q) = \frac{1}{|o|}\sum_{t=1}^{|o|} \min\left(\frac{\pi_\theta(o_t \mid q,o_{<t})}{\pi_{\theta_{\text{old}}}(o_t \mid q,o_{<t})}A^{(t)}, \text{clip}\left(\frac{\pi_\theta(o_t \mid q,o_{<t})}{\pi_{\theta_{\text{old}}}(o_t \mid q,o_{<t})}, 1-\epsilon, 1+\epsilon\right)A^{(t)}\right) - \beta\mathbb{D}_{\text{KL}}(\pi_\theta\|\pi_{\text{ref}}).$$

The GRPO objective for parameters θ is:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{\substack{q\sim P(Q) \\ \{o_i\}_{i=1}^{M}\sim\pi_{\theta_{\text{old}}}(q)}}\left[\frac{1}{M}\sum_{g=1}^{M}\ell(o_i;\theta)\right]. \tag{20}$$

Given n prompts and M rollouts per prompt, and letting $o_{i,g}$ be the g'th rollout from the i'th prompt, the empirical objective is:

$$\hat{\mathcal{J}}_{\text{GRPO}}(\theta) = \frac{1}{nM}\sum_{i=1}^{n}\sum_{g=1}^{M}\ell\left(o_{i,g};\theta,q_i\right). \tag{21}$$

In effect, the empirical GRPO objective is a finite sum objective over prompts and rollouts. If we sample according to another distribution $p \in \Delta^{nM}$, where we sample rollout pair $(q_i, o_{i,g})$ with probability $p_{(i,g)}$, the importance-weighted objective is:

$$\hat{\mathcal{J}}^{(p)}_{\text{GRPO}}(\theta) = \frac{1}{nM}\sum_{i=1}^{n}\sum_{g=1}^{M}\frac{1}{p_{(i,g)}}\ell\left(o_{i,g};\theta,q_i\right). \tag{22}$$

G.4 Hyperparameters

Table 3 summarizes the key hyperparameters used for both the GRPO baseline and our GRPO + ZVF method for main experiments. We report Scale-RL hyperparameters in Table 4, largely influenced by the findings of [Khatri et al., 2025].

Hyperparameter	**Qwen2.5-Math-1.5B-Instruct**	**Qwen3-4B**	**Qwen3-8B**
Algorithm	GRPO, GRPO + ZVF	GRPO	GRPO
Learning Rate	1×10^{-6}	1×10^{-6}	1×10^{-6}
Weight Decay	0.1	0.01	0.01
Train Batch Size	256	256	256
PPO Mini-Batch Size	64	128	128
Rollout Samples per Prompt	16	8	5
Rollout Temperature	1.0	1.0	1.0
Max Prompt Length	2048	1024	1024
Max Response Length	4096	8192	8192
KL Loss Coefficient	0.001	0.001	0.001
Clip Ratio Low	0.2	0.2	0.2
Clip Ratio High	0.28	0.2	0.2

Table 3: Key Training Hyperparameters for GRPO training across model sizes.

Hyperparameter	**Value**
Base Model	`Qwen/Qwen2.5-Math-1.5B-Instruct`
Advantage Estimator	GRPO
Policy Loss Type	CISPO
Learning Rate	5×10^{-7}
LR Warmup Steps	100
Weight Decay	0.01
Train Batch Size	512
PPO Mini-Batch Size	128
Gradient Clipping	1.0
KL Loss Coefficient	0.001
Entropy Coefficient	0
Clip Ratio Low	0.0
Clip Ratio High	5.0
Clip Ratio C	3.0
LM Head Precision	FP32

Table 4: Key Training Hyperparameters for ScaleRL/CISPO Optimized Script

G.5 Computational Requirements

Table 5 lists the hardware and approximate duration of each training run.

Model	Setting	GPUs per run	Approx. duration
Qwen2.5-Math-1.5B-Instruct	GRPO	4 × H100	48 hours
	GRPO + ZVF	4 × H100	72 hours
	CISPO	4 × H100	72 hours
Qwen3-4B	GRPO	8 × H100	2.5 days
Qwen3-8B	GRPO	8 × H100	3 days

Table 5: Hardware and approximate duration of each training run.

G.6 FLOPs accounting

For each optimizer step, we distinguish generation FLOPs, C_{gen}, from train FLOPs, C_{train}. Their sum is total FLOPs:

$$C_{\text{total}} = C_{\text{gen}} + C_{\text{train}}.$$

Let P be the parameter count, B the prompts drawn per step, M the rollouts per prompt, and $|o|$ a prompt-plus-response token count. We count FLOPs per training step as

$$C_{\text{gen}} = 2P\,T_{\text{gen}}, \qquad T_{\text{gen}} = R\,B\,M\,\overline{|o|}, \qquad C_{\text{train}} = 8P\,T_{\text{useful}}.$$

- **Generation** is charged one forward pass, at 2 FLOPs per parameter per token. R is the number of generation batches drawn to fill the step ($R > 1$ only under zero variance filtering; $R = 1$ for all non-ZVF runs).
- **Policy optimization** counts only the tokens actually trained on, T_{useful}: $6P$ per token for the forward and backward pass on the policy-gradient loss, plus $2P$ per token for the $\pi_{\theta_{\text{old}}}$ log-probability forward pass.

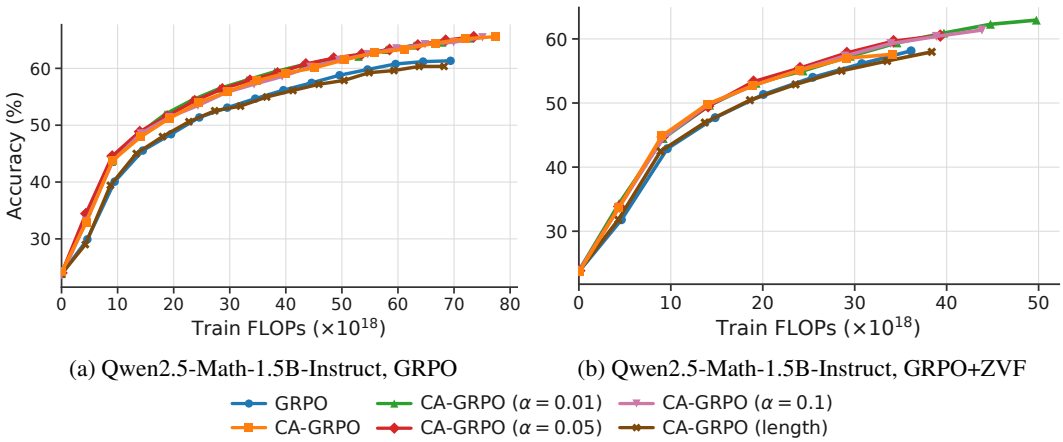
G.7 Response length analysis

Model	Setting	Method	Steps	Mean length	Δ vs. no sampling
Math-1.5B-IT	GRPO	No sampling	1–1400	888	—
	GRPO	p^*	1–1500	911	+2.6%
	ZVF	No sampling	1–700	932	—
	ZVF	p^*	1–700	930	−0.2%
Qwen3-4B	GRPO	No sampling	1–160	5,407	—
	GRPO	p^*	1–140	5,254	−2.8%
Qwen3-8B	GRPO	No sampling	1–200	5,415	—
	GRPO	p^*	1–160	5,225	−3.5%

Table 6: **Response length dynamics throughout training.** We evaluate to what extent the response length generated by the model is affected by the sampling strategy by reporting the average response length throughout training of each method. Each entry averages the per-step mean response length (tokens) over training steps up to that run's peak-accuracy checkpoint, i.e. the same runs and checkpoints as Table 1.

G.8 Ablation on variants of p^*

In this section we ablate the different variants of p^*. In particular, we also report p^*_{smooth} for $\alpha = 0.05, \alpha = 0.1$. Furthermore, we consider p^*_{LEN}, in which $(p^*_{\text{LEN}})_i = \frac{1}{\sqrt{c_i}}$. This sampling distribution only considers the length of sequences in its distribution. We report these variants both for GRPO and GRPO+ZVF. In particular with the GRPO setting, we see that p^*_{LEN} does not attain very strong performance. We hypothesize that this is because many shorter responses have 0 advantage, and hence there is little training signal. As in the main paper, we also report final performance for all methods.



(a) Qwen2.5-Math-1.5B-Instruct, GRPO (b) Qwen2.5-Math-1.5B-Instruct, GRPO+ZVF

Figure 7: Training FLOPs compared with AIME `pass@1/mean@32` accuracy throughout training for both GRPO and GRPO+ZVF methods.

Setting	Method	AIME	AMC	MATH500	GSM8K	Avg. Accuracy
GRPO	No sampling	61.3	64.1	**73.2**	**86.2**	71.2
GRPO	p^*	**65.6**	**71.2**	**73.2**	86.0	**74.0**
GRPO	$p^*_{\text{smooth}}(\alpha = 0.01)$	65.3	68.1	72.3	86.0	72.9
GRPO	$p^*_{\text{smooth}}(\alpha = 0.05)$	**65.6**	68.3	72.8	86.1	73.2
GRPO	$p^*_{\text{smooth}}(\alpha = 0.1)$	65.4	66.0	72.6	85.8	72.5
GRPO	p^*-LEN	60.3	64.8	73.0	86.0	71.0
ZVF	No sampling	58.1	64.7	73.2	86.1	70.6
ZVF	p^*	57.6	63.0	72.9	85.9	69.8
ZVF	$p^*_{\text{smooth}}(\alpha = 0.01)$	**62.9**	68.0	73.3	85.8	**72.5**
ZVF	$p^*_{\text{smooth}}(\alpha = 0.05)$	60.5	65.9	73.0	**86.4**	71.4
ZVF	$p^*_{\text{smooth}}(\alpha = 0.1)$	61.4	**68.3**	72.9	86.0	72.2
ZVF	p^*-LEN	58.0	63.3	**73.9**	86.2	70.3

Table 7: (Math-1.5B-Instruct, `pass@1/mean@32`): Complete results.

Setting	Method	AIME	AMC	MATH500	GSM8K	Avg. Accuracy
GRPO	No sampling	68.3	78.6	88.8	95.1	82.7
GRPO	p^*	**68.8**	79.4	**89.0**	95.1	83.0
GRPO	$p^*_{\text{smooth}}(\alpha = 0.01)$	**68.8**	**80.3**	88.6	94.9	**83.2**
GRPO	$p^*_{\text{smooth}}(\alpha = 0.05)$	68.7	79.7	**89.0**	**95.2**	83.1

Table 8: (Qwen3-8B, `pass@1/mean@32`): Complete results.

G.9 Full CISPO results

We report performance across benchmarks. The CISPO curves are reported in Figure 8 and the best checkpoint performance is reported in Table 9.

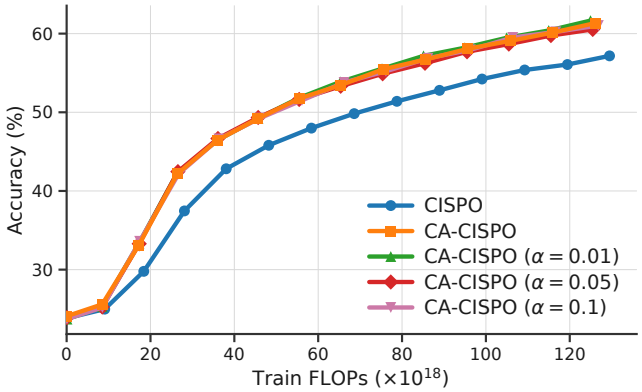


Figure 8: Full CISPO objective results for Qwen2.5-Math-1.5B-Instruct on AIME. We see robustness to smoothed cost-aware learning.

Setting	Method	AIME	AMC	MATH500	GSM8K	Avg. Accuracy
CISPO	No sampling	57.2	61.8	73.0	85.9	69.5
CISPO	p^*	61.3	65.7	**73.6**	**86.2**	**71.7**
CISPO	$p^*_{\text{smooth}}(\alpha = 0.01)$	**61.7**	65.4	73.5	86.0	71.6
CISPO	$p^*_{\text{smooth}}(\alpha = 0.05)$	60.5	**66.0**	73.0	86.1	71.4
CISPO	$p^*_{\text{smooth}}(\alpha = 0.1)$	61.0	65.2	**73.6**	86.0	71.5

Table 9: (Math-1.5B-Instruct, `pass@1/mean@32`): Complete results for CISPO.

G.10 Full sub-optimality results

In the main paper, we report the sub-optimality metrics for the 8B and 1.5B models under the GRPO setting. Here, we also include the sub-optimality metrics for the GRPO+ZVF setting and the CISPO settings in Figures 9 and 10 respectively.

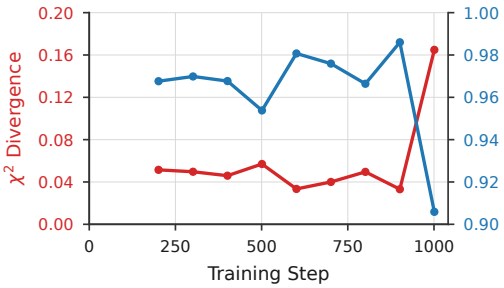


Figure 9: Sub-optimality metrics for 1.5B GRPO+ZVF training run.

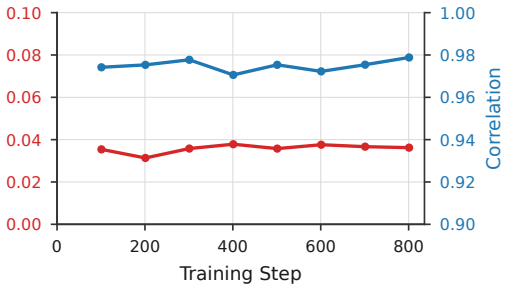


Figure 10: Sub-optimality metrics for 1.5B CISPO training objective run.

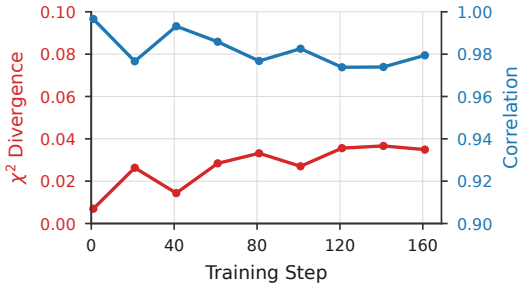
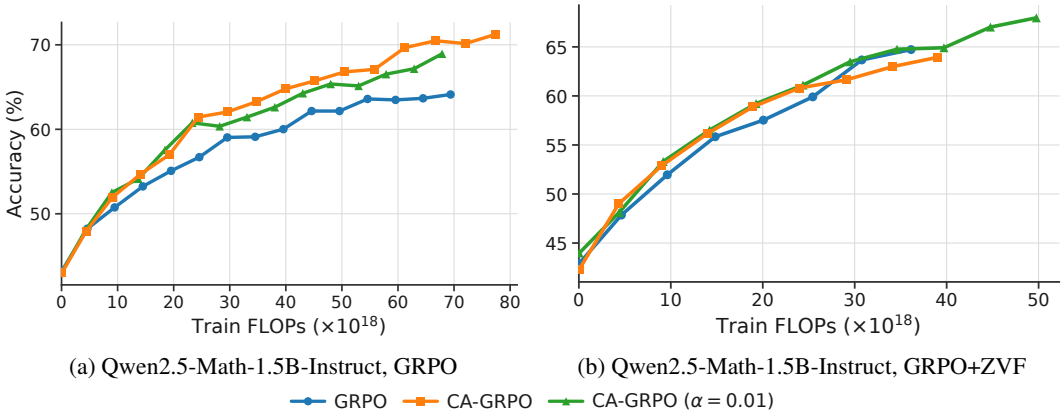


Figure 11: Sub-optimality metrics for 4B run.

G.11 Additional results for Qwen2.5-Math-1.5B-Instruct

We report AMC performance as a function of training FLOPs in Figure 12.



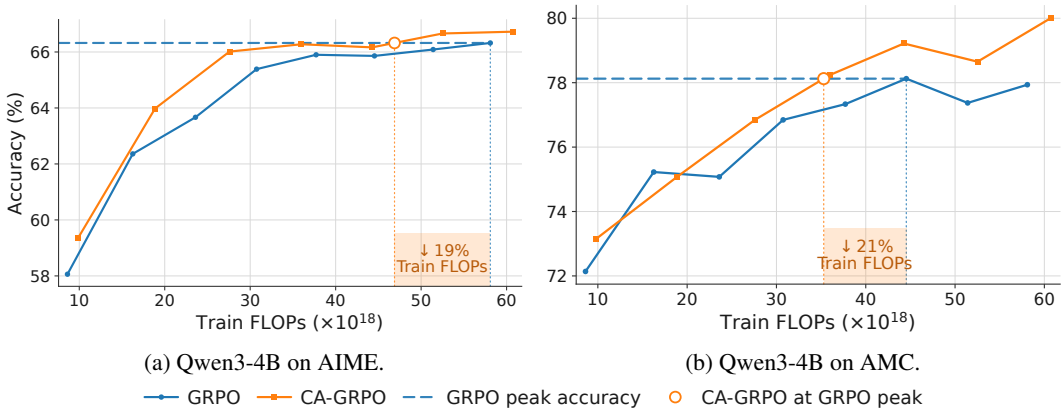
(a) Qwen2.5-Math-1.5B-Instruct, GRPO

(b) Qwen2.5-Math-1.5B-Instruct, GRPO+ZVF

Figure 12: Training FLOPs compared with AMC `pass@1/mean@32` accuracy throughout training for both GRPO and GRPO+ZVF settings for the 1.5B model.

G.12 Additional results for Qwen3-4B

We report results for Qwen3-4B on AIME and AMC below. We plot up until peak AIME accuracy is attained. We also report smoothed variants.



(a) Qwen3-4B on AIME. (b) Qwen3-4B on AMC.

Figure 13: Qwen3-4B on AIME and AMC.

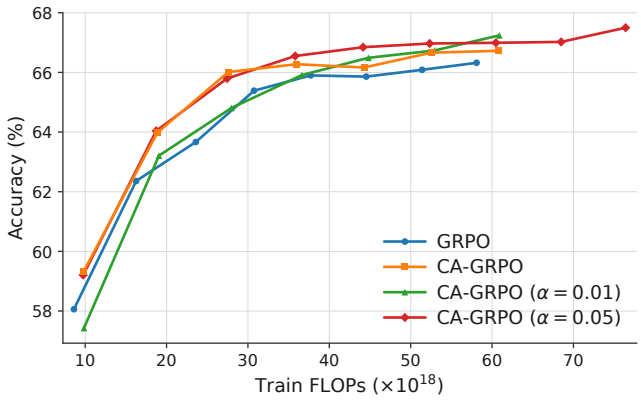
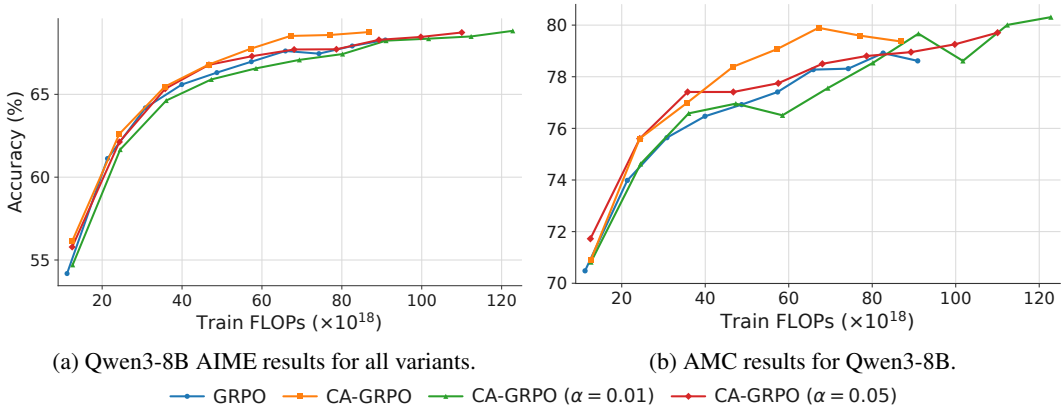


Figure 14: Qwen3-4B and smoothed variants on AIME.

G.13 Additional results for Qwen3-8B

We present the results for the smoothed variants of the 8B model on AIME in Figure 15a. Additionally, we plot AMC performance as a function of training FLOPs in Figure 15b. In Table 8, we report full results on best checkpoint accuracy for smoothed variants on four benchmarks.



(a) Qwen3-8B AIME results for all variants.

(b) AMC results for Qwen3-8B.

Figure 15: Qwen3-8B on AIME and AMC, all variants.