CANOPY: EXPLOITING PIECEWISE SMOOTH TREE PRIORS FOR MULTI-FIDELITY BANDITS

Anonymous authors
Paper under double-blind review

ABSTRACT

Many LLM inference problems, including model routing, prefix-cache management, prompt trimming, and test-time search, can be viewed as optimization over a tree. This structure arises naturally from autoregressive generation: every prefix defines a node, and its continuations form a subtree below it. Internal nodes of the tree provide cheap but biased estimates of a region's value, while leaf evaluations are expensive but accurate. Hierarchical bandit methods can exploit this structure, but typically require a specific smoothness schedule to be specified in advance, even though real objectives are often only piecewise smooth and their optima may lie near sharp boundaries. We introduce CANOPY, a multi-fidelity tree bandit that learns where the smoothness prior is valid rather than assuming it globally. CANOPY uses cheap random-path probes to construct an online certificate of local aggregation bias, then directs expensive leaf evaluations toward cells where the certificate detects a smoothness violation. We prove fixed-budget and regret guarantees whose additional cost is additive in the number of discontinuities, recovering the smooth-tree rate when no violations are present and approaching structure-blind search as violations become dense. Across routing, top-(k) identification, test-time search, caching, and prompt trimming, CANOPY consistently improves matched-budget performance, including $2.9\times$ higher top-10 recall on a 1000-model pool, $1.6\times$ more SWE-bench Verified issues resolved than best-of-(N), and $3.6\times$ lower median time-to-first-token with prefix caching.

1 INTRODUCTION

Modern LLM applications spend an increasing fraction of their compute at inference time (Welleck et al., 2024; Snell et al., 2024; Brown et al., 2024), and the algorithms that use this compute often revisit shared partial computations. For example, search procedures expand multiple traces from a common prefix (Yao et al., 2023), serving stacks reuse cached prefixes across requests (Kwon et al., 2023; Zheng et al., 2024), and prompt-trimming methods consider progressively compressed versions of a long prompt (Jiang et al., 2023). In each setting, the system must choose among more candidates than it can afford to evaluate exhaustively, since evaluating a candidate may require a generation, a test run, or a served request. These candidates admit a natural hierarchical organization: candidates that share more of their computation tend to behave more similarly, except at a small number of sharp boundaries.

This hierarchy induces a multi-fidelity feedback model (Figure 1a). An internal-node probe returns the subtree average, giving a cheap but biased proxy for the best leaf beneath it, while a leaf evaluation is expensive but unbiased. The key quantity is therefore the *aggregation bias*: how far the best leaf in a cell can exceed its mean. Classical hierarchical bandits control this bias using a smoothness schedule specified in advance (Bubeck et al., 2011; Munos, 2011; Azar et al., 2014). But the right schedule is difficult to know: assuming too much smoothness can prune promising regions, while being too conservative forfeits the savings from hierarchical search. Moreover, real objectives may be only piecewise smooth, with a small number of sharp boundaries where an innocuous subtree average hides an unusually valuable leaf. This raises our central question: can the learner use cheap observations to determine where the smoothness assumption is actually valid?

Contributions. We study a multi-fidelity bandit over a complete tree without access to the smoothness schedule that standard hierarchical bandits assume. Internal nodes provide cheap subtree aver-

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

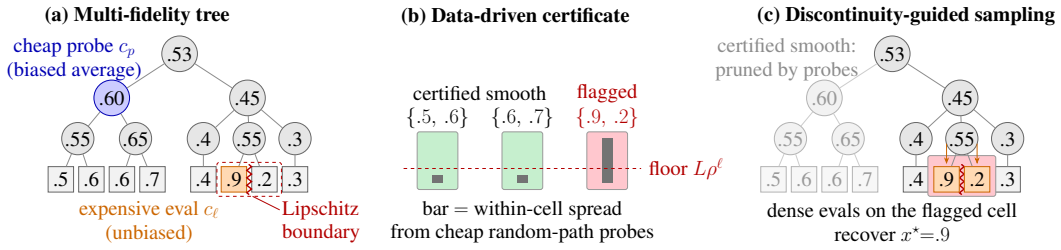


Figure 1: **Method overview.** (a) Leaves show the unknown values, and the red line marks the Lipschitz boundary splitting the cell. (b) The certificate measures each cell's within-cell spread from cheap probes against the Lipschitz floor. (c) On the same tree, the search prunes the certified-smooth subtree.

ages under a cost budget, while leaves provide expensive unbiased evaluations. We use the standard hierarchical descent of $\mathcal{X}$-armed bandits (Bubeck et al., 2011; Azar et al., 2014; Munos, 2011; Grill et al., 2015), but, to our knowledge, are the first to certify the required smoothness information directly from data rather than assume it or search over candidate schedules. We further show that once a smoothness violation is encountered, targeted leaf-level sampling is unavoidable, up to logarithmic factors, for identifying the best leaf (Proposition 2).

We make three key contributions. First, we develop an online, local-Lipschitz certificate of aggregation bias (Section 3.3). Second, we build a discontinuity-guided adaptive sampling technique that spends expensive evaluations only in regions where the certificate fails (Section 3.4). Third, we combine the two into a cost-budgeted framework (Section 3.1) that casts model routing, prefix caching, and test-time search as instances of the same multi-fidelity tree optimization problem.

Experimental overview. Across twelve open-source benchmarks, our method improves performance at matched cost or compute over others (Table 1). On RouterEval's 1000-model pool, the tree bandit reaches $2.9\times$ the top-10 recall of the strongest structure-blind baseline and is better at every budget. Value-guided search beats best-of-N by $+0.087$ on MATH, $+0.111$ on GPQA-Diamond, and $+0.123$ on SWE-bench Verified, with the strict budget cells of the model$\times$budget sweep reaching 1.8–$2.2\times$. In a serving-based setting, the regional router beats the flat learner in 4 of 5 independent τ-bench replicates, and reaches near-oracle quality at the lowest cost on live MMLU. On the Mooncake production trace, the adaptive cache reuses 78% more blocks per request than the engine-default LRU after a popularity shift. On a vLLM server, prefix caching cuts median time-to-first-token $3.6\times$, and adaptive eviction saves 83% more TTFT than LRU post-shift. The experiments also identify precisely when these gains are available and measure every boundary condition (Section 4).

2 Background

Hierarchical and $\mathcal{X}$-armed bandits. Optimizing a noisy function over a hierarchical partition of its domain is the classical $\mathcal{X}$-armed bandit problem (Bubeck et al., 2011; Munos, 2014). Optimistic tree-search methods differ primarily in how they control the gap between a cell's observed value and its best descendant. HOO (Bubeck et al., 2011) uses a known smoothness schedule, while SOO/StoSOO (Munos, 2011; Valko et al., 2013) avoid specifying one explicitly and POO (Grill et al., 2015) searches over a family of candidate schedules. Adaptation to unknown smoothness is impossible from leaf evaluations alone (Locatelli & Carpentier, 2018). Our setting provides additional feedback: a random-path probe samples the distribution of leaf values within a cell, allowing us to certify the corresponding maximum-minus-mean aggregation bias directly from data (Section 3.3).

Multi-fidelity optimization and best-arm identification. Multi-fidelity optimization combines cheap biased observations with expensive accurate ones (Kandasamy et al., 2016). Our setting instantiates this structure on a tree: internal-node probes return cheap subtree averages, while leaf evaluations provide expensive unbiased observations. Since a subtree average can underestimate its best leaf, the relevant fidelity bias is one-sided and is exactly the aggregation bias studied in Section 3.3. Fixed-budget best-arm identification provides the corresponding leaf-level problem, where

a problem-dependent hardness quantity determines how quickly identification error decreases with budget (Audibert et al., 2010; Kaufmann et al., 2016). Our method combines these two regimes: cheap structural search over regions where aggregation is reliable, and targeted leaf-level certification where it is not.

3 THEORY

We define the tree model of Section 3.1 and the optimistic descent of Section 3.2. Both are organized around how far a cell's best leaf can exceed its mean. The model makes that quantity the bias of a cheap probe, and the descent carries it as its optimism bonus. The guarantees of Section 3.4 are stated in terms of this gap, which the certificate of Section 3.3 bounds from data.

3.1 NOTATION AND PRELIMINARIES

The structure. We study a complete b-ary tree of depth D, with $N = b^D$ leaves. A node v at level $\ell \in \{0, \dots, D\}$ spans a contiguous block of $|\mathcal{L}(v)| = b^{D-\ell}$ leaves. Each leaf x has a true mean reward $\mu(x) \in [0, 1]$, and the value of any node is the average of its subtree's leaves,

$$f(v) \;=\; \frac{1}{|\mathcal{L}(v)|} \sum_{x \in \mathcal{L}(v)} \mu(x), \qquad \text{so} \qquad f(v) = \frac{1}{b} \sum_{c \in \text{children}(v)} f(c). \tag{1}$$

We denote $\mu^\star = \max_x \mu(x)$ for the best leaf. The aggregation bias, $\text{bias}(v)$, of an internal node is the gap between the largest leaf mean in its subtree, $\max_{x \in \mathcal{L}(v)} \mu(x)$, and the average $f(v)$ returned.

We measure the distance between leaves by their lowest common ancestor (LCA), $d(x, y) = \rho^{\text{level}(\text{LCA}(x,y))}$ with $\rho \in (0, 1)$. Two leaves are considered close when they share a deep ancestor. On a prefix tree, it is the longest-common-prefix distance. We define μ tree-Lipschitz with constant L when $|\mu(x) - \mu(y)| \le L\, d(x, y)$ for all leaves x, y, or equivalently when μ oscillates by at most $L\rho^\ell$ inside any level-ℓ subtree. Therefore, the assumed hiearchical bias schedule follows from tree-Lipschitzness alone,

$$\text{bias}(v) \;=\; \max_{x \in \mathcal{L}(v)} \mu(x) - f(v) \;\le\; \text{spread}(\ell) \;=\; L\,\rho^\ell, \qquad \ell = \text{level}(v). \tag{2}$$

Since $d(x, y)$ is exactly the fraction of all leaves underneath the LCA of x and y, we choose $\rho = 1/b$.

Because global smoothness is unrealistic, we work in the piecewise regime. We assume μ is tree-Lipschitz except on a set of K discontinuities satisfying a dispersion condition (Balcan et al., 2018). The bound in Equation (2) holds on any cell that contains no discontinuity. In a cell that contains a discontinuity the bound need not hold, and only the trivial bound $\text{bias}(v) \le 1$ from $\mu \in [0, 1]$ remains.

We write d for the near-optimality dimension of the smooth part. Thus, the number of ε-optimal cells at resolution ε grows as ε^{-d}.

Feedback and objectives. We provide the learner with two query types, a cheap probe of an internal node and an expensive evaluation of a leaf. A probe of node v follows a uniformly random path to a leaf $L \in \mathcal{L}(v)$ and returns $X = \mu(L) + \eta$ at cost c_p, so its expectation is the subtree average $f(v)$. A leaf evaluation of x returns $\mu(x) + \eta$ at cost $c_\ell \ge c_p$. In both cases the noise η is Gaussian, $\eta \sim \mathcal{N}(0, \sigma^2)$, and independent of the leaf drawn. The budget is measured in total cost spent rather than in the number of pulls.

We study two objectives on this model. The first is regret minimization, where the learner commits to a node v_t each round and receives a probe reward, scored by its cumulative regret against the best leaf, $R_n = \sum_{t \le n} (\mu^\star - f(v_t))$. The second is fixed-budget identification, where the learner spends a total cost budget B and returns the top-k leaves, scored by the misidentification probability $\mathbb{P}(\widehat{x} \neq x^\star)$. We focus here on situations where $k = 1$. The question for both objectives is when exploiting Equation (1) and Equation (2) beats a structure-blind learner that pays leaf cost everywhere. Section 3.4 answers it in terms of the probe to cost ratio c_p/c_ℓ and the violation count K. Both

objectives concern only the optimum. The learner is scored on the best leaf to which it commits or returns, so suboptimal regions may stay coarsely resolved.

3.2 The Optimistic Tree-Bandit Algorithm

Our regret-minimization algorithm is the standard optimistic tree bandit (Bubeck et al., 2011; Azar et al., 2014), modified to the average-backup tree of Section 3.1. We summarize it here because our contributions (Section 3.3, Section 3.4) modify its bias term and its sampling.

For a node v at level ℓ with $T(v)$ probes and empirical subtree-average $\widehat{\mu}(v)$, where the values are

$$U(v) = \widehat{\mu}(v) + c\sqrt{\tfrac{2\ln t}{T(v)}} + \mathrm{spread}(\ell), \qquad B(v) = \min\Big(U(v),\ \max_{c\in\mathrm{children}(v)} B(c)\Big). \tag{3}$$

The bias bonus $\mathrm{spread}(\ell)$ is the aggregation bound Equation (2), where the best leaf under v can exceed the mean we estimate. Thus, $U(v)$ is a valid upper bound on the best reachable leaf. Each round descends from the root to a leaf of the explored tree, following the child with the largest B-value at every step. It then probes that leaf and updates $\widehat{\mu}$, T, U, and B along the path.

The confidence radius $r(v) = c\sqrt{2\ln t/T(v)}$ shrinks with probes while the bias floor $\mathrm{spread}(\ell)$ is fixed, where

$$r(v) \le \mathrm{spread}(\ell) \quad\Longleftrightarrow\quad T(v) \ge \frac{c^2 \ln t}{\mathrm{spread}(\ell)^2}, \tag{4}$$

Proposition 1. *On a tree-Lipschitz instance of near-optimality dimension d with $\mathrm{spread}(\ell) = L\rho^\ell$, the descent incurs regret $R_n = \widetilde{O}\big(n^{(d+1)/(d+2)}\big)$, optimal up to logarithmic factors for this class. Equation (4) caps the explored tree at $\widetilde{O}\big(n^{d/(d+2)}\big)$ nodes.*

This gives the $\mathcal{X}$-armed-bandit guarantee (Bubeck et al., 2011; Azar et al., 2014).

Our contributions replace the two assumptions, the bias term $\mathrm{spread}(\ell)$, which Section 3.3 certifies from data, and the uniform descent, which Section 3.4 replaces with discontinuity-guided sampling.

3.3 Certifying Smoothness from Data

We describe here the replacement of the assumed schedule spread with a high-probability upper bound on the aggregation bias of Equation (2), computed only from random-path probes of v.

The bound. The certificate has three properties. First, a log-sum-exp bound, the smooth upper bound on the maximum, trades the maximum for an estimable function, since for any $\lambda > 0$,

$$\mathrm{bias}(v) \ \le\ \frac{1}{\lambda}\Big[\log m + \log M_\mu(\lambda)\Big] - f(v), \qquad M_\mu(\lambda) = \mathbb{E}_L e^{\lambda\mu(L)}. \tag{5}$$

where $M_\mu(\lambda)$ is the moment-generating function (MGF) of the leaf means. Second, the noise divides out of the observable MGF. A probe is $X = \mu(L) + \eta$ with $\eta \sim \mathcal{N}(0,\sigma^2)$ independent of L, so the MGF factors as $\mathbb{E}e^{\lambda X} = M_\mu(\lambda)\, M_\eta(\lambda)$ with $M_\eta(\lambda) = e^{\lambda^2\sigma^2/2}$ known exactly. Dividing gives $M_\mu(\lambda) = \mathbb{E}e^{\lambda X}\, e^{-\lambda^2\sigma^2/2}$, so any upper confidence bound on $\mathbb{E}e^{\lambda X}$ is an upper confidence bound on $M_\mu(\lambda)$. Exact knowledge of M_η is what makes this step valid. Under a sub-Gaussian upper bound alone the division would run the wrong way and yield only a lower bound on M_μ. Third, an empirical-Bernstein bound (Maurer & Pontil, 2009) turns n probes into a confidence interval for $\mathbb{E}e^{\lambda X}$. Since Gaussian probes are unbounded, we truncate each probe to $[-z\sigma,\ 1 + z\sigma]$ and assume the probability $2n\bar{\Phi}(z)$ that some probe leaves this range (Remark 4). Algorithm 1 lists the steps per iteration and returns the quantity that Theorem 1 certifies.

Theorem 1. *Fix a node v, let $\eta \sim \mathcal{N}(0,\sigma^2)$, and compute $\widehat{G}_\uparrow(\lambda)$ and ε_n from n probes truncated to $[-z\sigma,\ 1 + z\sigma]$. With probability at least $1 - \delta - 2n\bar{\Phi}(z)$,*

$$\mathrm{bias}(v) \ \le\ \min_{\lambda\in\Lambda}\Big\{\tfrac{1}{\lambda}\big[\log m + \log \widehat{G}_\uparrow(\lambda) - \tfrac{1}{2}\lambda^2\sigma^2\big] - \big(\bar{X} - \varepsilon_n\big)\Big\}, \tag{6}$$

where $\bar{X}$ is the probe mean and ε_n is a lower-confidence radius on $f(v) = \mathbb{E}[X]$. The level δ is split across the $|\Lambda| + 1$ confidence events at this node, and $2n\bar{\Phi}(z)$ is the probability that some probe leaves the truncation range. The bound needs only $O(|\Lambda|)$ running moments per node (Appendix A).

Algorithm 1 Aggregation-bias certificate at a node v. EB-UPPER and EB-LOWER are the empirical-Bernstein confidence bounds (Maurer & Pontil (2009)).

Require: probes $X_1, \dots, X_n$ from random paths under v; width $m = |\mathcal{L}(v)|$; noise scale σ; grid Λ; level δ

Ensure: $\widehat{\mathrm{bias}}(v) \geq \mathrm{bias}(v)$ with probability at least $1 - \delta$

$\bar{X} \leftarrow \frac{1}{n}\sum_{i \leq n} X_i$ ▷ probe mean, an unbiased estimate of $f(v)$

$\varepsilon_n \leftarrow \text{EB-LOWER}\big(X_{1:n},\, \delta/(|\Lambda|+1)\big)$ ▷ so $\bar{X} - \varepsilon_n \leq f(v)$

for $\lambda \in \Lambda$ **do**

  $\widehat{G}(\lambda) \leftarrow \text{EB-UPPER}\big(e^{\lambda X_{1:n}},\, \delta/(|\Lambda|+1)\big)$ ▷ upper bound on $\mathbb{E}e^{\lambda X}$

  $U(\lambda) \leftarrow \frac{1}{\lambda}\big[\log m + \log \widehat{G}(\lambda) - \frac{1}{2}\lambda^2\sigma^2\big] - \big(\bar{X} - \varepsilon_n\big)$ ▷ deconvolve, then apply (5)

end for

return $\widehat{\mathrm{bias}}(v) \leftarrow \min_{\lambda \in \Lambda} U(\lambda)$

Corollary 1. *Run the certificate at a set $\mathcal{V}$ of nodes with n_v probes and level δ_v at node v, where $\sum_{v \in \mathcal{V}} \delta_v \leq \delta$. Then the bounds of Theorem 1 hold simultaneously for all $v \in \mathcal{V}$ with probability at least $1 - \delta - 2\bar{\Phi}(z)\sum_{v \in \mathcal{V}} n_v$. The implementation splits δ evenly across the nodes it probes.*

The truncation level trades coverage against the width of the Bernstein range, and Remark 4 gives the numbers for the implementation's $z = 3$. The proof is mechanized in Lean 4 (Appendix A).

Properties and use. When the leaf means are sub-Gaussian with proxy σ_w, optimizing over λ gives the bound $\sigma_w\sqrt{2\log m}$. This is the classical light-tail rate, obtained here as a certified guarantee rather than the usual heuristic. In the worst case the log-sum-exp approaches the true maximum as λ grows, and the bound degrades to the maximum-minus-mean gap itself. The certificate is therefore valid on every instance and tight on the benign instances.

In a Lipschitz cell the measured within-cell spread obeys $\sigma_w \lesssim L\rho^\ell$. When σ_w instead exceeds this prediction by a constant factor, the certificate has established a smoothness violation in that cell, and we flag it. Under the piecewise-Lipschitz model of Section 3.1, smoothness fails only at the K discontinuities, so a flagged cell contains a discontinuity. Only flagged cells have their bias bonus inflated, from the tight floor $L\rho^\ell$ to the data-driven $\sigma_w\sqrt{2\log m}$. A single discontinuity flags one cell at every level it crosses, so the flagged set has $O(KD)$ cells rather than K, and the guarantees of Section 3.4 are stated in terms of that set.

3.4 GUARANTEES

We prove two guarantees. First, the fixed-budget identification bound quantifies the payoff of discontinuity-guided sampling. Second, the regret bound extends the guarantee of the tree-bandit algorithm of Section 3.2 to the piecewise-Lipschitz case. Each proof verifies the preconditions of a classical result and adds self-contained lemmas for detection and discontinuity accounting. Full proofs are in Appendix B and Appendix C.

Fixed-budget identification. The edge-targeted algorithm first detects the violation cells with cheap probes (Section 3.3). It then runs the optimistic descent with the schedule $\mathrm{spread}(\ell)$ outside the flagged set and with leaf-level certification inside it. Its complexity is

$$H_{\mathrm{edge}} \;=\; c_0(d, \Delta; c_p) \;+\; \sum_{k=1}^{K} c_\ell \cdot h_k, \qquad h_k = \sum_{x \in C_k} \frac{\sigma^2}{\max(\Delta_x, \Delta)^2}, \tag{7}$$

where c_0 is the cost of the structural search over the smooth part, governed by d and counted in cheap probes at cost c_p, and h_k is the fixed-budget hardness of leaf-certifying the k-th violation cell, counted in expensive evaluations at cost c_ℓ. Here Δ is the top-1 gap and $\Delta_x = \mu^\star - \mu(x)$. This clean split into a smooth term and a per-violation term holds on the event that the detector flags exactly the violation cells. The certificate does not deliver that event for free. Assumption 1 below is what guarantees it with probability $1 - \delta_{\mathrm{det}}$, and the δ_{det} term in Theorem 2 charges its failure.

Assumption 1. At the detection sample size, the within-cell spread of every violation cell exceeds the Lipschitz floor of its level by a margin $\gamma > 0$. The margin is needed for one direction only.

A one-sided confidence bound already prevents smooth cells from being flagged, and the margin is what makes every true violation statistically detectable. We write δ_{det} for the probability that this joint detection event fails, either by flagging a smooth cell or by missing a violation cell.

Theorem 2. *Under Assumption 1, there is $\kappa > 0$ such that at budget B the edge-targeted algorithm misidentifies the top leaf with probability*

$$\mathbb{P}(\textit{error}) \;\le\; \delta_{\mathrm{det}} \;+\; \widetilde{O}(N)\,\exp\!\big(-\kappa\, B/H_{\mathrm{edge}}\big). \tag{8}$$

Remark 1. Under Assumption 1, an empirical-Bernstein detector flags exactly the violation cells with probability at least $1-\delta_{\mathrm{det}}$. The smooth phase reduces to finite-arm successive rejects (Audibert et al., 2010) at probe cost. Leaf-certifying a flagged cell that contains $x^\star$ is standard fixed-budget best-arm identification (Kaufmann et al., 2016) with hardness h_k. A budget split and a union bound combine the pieces (Appendix B).

The proof is mechanized in Lean 4 (Appendix A).

Proposition 2. *There is a family of piecewise tree-Lipschitz instances, each with a single violation cell C of m leaves at the penultimate level and hardness $h = \sum_{x\in C} \sigma^2/\Delta_x^2$, on which any algorithm with access to both fidelities and cost budget $B \le c\, c_\ell h/\log m$ misidentifies the top leaf with constant probability. Hence, the leaf-certification component of H_{edge} is unavoidable up to a $\log m$ factor. The bound says nothing about the structural component c_0, whose necessity remains open.*

Remark 2. Within-cell leaf-mean permutations leave every cheap-probe distribution unchanged, since subtree averages and random-path mixtures depend only on the multiset of means. Cheap probes therefore carry no information about which leaf in C is best. The problem reduces to leaf-only fixed-budget identification over m arms with at most B/c_ℓ evaluations, where the lower bound (Carpentier & Locatelli (2016)) applies (Appendix B).

At $K = 0$ the complexity is c_0, and cheap structural search recovers the smooth rate. For sparse K it grows additively, with each violation adding a horizon-independent certification cost. The structure-blind complexity is $H_{\mathrm{blind}} = c_\ell \sum_x \sigma^2/\Delta_x^2$, in which every leaf is evaluated at full cost c_ℓ. Thus, $H_{\mathrm{edge}} \ll H_{\mathrm{blind}}$ exactly when probes are cheap ($c_p \ll c_\ell$) and violations are sparse. As K grows the identified set covers the tree and $H_{\mathrm{edge}} \to H_{\mathrm{blind}}$, matching the empirical crossover in Section 4. The algorithm runs with $\mathrm{spread}(\ell) = L\rho^\ell$ on smooth cells and with the bias on the $O(KD)$ discontinuous cells. Its regret then decomposes into the smooth rate plus a separate term for the discontinuities.

Theorem 3. *Let μ be tree-Lipschitz with near-optimality dimension d except at K dispersed discontinuities of height ≤ 1, and let $\Delta_{\min}$ be the smallest positive sub-optimality gap among jump cells. On the successful-detection event of Assumption 1, which holds with probability at least $1-\delta_{\mathrm{det}}$,*

$$R_n \;\le\; C_1\, n^{(d+1)/(d+2)} \;+\; C_2\, \frac{K\, D\, \log n}{\Delta_{\min}}, \tag{9}$$

where C_1 depends on (L, ρ, d) and C_2 is a constant.

Remark 3. On smooth cells the standard HOO analysis (Bubeck et al., 2011) remains unchanged and gives the first term of Theorem 3. Each suboptimal discontinuity contributes $O(\log n/\Delta_v)$ regret before its increased index falls below the optimum, and dispersion caps the number of jump cells at KD. For fixed K the jump term never changes the polynomial rate, and $K = 0$ recovers the clean Lipschitz rate (Appendix C).

4 Experiments

We evaluate our method on five downstream task families. Three of these are LLM-serving applications: regional routing, prefix-cache management, and prompt trimming. The other two are top-k identification on a 1000-model group and test-time search on reasoning and code. Table 1 consolidates the headline numbers, and the scope summary that follows maps every null, negative, and partial result onto its theoretical condition. The synthetic validation is in Appendix D and per-benchmark detail is in Appendix E.

	Benchmark	**Ours**	Best baseline	Δ
SELECTION	MMLU routing (6 models) quality @ cost	.977 @ .254	.943 @ .363 (flat)	+.034
	τ-bench retail (5×80 tasks) success, 5-replicate mean	.463 ± .156	.323 ± .213 (flat)	+.14 (4/5)
	Top-k id. (1000 models), B=600 recall@10	.370 ± .018	.126 ± .013 (s.e.)	2.9×
SEARCH	MATH (300, Llama-70B) accuracy	.617	.530	+.087 [+.04, +.14]
	GPQA-Diamond (Llama-70B) accuracy	.424	.313	+.111 [+.04, +.19]
	SWE-bench Verified (261) resolved	.318	.195	+.123 [+.08, +.17]
SYSTEMS	Mooncake trace, B=16, shift blocks/request	5.39	3.02 (LRU)	+78%
	vLLM caching (A10G) TTFT p50 (s)	.267	.961 (off)	3.6×
	vLLM eviction, post-shift TTFT saved (ms)	23.6	12.9 (LRU)	+83%
	LongBench trimming QA-F1 @ matched tokens	.362 @ 3544	.317 @ 3640 (fixed)	+.045
	BBH trimming (27 tasks) accuracy @ tokens	.368 @ 291	.356 @ 332 (fixed)	both axes

Table 1: Headline results at matched metric. In every row green marks the better and red the worse of ours against the baseline. Search rows report the paired same-item Δ with 95% confidence intervals. The τ-bench row compares the two learned routers. Oracle references, per-benchmark detail, and confidence intervals are in Appendix E.

4.1 Setup

Benchmarks. To evaluate our method on the downstream tasks, we chose twelve public benchmarks. Routing uses five benchmarks. RouterBench (Hu et al., 2024) and LLMRouterBench (Li et al., 2026) are offline routing suites with measured per-query quality and cost over fixed model pools, and RouterEval (Huang et al., 2025) provides the same over a 1000-model pool. We run the remaining two, MMLU (Hendrycks et al., 2021a) and τ-bench (Yao et al., 2024), in a live environment. We route over Amazon Web Service Bedrock (Amazon Web Services, 2026) models on MMLU, and we route per call inside τ-bench's long-horizon tool agents. Identification reuses RouterEval's pool. Search uses MATH (Hendrycks et al., 2021b), GSM8K (Cobbe et al., 2021), and GPQA-Diamond (Rein et al., 2024) for reasoning, HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) for code, and SWE-bench Verified (Jimenez et al., 2024) for repository-level issues. Caching uses a real prompt stream, the Mooncake production traces (Qin et al., 2025), request logs from a deployed serving system, and a server (Kwon et al., 2023). Trimming uses MMLU, LongBench (Bai et al., 2024) for long-context QA, and BBH (Suzgun et al., 2023) for few-shot reasoning.

Two criteria drove the benchmark selection. First, each task family tests a different part of the tree model. Routing runs the depth-one algorithm, and caching makes the tree the data structure itself. Trimming tests the arms-within-regions structure, identification runs the full multi-fidelity machinery, and search runs deep trees with real value functions. Second, within each family we pair a benchmark where the theory's precondition holds with one where it plausibly fails. Such examples include MATH against GSM8K tests saturation, RouterBench against LLMRouterBench tests cross-region heterogeneity, and BBH against LongBench and MMLU tests trim headroom.

Models. For the search and trimming tasks, we call five models through Amazon Bedrock (Amazon Web Services, 2026). Collectively, these demonstrate a scaling and capability scale from Llama-3.1-8B (Grattafiori et al., 2024) through Mistral-Large (Mistral AI, 2024), Llama-3.1-70B (Grattafiori et al., 2024), and Nova-Pro (Amazon Artificial General Intelligence, 2024) up to Claude-Sonnet-4.5 (Anthropic, 2025). We chose a ladder rather than a single model because capability is one of the two predicted axes of Section 3.4. The search gain needs a reachable optimum

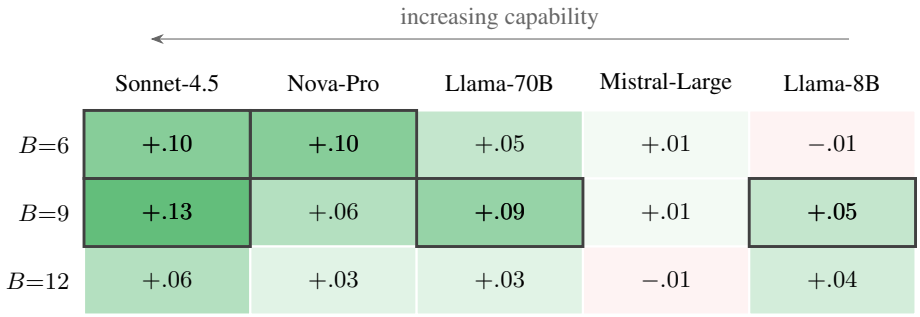


Figure 2: SWE-bench heatmap of capability against budget. Each cell reports the paired gap in resolved rate, Δ, or the value-guided minus best-of-N over 78–80 Verified issues. Shading is proportional to $|\Delta|$, green for positive and red for negative. Bold values with a dark border mark cells whose 95% confidence interval excludes zero. Full confidence intervals are in Table 6.

with headroom, so it should appear for capable models and vanish for weak or saturated ones, and the ladder measures both sides of that prediction.

The live systems experiments run Qwen2.5-7B (Yang et al., 2024) on a vLLM (Kwon et al., 2023) server. The routing pools come from the benchmarks themselves, eleven models in RouterBench and 1000 in RouterEval, replayed from their measured per-query outputs.

Metrics. Results in Table 1 compare methods at the same budget. Routing reports response quality at matched dollar cost. Identification reports recall@10 at a matched cost budget. Search reports task accuracy, or the benchmark's official resolved criterion, at matched generation calls. Caching reports tokens or blocks reused per request and time-to-first-token at matched cache memory. Trimming reports each benchmark's own accuracy at matched input tokens.

Baselines. For routing, the structure-blind baseline is our own method with the regional context removed. The gap measures the value of the regional structure alone. For identification, the baseline is successive elimination (Audibert et al., 2010). For search, the baseline is best-of-N, the standard test-time-compute method in the repeated-sampling literature (Brown et al., 2024; Snell et al., 2024). For caching, the baselines are LRU and LFU. LRU is the is the standard eviction policy of serving stacks, vLLM's PagedAttention (Kwon et al., 2023) and SGLang's RadixAttention (Zheng et al., 2024), and LFU is the standard frequency-based alternative in the caching literature. For trimming, the baselines are fixed trim levels at matched input tokens, so no gain can be explained by allowing more context.

4.2 Results

Table 1 summarizes the key experimental results. The regional router's cost and quality frontier improves every fixed model on RouterBench and reaches near-oracle quality on live MMLU. Hierarchical identification returns 2–3× the recall of structure-blind baselines at low-to-mid budgets. Value-guided search resolves 1.6× as many SWE-bench issues as best-of-N at matched compute, and gains $+0.087$ on MATH and $+0.111$ on GPQA. The adaptive cache beats the eviction policy within the serving engine (5.39 against 3.02 blocks reused per request on the Mooncake trace) and adaptive trimming beats every fixed trim at matched tokens.

Routing. Regional structure beats structure-blind learning wherever the regions are heterogeneous. Routing instantiates the depth-one algorithm of Section 3.2.

The structure-blind baseline, which we call flat, is the same learner restricted to a single region. On RouterBench the regional router's cost and quality frontier dominates every fixed model (Figure 8). On live MMLU it reaches near-oracle quality at the lowest cost of any realizable policy. Only the truth-based per-prompt oracle is cheaper. Inside long-horizon τ-bench agents, the regional learner beats the flat learner in 4 of 5 independent replicates, with mean success 0.463 against 0.323. Terminal-reward credit assignment makes single replicates noisy, so the gap is wide, $+0.14 \pm 0.27$, and all replicates appear in Appendix E.

Top-k identification. We arrange RouterEval's 1000-model pool as a branching-10, depth-3 tree. Here a leaf is a model, and an internal node is a group of similar models whose cheap probe returns the clustered group average. We run optimistic descent with the certified data-driven spread and discontinuity-guided targeting and the certificate's pre-pass added to the budget. The tree gives 2–3$\times$ the recall@10 of the structure-blind baselines at low-to-mid budgets, 0.253 against 0.124 at B=300 and 0.370 against 0.126 at B=600, and the edge narrows to 1.3$\times$ at the largest budget tested, 0.563 against 0.421 at B=2400 (Figure 8, Table 4). Absolute recall stays modest, since top-10 of 1000 is hard for every method, so the claim is the relative win.

Test-time search. Value-guided search beats best-of-N in the regimes predicted by the theory, namely when there is reachable headroom and an informative probe. Here a leaf is a complete trace graded against ground truth, and an internal node is a partial trace scored by a self-consistency or execution probe. The leaf grade is expensive and unbiased, while the probe is cheap and biased. We match budgets in generation calls and report paired per-item bootstrap intervals. Reasoning that is reachable, but unreliable, improves by $+0.087$ on MATH and $+0.111$ on GPQA (Figure 10). Repository-level code improves as well. On SWE-bench the gain is $+0.123$ over 261 issues (Table 5), and the method resolves 1.6$\times$ as many issues as best-of-N, rising to 1.8–2.2$\times$ in the significant tight-budget sweep cells (Table 6). The heatmap in Figure 2 bears out both predicted axes. The gain shrinks as the budget grows, and it appears only for capable models. The sharpest evidence is within-model. With the model and the probe held fixed, the paired gain moves from -0.005 on GSM8K to $+0.333$ on MATH to $+0.359$ on GPQA (Table 6).

Measuring the prior. We run the search to log each node's cheap probe value and its true graded value. The probe–truth correlation measures the smooth backbone, and the fraction of pivotal steps measures the violation count K. On MATH about 6% of steps are pivotal, and on those the cheap probe follows the best continuation 73% of the time against 33% for chance (Table 7). On SWE-bench patches, the execution probe correlates with the truth at $\rho = 0.86$ over 60 issues and 540 candidate patches. 10% of steps are pivotal, and the probe follows the truly best continuation on all 18 pivotal steps, again against 33% for chance (Table 7).

Serving systems. In prefix caching the token-prefix tree is the cache itself, which is an ancestor-closed subtree under a memory budget. For trimming the arms are trim levels and the regions are task categories. For all serving experiments, we use real request streams, production traces, or a live server. The adaptive cache matches LFU and the offline optimum on a stationary workload, re-adapts faster than LFU through a popularity shift, and matches it again at the post-shift steady state (Figure 11). It beats the engine-default LRU throughout, including on the real Mooncake tool-agent trace, 5.39 against LRU's 3.02 blocks per request at B=16 (Table 8). On a vLLM server, prefix caching cuts the median 3.6$\times$ (Table 8), and a GPU-calibrated replay shows our eviction roughly doubling the engine-default LRU's realized saving after the shift (Table 8). On BBH the adaptive trim strictly dominates every fixed trim on both axes (Table 9). On LongBench it beats every fixed trim at matched tokens, 0.362 against 0.317 QA-F1 at about 3.6k tokens, while no prefix trim, ours included, recovers full-context F1. The per-item oracle reaches 0.55 with 39% of the tokens, so the remaining headroom lives at finer-than-task granularity. Replacing positional prefix truncation with retrieval-scored chunk selection at the same budgets lifts every fixed trim, 0.417 against 0.317 at the 3.6k budget and 0.369 against 0.235 at 1.1k, and moves the adaptive point to 0.407 F1 at 2,701 tokens, within 0.03 of full context at 37% of the tokens.

Scope of the advantage. The advantage stops at the boundaries the theory names. Each null or negative lands where a condition of Section 3.4 fails, and together they trace the operating regime. When regions are homogeneous there is no structure to exploit. RouterEval ties the flat learner, and on LLMRouterBench a single cheap model Pareto-dominates every router. When a task is saturated there is nothing left to route. On GSM8K the gap is nearly zero, and on single-function code the shifts are $+0.01$ to $+0.02$ and -0.04 to -0.07, and none of these shifts is significant (Table 6). The probe is not to blame, since a multi-test probe correlates at $\rho = 0.96$ and still does no better than a single assert. When the optimum is out of reach, no search strategy can find it. On MATH, llama-3.1-8B loses 0.147. The hardest SWE-bench tier shows the same floor at small budgets. A pre-registered run on all 42 "1–4 hours" Verified issues left both arms near zero, at $\Delta = 0.000$ and $+0.024$, neither difference significant (Appendix E). A larger budget changes the picture, and the

SWE-bench gain stands at $+0.062$ $[-0.01, +0.14]$ at $B{=}12$. When the workload is stationary or prefixes barely overlap, an eviction policy has nothing to learn, and no policy separates from the others on the Mooncake conversation trace. When a benchmark leaves no room to trim, trimming cannot help, and on MMLU trimming beats only the verbose default.

5 Related Work

Unknown smoothness and certification. Hierarchical bandit methods differ in how they obtain the smoothness information used by their optimism bonus. HOO (Bubeck et al., 2011) assumes a known schedule, while POO (Grill et al., 2015) searches over a family of candidate schedules. Locatelli & Carpentier (2018) show that adaptation to unknown smoothness is impossible from leaf evaluations alone. Our setting provides additional feedback: random-path probes sample the distribution of leaf values within a cell, allowing us to bound its maximum-minus-mean aggregation bias directly from data (Section 3.3). Thus, rather than assuming or searching over a global schedule, the learner can certify locally where the smoothness prior is valid.

Piecewise-smooth and multi-fidelity optimization. Dispersion allows optimization of piecewise-smooth objectives when discontinuities are sufficiently sparse (Balcan et al., 2018). Our method differs in that it detects the cells where the smoothness prediction fails and concentrates expensive evaluations there. This yields a natural multi-fidelity decomposition: cheap subtree averages are used for structural search over smooth regions, while flagged cells reduce to leaf-level best-arm identification (Kandasamy et al., 2016; Audibert et al., 2010; Kaufmann et al., 2016). Our lower bound (Proposition 2) shows that this targeted leaf-level cost is unavoidable up to logarithmic factors.

LLM inference and serving. Several LLM inference problems expose hierarchical structure naturally. Routing chooses among models or model groups (Ong et al., 2024), prefix caching organizes requests by shared token prefixes (Zheng et al., 2024; Kwon et al., 2023), and test-time search branches from shared partial generations. Prompt trimming similarly compares related compressed contexts (Jiang et al., 2023). We treat these as instances of the same cost-budgeted multi-fidelity tree optimization problem rather than as separate application-specific settings.

6 Discussion

This paper shows that model routing, prefix-cache management, prompt trimming, top-k identification, and test-time search share a common structure: a budgeted tree bandit with cheap biased probes and expensive unbiased leaf evaluations. CANOPY bounds the resulting aggregation bias from data and concentrates expensive evaluations where the smoothness prior fails. It improves over structure-blind search when probes are cheap and violations are sparse, a regime reflected across twelve public benchmarks. Extensions to metric spaces, spherical embedding hierarchies, and DAGs appear in Appendix F.

Reproducibility Statement

Full proofs for all theoretical results appear in the appendix, and the statements marked as formally verified are accompanied by their Lean 4 entry-point theorems (Appendix A). Experimental details, including benchmarks, models, budgets, hyperparameters, and the number of replicates, are given in Section 4 and the appendix. The implementation of CANOPY and the scripts used to produce every figure and table will be released as open-source code upon publication.

The Use of Large Language Models

LLM-based tools were used during the writing process for editing and polishing prose. All technical content, proofs, experimental design, and analysis were developed and verified by the authors, who take full responsibility for the paper's contents.

REFERENCES

Amazon Artificial General Intelligence. The Amazon Nova family of models: Technical report and model card. https://www.amazon.science/publications/the-amazon-nova-family-of-models-technical-report-and-model-card, 2024. Technical report.

Amazon Web Services. Amazon bedrock. https://aws.amazon.com/bedrock/, 2026. Accessed September 2026.

Anthropic. Claude sonnet 4.5. https://www.anthropic.com/claude/sonnet, 2025. Model card.

Jean-Yves Audibert, Sébastien Bubeck, and Rémi Munos. Best arm identification in multi-armed bandits. In *Conference on Learning Theory (COLT)*, 2010.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Mohammad Gheshlaghi Azar, Alessandro Lazaric, and Emma Brunskill. Online stochastic optimization under correlated bandit feedback. In *International Conference on Machine Learning (ICML)*, 2014.

Yushi Bai et al. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of ACL*, 2024. arXiv:2308.14508.

Maria-Florina Balcan, Travis Dick, and Ellen Vitercik. Dispersion for data-driven algorithm design, online learning, and private optimization. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2018.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, et al. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Sébastien Bubeck, Rémi Munos, Gilles Stoltz, and Csaba Szepesvári. X-armed bandits. *Journal of Machine Learning Research*, 12:1655–1695, 2011.

Alexandra Carpentier and Andrea Locatelli. Tight (lower) bounds for the fixed budget best arm identification bandit problem. In *Conference on Learning Theory (COLT)*, 2016.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Zhuoming Chen et al. Sparse attention for efficient LLM reinforcement learning. OpenReview preprint, 2026. URL https://openreview.net/forum?id=vMrhhCAgCA.

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

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Jean-Bastien Grill, Michal Valko, and Rémi Munos. Black-box optimization of noisy functions with unknown smoothness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations (ICLR)*, 2021a.

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

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. RouterBench: A benchmark for multi-LLM routing system. *arXiv preprint arXiv:2403.12031*, 2024.

Zhongzhan Huang et al. Routereval: A comprehensive benchmark for routing llms to explore model-level scaling up in llms. In *Proceedings of EMNLP*, 2025. arXiv:2503.10657.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023. arXiv:2310.05736.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations (ICLR)*, 2024.

Kirthevasan Kandasamy, Gautam Dasarathy, Junier Oliva, Jeff Schneider, and Barnabás Póczos. Gaussian process bandit optimisation with multi-fidelity evaluations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Emilie Kaufmann, Olivier Cappé, and Aurélien Garivier. On the complexity of best-arm identification in multi-armed bandit models. *Journal of Machine Learning Research*, 17:1–42, 2016.

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 *ACM Symposium on Operating Systems Principles (SOSP)*, 2023.

Hao Li, Yiqun Zhang, Zhaoyan Guo, Chenxu Wang, Shengji Tang, Qiaosheng Zhang, Yang Chen, Biqing Qi, Peng Ye, Lei Bai, Zhen Wang, and Shuyue Hu. LLMRouterBench: A massive benchmark and unified framework for LLM routing. *arXiv preprint arXiv:2601.07206*, 2026.

Andrea Locatelli and Alexandra Carpentier. Adaptivity to smoothness in X-armed bandits. In *Conference on Learning Theory (COLT)*, 2018.

Andreas Maurer and Massimiliano Pontil. Empirical bernstein bounds and sample variance penalization. In *Conference on Learning Theory (COLT)*, 2009.

Mistral AI. Mistral large. https://mistral.ai/news/mistral-large/, 2024. Model card.

Rémi Munos. Optimistic optimization of a deterministic function without the knowledge of its smoothness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2011.

Rémi Munos. From bandits to Monte-Carlo tree search: The optimistic principle applied to optimization and planning. *Foundations and Trends in Machine Learning*, 7(1):1–129, 2014.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Zhang, Max Willsey, Ion Stoica, et al. RouteLLM: Learning to route LLMs with preference data. *arXiv preprint arXiv:2406.18665*, 2024.

Ruoyu Qin et al. Mooncake: Trading more storage for less computation — a KVCache-centric architecture for serving LLM chatbot. In *USENIX Conference on File and Storage Technologies (FAST)*, 2025. arXiv:2407.00079.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level Google-proof Q&A benchmark. In *Conference on Language Modeling (COLM)*, 2024. arXiv:2311.12022.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Mirac Suzgun et al. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL*, 2023. arXiv:2210.09261.

Michal Valko, Alexandra Carpentier, and Rémi Munos. Stochastic simultaneous optimistic optimization. In *International Conference on Machine Learning (ICML)*, 2013.

Sean Welleck, Amanda Bertsch, Matthew Finlayson, Hailey Schoelkopf, Alex Xie, Graham Neubig, Ilia Kulikov, and Zaid Harchaoui. From decoding to meta-generation: Inference-time algorithms for large language models. *arXiv preprint arXiv:2406.16838*, 2024.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2305.10601.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, et al. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

APPENDIX

A CONCENTRATION AND SELF-CERTIFICATION

This appendix derives the data-driven smoothness certificate of Section 3.3 in full. The goal is a high-probability upper bound on the aggregation bias, where $\mathrm{bias}(v) = \max_{x \in \mathcal{L}(v)} \mu(x) - f(v)$, computed from a finite sequence of random-path probes.

Mechanization convention Where a statement is machine-checked, the main text says so below it, and the list at the end of this paragraph names the entry-point theorem in our Lean 4 development. The mechanization certifies the mathematical claim under its stated hypotheses. The Gaussian deconvolution is itself mechanized. The remaining cited results, empirical-Bernstein concentration, successive rejects, and HOO, enter the mechanized compositions as named hypotheses at exactly the point they are invoked. The full-theorem algebra is additionally checked symbolically and on 20k randomized exact instances. The Lean sources ship as supplementary material. The four mechanized main-text statements and their entry points:

- Theorem 1: Steps 1 and 2 and the full inequality chain, with the empirical-Bernstein confidence events supplied as hypotheses.
- The light-tail rate of Section 3.3: the optimizer's value $\sigma_w \sqrt{2 \log m}$ and its minimality over all $\lambda > 0$.
- Theorem 2: budget-split algebra and union-bound synthesis, with the per-phase successive-rejects guarantees supplied as hypotheses.
- Theorem 3: the dispersion count ($\leq K$ jump cells per level) and the jump-regret summation, with the per-cell UCB visit bound supplied as a hypothesis.

Remark 4. The Bernstein step needs bounded $e^{\lambda X}$, so the range is taken over $[-z\sigma,\, 1 + z\sigma]$ for a truncation level z, and the implementation uses $z = 3$. A Gaussian probe escapes this range with probability $2\bar{\Phi}(z)$, which is the $2n\bar{\Phi}(z)$ coverage term in Theorem 1 after a union bound over n probes. The escape probability is about 0.27% per probe at z=3. A larger z tightens it at the cost of a wider Bernstein range.

A.1 SETUP AND ASSUMPTIONS

Consider a node v whose subtree contains a set of leaves $\mathcal{L}(v)$ with size $m = |\mathcal{L}(v)|$. Each leaf $x \in \mathcal{L}(v)$ has an unknown true mean reward $\mu(x) \in [0, 1]$, and the true subtree average is $f(v) = \frac{1}{m} \sum_{x \in \mathcal{L}(v)} \mu(x)$.

A probe of v selects a leaf L uniformly at random from $\mathcal{L}(v)$ and observes the noisy reward $X = \mu(L) + \eta$. The certificate consumes n independent probes $X_1, \ldots, X_n$ drawn this way.

We assume the noise η is independent of the leaf draw L and distributed as $\mathcal{N}(0, \sigma^2)$. The Gaussian form matters. The deconvolution in Step 2 below divides the observed MGF by the noise MGF, and that division yields an upper bound on the leaf-mean MGF only when the noise MGF is known exactly. A sub-Gaussian upper bound on the noise MGF would give a lower bound on the leaf-mean MGF instead, which is the wrong direction for a certificate. Extending the argument to general sub-Gaussian noise is left to future work.

A.2 FORMAL PROOF OF THEOREM 1

Theorem 4 (Data-Driven Aggregation-Bias Certificate)**.** *Fix a node v. Let Λ be a finite grid of positive constants $\lambda > 0$. Assume the leaf means satisfy $\mu(x) \in [0, 1]$ and the noise is $\eta \sim \mathcal{N}(0, \sigma^2)$. Truncate each probe to the range $[-z\sigma,\, 1 + z\sigma]$ for a level $z > 0$, and write $X_{max} = 1 + z\sigma$ for its upper end. With probability at least $1 - \delta - 2n\bar{\Phi}(z)$, simultaneously over all $\lambda \in \Lambda$, the aggregation bias $\mathrm{bias}(v)$ satisfies:*

$$\mathrm{bias}(v) \leq \min_{\lambda \in \Lambda} \left\{ \frac{1}{\lambda} \left[\log m + \log \widehat{G}_{\uparrow}(\lambda) - \frac{1}{2} \lambda^2 \sigma^2 \right] - \left(\bar{X} - \varepsilon_n \right) \right\}$$

where $\widehat{G}_{\uparrow}(\lambda)$ is an empirical Bernstein upper confidence bound on $\mathbb{E}[e^{\lambda X}]$, $\bar{X}$ is the empirical probe mean, and ε_n is the empirical-Bernstein lower-confidence radius on $f(v) = \mathbb{E}[X]$, so that $\bar{X} - \varepsilon_n \le f(v)$ on the mean's confidence slice, matching Theorem 1.

Proof. The proof proceeds in three distinct steps: bounding the maximum via the soft-maximum, deconvolving the observation noise, and applying empirical concentration.

Step 1: The Log-Sum-Exp Bound.
By the definition of the maximum and the monotonicity of the exponential function, for any $\lambda > 0$, the maximum leaf value is bounded by the log-sum-exp function:

$$\max_{x \in \mathcal{L}(v)} \mu(x) \le \frac{1}{\lambda} \log \left(\sum_{x \in \mathcal{L}(v)} e^{\lambda \mu(x)} \right)$$

Multiplying and dividing the argument of the logarithm by m, we rewrite the sum as an expectation over a uniformly random leaf $L \sim \mathrm{Unif}(\mathcal{L}(v))$:

$$\max_{x \in \mathcal{L}(v)} \mu(x) \le \frac{1}{\lambda} \log \left(m \cdot \frac{1}{m} \sum_{x \in \mathcal{L}(v)} e^{\lambda \mu(x)} \right) = \frac{1}{\lambda} \left[\log m + \log \mathbb{E}_L[e^{\lambda \mu(L)}] \right]$$

Subtracting the subtree average $f(v)$ from both sides yields a bound on the aggregation bias:

$$\mathrm{bias}(v) \le \frac{1}{\lambda} \left[\log m + \log M_\mu(\lambda) \right] - f(v)$$

where $M_\mu(\lambda) = \mathbb{E}_L[e^{\lambda \mu(L)}]$ is the MGF of the true leaf-mean mixture.

Step 2: Noise Deconvolution.
The algorithm does not observe $\mu(L)$ directly, but rather $X = \mu(L) + \eta$. Because L and η are independent, the MGF of the observation X factors perfectly:

$$\mathbb{E}[e^{\lambda X}] = \mathbb{E}_L[e^{\lambda \mu(L)}] \cdot \mathbb{E}_\eta[e^{\lambda \eta}]$$

Under the assumption that $\eta \sim \mathcal{N}(0, \sigma^2)$, its MGF is exactly $e^{\lambda^2 \sigma^2 / 2}$. Rearranging to isolate the unknown leaf-mixture MGF, we obtain:

$$M_\mu(\lambda) = \mathbb{E}[e^{\lambda X}] e^{-\frac{1}{2}\lambda^2 \sigma^2}$$

Step 3: Empirical Concentration.
We estimate $\mathbb{E}[e^{\lambda X}]$ from the n probes $X_1, \ldots, X_n$. Define $Y_i = e^{\lambda X_i}$. The empirical Bernstein inequality (Maurer & Pontil, 2009) needs a bounded range, and a Gaussian probe is unbounded, so we work on the event E_{rng} that every probe lies in $[-z\sigma,\, 1 + z\sigma]$. Since $\mu(L) \in [0, 1]$, a probe leaves this range only when $|\eta| > z\sigma$, which has probability $2\bar{\Phi}(z)$, so $\mathbb{P}(E_{\mathrm{rng}}^c) \le 2n\bar{\Phi}(z)$ by a union bound over the n probes. On E_{rng} each $Y_i \in [0, e^{\lambda X_{max}}]$ with $X_{max} = 1 + z\sigma$, and the inequality applies. Let $\bar{Y} = \frac{1}{n}\sum_{i=1}^n Y_i$ be the sample mean and $\hat{V}_n = \frac{1}{n(n-1)} \sum_{1 \le i < j \le n} (Y_i - Y_j)^2$ the sample variance. For a fixed λ, with probability at least $1 - \delta/(|\Lambda| + 1)$,

$$\mathbb{E}[e^{\lambda X}] \le \bar{Y} + \sqrt{\frac{2\hat{V}_n \log((|\Lambda| + 1)/\delta)}{n}} + \frac{7 e^{\lambda X_{max}} \log((|\Lambda| + 1)/\delta)}{3(n-1)} := \widehat{G}_{\uparrow}(\lambda)$$

This accounts for $|\Lambda|$ of the $|\Lambda|+1$ confidence slices. The last slice covers the mean. The true average satisfies $f(v) = \mathbb{E}[X]$, and we estimate it by the sample mean $\bar{X}$ with an empirical-Bernstein lower-confidence radius ε_n at level $\delta/(|\Lambda| + 1)$. The mean needs its own slice because subtracting $\bar{X}$ alone would under-cover whenever $\bar{X} > f(v)$, which is the $|\Lambda| + 1$ split stated in Theorem 1. A union bound over the $|\Lambda| + 1$ confidence events and the range event E_{rng} gives coverage at level $1 - \delta - 2n\bar{\Phi}(z)$ for the fixed node v. To hold simultaneously over a set $\mathcal{V}$ of probed nodes, we allocate δ_v across nodes with $\sum_v \delta_v \le \delta$ and repeat the per-node argument unchanged, which is Corollary 1. The range events also add up across nodes, giving the $2\bar{\Phi}(z)\sum_v n_v$ term there. The implementation splits δ across the nodes actually probed. Substituting the deconvolution bound of Step 2 and $\widehat{G}_{\uparrow}(\lambda)$ into the bound from Step 1 gives the stated inequality. □

B Complexity of Fixed-Budget Identification

This appendix proves the fixed-budget identification guarantee, Theorem 2. The edge-targeted algorithm uses the certificate of Appendix A to partition the tree into smooth cells and jump cells, and it allocates the budget accordingly.

B.1 Complexity Definitions

Let $\Delta = \mu^* - \max_{x \neq x^*} \mu(x)$ be the top-1 gap, and $\Delta_x = \mu^* - \mu(x)$ be the sub-optimality gap for any leaf x.

The identification complexity has two components, one per query fidelity.

Structural complexity c_0. This is the hardness of localizing the optimum's cell over the smooth part of the tree with cheap probes at cost c_p. We make it concrete by a reduction. Fix the certified resolution ℓ^*, the deepest level at which the certificate validates $\mathrm{spread}(\ell)$ off the flagged set, and let $\mathcal{U}$ be the level-ℓ^* cells that survive sound pruning. Each $u \in \mathcal{U}$ is an arm whose value is its subtree average $f(u)$, observed by cheap probes with Gaussian noise of scale σ. The cell gaps are $\Delta_u = \max_{u'} f(u') - f(u)$, deflated by the certified within-cell spread. Define

$$c_0 \;=\; c_p \sum_{u \in \mathcal{U}} \frac{\sigma^2}{\max(\Delta_u - 2\,\mathrm{spread}(\ell^*),\, \Delta)^2},$$

the standard finite-arm fixed-budget complexity over the certified-level cells, counted at probe cost. This is the two-phase structure the implementation uses, a coarse descent followed by a race over the survivors.

Leaf-certification complexity h_k. This is the hardness of identifying the best leaf within a detected jump cell C_k with expensive leaf evaluations at cost c_ℓ. For the k-th jump cell it is the standard unstructured best-arm identification complexity

$$h_k = \sum_{x \in C_k} \frac{\sigma^2}{\max(\Delta_x, \Delta)^2}.$$

The edge-targeted complexity is the sum of the structural search and the localized certifications,

$$H_{\mathrm{edge}} = c_0(d, \Delta; c_p) + \sum_{k=1}^{K} c_\ell \cdot h_k$$

B.2 Formal Proof of Theorem 2 (Main Text)

Theorem 5 (Fixed-Budget Error)**.** *Let B be the total cost budget. Suppose the function μ is piecewise tree-Lipschitz with K dispersed discontinuities satisfying the detectability margin (Assumption 1). The edge-targeted algorithm misidentifies the optimal leaf x^* with probability bounded by:*

$$\mathbb{P}(\widehat{x} \neq x^*) \leq \delta_{\mathrm{det}} + O(|V|) \exp\left(-\kappa \frac{B}{H_{\mathrm{edge}}}\right)$$

where δ_{det} is the detection-failure probability, $|V|$ is the number of explored nodes, and $\kappa > 0$ is a universal constant. Since a complete b-ary tree with N leaves has fewer than $2N$ nodes, $O(|V|)$ is the $\widetilde{O}(N)$ of Theorem 2.

Proof. The failure event $E_{fail} = \{\widehat{x} \neq x^*\}$ can be bounded by analyzing three constituent events: detection failure, structural search failure, and leaf certification failure.

Step 1: The Detection Event.
Let E_{det} be the event that the detector flags every violation cell and no smooth cell. The two directions need different arguments. For no false flags, the one-sided empirical-Bernstein confidence on each cell's within-cell spread controls the probability that a smooth cell's estimate exceeds the

Lipschitz floor, and a union bound over the $O(b^l)$ tested cells gives failure probability at most $\delta_{\text{det}}/2$. For no missed violations, the argument is not free. A violation whose spread exceeds the floor by an arbitrarily small amount cannot be caught at any finite sample size. Assumption 1 supplies the margin, every violation cell's spread exceeds the floor by $\gamma > 0$, and then the same Bernstein concentration at the detection sample size $n_{det} = O(\log(KD/\delta_{\text{det}})/\gamma^2)$ keeps every violation's estimate above the floor with probability at least $1 - \delta_{\text{det}}/2$. Together $\mathbb{P}(E_{det}^c) \le \delta_{\text{det}}$, and on E_{det} the flagged set is exactly the $O(KD)$ violation cells.

Step 2: Structural Search off the Flagged Set.
On E_{det} the unflagged cells obey the smoothness schedule $\text{spread}(l) = L\rho^l$, so every unflagged level-ℓ^* cell u satisfies the certified bound $|\max_{x\in u} \mu(x) - f(u)| \le \text{spread}(\ell^*)$. The smooth phase is then a finite-arm fixed-budget best-arm identification over the arms $\mathcal{U}$. A cheap probe of u is an unbiased Gaussian observation of $f(u)$, and identifying the cell containing x^* requires separating cells whose deflated gaps are $\max(\Delta_u - 2\,\text{spread}(\ell^*), \Delta)$. Applying the fixed-budget successive-rejects guarantee of Audibert et al. (2010) to these arms with B_{smooth}/c_p observations gives

$$\mathbb{P}(E_{smooth}^c \mid E_{det}) \le O(|\mathcal{U}|) \exp\left(-\kappa \frac{B_{smooth}}{c_0}\right),$$

with c_0 as defined above and κ universal. The descent that produces $\mathcal{U}$ uses sound pruning only, and on E_{det} it never discards the optimum's branch, so its cost is counted inside B_{smooth} and it only shrinks $|\mathcal{U}|$.

Step 3: Leaf Certification on the Flagged Set.
If x^* lies within one of the K flagged jump cells, the algorithm must identify it using expensive leaf evaluations. The algorithm runs a successive elimination or uniform exploration routine restricted to the leaves of the K flagged cells. Let B_{cert} be the portion of the budget spent on these expensive evaluations. Because the K cells are disjoint, the hardness of identifying x^* among them is exactly the sum of their individual hardnesses. The probability of returning a suboptimal leaf from the flagged set is bounded by standard unstructured best-arm identification guarantees (Audibert et al., 2010):

$$\mathbb{P}(E_{cert}^c \mid E_{det}) \le O(K) \exp\left(-\kappa \frac{B_{cert}}{\sum_{k=1}^K c_\ell h_k}\right)$$

Step 4: Budget Allocation and Synthesis.
The total budget is $B = B_{smooth} + B_{cert}$. The algorithm statically or dynamically allocates the budget proportional to the empirical complexities of the respective phases. To minimize the maximum of the two error probabilities, the budget splits such that the exponents are equalized:

$$\frac{B_{smooth}}{c_0} = \frac{B_{cert}}{\sum_k c_\ell h_k} = \frac{B}{c_0 + \sum_k c_\ell h_k} = \frac{B}{H_{\text{edge}}}$$

Applying a union bound over the failure modes, the total probability of misidentification is:

$$\mathbb{P}(\widehat{x} \ne x^*) \le \mathbb{P}(E_{det}^c) + \mathbb{P}(E_{smooth}^c \mid E_{det}) + \mathbb{P}(E_{cert}^c \mid E_{det})$$

Substituting the bounds derived in Steps 1 through 3:

$$\mathbb{P}(\widehat{x} \ne x^*) \le \delta_{\text{det}} + O(|V|) \exp\left(-\kappa \frac{B}{H_{\text{edge}}}\right) + O(K) \exp\left(-\kappa \frac{B}{H_{\text{edge}}}\right)$$

Because $K \le |V|$ (the number of jump cells cannot exceed the number of explored nodes), we absorb the $O(K)$ term into the $O(|V|)$ leading factor, yielding the final bound:

$$\mathbb{P}(\widehat{x} \ne x^*) \le \delta_{\text{det}} + O(|V|) \exp\left(-\kappa \frac{B}{H_{\text{edge}}}\right)$$

□

B.3 Proof of the Partial Lower Bound (Proposition 2)

Proof. Fix a violation cell C of m leaves at the penultimate level, and consider the instance family $\{\mu_\pi\}$ indexed by permutations π of a fixed gap profile $(\Delta_1, \ldots, \Delta_m)$ within C: leaf x_i of C has

mean $\mu^* - \Delta_{\pi(i)}$ (with $\Delta_{\pi^{-1}(1)} = 0$ the unique optimum), and every leaf outside C has a common constant mean strictly below $\mu^* - \max_i \Delta_i$. All instances in the family share identical leaf means outside C and an identical multiset of means inside C.

Step 1: Cheap probes carry no information about the optimum's identity.
Consider any internal-node probe, whether it returns the subtree average directly or a random-path sample. A direct-average probe of a node v returns $f(v) + \eta$. If $C \subseteq v$ then $f(v)$ depends only on the sum of the means in C, which is permutation-invariant, and if $v \cap C = \emptyset$ it does not depend on π at all. No other case exists, since C sits at the penultimate level and its only descendants are leaves, which are expensive evaluations by definition. A random-path probe of v returns $\mu(L) + \eta$ for L uniform on the leaves under v, and its law is the uniform mixture over those leaves' means, again a function of the multiset only. Hence the joint distribution of any sequence of cheap probes is the same for every π, and cheap observations are ancillary for identifying the $\arg\max$ within C.

Step 2: Reduction to leaf-only identification.
Any algorithm $\mathcal{A}$ with cost budget B makes at most $n_\ell = \lfloor B/c_\ell \rfloor$ leaf evaluations. Conditioning on the ancillary cheap observations, $\mathcal{A}$ induces a leaf-only fixed-budget identification algorithm over the m arms of C with budget n_ℓ. Leaf evaluations outside C are known constants and carry no information about π, and an algorithm that randomizes using the ancillary observations cannot decrease the minimax error. By the fixed-budget lower bound of Carpentier & Locatelli (2016), for every such algorithm there is a permutation π under which

$$\mathbb{P}(\widehat{x} \neq x^*) \;\geq\; c' \exp\!\left(-\frac{c'' \, n_\ell \log m}{h}\right), \qquad h = \sum_i \frac{\sigma^2}{\Delta_i^2},$$

which is a constant whenever $n_\ell \leq c\, h/\log m$, that is, whenever $B \leq c\, c_\ell h/\log m$.

Step 3: What the bound does and does not show.
On this family $H_{\mathrm{edge}} = c_0 + c_\ell h$, and the certification term dominates because the smooth part is flat. The certification component of H_{edge} is therefore necessary up to the $\log m$ factor. The argument says nothing about the structural component c_0, which is negligible on this family by construction, and that is why the lower bound is partial. □

C Regret Analysis in the Piecewise-Lipschitz Regime

This appendix proves the regret bound for the optimistic tree bandit on a piecewise-Lipschitz objective. Inflating the optimism bonus only on the detected discontinuity cells makes the regret decompose into the smooth rate plus an additive penalty in the number of discontinuities.

C.1 The Tree-Dispersion Condition

Let $\mathcal{X}$ be the leaf space with the LCA distance $d(x, y) = \rho^{\text{level}(\text{LCA}(x,y))}$ of Section 3.1. A function $\mu : \mathcal{X} \to [0, 1]$ is globally tree-Lipschitz with constant L if $|\mu(x) - \mu(y)| \leq L d(x, y)$ for all $x, y \in \mathcal{X}$.

In our setting μ is only piecewise tree-Lipschitz. We define this through the dispersion condition of Balcan et al. (2018), adapted to the tree metric.

Definition 1 (Tree-Dispersion)**.** A function μ is piecewise tree-Lipschitz with K discontinuities if there exists a set of boundary points $\mathcal{D} \subset \mathcal{X}$ with $|\mathcal{D}| \leq K$ such that for any cell $\mathcal{C}(v)$ that does not contain a point in $\mathcal{D}$, the restriction of μ to $\mathcal{C}(v)$ is tree-Lipschitz.

Consequently, at any level $l \in \{1, \ldots, D\}$, at most K cells straddle a discontinuity. Summing over all depths, the total number of "jump cells" in the entire tree is bounded by $K \cdot D$.

C.2 Formal Proof of Theorem 3

Theorem 6 (Regret with K Discontinuities)**.** *Assume μ is piecewise tree-Lipschitz with K dispersed discontinuities of height at most 1, and the smooth regions have a near-optimality dimension d. Let the detection mechanism identify the $O(KD)$ jump cells with probability $1 - \delta_{\text{det}}$, and let $\Delta_{\min}$ be the smallest positive sub-optimality gap among jump cells. Conditioned on successful detection, the cumulative regret R_n is bounded by:*

$$R_n \leq C_1 n^{\frac{d+1}{d+2}} + C_2 \frac{KD \log n}{\Delta_{\min}}$$

where C_1 depends on the Lipschitz constant L, discount ρ, and dimension d, and C_2 is a universal constant.

Proof. Let $\mathcal{V}$ be the set of all nodes in the tree. We partition $\mathcal{V}$ into two disjoint sets: the smooth cells $\mathcal{V}_{smooth}$ and the jump cells $\mathcal{V}_{jump}$. By the tree-dispersion condition, $|\mathcal{V}_{jump}| \leq K \cdot D$.

The cumulative regret over n steps can be decomposed based on the type of cell the algorithm commits to. Consistently with the definition in Section 3.1, regret is measured against the value of the committed cell v_t (a random-path probe of v_t has expectation $f(v_t)$, so this coincides with the expected sampled-leaf regret):

$$R_n = \sum_{t=1}^{n} (\mu^* - f(v_t)) = \sum_{t \in \mathcal{T}_{smooth}} (\mu^* - f(v_t)) + \sum_{t \in \mathcal{T}_{jump}} (\mu^* - f(v_t))$$

where $\mathcal{T}_{smooth}$ (resp. $\mathcal{T}_{jump}$) is the set of rounds where the committed cell v_t was a smooth (resp. jump) cell.

Step 1: Bounding Regret on Smooth Cells.
For a node $v \in \mathcal{V}_{smooth}$ the subtree contains no discontinuity, so the aggregation bias $\text{bias}(v)$ is bounded by the schedule $\text{spread}(l) = L\rho^l$. On the successful detection event, the optimistic index $U(v)$ is therefore a valid upper bound for every $v \in \mathcal{V}_{smooth}$.

Because the confidence bounds are valid and the smoothness schedule holds on this subset, the standard HOO analysis of Bubeck et al. (2011) applies unchanged. The number of pulls allocated to suboptimal smooth cells is controlled by the near-optimality dimension d. Summing the regret over $\mathcal{T}_{smooth}$ yields the classical rate

$$\sum_{t \in \mathcal{T}_{smooth}} (\mu^* - f(v_t)) \leq C_1 n^{\frac{d+1}{d+2}}$$

where C_1 absorbs the logarithmic terms and dependencies on L, ρ, and d.

Step 2: Bounding the Discontinuity Penalty.
For a node $v \in \mathcal{V}_{jump}$, the local variation may violate $\mathrm{spread}(l)$. However, the algorithm identifies these cells and inflates their bias bonus to the empirical worst-case bound (Appendix A), so the inflated index $U(v)$ is a valid upper confidence bound on the best leaf under v.

Fix a suboptimal jump cell v with sub-optimality gap $\Delta_v = \mu^* - \max_{x \in \mathcal{C}(v)} \mu(x) > 0$. With the inflated and valid bonus, the standard UCB argument applies to v as to a single arm with reward range $[0, 1]$. After $T_v(t)$ visits its confidence radius is $O(\sqrt{\log t / T_v(t)})$, and once $T_v(t) \geq c \log n / \Delta_v^2$ for a universal c the optimistic index of v falls below μ^* with high probability, so v is no longer selected. Each visit to v incurs instantaneous regret at most $\Delta_v + \mathrm{spread}(l_v)$, which is $O(\Delta_v)$ once the bonus is valid, since the cell's internal spread is then dominated by its gap and the discontinuity height of at most 1 bounds the worst case. The aggregate regret charged to v over the horizon is therefore

$$C_v \;\leq\; O\!\left(\frac{\log n}{\Delta_v^2}\right) \cdot O(\Delta_v) \;=\; O\!\left(\frac{\log n}{\Delta_v}\right).$$

By the tree-dispersion condition $|\mathcal{V}_{jump}| \leq K \cdot D$, so summing over jump cells,

$$\sum_{t \in \mathcal{T}_{jump}} (\mu^* - f(v_t)) \;\leq\; \sum_{v \in \mathcal{V}_{jump}} C_v \;\leq\; C_2 \, \frac{KD \log n}{\Delta_{\min}},$$

with $\Delta_{\min} = \min_{v \in \mathcal{V}_{jump} : \Delta_v > 0} \Delta_v$ and C_2 universal. (A jump cell containing the optimum has $\Delta_v = 0$ and contributes no regret.)

Step 3: Synthesis.
Combining the bounds for $\mathcal{T}_{smooth}$ and $\mathcal{T}_{jump}$ gives the decomposition

$$R_n \leq C_1 n^{\frac{d+1}{d+2}} + C_2 \frac{KD \log n}{\Delta_{\min}}.$$

For any fixed K the jump term grows only logarithmically in n, so the polynomial term dominates as $n \to \infty$. The K dispersed discontinuities change constants and a $\log n$ factor, not the learning rate, and $K = 0$ recovers the clean Lipschitz rate. □

D SYNTHETIC VALIDATION OF THE MULTI-FIDELITY MACHINERY

These controlled studies test the core bandit claims under conditions a fixed real benchmark cannot provide. We sweep the probe/leaf cost ratio, the number of discontinuities K, and the assumed smoothness. Every synthetic result is averaged over many seeds and reported with 95% confidence bands.

The tree prior pays off only when probes are cheap. We run top-5 identification on a 1024-leaf tree over 60 seeds against a strong structure-blind successive-elimination baseline (Figure 3). When an internal probe costs as much as a leaf, the descent is pure overhead and the tree loses. Once probes are at least $2\times$ cheaper the tree overtakes the baseline, and it approaches full recall as probes get cheaper still. Figure 3 plots two hierarchical variants to show what carries the win. The practical beam variant, best-first focusing with beam width 20, overtakes successive elimination once probes are cheap, 0.84 against 0.57 at budget 1500. The sound variant runs exactly the worst-case-valid pruning that Theorem 2 analyzes, and on this smooth family it only matches the baseline, 0.62 against 0.57, because the conservative spread bound prunes little. The empirical advantage therefore comes from the beam focusing. The sound algorithm is the certified never-worse floor, and sharper certified pruning remains open.

Regret versus memory. We run online regret minimization on a 256-leaf tree of depth 4 over a 12k-round horizon and 16 seeds (Figure 4). Fixed-resolution play traces a clean frontier, in which deeper trees have less bias but exponentially more memory. The expand-versus-refine rule reaches near full-resolution regret at about a third of the memory. The variance-aware variant, which estimates the bias from data rather than assuming spread, reaches it at about $6\times$ less and sits below the fixed-depth frontier.

Graceful degradation in the violation count. We measure top-1 accuracy at a fixed tight budget as the number of discontinuities K grows, with branching 4, depth 5, budget 400, $c_p/c_\ell = 0.05$, and 30 seeds (Figure 6). The data-driven hybrid degrades smoothly from near-perfect accuracy when violations are few toward the structure-blind floor as they proliferate, and it strictly dominates both alternatives. Pure assumed smoothness is misled by the spikes, and the blind baseline cannot afford enough leaf evaluations. This is the empirical form of $H_{\mathrm{edge}} = c_0 + \sum_{k \le K} c_\ell h_k$ saturating to H_{blind}.

Finite-state compression. On the infinite-depth tree, with arity 3, a 15k horizon, and the memory-bounded rule, the tree grows from 121 to nearly $800{,}000$ nodes while the explored memory stays near 70 and the regret stays between 663 and 1702. This is the discrete witness of the depth-independent memory in Proposition 1.

Estimating the smoothness. The optimistic descent is inherited, so this ablation isolates the paper's contribution, estimating the local Lipschitz constant from data rather than assuming a schedule. The instance is a heterogeneous-smoothness tree whose left half is smooth and whose right half is rough, so no single global constant fits. Sweeping the assumed constant L traces a U-shaped tuning curve. A constant that is too tight under-explores the rough half and misses the optimum, and a constant that is too loose over-explores everywhere. The data-driven per-subtree estimator of Section 3.3 uses no tuning at all (Figure 5, Table 2). It matches and slightly beats the oracle-tuned fixed constant, 3029 against 3127 final regret at the swept minimum, and it far outperforms a mis-specified tight constant at 4076, because it adapts a local constant per subtree that no global L can match. The win comes from the certified and estimated smoothness, not from the inherited optimistic descent.

From the rate to the data. Two checks connect the regret bound of Section 3.4 to the experiments (Figure 7). First, on a tree-Lipschitz value function the average regret $R(n)/n$ decays toward zero as the horizon grows. The measured cumulative-regret exponent is $\alpha \approx 0.77 < 1$, with the near-optimality dimension estimated from the value function at $\hat{d} \approx 0$, which is the empirical form of the sublinear $\widetilde{O}(n^{(d+1)/(d+2)})$ rate. Second, we ask whether the prior holds on real routing data. A variance decomposition of the RouterBench value function (Hu et al., 2024), over 36k prompts, 11 models, and 86 categories, shows that the region tree captures the exploitable regional signal. The

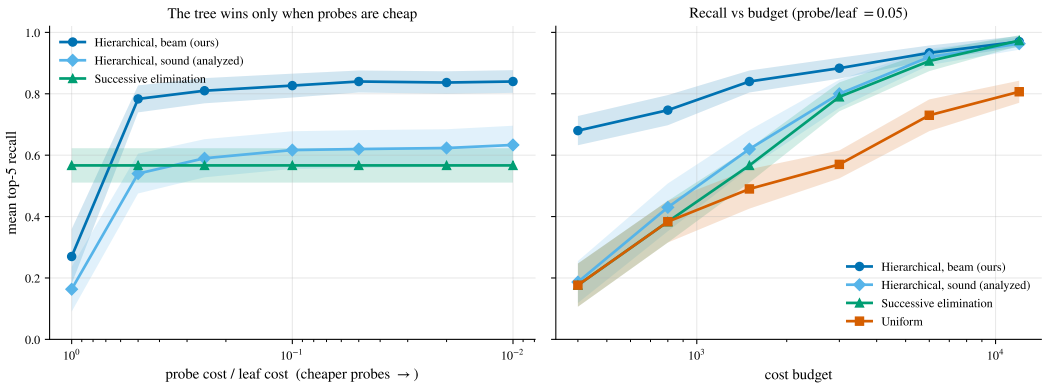


Figure 3: Multi-fidelity top-k on a 1024-leaf tree (60 seeds, 95% CI bands). Left: recall vs. probe/leaf cost ratio at a fixed budget. Right: recall vs. cost budget with cheap probes, for the beam and sound hierarchical variants against successive elimination.

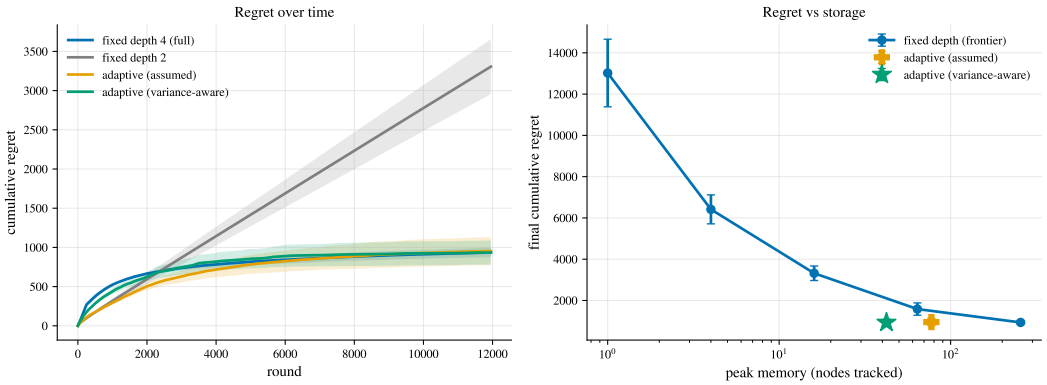


Figure 4: Regret vs. storage, isolated on controlled instances (256 leaves, depth 4, horizon 12k, 16 seeds). The data-driven expand-vs-refine rule reaches near full-resolution regret at $\approx 6\times$ less memory than the fixed-depth frontier.

category tree explains about 31% of the per-prompt value variance, and the benchmark-family tree explains about 53% of the region-value variance. The large remaining within-region variation is per-prompt difficulty, observation noise the bandit averages over. The learner needs only the regional optimum, not the full noisy per-prompt value function (Section 3.1), so the routing value function is tree-Lipschitz at the regional scale rather than finely smooth. The finer almost-tree-K-Lipschitz characterization, in which a partial trace's cheap value predicts its completion and the violations are the pivotal decision steps, lives in the reasoning traces and is reported in Section 4.2.

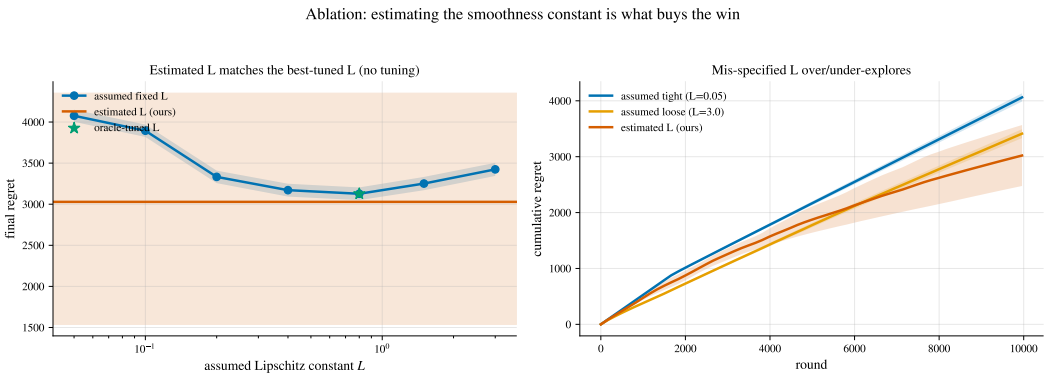


Figure 5: The Lipschitz-constant ablation on a heterogeneous-smoothness tree (10 seeds). The assumed-L tuning curve is U-shaped, while the estimated-L policy (ours) matches the oracle-tuned minimum with no tuning (3029 vs. 3127 final regret).

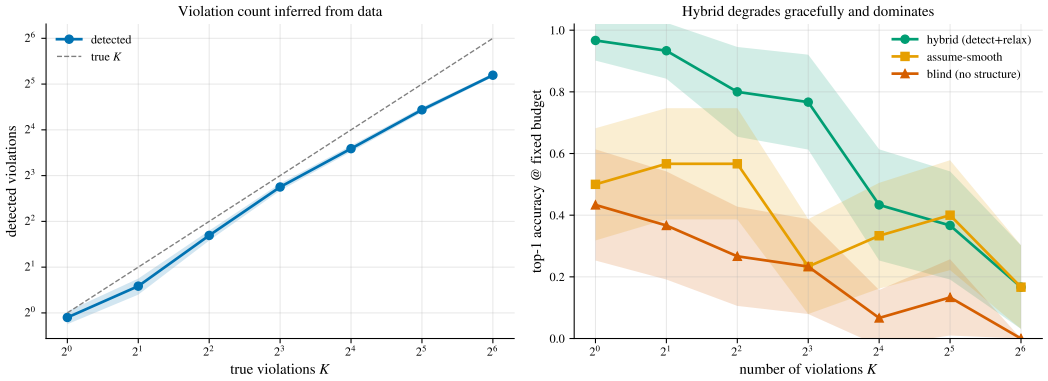


Figure 6: Value of structure vs. number of Lipschitz violations K (30 seeds, 95% CI bands). Left: the violation count is recovered from data. Right: top-1 accuracy at a fixed budget, where the detect-and-relax hybrid degrades gracefully and dominates both baselines.

Policy	Final regret
Assumed L (too tight, $L = 0.05$)	4076
Assumed L (too loose, $L = 3.0$)	3424
Oracle-tuned fixed $L = 0.8$	3127
Estimated L (ours)	**3029**

Table 2: Lipschitz-constant ablation: final regret for mis-specified fixed L, the oracle-tuned fixed L, and the data-driven estimator (ours).

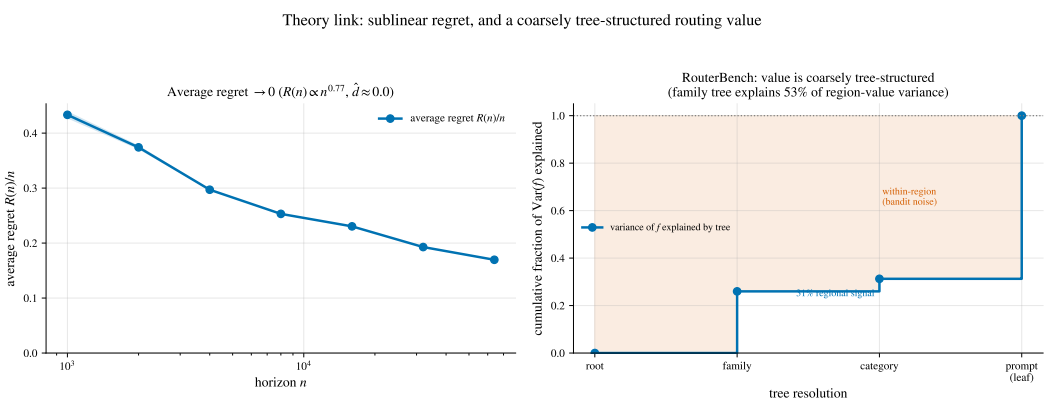


Figure 7: Linking theory to data (8 seeds). Left: on a tree-Lipschitz value function the average regret $R(n)/n$ decays toward zero ($R(n) \propto n^{\alpha}$, $\alpha \approx 0.77 < 1$; $\hat{d} \approx 0$). Right: variance decomposition of the real RouterBench routing value function. The region tree captures the coarse regional signal, and the within-region remainder is per-prompt noise the bandit averages over.

E Additional Experimental Results

This appendix collects the per-benchmark tables, secondary figures, and negative results behind Section 4.

E.1 Routing

Live models on MMLU. We route six Bedrock models spanning roughly a $100\times$ cost gradient, with MMLU subjects as regions. The regional router reaches near-oracle quality at the lowest cost of any realizable policy, and only the truth-based per-prompt oracle is cheaper. It beats both the structure-blind flat learner and the strongest fixed model (Table 3).

Per-call routing inside long-horizon agents. On τ-bench the region is a small discrete difficulty-and-phase feature of the turn, the arm is a model, and the quality signal is the episode's terminal reward, credited to every region-model pull along the episode. A single trial's bootstrap interval captures task-to-task spread but not seed variance, so we ran 5 independent replicates of the two learned routers over the same 80 tasks (Table 3). The regional learner beats the flat learner in 4 of 5 replicates, with mean success 0.463 ± 0.156 against 0.323 ± 0.213 and gap $+0.14 \pm 0.27$. The variance is dominated by which models each learner converges to under terminal-reward credit assignment. In the one reversed replicate the flat learner drifted to expensive models and the regional learner did not. We therefore present τ-bench as directional evidence and let RouterBench and live MMLU carry the routing claim. For completeness, the strongest fixed policy, always claude-sonnet-4.5, attains higher raw success at 0.600 than either learned router, at roughly $1.6\times$ the regional router's realized cost per task (Table 3). The learned routers' advantage on this benchmark is therefore cost-adjusted, not absolute. A per-turn value signal would be tighter and is future work.

E.2 Top-k identification

Table 4 reports the full budget sweep behind the identification row of Table 1 and Figure 8. The pool is RouterEval's 1000 models arranged as a branching-10, depth-3 tree, the metric is mean recall@10 over 20 seeds and 9 evaluation splits, and a probe costs 0.05 of a leaf evaluation. The charged pre-pass that measures the spread is included in our method's cost. Two patterns in the sweep support the theory. First, successive elimination never separates from uniform sampling, staying within one standard error of it at every budget, so the win comes from the tree structure rather than from an elimination schedule. Second, the hierarchical gain is largest at low-to-mid budgets, $2.9\times$ recall at B=600, and narrows to $1.3\times$ at B=2400 as the structure-blind baselines begin to catch up.

E.3 Test-time search

This subsection collects the accuracy-vs-budget curves, per-benchmark tables, the capability ladder, and the tree-Lipschitz instrumentation behind the search and measured-prior experiments.

Pre-registered hard-tier run. A pre-registered follow-up tested whether the relative gain widens on the hardest tier. The protocol was committed to the repository before the run, and the setup covers all 42 "1–4 hours" SWE-bench Verified issues with claude-sonnet-4.5 at matched budgets 6 and 9, with machinery identical to the main sweep. The directional hypothesis failed, and both arms fall to near floor. At B=6 each arm resolves $2/42$, with $\Delta = 0.000$ $[-0.07, +0.07]$. At B=9 the arms resolve $2/42$ and $3/42$, with $\Delta = +0.024$ $[-0.05, +0.10]$, and we do not report ratios, per the pre-registered instability rule. This run measures the unreachable regime of the scope paragraph directly on code. When issues are beyond the model's reach at the given budget, guidance has nothing to steer, exactly as the reachable-headroom condition predicts.

E.4 Prefix caching and systems

This subsection details the serving experiments. The prompt-stream and Mooncake trace tables come first, then the live-server vLLM measurements, which cover caching on and off, the KV-budget frontier, and the GPU-calibrated eviction-policy comparison.

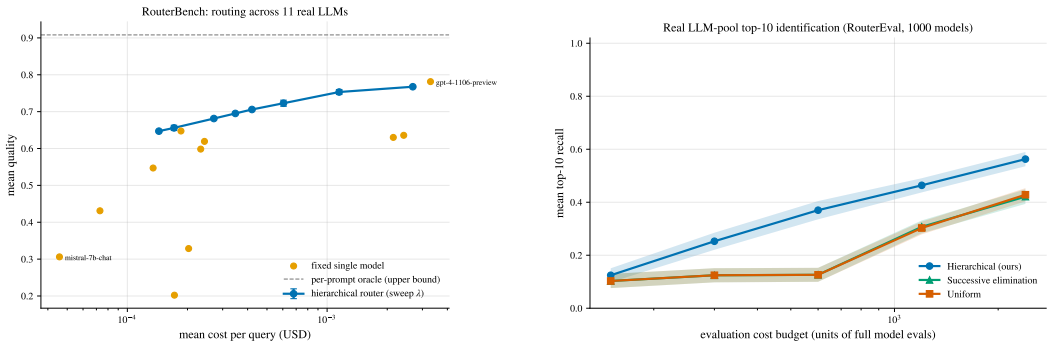


Figure 8: Structure against structure-blind baselines on real model pools. Left: RouterBench routing. Sweeping the cost weight λ traces a router frontier (blue) that dominates every fixed model (orange), and the dashed line is the per-prompt oracle. Right: top-10 recall vs. evaluation cost on RouterEval's 1000-model pool (9 evals $\times$ 20 seeds, 95% CI bands), where the tree bandit dominates both structure-blind baselines at every budget.

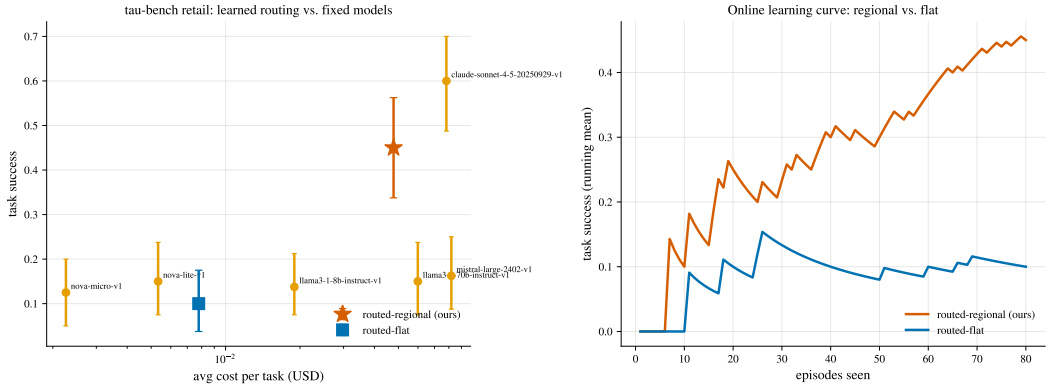


Figure 9: Per-call routing inside long-horizon τ-bench agents. Left: retail cost/quality, where the learned regional router (red star) reaches the fixed-model cloud at low cost. Right: the online curve separates the regional from the flat learner.

E.5 Prompt trimming

Trimming spans three regimes: little headroom on MMLU, a real quality/token tradeoff on LongBench, and strict dominance on BBH.

MMLU: LIVE BEDROCK ROUTING

Policy	Regret	Avg. quality	Rel. cost/query
Hierarchical (ours)	472 ± 3	0.977 ± 0.001	0.254 ± 0.002
Flat (structure-blind)	878 ± 8	0.943 ± 0.001	0.363 ± 0.006
Best single (oracle ref.)	831 ± 8	0.936 ± 0.001	0.316 ± 0.000
Highest-quality (oracle ref.)	843 ± 4	0.969 ± 0.001	0.432 ± 0.000
Per-prompt oracle	0 ± 0	1.000 ± 0.000	0.067 ± 0.001

τ-BENCH RETAIL: PER-CALL ROUTING IN LONG-HORIZON AGENTS

Per-policy results (main run)

Policy	Task success [95% CI]	Avg. cost/task ($)
routed-flat	**0.100 [0.04, 0.17]**	**0.00778**
routed-regional (ours)	**0.450 [0.34, 0.56]**	**0.04781**
nova-micro-v1	0.125 [0.05, 0.20]	0.00226
nova-lite-v1	0.150 [0.07, 0.24]	0.00533
llama3-1-8b-instruct-v1	0.138 [0.07, 0.21]	0.01897
llama3-1-70b-instruct-v1	0.150 [0.07, 0.24]	0.05993
claude-sonnet-4-5-20250929-v1	0.600 [0.49, 0.70]	0.07820
mistral-large-2402-v1	0.163 [0.09, 0.25]	0.08188

Seed-variance study (5 independent replicates)

	regional (ours)		flat	
Replicate	success	$/task	success	$/task
1	0.450	0.0478	0.100	0.0078
2	0.537	0.0620	0.225	0.0141
3	0.225	0.0173	0.537	0.0672
4	0.450	0.0367	0.188	0.0168
5	0.650	0.0613	0.562	0.0703
mean $\pm$ s.d.	0.463 ± 0.156		0.323 ± 0.213	

Table 3: Routing detail. Top: routing over six real Bedrock models on MMLU, with 8 subjects as regions and a horizon of 6k. Only the hierarchical and flat policies learn online. The rest are computed from the ground truth and pay no exploration cost. Middle: τ-bench retail, the online contextual UCB router (regional) against the structure-blind flat learner and the fixed models. Bottom: the seed-variance study, 5 independent replicates of the two learned routers over the same 80 tasks, of which replicate 1 is the main run above.

Cost budget	150	300	600	1200	2400
Hierarchical (ours)	$\mathbf{0.124 \pm 0.013}$	$\mathbf{0.253 \pm 0.016}$	$\mathbf{0.370 \pm 0.018}$	$\mathbf{0.464 \pm 0.014}$	$\mathbf{0.563 \pm 0.014}$
Successive elimination	0.103 ± 0.014	0.124 ± 0.014	0.126 ± 0.013	0.307 ± 0.012	0.421 ± 0.014
Uniform	0.103 ± 0.014	0.124 ± 0.014	0.126 ± 0.013	0.303 ± 0.012	0.428 ± 0.013

Table 4: Top-10 identification on the RouterEval 1000-model pool: mean recall@10 ($\pm$ SEM over 20 seeds and 9 evaluation splits) as the cost budget grows. The probe/leaf cost ratio is 0.05, and the hierarchical method's cost includes the certificate pre-pass.

Budget (calls)	best-of-N resolved [95% CI]	value-guided resolved [95% CI]	Δ (paired) [95% CI]
9	0.195 [0.15, 0.25]	0.318 [0.26, 0.38]	+0.123 [+0.08, +0.17]

Table 5: Repository-level code: value-guided patch search vs. best-of-N at matched compute on SWE-bench Verified (261 issues, "15 min–1 hour" tier, claude-sonnet-4.5). Resolved rate is the official criterion, and Δ is the paired same-issue gap with a 95% bootstrap CI over issues.

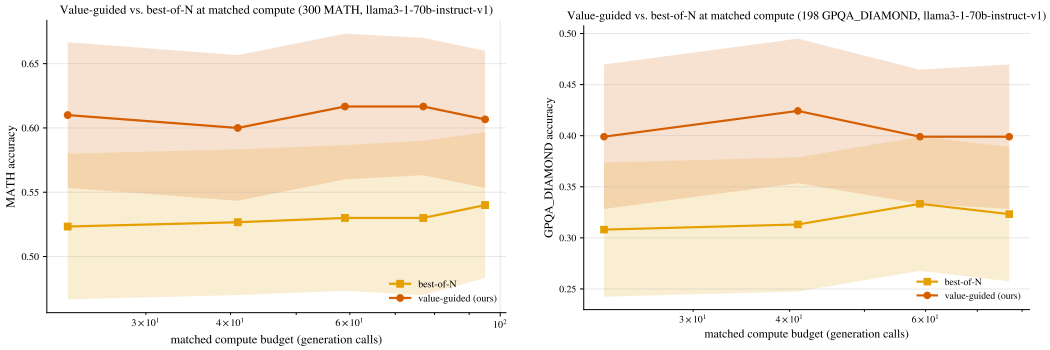


Figure 10: Accuracy vs. matched compute budget (generation calls) for value-guided search vs. best-of-N, with bootstrap 95% CI bands (llama-3.1-70B). Left: MATH. Right: GPQA-Diamond.

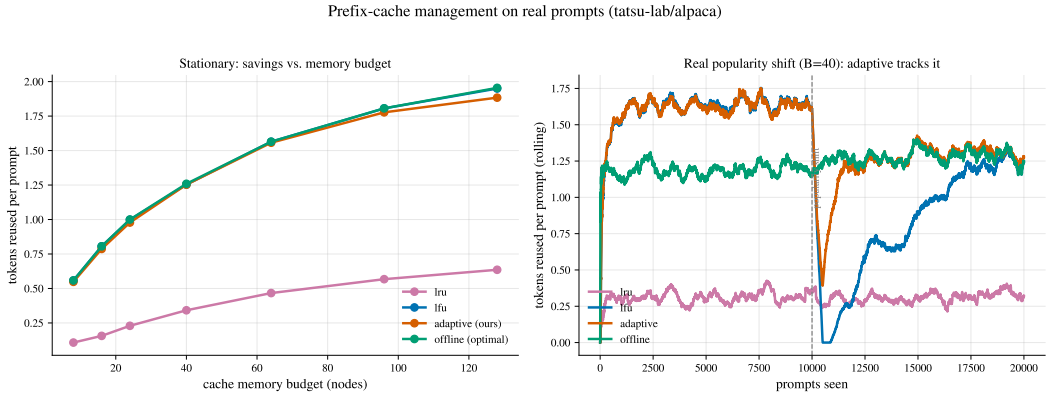


Figure 11: Prefix caching on a real prompt stream. Left: savings vs. memory budget (stationary), where adaptive tracks LFU and the offline optimum. Right: savings over time across a real popularity shift, where adaptive re-adapts fastest, LFU recovers by the end of the stream (1.25 vs. adaptive's 1.28 tokens/prompt at the final point), and LRU stays stale (0.32).

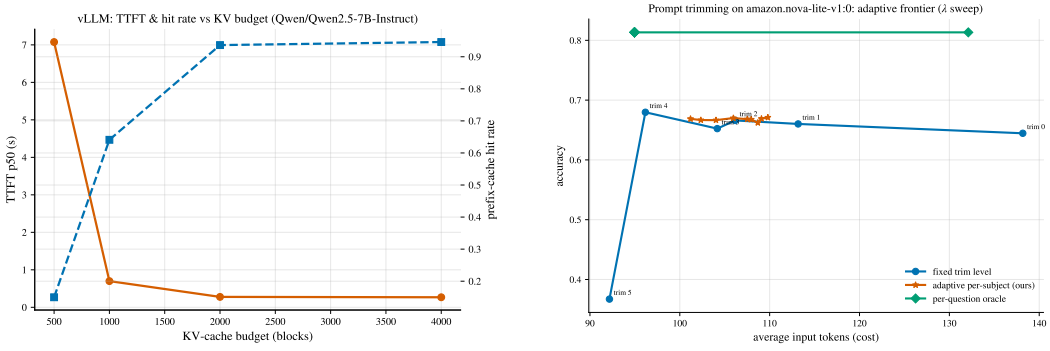


Figure 12: Serving knobs on real systems. Left: the KV-cache budget frontier on live vLLM, where median TTFT (log scale) falls and prefix-cache hit rate rises as the KV budget grows, the hardware analog of savings vs. memory budget. Right: prompt trimming, accuracy vs. average input tokens. The fixed trim levels (blue) trace a tradeoff curve, sweeping λ gives the adaptive per-subject frontier (red), which dominates the naive always-verbose default but sits near the best fixed trim at this scale, and the per-question oracle (green) is an upper bound.

CAPABILITY LADDER

MATH

Model	best-of-N [95% CI]	value-guided [95% CI]	Δ [95% CI]
mistral-large	0.084 [0.05, 0.12]	0.140 [0.10, 0.18]	+0.057 [+0.01, +0.10]
sonnet45	0.503 [0.45, 0.56]	0.837 [0.79, 0.88]	+0.333 [+0.28, +0.39]
llama8b	0.537 [0.48, 0.59]	0.390 [0.34, 0.45]	−0.147 [-0.21, -0.09]
llama3-1-70b	0.540 [0.48, 0.59]	0.607 [0.55, 0.66]	+0.067 [+0.02, +0.12]
nova-pro	0.623 [0.57, 0.68]	0.753 [0.70, 0.80]	+0.130 [+0.08, +0.18]

GPQA-Diamond

Model	best-of-N [95% CI]	value-guided [95% CI]	Δ [95% CI]
llama8b	0.258 [0.20, 0.32]	0.263 [0.20, 0.32]	+0.005 [-0.07, +0.08]
sonnet45	0.313 [0.25, 0.38]	0.672 [0.61, 0.74]	+0.359 [+0.27, +0.44]
llama3-1-70b	0.323 [0.26, 0.39]	0.399 [0.33, 0.46]	+0.076 [+0.00, +0.15]
nova-pro	0.354 [0.29, 0.42]	0.414 [0.35, 0.48]	+0.061 [-0.01, +0.13]
mistral-large	0.374 [0.31, 0.44]	0.333 [0.27, 0.40]	−0.040 [-0.11, +0.04]

GSM8K (near-saturated control)

Model	best-of-N [95% CI]	value-guided [95% CI]	Δ [95% CI]
llama8b	0.925 [0.89, 0.96]	0.850 [0.80, 0.90]	−0.075 [-0.12, -0.03]
nova-pro	0.945 [0.91, 0.97]	0.970 [0.94, 0.99]	+0.025 [+0.01, +0.05]
mistral-large	0.955 [0.93, 0.98]	0.840 [0.79, 0.89]	−0.115 [-0.16, -0.07]
llama3-1-70b	0.960 [0.93, 0.98]	0.960 [0.93, 0.98]	+0.000 [-0.03, +0.03]
sonnet45	0.985 [0.96, 1.00]	0.980 [0.96, 0.99]	−0.005 [-0.02, +0.00]

REPOSITORY-LEVEL CODE

SWE-bench Verified, model × budget sweep

Model	$B = 6$	$B = 9$	$B = 12$
claude-sonnet-4.5	+0.100 **[+0.02, +0.19]**	+0.125 **[+0.04, +0.21]**	+0.062 [-0.01, +0.14]
nova-pro	+0.100 **[+0.03, +0.18]**	+0.062 [-0.01, +0.15]	+0.025 [-0.04, +0.09]
llama-3.1-70B	+0.050 [-0.01, +0.12]	+0.087 **[+0.02, +0.16]**	+0.025 [-0.04, +0.09]
mistral-large	+0.013 [-0.04, +0.06]	+0.013 [-0.04, +0.06]	−0.013 [-0.06, +0.04]
llama-3.1-8B	−0.013 [-0.08, +0.05]	+0.050 [+0.00, +0.11]	+0.037 [-0.02, +0.10]

CODE SYNTHESIS (PRE-REGISTERED NULLS)

HumanEval

Budget (calls)	best-of-N acc [95% CI]	value-guided acc [95% CI]	Δ (paired) [95% CI]
23	0.811 [0.75, 0.87]	0.835 [0.77, 0.89]	+0.024 [-0.03, +0.08]
32	0.811 [0.75, 0.87]	0.835 [0.78, 0.89]	+0.024 [-0.03, +0.08]
41	0.811 [0.75, 0.87]	0.823 [0.76, 0.88]	+0.012 [-0.04, +0.07]

MBPP

Budget (calls)	best-of-N acc [95% CI]	value-guided acc [95% CI]	Δ (paired) [95% CI]
23	0.787 [0.72, 0.85]	0.744 [0.68, 0.81]	−0.043 [-0.09, +0.00]
32	0.787 [0.73, 0.85]	0.720 [0.65, 0.79]	−0.067 [-0.12, -0.02]
41	0.787 [0.73, 0.85]	0.713 [0.64, 0.78]	−0.073 [-0.12, -0.03]

Table 6: Test-time search, value-guided against best-of-N at matched compute, with 95% bootstrap confidence intervals, Δ the paired same-item gap, and bold where the interval excludes zero. Capability ladder: five models on MATH and GPQA-Diamond, with GSM8K as the near-saturated control. The gain is largest for capable but unsaturated models and vanishes or reverses for the weakest, which is the unreachable regime. Mistral-Large's low absolute MATH accuracy reflects answer-format non-compliance rather than reasoning ability. Repository-level code: the SWE-bench model × budget sweep over 78–80 Verified "15 min–1 hour" issues per cell, with instances whose evaluation harness errored dropped from both arms. Code synthesis: the HumanEval and MBPP nulls for Llama-3.1-70B. The public-test cheap probe is weak and both benchmarks are near-saturated, a limit fixed before the results were seen.

MATH

Quantity	Value
Cheap-vs-true node value (Spearman ρ)	0.485
Cheap-vs-true node value (Pearson r)	0.427
Edge-following hit rate, all steps (chance 0.33)	0.940
Edge-following hit rate, pivotal steps only	0.734 [0.64, 0.82] (n=109)
Mean true value lost per step (edge gap)	0.025
Pivotal-step fraction	τ=0.25: 11%; τ=0.5: 6%; τ=0.75: 4%

SWE-bench Verified

Quantity	Value
Cheap-vs-true patch value (Spearman ρ)	0.863
Cheap-vs-true patch value (Pearson r)	0.875
Edge-following hit rate, all steps (chance 0.33)	1.000
Edge-following hit rate, pivotal steps only	1.000 [1.00, 1.00] (n=18)
Pivotal-step fraction	τ=0.25: 10%; τ=0.5: 10%; τ=0.75: 10%

Table 7: The value function as almost tree-K-Lipschitz, measured in the wild: cheap-vs-true node-value correlation (the tree-Lipschitz backbone) and the count of pivotal steps K (violations). Top: MATH partial traces (cheap self-consistency probe). Bottom: the real-code analog on SWE-bench Verified (60 issues and 540 candidate patches, cheap FAIL_TO_PASS test probe vs. the official resolved criterion). The empirical counterpart of the synthetic violation-family task (Appendix D).

MOONCAKE PRODUCTION TRACE

Policy	Stationary (B=64)	After shift (B=16)
LRU	4.74	3.02
LFU	5.92	5.39
Adaptive (ours)	**5.89**	**5.39**
Offline-optimal (static)	5.92	5.39

LIVE VLLM: PREFIX CACHING ON VS. OFF

Metric	Caching ON	Caching OFF	Gain
TTFT p50 (s)	0.267	0.961	+72%
TTFT mean (s)	0.275	1.086	+75%
Latency mean (s)	2.984	11.293	+74%
Throughput (tok/s)	678.4	180.8	+275%
Prefix-cache hit rate	0.947	0.000	—

GPU-CALIBRATED EVICTION POLICIES

Policy	TTFT saved, stationary (ms)	TTFT saved, post-shift (ms)
LRU (engine default)	12.3	12.9
LFU	24.1	8.8
Adaptive (ours)	**24.0**	**23.6**
Offline-optimal	20.4	12.7

Table 8: Caching and systems detail. Mooncake: prefix-cache management on the real tool-agent production trace of 20k requests, measured in blocks reused per request. Adaptive matches the offline optimum and beats the engine-default LRU, most at tight budgets. Live vLLM: automatic prefix caching on against off, reporting TTFT, latency, throughput, and prefix-cache hit rate on a shared-prefix workload with the built-in LRU eviction. Eviction policies: realized TTFT saved in milliseconds on the GPU-calibrated shared-prefix workload, stationary and post-shift. Adaptive beats the engine-default LRU throughout and dominates after the shift.

LONGBENCH: CONTEXT TRIMMING

Positional truncation

Policy	QA-F1	Avg. input tokens
Full context (keep 1.00)	0.440	7239
Best fixed trim	0.440	7239
Adaptive per-task (ours)	**0.362**	**3544**
Per-item oracle	0.542	2810

Retrieval-scored chunk selection (pre-registered follow-up)

Policy	QA-F1	Avg. input tokens
Full context (keep 1.00)	0.440	7239
Best fixed trim	0.440	7239
Adaptive per-task (ours)	**0.407**	**2701**
Per-item oracle	0.563	2239

BBH: FEW-SHOT TRIMMING

Policy	Accuracy	Avg. input tokens
Full 3-shot (verbose)	0.346	467
Best fixed trim	0.356	332
Adaptive per-task (ours)	**0.368**	**291**
Per-item oracle	0.566	208

Table 9: Prompt trimming on real benchmarks: adaptive per-task selection vs. fixed trims and the per-item oracle. LongBench (QA-F1 vs. input tokens): the top block trims by positional truncation. The middle block is the pre-registered follow-up with the identical setup but retrieval-scored chunk selection at the same word budgets. The better primitive lifts every fixed trim at matched tokens ($+0.10$ to $+0.13$ F1 at tight budgets) and moves the adaptive operating point to 0.407 F1 at $2{,}701$ tokens. BBH exact-match vs. input tokens shows the adaptive per-task few-shot selection.

F FUTURE WORK

The structure our method exploits is not specific to trees. Two ingredients drive everything in Section 3.4: a metric under which the objective is smooth, and an aggregation map whose value on a region is a (weighted) average of the points it contains. The tree supplies the LCA distance together with the subtree average, but the optimistic engine, the data-driven smoothness certificate, and the discontinuity-guided sampling all carry over to any index set equipped with these two ingredients.

General metric spaces. Replacing the LCA distance by an arbitrary metric and the dyadic tree by a hierarchical cover, as in zooming and $\mathcal{X}$-armed bandits over metric spaces, preserves the near-optimality-dimension analysis. The certificate of Section 3.3 bounds the within-cell maximum-minus-mean from samples regardless of how the cells are formed, so it transfers unchanged.

Spheres and directional embeddings. The most relevant high-dimensional instance for the language-model applications is the unit sphere under the angular (cosine) metric, since prompt and model embeddings live there. Cells become spherical caps, the aggregation map averages over a cap, and tree-Lipschitzness becomes Lipschitzness in geodesic distance. This would replace the token-prefix tree of Section 3.1, which is sparse and brittle for routing, with an embedding-induced hierarchy on the sphere, and connects the bandit view developed here to angular reconstruction of a reward defined over the sphere.

DAGs and weighted aggregation. When a region's value is any linear functional of its underlying points (a weighted subset average rather than a uniform one), the noise-deconvolved moment-generating certificate is unchanged, so the framework extends from trees to directed acyclic and lattice-structured index sets.

Certified sparse attention for long-context RL. A further application targets a failure mode recently identified in reinforcement learning of long-context language models (Chen et al., 2026): applying static or heuristic sparse attention during on-policy rollouts injects an approximation error that biases the token likelihoods, and this error compounds over the autoregressive trajectory into an actor–policy distribution mismatch large enough to inflate the PPO/GRPO importance ratio and collapse training. The pathology is not sparsity per se but *uncontrolled, inconsistent per-step error*. Our framework suggests a route to controlling it: cast the choice of which KV-cache blocks to attend to at each step as a hierarchical bandit over the context tree (leaves are tokens, internal nodes are block summaries), whose arm value is a block's contribution to the attention output and whose reward is the noisy downstream return. Because the value is now measured through a stochastic reward rather than computed exactly, the full optimistic apparatus of Section 3.2–3.4 applies, not merely the multi-fidelity certificate. Two smoothness axes structure the problem: the *spatial* tree-Lipschitzness over the context hierarchy studied here — with the K violations being exactly the isolated "needle" blocks that produce error spikes — and a *temporal* assumption that block relevance drifts smoothly across generation steps, which would bound how much re-exploration each step requires. The central open result is a bridge theorem: a certified per-step bound on the selection error (of the form $L\rho^{\ell} + K \cdot D$ supplied by Section 3.3) implies a bounded total-variation gap between the sparse and dense policies, hence a clip-safe importance ratio and provably stable training — a mathematically grounded alternative to the empirical distillation used to patch this instability today. We leave the temporal-smoothness lemma and the importance-ratio bound to future work.