2FFS: 2-fidelity best-arm identification for minimax tree search

Peter Chen
University of California, Berkeley
pllc@eecs.berkeley.edu

Xi Chen
New York University
xc13@stern.nyu.edu

Abstract

We study fixed-confidence best-arm identification (BAI) in stochastic minimax trees. This problem is increasingly relevant in modern AI planning, where deep minimax search and Monte Carlo Tree Search (MCTS) with language model long rollouts face a fundamental tradeoff: heuristic evaluations are cheap but biased, while accurate rollouts are reliable but prohibitively expensive. We propose 2FFS, a 2-fidelity minimax tree search paradigm that brings multi-fidelity flat bandit ideas into trees. 2FFS combines minimax-style fast expansion with MCTS-style stochastic sampling, adaptively deciding when to exploit cheap biased evaluations and when to invoke expensive accurate evaluations for local certification. We prove fixed-confidence correctness, establish finite stopping for exact identification, and give a polynomial-depth cost upper bound for general-depth trees. Guarantees are further extended to real-world multi-agent reinforcement learning (MARL). Across numerical stochastic-tree experiments, 2FFS uses substantially fewer samples and computational operations comparing to existing BAI-MCTS baseline[1].

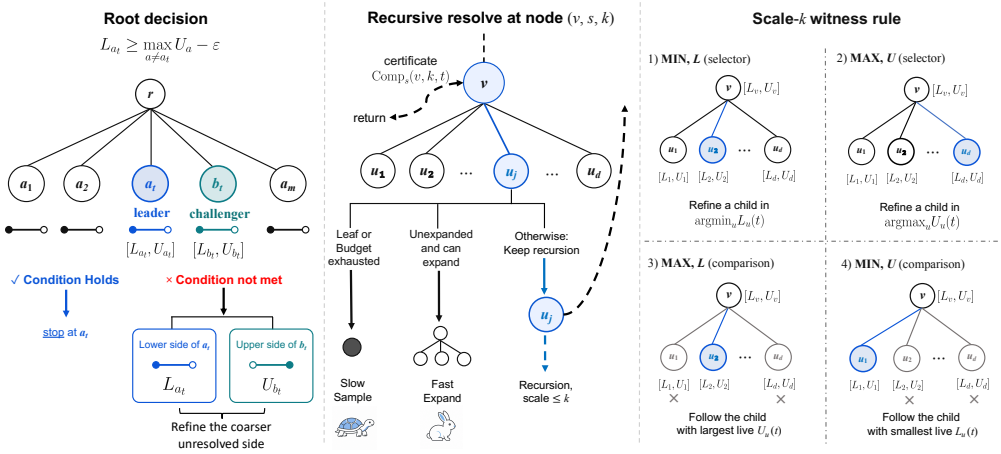

Figure 1: Algorithmic pipeline for 2-fidelity fast-slow (2FFS) minimax tree search.

1 Introduction

Modern decision-making systems often face a fundamental budget-allocation tradeoff: when reasoning over a large search space, should computation be used to expand the search tree more deeply, or to improve the reliability of the evaluations assigned to actions already under consideration? In two-player zero-sum planning and multi-agent reinforcement learning, this tradeoff is exemplified by the

[1]Numerical implementation: https://github.com/PeterLauLukChen/2FFS

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

contrast between minimax-style tree search, which pushes depth using fast heuristic leaf evaluations, and Monte Carlo Tree Search (MCTS), which allocates computation to repeated stochastic sampling and consequently grows a narrower, more selective tree.

A closely related tradeoff also arises in long-horizon reasoning and planning with large language models, where exhaustive tree expansion quickly becomes infeasible and each rollout, especially one involving a long chain-of-thought, can be costly. At a high level, this setting points to a common abstraction: a learner must allocate a limited sampling budget between *cheap but biased* information and *expensive but accurate* information. This viewpoint is closely connected to multi-fidelity bandits, in which an agent may query each arm at different fidelities, trading cost against accuracy in estimating its value. While the multi-fidelity multi-armed bandit (`MF-MAB`) literature [1–4] formalizes this tradeoff in flat action spaces, and best-action identification in MCTS [5] studies fixed-confidence identification in minimax trees under a single stochastic oracle, the corresponding problem of *multi-fidelity best-action identification in minimax trees* remains largely unexplored.

In this work, we formalize this tradeoff in a minimax tree with two complementary sources of information: a *fast* oracle, which is cheap but biased, and a *slow* oracle, which is more costly but statistically accurate. The agent must adaptively allocate computation across the tree, deciding whether to expand the search using fast evaluations or to spend additional budget refining the values of nodes already on the frontier. Our objective is to identify the optimal root action under a fixed-confidence requirement while minimizing total evaluation cost. This exposes a principled crossover between α-β-style expansion and MCTS-style sampling: at a frontier node, should the algorithm go one level deeper using the cheap biased fast oracle, or stay and reduce uncertainty using the expensive unbiased slow oracle?

Contributions. Our work fills this gap by proposing 2FFS, a two-fidelity search algorithm for fixed-confidence best-action identification in stochastic minimax trees. 2FFS adaptively compares two routes at each decision-critical node: recursively expanding the tree using cheap biased fast evaluations, or locally certifying the node value using expensive but statistically accurate slow evaluations. On the theoretical side, we prove (ε, δ)-PAC correctness and, for exact identification under a local regularity assumption, establish finite stopping together with a general-depth cost upper bound whose overhead is polynomial in the tree depth. The same exact-identification upper bound also gives a conservative bound for $\varepsilon > 0$ runs, since the approximate stopping rule can only terminate earlier. To our knowledge, this is the first analytic framework for multi-fidelity bandits in a minimax-tree setting. Empirically, on synthetic stochastic minimax trees, we show that 2FFS substantially improves both sampling and computational efficiency compared with existing baseline.

Related works. Viewed through the compute-allocation lens above, classical perfect-information two-player zero-sum search is dominated by minimax with α-β pruning, where efficiency is driven by deeper expansion and pruning under deterministic leaf evaluation [6, 7]. MCTS takes the opposite perspective, treating action selection as an exploration–exploitation problem guided by bandit principles (UCB) and stochastic samples [8]. Empirical hybrids such as implicit minimax backups suggest these paradigms are complementary, but also echo classic game-tree pathology results: applying minimax to noisy point estimates can mis-rank moves as depth grows [9, 10]. On the theory side, BAI-MCTS formalizes root-action selection as fixed-confidence BAI in minimax trees with an unbiased sampling oracle [5]; in parallel, multi-fidelity bandits and Bayesian optimization study cost–accuracy tradeoffs between cheap biased approximations and expensive accurate evaluations [1, 11, 12, 2–4]. We defer further related works to Appendix A.

2 Problem Setup and Preliminaries

In this section, we formalize fixed-confidence BAI in two-fidelity stochastic minimax trees and introduces the notation used by the algorithm and analysis. We first define the tree model, the fast and slow oracles, and the frontier maintained by the learner. Then, we define local and propagated confidence intervals, the effective gaps that determine decision-relevant precision, and the recursive oracle complexity and dyadic scale surrogates used in the upper-bound proof.

2.1 Minimax tree with two-fidelity oracles

Stochastic minimax tree. Let $\mathcal{T}$ be a finite rooted tree with root r and internal nodes alternating between MAX and MIN, with root r as MAX node. For each node $v \in \mathcal{T}$, we denote $\mathrm{Ch}(v)$ as its

set of children and $d(v)$ as its depth, and we define *remaining depth* of node v as $h(v) := D - d(v)$, where D is the depth of $\mathcal{T}$. For notational simplicity, we assume the tree has balanced depth: a node v is a leaf if and only if $h(v) = 0$.

Let $\mathcal{L}$ be the set of leaves, and let each $\ell \in \mathcal{L}$ have mean payoff μ_ℓ, then its minimax value $V^*(v)$ is defined by $V^*(\ell) = \mu_\ell$. For every internal node v,

$$V^*(v) = \begin{cases} \max_{u \in \mathrm{Ch}(v)} V^*(u), & \text{if } v \text{ is a MAX node,} \\ \min_{u \in \mathrm{Ch}(v)} V^*(u), & \text{if } v \text{ is a MIN node.} \end{cases}$$

For each root action $a \in \mathrm{Ch}(r)$, we write $V^*(r, a) := V^*(a)$. Thus, $V^*(\cdot)$ denotes node values, while $V^*(r, \cdot)$ denotes root-action values. We assume throughout that $|\mathrm{Ch}(r)| \geq 2$, so the BAI problem is nontrivial, and that the optimal root action is unique, which is given by $a^* \in \arg\max_{a \in \mathrm{Ch}(r)} V^*(r, a)$, along with the *root gap* $\Delta_* := V^*(r, a^*) - \max_{a \neq a^*} V^*(r, a)$.

Definition 2.1 (Frontier)**.** *A frontier $\mathcal{F} \subseteq \mathcal{T} \setminus \{r\}$ is a complete cut of the tree, meaning that every root-to-leaf path intersects $\mathcal{F}$ in exactly one node.*

At each round t, the algorithm has an explored subtree $\mathcal{T}_t \subseteq \mathcal{T}$ containing the root. A node becomes *exposed* when it first enters $\mathcal{T}_t$. The current frontier $\mathcal{F}_t$ consists of the exposed nodes at which recursive expansion has not yet proceeded; it forms a complete cut of the full tree. Expanding an exposed internal node reveals all of its children, queries the fast oracle at those children, and replaces the node by its children in the frontier. Any local slow samples already collected at the expanded node remain available for all future interval computations.

Two-fidelity node-wise evaluations. Each exposed non-root node v can be evaluated through two resources. In the multi-fidelity bandit viewpoint [1, 2, 4], querying an arm at fidelity m produces a sample from a distribution $\nu_{v,m}$ with mean $\mu_{v,m}$ and known query cost λ_m. In our tree setting, the node v plays the role of the arm, and we use two fidelities. The *fast* oracle $(m = 1)$ is cheap and deterministic, but may be biased. The *slow* oracle $(m = 2)$ is more expensive and stochastic, but unbiased for the true minimax value $V^*(v)$. Thus, the fast oracle provides a low-fidelity estimate and the slow oracle provides a high-fidelity estimate of the same node value.

Note that the fast-oracle bias is controlled by the remaining depth $h(v)$ through a known calibration envelope $B : \{0, \ldots, D\} \to \mathbb{R}_{\geq 0}$, which satisfies

$$B(0) = 0, \qquad B(h+1) \geq B(h) \quad \text{for every } h \in \{0, \ldots, D-1\}, \tag{1}$$

We normalize query costs as $\lambda_1 = 1$ and $\lambda_2 = c \geq 1$. The two oracles are defined formally below.

Definition 2.2 (Fast oracle F)**.** *The fast oracle returns a deterministic value $V_F(v)$. Equivalently, $\nu_{v,1}$ is a degenerate distribution at $V_F(v)$, so $\mu_{v,1} = V_F(v)$ and $|V_F(v) - V^*(v)| \leq B(h(v))$.*

Definition 2.3 (Slow oracle S)**.** *For each node v, repeated slow-oracle queries return independent σ-sub-Gaussian samples $Y_{v,s}$ with distribution $\nu_{v,2}$ and mean $\mu_{v,2} = V^*(v)$.*

Remark 2.1. *The unbiased slow oracle setup is analogous to that of Kaufmann and Koolen [5], which defines the sampling oracle as mean-correct $\mathbb{E}[Y_{l,s}] = V^*(l)$; in §4, we instantiate it by adding actual noise to true minimax values. Also, unlike classical BAI-MCTS [5], where stochastic samples are attached to leaf rollouts, our slow oracle may be queried at any exposed non-root node. In §3.2, we add further worst-case guarantee to real-world applications, where slow-oracles are biased.*

Remark 2.2 (Fast-oracle envelop)**.** *We note that the envelope structure in Eq. (1) can be motivated by finite horizon MARL. Specifically, for a critic $\widehat{V}_t$ with the correct terminal value, discount $0 \leq \gamma \leq 1$, and minimax Bellman residual $\eta_t = \|\widehat{V}_t - \mathcal{T}_t \widehat{V}_{t+1}\|_\infty$, Bellman nonexpansiveness gives $\|\widehat{V}_t - V_t^*\|_\infty \leq \sum_{j=t}^{H-1} \gamma^{j-t} \eta_j$. A known bound $\eta_j \leq \bar{\eta}$ provides the nondecreasing envelope $B(h) = \bar{\eta} \sum_{i=0}^{h-1} \gamma^i$.*

2.2 Intervals, effective gaps, and recursive oracle complexity

In the fixed-confidence setting, a strategy adaptively chooses exposed nodes and query types, stops at a stopping time τ, and outputs a recommended action $\hat{a}_\tau \in \mathrm{Ch}(r)$. Let $N^F(\tau)$ denote the total number of fast-oracle queries issued by time τ, and let $N_v^S(\tau)$ denote the number of slow-oracle queries issued at node v. The total searching cost is

$$C_\tau := N^F(\tau) + c \sum_{v \in \mathcal{T}} N_v^S(\tau),$$

where c is the slow-oracle cost λ_2. Given $(\varepsilon, \delta) \in [0, \infty) \times (0, 1)$, we seek strategies that satisfy the (ε, δ)-PAC guarantee: $\mathbb{P}(\tau < \infty,\ V^*(r, a^*) - V^*(r, \hat{a}_\tau) > \varepsilon) \leq \delta$, while seeking low evaluation cost; our cost bounds hold with high probability.

We now introduce the interval objects used by 2FFS and the instance-dependent oracle complexity used in the upper bound. The construction has three layers: (i) local intervals combine the cheap fast evaluation with slow statistical evidence at the same node; (ii) propagated intervals carry child information upward through the minimax recursion; (iii) finally, we establish the ideal amount of oracle complexity needed to certify only the precision that can affect the root decision.

Confidence allocation & intervals. We first fix a feasible node-wise confidence allocation $\boldsymbol{\delta} = (\delta_v)_{v \in \mathcal{T} \setminus \{r\}}$, which indicates that $\delta_v \in (0, 1)$ and $\sum_{v \in \mathcal{T} \setminus \{r\}} \delta_v \leq \delta$. For every exposed non-root node v, the fast oracle gives the deterministic interval

$$I_v^F = [V_F(v) - B(h(v)),\ V_F(v) + B(h(v))]. \tag{2}$$

For slow samples, let $\widehat{V}_{v,n}$ be the empirical mean of the first n slow-oracle samples at v. Following from fixed-confidence bandit literature [13, 5, 14, 15, 2], we use the time-uniform radius $\beta_v(\cdot, \delta_v)$ satisfying

$$\mathbb{P}\left(\exists\, n \geq 1 :\ \left|\widehat{V}_{v,n} - V^*(v)\right| > \beta_v(n, \delta_v)\right) \leq \delta_v, \tag{3}$$

where $n \mapsto \beta_v(n, \delta_v)$ is non-increasing and $\beta_v(n, \delta_v) \to 0$ as $n \to \infty$. Let $N_v^S(t)$ be the number of slow queries issued at v by round t, the slow interval is then the running intersection

$$I_v^S(t) = \begin{cases} \bigcap_{j=1}^{N_v^S(t)} \left[\widehat{V}_{v,j} - \beta_v(j, \delta_v),\ \widehat{V}_{v,j} + \beta_v(j, \delta_v)\right], & N_v^S(t) \geq 1, \\ (-\infty, +\infty), & N_v^S(t) = 0, \end{cases} \tag{4}$$

which further gives the direct local interval $I_v^{\mathrm{loc}}(t) := I_v^F \cap I_v^S(t)$. Finally, we define the simultaneous-validity event by Eq. (5). Also, by Eq. (3) and the union bound, we always have $\mathbb{P}(\mathcal{E}_\delta) \geq 1 - \delta$.

$$\mathcal{E}_\delta := \bigcap_{v \in \mathcal{T} \setminus \{r\}} \bigcap_{n \geq 1} \left\{ \left|\widehat{V}_{v,n} - V^*(v)\right| \leq \beta_v(n, \delta_v) \right\}. \tag{5}$$

Propagated intervals. If an internal node v has been expanded, its children are exposed and carry effective intervals $I_u(t) = [L_u(t), U_u(t)]$. The child-backup interval is

$$I_v^{\mathrm{ch}}(t) = \begin{cases} \left[\max_{u \in \mathrm{Ch}(v)} L_u(t),\ \max_{u \in \mathrm{Ch}(v)} U_u(t)\right], & \text{if } v \text{ is a MAX node}, \\ \left[\min_{u \in \mathrm{Ch}(v)} L_u(t),\ \min_{u \in \mathrm{Ch}(v)} U_u(t)\right], & \text{if } v \text{ is a MIN node}. \end{cases} \tag{6}$$

If v is a leaf or not expanded yet, we set $I_v^{\mathrm{ch}}(t) = (-\infty, +\infty)$. For every explored non-root node, the effective interval combines direct local evidence and descendant evidence: $I_v(t) = I_v^{\mathrm{loc}}(t) \cap I_v^{\mathrm{ch}}(t)$. We note that the minimax backup in Eq. (6) preserves validity and does not increase interval width. The following two lemmas formalize these properties; their proofs are given in Appendix B.2.

Lemma 2.3. *Fix a frontier $\mathcal{F}$ and suppose that each frontier interval $[L_v, U_v]$, $v \in \mathcal{F}$, contains $V^*(v)$. Under the recursive backup in Eq. (6),* (i) *every ancestor u of a frontier node satisfies $V^*(u) \in [L_u, U_u]$; and* (ii) *if every frontier interval has half-width at most w, then every propagated ancestor interval also has half-width at most w.*

Lemma 2.4 (Simultaneous interval validity)**.** *On $\mathcal{E}_\delta$, every effective interval constructed from Eqs. (2)–(6) is valid at every round t. In particular, $V^*(v) \in I_v(t)$ for every explored non-root node v, and the intervals of root children and the root backup are valid as well.*

Effective gap. A node does not necessarily need to be estimated down to its own local sibling gap. The value of a node v can affect the final recommendation only if its uncertainty propagates through every ancestor and changes the comparison between root actions. Along this path, any large value separation creates a bottleneck: once the uncertainty below v is smaller than that separation, further refinement cannot change the root decision. We therefore define the *effective gap* of a node as the precision scale needed to certify that node for the purpose of root-action identification.

Definition 2.4 (Effective gap)**.** *Set the root effective gap to* $\Delta_r^{\mathrm{eff}} := \Delta_*$*. For every non-root node* v*, let* $p(v)$ *denotes the parent of* v*, we define recursively*

$$\Delta_v^{\mathrm{eff}} := \max\Big\{\Delta_{p(v)}^{\mathrm{eff}},\, |V^*(v) - V^*(p(v))|\Big\}.$$

Thus, Δ_v^{eff} records the largest bottleneck encountered on the path from the root to v. If this quantity is large, then v only needs to be estimated coarsely: Proposition B.10 shows that a call at scale k requires $\Delta_v^{\mathrm{eff}} \le 2\rho_k$.

Reference recursive oracle complexity. To derive cost guarantees, we first define an idealized reference oracle complexity H^* under perfect gap information within our recursive certification model. Our main theorem shows that 2FFS, despite not knowing the effective gaps or the cheaper route in advance, has cost within a polynomial-depth factor of this ideal reference complexity.

For a target precision $\rho > 0$, we first define the local cost of certifying a single exposed node v. Let

$$m_v(\rho, \delta_v) := \inf\big\{n \in \mathbb{N} :\ n \ge 1,\ \beta_v(n, \delta_v) \le \tfrac{1}{4}\rho\big\}.$$

After $m_v(\rho, \delta_v)$ slow samples, the slow confidence interval has width at most $\rho/2$. The factor $1/4$ is chosen so that either the fast interval alone, $2B(h(v)) \le \rho/2$, or the slow confidence interval alone, $2\beta_v(n, \delta_v) \le \rho/2$, is sufficient to make the direct local interval I_v^{loc} have width at most $\rho/2$.

The corresponding local certification cost is

$$\Gamma_v(\rho, \delta_v) = \begin{cases} 0, & \text{if } B(h(v)) \le \frac{1}{4}\rho, \\ c \cdot m_v(\rho, \delta_v), & \text{if } B(h(v)) > \frac{1}{4}\rho. \end{cases} \tag{7}$$

Thus, $\Gamma_v(\rho, \delta_v)$ is the slow-oracle cost needed to certify the value of v locally at precision ρ, after v has already been exposed. The first case means that the fast interval is already narrow enough, so no slow sample is needed at v.

We now define the recursive subtree oracle complexity. For a fixed confidence allocation $\boldsymbol{\delta}$, $J_v^*(\boldsymbol{\delta})$ is a reference cost for the subtree rooted at v, using effective-gap information. At each internal node, it compares two routes: certify v locally, or expand v and add all child reference costs:

$$J_v^*(\boldsymbol{\delta}) = \begin{cases} \Gamma_v(\Delta_v^{\mathrm{eff}}, \delta_v), & \text{if } h(v) = 0, \\ \min\Big\{\Gamma_v(\Delta_v^{\mathrm{eff}}, \delta_v),\ |\mathrm{Ch}(v)| + \sum_{u \in \mathrm{Ch}(v)} J_u^*(\boldsymbol{\delta})\Big\}, & \text{if } h(v) \ge 1. \end{cases} \tag{8}$$

This induces the root oracle complexity $H(\boldsymbol{\delta}) := |\mathrm{Ch}(r)| + \sum_{a \in \mathrm{Ch}(r)} J_a^*(\boldsymbol{\delta})$, where $|\mathrm{Ch}(r)|$ is the cost of initially exposing all root actions, and gives us the optimized recursive oracle complexity H^*:

$$H^* := \inf_{\boldsymbol{\delta}\ \text{feasible}} H(\boldsymbol{\delta}). \tag{9}$$

Remark 2.5. *We note that instance-dependent complexity measures are standard in fixed-confidence BAI. Classical bounds are often stated in terms of gap-dependent hardness quantities such as* $H_1 = \sum_{i \ne i^*} \Delta_i^{-2}$*, while refined analyses use information-theoretic characteristic times to quantify the intrinsic difficulty of an instance [16–18]. BAI-MCTS and multi-fidelity BAI follow the same convention, using tree- or fidelity-dependent oracle complexities to state adaptive cost guarantees [5, 2, 4]. Our* H^* *serves as an analysis-only recursive oracle complexity under perfect gap information, rather than an information-theoretic lower bound; see Appendix B.8.*

Dyadic scale surrogates. Following from Remark 2.5, the algorithm does not know the effective gaps and therefore cannot target $J_v^*(\boldsymbol{\delta})$ directly. Instead, 2FFS works on a dyadic precision grid, which initializes by exposing the root children with fast queries, then sets

$$\rho_0 := \max_{a \in \mathrm{Ch}(r)} \big(U_a(0) - L_a(0)\big), \qquad \rho_{k+1} := \tfrac{1}{2}\rho_k, \qquad k \ge 0.$$

If $\rho_0 = 0$, the root-child values are already known exactly and the algorithm stops immediately; otherwise the following scales are used. A node v is *scale-k safe* at round t if $U_v(t) - L_v(t) \le \frac{1}{2}\rho_k$.

The scale-k local certification cost is $\Gamma_v^{(k)}(\delta_v) := \Gamma_v(\rho_k, \delta_v)$, and the operational rules that decide which scale and which node to refine are given in Algorithm 1.

Algorithm 1 2FFS algorithm skeleton

Input: confidence schedule (δ_v), accuracy ε, depth budgets $(\alpha_h)_{h=0}^{D}$
Initialize the root children with fast-oracle intervals and propagate intervals to the root
while no root child $\hat{a}$ satisfies $L_{\hat{a}} \geq \max_{a \neq \hat{a}} U_a - \varepsilon$ **do**
    Let $a_t \leftarrow \operatorname{argmax}_a L_a(t)$ be the leader and $b_t \leftarrow \operatorname{argmax}_{a \neq a_t} U_a(t)$ be the challenger
    Choose the coarser unresolved root side: L_{a_t} or the challenger's coarsest unresolved side
    $\triangleright$ x is the selected node, s the selected side, and k the active scale.
    RESOLVE(x, s, k)
end while
return any separated root child
procedure RESOLVE(v, s, k)
    if the side certificate $\mathsf{Comp}_s(v, k, t)$ already holds **then**
        return
    end if
    if v is a leaf, its recursive budget is exhausted, or it is unexpanded and expansion is unaffordable **then**
        Take one slow local sample at v and update affected intervals and certificates
    else if v is unexpanded and expansion fits the remaining recursive budget **then**
        Expand v, query the fast oracle at its children, and update affected intervals and certificates
    else
        Select the current live child using selector-comparison rule in Figure 1 (right).
        Recursively resolve the selected child at an unresolved scale no larger than k
        if the child call is blocked only by the inherited parent cap **then**
            Return the blocked status upward
        else if the child call is blocked by v's own recursive budget **then**
            Take one slow local sample at v
        end if
    end if
end procedure

3 2FFS Tree Search: Two-Fidelity Fast-Slow Tree Search

2FFS maintains a confidence interval $[L_v(t), U_v(t)]$ for each exposed node and repeatedly certifies the root decision. At each round, the algorithm first checks whether some root action is already ε-optimal according to the current intervals. If not, it selects one decision-relevant side of one root child and calls a recursive resolver RESOLVE(v, s, k), where $s \in \{L, U\}$ is the endpoint to certify and k is the active dyadic scale. Figure 1 visualizes these three pieces: the root test, the recursive resolver, and the scale-k child-selection rules.

At the root, let $a_t \in \operatorname{argmax}_{a \in \mathrm{Ch}(r)} L_a(t)$ and $b_t \in \operatorname{argmax}_{a \in \mathrm{Ch}(r) \setminus \{a_t\}} U_a(t)$ be the current leader and challenger. 2FFS stops once a root action is certified to be ε-optimal: $L_{\hat{a}_t}(t) \geq \max_{a \neq \hat{a}_t} U_a(t) - \varepsilon$. If this test fails, the leader's lower endpoint and the challenger's unresolved endpoint information are the two quantities that can still change the root decision. 2FFS refines whichever of these two root-side certificates is currently coarser. For the leader this means the lower side L_{a_t}; for the challenger it means the coarsest unresolved side among L_{b_t} and U_{b_t}.

The call RESOLVE(v, s, k) tries to certify endpoint s of node v at scale ρ_k. It has two competing routes. The local route takes slow samples directly at v. The recursive route expands below v, uses fast evaluations on the children, and continues recursively only along the currently relevant child. These routes remain available throughout the run: even after v has been expanded, 2FFS may still take slow samples at v. This *local reversibility* is important because the cheaper route is not known in advance. Recursive spending at node v and scale k is capped by $\mathsf{B}^{\mathrm{rec}}_{v,k} = \alpha_{h(v)} \Gamma_v^{(k)}(\delta_v)$, and every child call uses a scale no finer than the parent scale k. If the recursive route cannot make admissible progress within this budget, the resolver falls back to one slow local query at v.

Inside an expanded node, the recursive route follows the live child rule in Figure 1 (right). In selector cases, the live child is the endpoint that currently determines the minimax backup. In comparison cases, 2FFS does not force all children to become fully certified. Instead, it lazily removes a child once the child is already outside the scale-k comparison margin, or once the child has the required

same-side certificate. Among the remaining live children, only the current endpoint-most blocker is refined recursively. The comparison rows use the following discharge margins. For a MAX node on side L, write $\lambda_L(v,t) = \max_u L_u(t)$; a child is no longer live if

$$U_u(t) \le \lambda_L(v,t) + \rho_k/2 \quad \text{or} \quad \mathsf{Comp}_L(u,k,t).$$

For a MIN node on side U, write $\lambda_U(v,t) = \min_u U_u(t)$; a child is no longer live if

$$L_u(t) \ge \lambda_U(v,t) - \rho_k/2 \quad \text{or} \quad \mathsf{Comp}_U(u,k,t).$$

The formal implementation maintains scale-local counters, monotone side flags $\mathsf{Done}_s(v,k)$, observable certificates $\mathsf{Comp}_s(v,k,t)$, and capped unresolved scales $K_s^{\le k}(v,t)$. These details are given in Appendix B.3, which are not needed to understand the main flow above but important for the proofs.

3.1 Theoretical results

We present three results: algorithmic correctness and a general-depth cost upper bound guarantee along with finite stopping time guarantee. First, we provide algorithm's (ε,δ)-PAC correctness:

Theorem 3.1 (PAC correctness)**.** *Fix any feasible confidence allocation $\boldsymbol{\delta} = (\delta_v)_{v\in\mathcal{T}\setminus\{r\}}$ and any $\varepsilon \ge 0$. Consider* Algorithm 1 *run with accuracy ε, let τ be its stopping time, and, on the event $\{\tau<\infty\}$, let $\hat{a}_\tau$ be the returned root child. Then, $\mathbb{P}(\tau<\infty$ and $V^*(r,a^*) - V^*(r,\hat{a}_\tau) > \varepsilon) \le \delta$. In fact, on the simultaneous-validity event $\mathcal{E}_\delta$, every finite-time recommendation made by the stopping rule is ε-optimal.*

Remark 3.2. *We note that* Theorem 3.1 *uses only simultaneous interval validity and the root stopping rule and does not use any assumptions in the later upper-bound proof. Its proof is very straightforward, and we defer it to* Appendix B.4. *The finite stopping guarantee is elaborated in* Theorem 3.6.

Now, we move to establish the upper bound for cost using the reference oracle complexity H^*. Before that, we first explicitly state several required regular assumptions along with their discussions.

Assumption 3.3 (Local dyadic scale regularity)**.** *Fix the confidence allocation $\boldsymbol{\delta}$, we assume there exist constants $\Lambda_{\mathrm{pre}} \ge 1$ and $\Lambda_{\mathrm{gap}} \ge 1$ such that, for every non-root node v and every $K \ge 0$,*

$$\sum_{j=0}^{K} \Gamma_v^{(j)}(\delta_v) \le \Lambda_{\mathrm{pre}} \Gamma_v^{(K)}(\delta_v), \tag{10}$$

and, for every non-root node v,

$$\Gamma_v\left(\frac{\Delta_v^{\mathrm{eff}}}{2}, \delta_v\right) \le \Lambda_{\mathrm{gap}}\, \Gamma_v(\Delta_v^{\mathrm{eff}}, \delta_v). \tag{11}$$

We write $\Lambda_{\mathrm{loc}} \coloneqq \Lambda_{\mathrm{pre}}\Lambda_{\mathrm{gap}}$.

Remark 3.4. Assumption 3.3 *is purely local: it does not compare the raw recursive dyadic DP with J^*. The prefix condition is the standard dyadic regularity of a statistical sample-complexity curve; it holds with a constant for polynomial laws, e.g., $m_v(\rho,\delta_v) \asymp \rho^{-2}\log(1/\delta_v)$ with uniform comparison constants over the positive-cost scales. The second condition only asks that moving from the effective gap to the neighboring dyadic half-scale changes the local stop cost by at most a constant factor. This is the only place where the deterministic fast-bias cutoff in* Eq. (7) *can create a knife-edge, which is ruled out by the assumption at the exact effective gap.*

Lemma 3.5. *For a fixed nondecreasing depth-budget sequence $(\alpha_h)_{h=0}^D$ with $\alpha_h \ge 1$, define $P_0 = \Lambda_{\mathrm{loc}}$ and, for $h \ge 1$,*

$$P_h \coloneqq \max\left\{(1+\alpha_h)\Lambda_{\mathrm{loc}},\ \left(1 + \frac{\Lambda_{\mathrm{pre}}}{\alpha_h}\right) P_{h-1}\right\}. \tag{12}$$

With the concrete choice $\alpha_h = (h+1)^2$, this recursion gives $P_D = \mathcal{O}_{\Lambda_{\mathrm{pre}},\Lambda_{\mathrm{gap}}}(D^2)$.

Proof. We denote $A = \Lambda_{\mathrm{loc}}$ and $B = \Lambda_{\mathrm{pre}}$, and unrolling the recursion shows that P_D is at most the largest local term $(1+\alpha_i)A$, multiplied by the product of later recursive multipliers:

$$P_D \le A \max_{0\le i\le D}\left[(1+\alpha_i)\prod_{j=i+1}^{D}\left(1+\frac{B}{\alpha_j}\right)\right].$$

For $\alpha_j = (j+1)^2$, the product is bounded by

$$\prod_{j=1}^{D}\left(1+\frac{B}{(j+1)^2}\right) \le \exp\left(B\sum_{j=1}^{\infty}\frac{1}{(j+1)^2}\right),$$

which is a constant depending only on B. Since $\max_{i\le D}(1+\alpha_i) = O(D^2)$, we obtain $P_D = O_{\Lambda_{\mathrm{pre}},\Lambda_{\mathrm{gap}}}(D^2)$, which is the key ingredient for the following general-depth cost bound. □

Theorem 3.6 (General-depth upper bound)**.** *Fix any feasible confidence allocation $\boldsymbol{\delta} = (\delta_v)_{v\in\mathcal{T}\setminus\{r\}}$ satisfying* Assumption 3.3*, and run* Algorithm 1 *with $\varepsilon = 0$. On $\mathcal{E}_\delta$, the algorithm is guaranteed to stop at a finite time τ, returns $\hat{a}_\tau = a^*$, and satisfies $C_\tau \le P_D\, H(\boldsymbol{\delta})$.*

With the depth budgets $\alpha_h = (h+1)^2$, the cost $C_\tau \le \mathcal{O}_{\Lambda_{\mathrm{pre}},\Lambda_{\mathrm{gap}}}\big(D^2 H(\boldsymbol{\delta})\big)$. Consequently, suppose the regularity constants Λ_{pre} and Λ_{gap} are uniform along a near-optimal feasible sequence: for every $\eta > 0$, there exists a feasible allocation $\boldsymbol{\delta}^\eta$ satisfying Assumption 3.3 *with these same constants and*

$$H(\boldsymbol{\delta}^\eta) \le H^* + \eta.$$

Then, the corresponding run satisfies, with probability at least $1-\delta$,

$$\hat{a}_\tau = a^*, \qquad C_\tau \le P_D\big(H^* + \eta\big).$$

In particular, when $\alpha_h = (h+1)^2$ and the local regularity constants are uniform constants,

$$C_\tau \le \mathcal{O}_{\Lambda_{\mathrm{pre}},\Lambda_{\mathrm{gap}}}\big(D^2(H^* + \eta)\big),$$

and, since $H^* \ge |\mathrm{Ch}(r)| \ge 2$, one obtains $\mathcal{O}_{\Lambda_{\mathrm{pre}},\Lambda_{\mathrm{gap}}}(D^2 H^*)$ by taking $\eta \le H^*$, or $\eta = 0$ whenever the infimum in Eq. (9) *is attained with the same regularity constants.*

Proof sketch. To our knowledge, this is the first theoretical result to establish a bound for multi-fidelity bandit in the tree setting. We therefore hope the proof structure sets up a paradigm for future work, and Theorem 3.6 is proved formally in Appendix B.6. Overall, we have one main goal: charge the actual oracle calls made by 2FFS to the ideal recursive oracle complexity $H(\boldsymbol{\delta})$. First, every certificate used by 2FFS is shown to be sound: once a side of a node is certified at scale k, the corresponding endpoint error is at most $\rho_k/2$. Second, a recursive call is shown to spend effort only on children that can still affect the parent certificate at the current scale; since child calls are capped by the parent scale, the algorithm avoids irrelevant fine-scale work below already separated parts of the tree. Third, for each node and scale, the budgeted resolver spends at most a local certification budget plus its recursive budget, and the same node-side-scale is never charged again after certification. Summing these charges over active scales, while reserving the cap needed for the next step, gives a subtree bound with overhead $P_{h(v)}$. Lemma 3.5 then shows that this overhead is only quadratic in depth for $\alpha_h = (h+1)^2$. Applying the subtree bound to the root children gives a finite global cost bound; since every non-stopping round makes positive progress, the algorithm must stop, and interval validity gives correctness.

3.2 Guarantees for real-world multi-agent RL application

In many real-world applications, e.g., multi-agent RL (MARL) with minimax trees, slow oracles may not provide perfectly unbiased estimates of the minimax value V^*, and a tight, accurate fast-oracle envelope B may be unavailable and must instead be approximated by a proxy. In this section, we establish additional theoretical guarantees for such imperfect real-world settings.

Biased slow oracles. We show that pointwise slow-oracle bias can be handled without accumulation across tree depths. Specifically, Theorem 3.7 gives a correctness guarantee using a known bias bound.

Theorem 3.7. *Suppose the slow oracle has fixed mean $V^*(v) + b_v$, with known $\xi \ge \max_v |b_v|$, and its samples satisfy Eq. (3) around this mean. Replacing each slow radius β_v by $\beta_v + \xi$ preserves the (ε,δ) finite-output correctness of 2FFS. The bias inflation does not accumulate with tree depth.*

Loose fast-oracle envelope. In practical MARL applications, the true minimax value is unavailable, and thus a tight fast-oracle envelope cannot be constructed. Instead, one can use a calibrated proxy envelope that shrinks with the remaining horizon, for example,

$$\hat{B}(h) = 1 + \frac{h-1}{D-1}, \qquad \hat{B}(0) = 0,$$

where $D \geq 2$ is the full tree depth and $h \geq 1$ is the remaining depth, provided $\hat{B}(h) \geq B(h)$. We show that such a loose envelope preserves the PAC guarantee.

Theorem 3.8. *Let $B(h)$ denote the true fast-oracle envelope, and let $\hat{B}(h)$ be a nondecreasing proxy envelope with $\hat{B}(0) = 0$ and $\hat{B}(h) \geq B(h)$ for all h. Replacing $B(h)$ with $\hat{B}(h)$ preserves the finite-output PAC guarantee and, on $\mathcal{E}_\delta$ for the same query history, can only widen the resulting confidence intervals.*

The proofs and qualifications for both results are given in Appendix B.7.

4 Numerical Experiments

Table 1: Results over stochastic minimax trees, with mean $\pm$ standard deviation reported.

Method	Stopping (↑)	Accuracy (↑)	Sampling count (↓)	Operation count (↓)
(i) $D=5, b=8$; 37,449 nodes				
`2FFS`	1.00	1.00	$5.39\times10^3 \pm 1.27\times10^3$	$8.38\times10^7 \pm 4.05\times10^7$
`BAI-MCTS`	1.00	0.99	$8.80\times10^5 \pm 3.39\times10^5$	$2.38\times10^8 \pm 9.49\times10^7$
`Minimax-fast`	1.00	0.91	$1.49\times10^4 \pm 3.35\times10^3$	$8.20\times10^5 \pm 1.91\times10^5$
`Slow-only`	0.47	0.47	$5.87\times10^5 \pm 4.86\times10^5$	$8.62\times10^7 \pm 7.15\times10^7$
(ii) $D=7, b=6$; 335,923 nodes				
`2FFS`	1.00	1.00	$1.77\times10^4 \pm 2.64\times10^3$	$1.06\times10^9 \pm 3.02\times10^8$
`BAI-MCTS`	0.98	0.98	$1.75\times10^7 \pm 6.71\times10^6$	$4.85\times10^9 \pm 1.87\times10^9$
`Minimax-fast`	1.00	0.88	$5.52\times10^4 \pm 7.08\times10^3$	$3.48\times10^6 \pm 4.59\times10^5$
`Slow-only`	0.73	0.70	$4.22\times10^6 \pm 9.16\times10^6$	$1.49\times10^9 \pm 4.13\times10^8$
(iii) $D=10, b=3$; 88,573 nodes				
`2FFS`	1.00	1.00	$1.31\times10^4 \pm 2.34\times10^3$	$2.49\times10^9 \pm 8.64\times10^8$
`BAI-MCTS`	0.98	0.98	$1.91\times10^7 \pm 4.19\times10^6$	$4.62\times10^9 \pm 9.34\times10^8$
`Minimax-fast`	1.00	0.90	$4.69\times10^4 \pm 5.67\times10^3$	$4.56\times10^6 \pm 5.81\times10^5$
`Slow-only`	0.77	0.77	$5.60\times10^6 \pm 8.98\times10^6$	$1.63\times10^9 \pm 4.22\times10^8$

We evaluate 2FFS on stochastic minimax trees, focusing on whether it can reduce search effort by adaptively combining cheap biased evaluations with expensive accurate samples. We compare against three fixed-confidence baselines: (i) BAI-MCTS sampling (`BAI-MCTS`) [5], which uses stochastic leaf samples; (ii) minimax fast-oracle (`Minimax-fast`), which uses only the deterministic fast oracle; and (iii) fixed-depth expansion followed by slow-oracle sampling on the exposed frontier (`Slow-only`). The second and third baselines are controlled ablations: they test, respectively, the limits of relying only on fast expansion and of committing to a non-adaptive expansion depth before slow sampling. We use the abbreviated names in parentheses in all tables. We note that BAI-MCTS [5, §4] has shown that it already substantially improves ($15\times$) over earlier tree-based BAI methods, including UGapE-MCTS [19], LUCB-MCTS [20], and FindTopWinner [21].

Setup and evaluation. Our main experiments use balanced b-ary minimax trees with $(D, b) \in \{(5, 8), (7, 6), (10, 3)\}$, with 100 independently generated trees per setting. These configurations cover both wide shallow trees and deeper narrower trees, and are much larger than the depth-3, 10-ary setting (only 1111 nodes) in BAI-MCTS [5], with more explicit evidence of efficiency improvement.

For evaluation, we report two complementary metrics. The first is *sampling count*, the total number of nodes sampled and visited. The second is *operation count*, the total number of bookkeeping operation units, where one operation count is one $\mathcal{O}(1)$ scalar state access, update, or comparison. This distinction is important because sampling one node does not necessarily induce the same amount of computation across methods. For example, after a stochastic leaf sample, BAI-MCTS mainly updates the sampled leaf interval and propagates confidence bounds along the leaf-to-root path. In contrast, after a slow oracle sample, 2FFS updates local slow intervals, intersects them with fast intervals, propagates minimax bounds, and maintains scale-dependent comparison certificates. Thus, sampling count measures efficiency of repetitive nodes visit, while operation count measures the overall computation efficiency.

Results. Table 1 shows that `2FFS` substantially reduces search effort relative to the `BAI-MCTS` baseline: across the reported settings, it uses 160–1450$\times$ fewer sampled or visited nodes and 1.9–4.6$\times$ fewer bookkeeping operations. The ablations illustrate the two sides of the fast–slow tradeoff.

`Minimax-fast` is computationally cheap because it relies only on fast expansion, but this efficiency comes at the cost of accuracy since the fast oracle is biased. Conversely, `Slow-only` uses accurate slow-oracle samples and can be accurate when enough samples are collected, but it requires many more samples and often fails to stop within the finite budget. Overall, these results suggest that 2FFS gains its efficiency by adaptively choosing when to trust fast expansion and when to spend slow samples on decision-critical nodes.

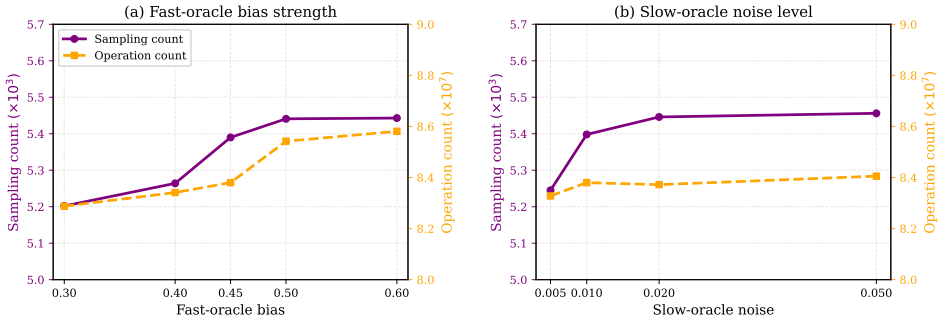

Figure 2: Oracle-bias ablation results for 2FFS: (a) fast-oracle bias and (b) slow-oracle noise.

Ablations. We further test the sensitivity of 2FFS to the fast-oracle bias and slow-oracle noise. Using $D = 5$, 8-ary trees, we sweep the fast-oracle bias with mean 0.45 and the slow-oracle noise with mean 0.01. Figure 2 shows that 2FFS is relatively stable across both ranges, suggesting that its performance is not overly sensitive to moderate changes in either oracle parameter.

5 Conclusion

In this work, we introduced 2FFS, a two-fidelity algorithm for fixed-confidence best-action identification in stochastic minimax trees. Our work bridges multi-fidelity bandits and minimax tree search by providing a principled mechanism for adaptively trading off cheap biased expansion against expensive accurate sampling. Theoretically, we prove PAC correctness, finite stopping, and a general-depth cost bound relative to an ideal recursive oracle complexity. Empirically, 2FFS achieves substantial gains in sampling efficiency over BAI-MCTS-style baselines while maintaining reliable root-action identification. These results suggest that two-fidelity tree search may be a useful abstraction beyond the current theoretical framework, with potential applications in multi-agent reinforcement learning, game-tree planning, and LLM policy-rollout systems, where accurate rollouts are expensive but cheap heuristic evaluations are widely available, a further discussion to these is provided in Appendix A.

Follow-up work. We note that 2FFS opens up a broad range of potential directions for future work. On the empirical side, we suggest investigating how 2FFS can be integrated with policy-gradient-based MARL, particularly how to design approximate envelopes that ensure the effectiveness of fast oracles. Such extensions could also be explored for training LLMs in multi-agent environments, such as chess playing and adversarial training. On the theoretical side, we first look forward to seeing how 2FFS can be adapted to non-minimax trees, which could facilitate extensions to tree-of-thought rollout structures for expansive LLM reasoning trajectories. Beyond this direction, within minimax trees, we also look forward to further developments in fast-oracle envelope design, as well as weaker assumptions for the cost upper bound, or even tighter upper bounds.

Acknowledgment

The authors declare no competing interests and have not receiving external funding for this work.

References

[1] K. Kandasamy, G. Dasarathy, B. Poczos, and J. Schneider. The multi-fidelity multi-armed bandit. *NeurIPS*, 2016.

[2] R. Poiani, A. M. Metelli, and M. Restelli. Multi-fidelity best-arm identification. *NeurIPS*, 2022.

[3] X. Wang, Q. Wu, W. Chen, and J. Lui. Multi-fidelity multi-armed bandits revisited. *NeurIPS*, 2023.

[4] R. Poiani, R. Degenne, E. Kaufmann, A. M. Metelli, and M. Restelli. Optimal multi-fidelity best-arm identification. *NeurIPS*, 2024.

[5] E. Kaufmann and W. M. Koolen. Monte-carlo tree search by best arm identification. *NeurIPS*, 2017.

[6] D. E. Knuth and R. W. Moore. An analysis of alpha-beta pruning. *Artificial Intelligence*, 1975.

[7] G. M. Baudet. An analysis of the full alpha-beta pruning algorithm. *STOC*, 1978.

[8] L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. *ECML*, 2006.

[9] M. Lanctot, M. H. M. Winands, T. Pepels, and N. R. Sturtevant. Monte carlo tree search with heuristic evaluations using implicit minimax backups. *2014 IEEE Conference on Computational Intelligence and Games*, 2014.

[10] J. Pearl. On the nature of pathology in game searching. *Artificial Intelligence*, 1983.

[11] K. Kandasamy, G. Dasarathy, J. B. Oliva, J. Schneider, and B. Poczos. Gaussian process bandit optimisation with multi-fidelity evaluations. *NeurIPS*, 2016.

[12] K. Kandasamy, G. Dasarathy, J. Schneider, and B. Póczos. Multi-fidelity bayesian optimisation with continuous approximations. *ICML*, 2017.

[13] A. Garivier and E. Kaufmann. Optimal best arm identification with fixed confidence. *COLT*, 2016.

[14] S. R. Howard, A. Ramdas, J. McAuliffe, and J. Sekhon. Time-uniform, nonparametric, nonasymptotic confidence sequences. *The Annals of Statistics*, 2021.

[15] E. Kaufmann and W. M. Koolen. Mixture martingales revisited with applications to sequential tests and confidence intervals. *JMLR*, 2021.

[16] J.-Y. Audibert, S. Bubeck, and R. Munos. Best arm identification in multi-armed bandits. *COLT*, 2010.

[17] K. Jamieson, M. Malloy, R. Nowak, and S. Bubeck. lil' ucb: An optimal exploration algorithm for multi-armed bandits. *COLT*, 2014.

[18] E. Kaufmann, O. Cappé, and A. Garivier. On the complexity of best-arm identification in multi-armed bandit models. *JMLR*, 2016.

[19] V. Gabillon, M. Ghavamzadeh, and A. Lazaric. Best arm identification: A unified approach to fixed budget and fixed confidence. *NeurIPS*, 2012.

[20] S. Kalyanakrishnan, A. Tewari, P. Auer, and P. Stone. Pac subset selection in stochastic multi-armed bandits. *ICML*, 2012.

[21] E. Even-Dar, S. Mannor, and Y. Mansour. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *JMLR*, 2006.

[22] M. Poloczek, J. Wang, and P. Frazier. Multi-information source optimization. *NeurIPS*, 2017.

[23] S. Li, W. Xing, R. Kirby, and S. Zhe. Multi-fidelity bayesian optimization via deep neural networks. *NeurIPS*, 2020.

[24] R. Sen, K. Kandasamy, and S. Shakkottai. Multi-fidelity black-box optimization with hierarchical partitions. *ICML*, 2018.

[25] C. Fiegel, V. Gabillon, and M. Valko. Adaptive multi-fidelity optimization with fast learning rates. *AISTATS*, 2020.

[26] R. Sen, K. Kandasamy, and S. Shakkottai. Noisy blackbox optimization using multi-fidelity queries: A tree search approach. *AISTATS*, 2019.

[27] R. Degenne, W. M. Koolen, and P. Ménard. Non-asymptotic pure exploration by solving games. *NeurIPS*, 2019.

[28] D. Ghosh, M. K. Hanawal, and N. Zlatanov. Fixed budget best arm identification in unimodal bandits. *TMLR*, 2024.

[29] R. Poiani, M. Jourdan, E. Kaufmann, and R. Degenne. Best-arm identification in unimodal bandits. *AISTATS*, 2025.

[30] K. Kanarios, Q. Zhang, and L. Ying. Cost aware best arm identification. *RLC*, 2024.

[31] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *JMLR*, 2018.

[32] K. Jamieson and A. Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. *AISTATS*, 2016.

[33] S. Falkner, A. Klein, and F. Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. *ICML*, 2018.

[34] L. Li, K. Jamieson, A. Rostamizadeh, E. Gonina, J. Ben-Tzur, M. Hardt, B. Recht, and A. Talwalkar. A system for massively parallel hyperparameter tuning. *MLSys*, 2020.

[35] H. E. Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58:527–535, 1952.

[36] L. Bonfiglio, P. Perdikaris, S. Brizzolara, and G. E. Karniadakis. Multi-fidelity optimization of super-cavitating hydrofoils. *Computer Methods in Applied Mechanics and Engineering*, 2018.

[37] P.-E. Réthoré, P. Fuglsang, G. C. Larsen, T. Buhl, T. J. Larsen, and H. A. Madsen. Topfarm: Multi-fidelity optimization of wind farms. *Wind Energy*, 2014.

[38] L. Zheng, T. L. Hedrick, and R. Mittal. A multi-fidelity modelling approach for evaluation and optimization of wing stroke aerodynamics in flapping flight. *Journal of Fluid Mechanics*, 2013.

[39] Z. Tang, D. Rybin, and T.-H. Chang. Zeroth-order optimization meets human feedback: Provable learning via ranking oracles. *ICLR*, 2024.

[40] P. Chen, X. Chen, W. Yin, and T. Lin. ComPO: Preference alignment via comparison oracles. *NeurIPS*, 2025.

[41] T. Lin, C. Jin, and M. I. Jordan. Two-timescale gradient descent ascent algorithms for nonconvex minimax optimization. *JMLR*, 2025.

[42] D. Guo, D. Yang, H. Zhang, J. Song, P. Wang, Q. Zhu, R. Xu, R. Zhang, S. Ma, X. Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081): 633–638, 2025.

[43] P. Chen, X. Li, Z. Li, X. Chen, and T. Lin. Stepwise guided policy optimization: Coloring your incorrect reasoning in GRPO. *TMLR*, 2026.

[44] Z. Li, C. Chen, T. Yang, T. Ding, R. Sun, G. Zhang, W. Huang, and Z.-Q. Luo. Knapsack RL: Unlocking exploration of LLMs via optimizing budget allocation. *ICML*, 2026.

[45] P. Chen, X. Li, Z. Li, W. Yin, X. Chen, and T. Lin. Exploration vs exploitation: Rethinking rlvr through clipping, entropy, and spurious reward. *ICLR*, 2026.

[46] P. Chen, X. Li, X. Chen, and T. Lin. Reward-free alignment for conflicting objectives. *ICML*, 2026.

[47] C. Park, Z. Chen, A. Ozdaglar, and K. Zhang. Post-training llms as better decision-making agents: A regret-minimization approach. *ICML*, 2026.

A Further Related Works and Limitations

Multi-fidelity bandits and optimization. Multi-fidelity methods study how to allocate queries across information sources with different costs and accuracies. In flat bandit models, the goal is typically to identify or compete with the best arm while using cheap biased fidelities when they are informative and reserving expensive accurate fidelities for difficult comparisons [1–4]. Related ideas also appear in multi-fidelity Bayesian and black-box optimization, where low-cost simulators, subsampled training runs, or surrogate evaluations are used to accelerate optimization under a total cost budget [11, 12, 22–25]. These works provide the broad cost–accuracy motivation for our two-oracle model, but they do not address minimax value propagation through an adversarial tree or fixed-confidence certification of a root action.

Relation to multi-fidelity tree search for noisy black-box optimization. Sen et al. [26] study noisy black-box optimization with multi-fidelity queries using a tree-search approach. Their tree is a hierarchical partition of the input domain of an unknown function; a node represents a region of the search space, and querying at a fidelity gives a biased and noisy estimate useful for optimistic optimization. Their algorithms, MFHOO and MFPOO, adapt HOO/POO-style optimistic tree search to continuous fidelities and are analyzed through simple regret under a finite cost budget.

This is related in spirit because both settings combine tree search with multi-fidelity information, but it is not the same problem. In our setting, the tree is not an auxiliary partition of a black-box domain: it is the stochastic minimax tree itself, with alternating MAX/MIN backups and a root-action decision. Our objective is fixed-confidence (ε, δ)-PAC best-action identification, not simple-regret minimization after a budget is exhausted. Methodologically, 2FFS maintains valid confidence intervals at minimax nodes, propagates them through adversarial backups, and adaptively chooses between local slow certification and recursive fast expansion. This differs from the optimistic partition-refinement mechanism of MFHOO/MFPOO, where the tree is used to explore a continuous domain and there is no minimax alternation, no root-child certification problem, and no two-route local-versus-recursive certificate of the kind used here.

Best-action identification and tree-based planning. Pure-exploration bandits distinguish fixed-budget regret-style objectives from fixed-confidence identification objectives. The flat best-arm identification literature characterizes instance-dependent sample complexity for identifying the best arm [18, 27], with extensions to structured settings such as unimodal bandits [28, 29] and cost-aware sampling [30]. In tree-based planning, classical minimax search and α-β pruning exploit deterministic backups in adversarial trees [6, 7], while MCTS uses stochastic sampling and bandit-style selection rules [8]. BAI-MCTS [5] brings fixed-confidence identification to minimax trees, but assumes a single unbiased sampling oracle. Our contribution combines this fixed-confidence root-action perspective with a calibrated cheap evaluator and an expensive accurate evaluator.

Applications and oracle-based learning. Multi-fidelity ideas are widely used in hyperparameter optimization [31–34], medical trial design [35], and engineering design problems such as hydrofoil, wind-farm, and aerodynamic optimization [36–38]. More broadly, oracle-based learning studies how decision-making systems can exploit structured feedback such as ranking or comparison oracles [39, 40]. Our work fits this broader line by studying how cheap heuristic information and expensive reliable feedback should be allocated in a structured decision tree and GDA-style minimax tree [41].

Future work. We acknowledge several limitations, which also open a broad space for follow-up work from both empirical and theoretical perspectives. Empirically, our experiments focus on simulated finite minimax trees, while many modern planning problems involve richer interactive environments (we note that adapting 2FFS to these areas would require the effort of a fully new follow-up work). Extending 2FFS to multi-agent reinforcement learning, policy-gradient-based training, or neural-guided MCTS would be an important step toward broader applications, but is beyond the scope of the present paper. In addition, our current formulation assumes finite trees with fixed depth (which aligns with the setup from previous tree-based BAI works); adapting the method to dynamically generated trees, progressive widening, or very large action spaces is a natural next direction. We also note such minimax planning can be smoothly integrated with the recent group-based language model policy optimization for language model reasoning [42–45] and language model alignment and decision making problems [46, 47].

Theoretically, our general-depth upper bound relies on mild local regularity assumptions on the sampling complexity. Understanding whether these conditions can be weakened, verified adaptively, or replaced by fully data-dependent stopping criteria would further sharpen the theory. Another important direction is to close the remaining gap between the current polynomial-depth factor and the ideal recursive oracle complexity, potentially leading to tighter depth dependence or sharper instance-dependent bounds.

B Missing Proofs and Further Technical Details

This section gives the proofs and algorithmic details omitted from the main text. We begin with a notation table in Appendix B.1.

B.1 Notation Glossary

Table 2: Glossary of notation used in the main text and appendix.

Notation	Meaning
1) **Tree, values, and root decision**	
$\mathcal{T}, r$	Finite game tree and its root. The root is a MAX node.
$\mathrm{Ch}(v)$	Set of children of node v.
$d(v)$, D, $h(v)$	Depth of v, total tree depth, and remaining depth $h(v) = D - d(v)$.
$\mathcal{L}, \ell$	Leaf set and a generic leaf.
μ_ℓ, $V^*(v)$	Mean payoff at leaf ℓ and true minimax value of node v.
$V^*(r, a)$	Root-action value, defined as the node value $V^*(a)$ for $a \in \mathrm{Ch}(r)$.
a^*, Δ_*	Unique optimal root action and root gap $V^*(r, a^*) - \max_{a \neq a^*} V^*(r, a)$.
$\mathcal{F}$, $\mathcal{F}_t$	A complete frontier cut and the frontier maintained at round t.
$\mathcal{T}_t$	Explored subtree at round t.
$p(v)$	Parent of node v.
Δ_v^{eff}	Effective gap of node v, i.e., the largest root-to-v bottleneck scale that can affect root-action identification.
2) **Two-fidelity oracles and sampling**	
F, S	Fast deterministic oracle and slow stochastic oracle.
$V_F(v)$	Fast-oracle value returned at node v.
$B(h)$	Known fast-oracle bias envelope at remaining depth h, with $B(0) = 0$.
$\nu_{v,m}$, $\mu_{v,m}$	Distribution and mean of fidelity-m observations at node v.
λ_1, λ_2, c	Fidelity costs, normalized as $\lambda_1 = 1$ and $\lambda_2 = c \geq 1$.
$Y_{v,s}$, σ	The s-th slow-oracle sample at v, assumed σ-sub-Gaussian around $V^*(v)$.
$N^F(t)$, $N_v^S(t)$	Total number of fast queries and number of slow queries at node v by round t.
C_t	Total search cost $N^F(t) + c\sum_v N_v^S(t)$.
τ, $\hat{a}_\tau$	Stopping time and returned root action.
ε, δ	PAC accuracy and failure probability.
$\boldsymbol{\delta}$, δ_v	Node-wise confidence allocation with $\sum_{v \neq r} \delta_v \leq \delta$.
$\widehat{V}_{v,n}$	Empirical mean of the first n slow samples at v.
$\beta_v(n, \delta_v)$	Time-uniform confidence radius for node v.
$\mathcal{E}_\delta$	Simultaneous-validity event on which all slow confidence intervals are valid.
3) **Intervals and local certification**	
I_v^F	Fast interval $[V_F(v) - B(h(v)),\ V_F(v) + B(h(v))]$.
$I_v^S(t)$	Running intersection of slow confidence intervals at node v.
$I_v^{\mathrm{loc}}(t)$	Direct local interval $I_v^F \cap I_v^S(t)$.
$I_v^{\mathrm{ch}}(t)$	Child-backup interval obtained by minimax propagation from children.
$I_v(t) = [L_v(t), U_v(t)]$	Effective interval used by the algorithm at node v.
$\mathrm{width}(I)$	Length of an interval I. In particular, $\mathrm{width}(I_v(t)) = U_v(t) - L_v(t)$.
$m_v(\rho, \delta_v)$	Minimum slow sample count needed so $\beta_v(n, \delta_v) \leq \rho/4$.

Continued on next page

Table 2: Glossary of notation used in the main text and appendix (continued).

Notation	Meaning
$\Gamma_v(\rho, \delta_v)$	Local slow-oracle cost to certify node v at precision ρ; it is zero if the fast interval is already narrow enough.
$J_v^*(\boldsymbol{\delta})$	Ideal recursive cost to certify subtree v at precision Δ_v^{eff}.
$H(\boldsymbol{\delta})$, H^*	Root recursive oracle complexity for allocation $\boldsymbol{\delta}$, and its infimum over feasible allocations.
4) **Dyadic scales and algorithmic certificates**	
ρ_k	Dyadic precision grid, initialized at ρ_0 and updated by $\rho_{k+1} = \rho_k/2$.
$\Gamma_v^{(k)}(\delta_v)$	Scale-k local certification cost $\Gamma_v(\rho_k, \delta_v)$.
$s \in \{L, U\}$, $\bar{s}$	Targeted endpoint side and its opposite side.
a_t, b_t	Current root leader $\mathrm{argmax}_a\, L_a(t)$ and challenger $\mathrm{argmax}_{a \neq a_t}\, U_a(t)$.
$\lambda_L(v,t)$, $\lambda_U(v,t)$	Comparison endpoints $\max_{u \in \mathrm{Ch}(v)} L_u(t)$ and $\min_{u \in \mathrm{Ch}(v)} U_u(t)$.
$A_s(v,k,t)$	Scale-aware active child set for certifying side s of node v at scale k.
$b_s(v,k,t)$	Current blocking child in a comparison case.
$\mathsf{Cert}_s(v,k,t)$	Observable side certificate at node v, side s, and scale k.
$\mathsf{Done}_s(v,k,t)$	Latched monotone flag recording that side s at scale k was certified earlier.
$\mathsf{Comp}_s(v,k,t)$	Completion predicate $\mathsf{Done}_s(v,k,t)$ or $\mathsf{Cert}_s(v,k,t)$.
$w_x(t)$, $\underline{k}(x,t)$	Current width $U_x(t) - L_x(t)$ and the dyadic width scale satisfying $\rho_k/2 < w_x(t) \leq \rho_k$.
$K_s(x,t)$	Node-local active scale: the coarsest unresolved scale for side s at node x.
$K_s^{\leq k}(x,t)$	Capped unresolved scale for side s, restricted to scales no finer than parent scale k.
$\sigma(u,t)$, $K_\pm(u,t)$	Root-challenger contender side and its corresponding contender scale.
$\textsc{Resolve}(v,s,k,B)$	Recursive resolver call for node v, side s, scale k, and invocation cost cap B.
5) **Budgets, counters, and proof accounting**	
α_h	Depth-budget multiplier for remaining depth h, e.g., $(h+1)^2$.
$\mathsf{B}_{v,k}^{\mathrm{rec}}$	Recursive race budget $\alpha_{h(v)}\Gamma_v^{(k)}(\delta_v)$ for node v and scale k.
$M_{v,k}^{\mathrm{loc}}(t)$	Cost of slow local samples charged to scale k at node v by round t.
$M_{v,k}^{\mathrm{exp}}(t)$	Cost spent below v while resolving the scale-k recursive obligation at v.
Λ_{pre}, Λ_{gap}, Λ_{loc}	Local regularity constants; $\Lambda_{\mathrm{loc}} = \Lambda_{\mathrm{pre}}\Lambda_{\mathrm{gap}}$.
P_h	Depth-overhead recursion used in the general-depth cost bound.
$\mathcal{K}_v^{\mathrm{act}}$	Active scales for v: $\{k \geq 0 : \Delta_v^{\mathrm{eff}} \leq 2\rho_k\}$.
$\mathfrak{C}_v(t)$	Total cost returned through node v, excluding the fast query exposing v.
$d_v(s,k;t)$	Minimum inherited cap permitting positive progress on the current unresolved obligation.
Q_v	Recursive comparison budget $P_{h(v)-1}J_v^*(\boldsymbol{\delta})$ when the recursive reference route is chosen.
$q_{\min}$	Minimum positive oracle-action cost, $\min\{c, 1\}$.

B.2 Proofs for Preliminaries

B.2.1 Proof of Lemma 2.3

Proof. We propagate validity and the width bound upward from the frontier. For any node u that belongs to $\mathcal{F}$ or is an ancestor of a node in $\mathcal{F}$, define

$$m(u) \coloneqq \max\{d(z) - d(u) : z \in \mathcal{F},\ u \text{ is an ancestor of } z \text{ or } u = z\}.$$

The frontier nodes are exactly those with $m(u) = 0$, where validity holds by assumption. We proceed by induction on $m(u)$.

Let $m(u) \geq 1$, and assume validity for all smaller values of m. Since u lies on a path to a frontier node, no strict ancestor of u belongs to $\mathcal{F}$; otherwise, that path would meet $\mathcal{F}$ twice. For every child $x \in \mathrm{Ch}(u)$, each root-to-leaf path through x must therefore meet $\mathcal{F}$ at or below x. Thus x either belongs to $\mathcal{F}$ or has a descendant in $\mathcal{F}$. Its propagated interval is valid by the induction hypothesis.

If u is a MAX node, taking maxima of the child bounds gives

$$\max_x L_x \le \max_x V^*(x) = V^*(u) \le \max_x U_x,$$

which is exactly $V^*(u) \in [L_u, U_u]$.

If u is a MIN node, taking minima gives

$$\min_x L_x \le \min_x V^*(x) = V^*(u) \le \min_x U_x.$$

This proves the validity statement.

For the half-width bound, use the same induction. Suppose every child interval of u has width at most $2w$. If u is a MAX node, choose $x^* \in \operatorname{argmax}_x U_x$. Then

$$U_u - L_u = U_{x^*} - \max_x L_x \le U_{x^*} - L_{x^*} \le 2w.$$

If u is a MIN node, choose $x^* \in \operatorname{argmin}_x L_x$ to obtain

$$U_u - L_u = \min_x U_x - L_{x^*} \le U_{x^*} - L_{x^*} \le 2w.$$

Every propagated ancestor interval therefore has half-width at most w. □

B.2.2 Proof of Lemma 2.4

Proof. Fix a round t. On $\mathcal{E}_\delta$, the slow interval is valid by Eq. (4), while the fast interval I_v^F is valid deterministically by Definition 2.2. Their intersection therefore satisfies

$$V^*(v) \in I_v^{\mathrm{loc}}(t) = I_v^F \cap I_v^S(t),$$

which holds for every exposed non-root node v.

It remains to check the recursive backups. We use induction upward through the explored tree. If a non-root node v is unexpanded, then $I_v^{\mathrm{ch}}(t) = (-\infty, +\infty)$, so validity of $I_v(t) = I_v^{\mathrm{loc}}(t) \cap I_v^{\mathrm{ch}}(t)$ reduces to local validity.

If v is an expanded non-root node, all children of v are explored and have valid effective intervals by the induction hypothesis. For a MAX node v, this gives

$$\max_{u \in \mathrm{Ch}(v)} L_u(t) \le \max_{u \in \mathrm{Ch}(v)} V^*(u) = V^*(v) \le \max_{u \in \mathrm{Ch}(v)} U_u(t),$$

so $V^*(v) \in I_v^{\mathrm{ch}}(t)$.

The MIN case is identical with maxima replaced by minima. Since $I_v^{\mathrm{loc}}(t)$ is also valid, the intersection $I_v(t) = I_v^{\mathrm{loc}}(t) \cap I_v^{\mathrm{ch}}(t)$ remains valid.

At the root, the interval consists only of the child backup. Applying the same MAX argument to its valid child intervals gives $V^*(r) \in I_r(t)$, completing the induction. □

B.3 Full 2FFS Details

We now give the full version of Algorithm 1. The resolver RESOLVE(v, s, k, B) includes an invocation-level cost cap B, omitted from the main-text pseudocode. This cap prevents a child call from exceeding the remaining budget inherited from its parent. All ties use fixed deterministic node and side orders, independent of the invocation cap and of ε.

To define the algorithm on every sample path, fix a default root action in advance. If an interval intersection becomes empty, terminate the entire run immediately and return this default action, before evaluating any widths or certificates. This exceptional rule never applies on $\mathcal{E}_\delta$ by Lemma 2.4; any error caused by it is therefore included in the confidence-failure event. All subsequent interval and scale operations are performed only while the intervals are nonempty.

The algorithm allows *local reversibility*: once a node v has been exposed, it may receive direct slow samples at v even after expansion. Thus the two certification routes at v,

- the *local route*, which shrinks $I_v^{\mathrm{loc}}(t)$ through slow samples at v, and

- the *recursive route*, which shrinks $I_v^{\mathrm{ch}}(t)$ by expanding below v and recursively refining its children,

remain available throughout the run. Work performed below v and samples taken at v both contribute to its effective interval.

The algorithm uses the dyadic target diameters $(\rho_k)_{k\ge 0}$ from §2: scale k means target width $\rho_k/2$, and $\Gamma_v^{(k)}(\delta_v) = \Gamma_v(\rho_k, \delta_v)$ is the corresponding local stopping cost.

Each outer-loop iteration computes an *active local scale* for the current leader and challenger, then selects a root-side obligation as specified below. A node v is resolved at scale k once the targeted side has an observable certificate at tolerance $\rho_k/2$.

The choice between local sampling and recursion is governed by a budget. Fix a nondecreasing polynomial depth-budget sequence

$$\alpha_h \ge 1, \qquad h = 0, 1, \dots, D,$$

for example $\alpha_h = (h+1)^2$. For an internal node v, the recursive route at scale k is allowed to spend at most

$$\mathsf{B}_{v,k}^{\mathrm{rec}} := \alpha_{h(v)}\, \Gamma_v^{(k)}(\delta_v)$$

below v, before the obligation falls back to direct slow sampling at v. The resolver first attempts recursive work and falls back to local sampling when its recursive budget is insufficient. To track spending at each explored non-root node v and scale k, define

$$M_{v,k}^{\mathrm{loc}}(t) := c \times \big(\text{number of slow queries issued at } v \text{ while resolving scale } k\big),$$

and $M_{v,k}^{\mathrm{exp}}(t)$ as the total cost already spent strictly below v while resolving the scale-k recursive obligation at v.

These counters record costs already incurred. The recursive counter enforces the budget, while the current certificates determine which child to refine. Together, these rules limit spending below a node and direct it toward children that still affect the required comparison.

At the root, let

$$a_t \in \operatorname*{argmax}_{a\in \mathrm{Ch}(r)} L_a(t), \qquad b_t \in \operatorname*{argmax}_{a\in \mathrm{Ch}(r)\setminus\{a_t\}} U_a(t)$$

be the current leader and challenger. 2FFS stops once some root action beats every competitor up to ε, i.e.,

$$\text{stop if there exists } \hat{a}_t \in \mathrm{Ch}(r) \text{ such that } L_{\hat{a}_t}(t) \ge \max_{a\ne \hat{a}_t} U_a(t) - \varepsilon.$$

Otherwise, 2FFS refines one uncertified decision-critical root side.

The challenger may either be excluded or become the leader. The algorithm therefore computes the current lower-side scale for the leader a_t and the current full-contender scale for the challenger b_t, then selects the obligation with the coarser unresolved diameter.

Inside the selected subtree, the algorithm first attempts recursive work. If the node's own recursive budget prevents progress, the resolver falls back to a direct slow sample, provided its inherited cap permits that sample. If the inherited cap prevents the chosen action, the call returns without spending. In the two selector cases $(\textsc{Min}, L)$ and $(\textsc{Max}, U)$, the recursive route selects an unresolved endpoint witness. In the two comparison cases $(\textsc{Max}, L)$ and $(\textsc{Min}, U)$, it selects the child whose endpoint most strongly obstructs the parent-side comparison certificate.

An internal blocker is refined through obligations capped at the parent scale. A call may improve either the parent's comparison side or the opposite endpoint used to exclude the blocker. It cannot use a finer scale than the parent obligation. This restriction is what allows the localization bound below to remain independent of depth.

For an expanded internal node v, define the scale-aware active child set $A_s(v,k,t) \subseteq \mathrm{Ch}(v)$ as follows. In the two selector cases, we use the endpoint witness sets:

- if $s = L$ and v is a MIN node, then

$$A_L(v,k,t) = \operatorname*{argmin}_{u\in \mathrm{Ch}(v)} L_u(t);$$

- if $s = U$ and v is a MAX node, then

$$A_U(v,k,t) = \operatorname*{argmax}_{u \in \mathrm{Ch}(v)} U_u(t);$$

In the two comparison cases we instead compare against the current best comparison endpoint and define a lazy set of critical children:

- if $s = L$ and v is a MAX node, let

$$\lambda_L(v,t) \coloneqq \max_{u \in \mathrm{Ch}(v)} L_u(t),$$

and set

$$A_L(v,k,t) = \big\{u \in \mathrm{Ch}(v) : U_u(t) > \lambda_L(v,t) + \rho_k/2 \text{ and } \neg \mathsf{Comp}_L(u,k,t)\big\};$$

- if $s = U$ and v is a MIN node, let

$$\lambda_U(v,t) \coloneqq \min_{u \in \mathrm{Ch}(v)} U_u(t),$$

and set

$$A_U(v,k,t) = \big\{u \in \mathrm{Ch}(v) : L_u(t) < \lambda_U(v,t) - \rho_k/2 \text{ and } \neg \mathsf{Comp}_U(u,k,t)\big\}.$$

Therefore, $A_s(v,k,t)$ contains only the children that are still unresolved at the current scale for the comparison required by the parent. In a comparison case, a child is discharged either by endpoint exclusion, namely falling outside the displayed margin, or by same-side completion at scale k. A discharged child is no longer recursively refined for that parent-scale obligation. Equivalently, the two comparison certificates use the following dual discharge rules:

$$\begin{aligned} &\text{MAX, } L\text{:} \quad u \text{ is discharged if } U_u(t) \le \lambda_L(v,t) + \rho_k/2 \text{ or } \mathsf{Comp}_L(u,k,t),\\ &\text{MIN, } U\text{:} \quad u \text{ is discharged if } L_u(t) \ge \lambda_U(v,t) - \rho_k/2 \text{ or } \mathsf{Comp}_U(u,k,t). \end{aligned}$$

So the first line excludes a MAX-node child through its upper endpoint or certifies it on the lower side, while the second line is the exact dual for MIN nodes.

When (v,s) is a comparison case and $A_s(v,k,t) \neq \varnothing$, define the current blocking contender by

$$\begin{cases} b_L(v,k,t) \in \operatorname{argmax}_{u \in A_L(v,k,t)} U_u(t), & \text{if } v \text{ is a MAX node and } s = L,\\ b_U(v,k,t) \in \operatorname{argmin}_{u \in A_U(v,k,t)} L_u(t), & \text{if } v \text{ is a MIN node and } s = U. \end{cases}$$

Recursive descent works on this blocker and recomputes it after every unit of work. Other active children are considered when they become the blocker.

2FFS maintains scale-local side-completion flags $\mathsf{Done}_s(v,k) \in \{0,1\}$. These flags are initialized to 0 when the pair (v,k) is first used, and are only changed from 0 to 1. When used as a predicate, $\mathsf{Done}_s(v,k,t)$ denotes that the flag is equal to 1 at round t. An unstored flag is interpreted as zero; storage is initialized before any assignment to it.

We call (v,s) a comparison case if either v is a MAX node with $s = L$, or v is a MIN node with $s = U$. The selector cases are (MIN, L) and (MAX, U). In a comparison case, the children in $A_s(v,k,t)$ are exactly those that are neither endpoint-excluded nor same-side complete at scale k.

Among them, only the current blocker $b_s(v,k,t)$ is eligible for recursive comparison work. The blocker may be refined either on the parent-targeted side s, which supports promotion or same-side safety, or on the opposite side $\bar{s}$, which supports endpoint exclusion, but both calls are capped by the current parent scale k. Note that no child is required to become fully complete merely because it is currently active.

For each fixed scale k and round t, we define the predicates $\mathsf{Cert}_s(v,k,t)$ and $\mathsf{Comp}_s(v,k,t)$ upward from the leaves of the currently explored tree. The observable side certificate $\mathsf{Cert}_s(v,k,t)$ holds if either

$$U_v(t) - L_v(t) \le \rho_k/2,$$

or

$$\mathrm{width}\big(I_v^{\mathrm{loc}}(t)\big) \le \rho_k/2,$$

or v is an expanded internal node and one of the following recursive alternatives holds:

$$\begin{cases} \text{for every } u \in \mathrm{Ch}(v),\ \big[U_u(t) \le \lambda_L(v,t) + \rho_k/2 \text{ or } \mathsf{Comp}_L(u,k,t)\big], & \text{if } v \text{ is MAX and } s = L, \\ \text{for every } u \in \mathrm{Ch}(v),\ \big[L_u(t) \ge \lambda_U(v,t) - \rho_k/2 \text{ or } \mathsf{Comp}_U(u,k,t)\big], & \text{if } v \text{ is MIN and } s = U, \\ \mathsf{Comp}_s(u,k,t) \quad \text{for every } u \in A_s(v,k,t), & \text{if } (v,s) \text{ is a selector case.} \end{cases}$$

For leaves, only the first two alternatives can apply. Finally, the following are equivalent

$$\mathsf{Comp}_s(v,k,t) \quad \Longleftrightarrow \quad \mathsf{Done}_s(v,k,t) \ \text{ or } \ \mathsf{Cert}_s(v,k,t).$$

This joint definition of Cert, Comp, the comparison endpoints $\lambda_s(v,t)$, and the lazy active sets $A_s(v,k,t)$ is well-founded, as the only recursive certificate reference is from a node to its children. Every evaluation at a finite scale k, including evaluations inside the scale searches below, first initializes the scale storage at all exposed non-root nodes and then evaluates the predicates in bottom-up order. Whenever a true $\mathsf{Cert}_s(v,k,t)$ is observed, it is immediately latched by setting $\mathsf{Done}_s(v,k) \leftarrow 1$. Thus a certificate observed during a scale search is retained just as one observed after an oracle update. The routine EVALUATESCALE(k) below implements this convention.

For any explored non-root node x with width $w_x(t) = U_x(t) - L_x(t) > 0$, let $\underline{k}(x,t)$ be the unique integer $k \ge 0$ such that

$$\rho_k/2 < w_x(t) \le \rho_k.$$

This integer exists: balanced depth gives $\rho_0 = 2B(D-1)$, and every exposed node has $w_x(t) \le 2B(h(x)) \le \rho_0$, because its effective interval is contained in its fast interval. When $\rho_0 = 0$, initialization already determines all root-action values and the stopping test succeeds. The node-local active scale of side $s \in \{L,U\}$ at x is

$$K_s(x,t) \coloneqq \min\{k \ge \underline{k}(x,t) :\ \neg\mathsf{Comp}_s(x,k,t)\},$$

with the convention $K_s(x,t) = +\infty$ if $w_x(t) = 0$ or if the set is empty. Thus 2FFS never refines a side at a scale where that side is already certified; if the width-derived scale is already complete, it moves directly to the next unresolved finer scale for the same side of the same node.

Finite implementation of the uncapped scale search. The convention $K_s = +\infty$ does not require an infinite search. Define $Z_s(v,t)$ by the same bottom-up formula as Cert_s, replacing the tolerance $\rho_k/2$ by zero and every child predicate $\mathsf{Comp}_{s'}(u,k,t)$ by $Z_{s'}(u,t)$. This auxiliary evaluation ignores all Done flags and neither initializes nor changes storage. If $Z_s(x,t)$ holds, monotonicity of these formulas in their tolerance shows that $\mathsf{Comp}_s(x,k,t)$ holds for every finite k, so the search returns $+\infty$.

Otherwise, the search has a finite upper limit. To see this, collect the strictly positive values among all current effective and local interval widths and all comparison differences $U_u - \lambda_L(v,t)$ and $\lambda_U(v,t) - L_u$ appearing in the certificate formulas. This is a finite set, and it is nonempty when $w_x(t) > 0$; let $b(t) > 0$ be its minimum. Let k_{stored} be the largest scale currently stored, or -1 if none is stored. Choose a finite $k_{\mathrm{test}} \ge \underline{k}(x,t)$ with $k_{\mathrm{test}} > k_{\mathrm{stored}}$ and $\rho_{k_{\mathrm{test}}}/2 < b(t)$. At this scale all initial flags are zero, and every primitive comparison has the same truth value as at zero tolerance. A bottom-up evaluation therefore gives $\mathsf{Comp}_s(x,k_{\mathrm{test}},t) = Z_s(x,t) = 0$. Evaluating the finitely many scales from $\underline{k}(x,t)$ through k_{test} returns the first unresolved one. Latching during this finite scan does not alter this argument, since flags at different scales are distinct. Consequently every uncapped scale search terminates after finitely many arithmetic and predicate operations.

The root selection is also well-defined outside $\mathcal{E}_\delta$ while all intervals remain nonempty. Indeed, bottom-up induction gives

$$Z_L(v,t) \text{ and } Z_U(v,t) \quad \Longrightarrow \quad U_v(t) = L_v(t).$$

If either zero-tolerance predicate uses a width alternative, this is immediate. Otherwise, suppose both predicates hold recursively but v has positive width. At a MAX node, choose $u \in \mathrm{argmax}_x\, U_x$. Since $I_v \subseteq I_v^{\mathrm{ch}}$, positive width implies $U_u > \max_x L_x$. The upper selector certificate gives $Z_U(u,t)$, and the lower comparison certificate then requires $Z_L(u,t)$. By induction $U_u = L_u \le \max_x L_x$, a contradiction. The MIN case is dual, using a child in $\mathrm{argmin}_x\, L_x$. An unexpanded node has no recursive certificate, which supplies the induction base. The finite search above therefore shows that a positive-width node cannot have both $K_L = K_U = +\infty$. Whenever the root stopping test fails,

$U_{b_t} > L_{a_t} + \varepsilon \geq L_{b_t}$, so the challenger has positive width and a finite contender scale. Thus the selected root obligation is finite on every path on which the algorithm has not already terminated.

For a finite parent scale k, define the capped unresolved scale

$$K_s^{\leq k}(x,t) \coloneqq \min\{0 \leq j \leq k : \neg\mathsf{Comp}_s(x,j,t)\},$$

with the convention $K_s^{\leq k}(x,t) = +\infty$ if the set is empty.

A recursive call made while resolving a parent obligation at scale k may only call a child at a capped scale $K_{s'}^{\leq k}(u,t)$.

Thus, recursive descent can refine a child at a coarser unresolved scale, but never at a scale finer than the current parent obligation.

The root challenger has no parent scale to cap its obligations. Its contender side is defined as follows. If at least one of $K_L(u,t)$ and $K_U(u,t)$ is finite, set

$$\sigma(u,t) \in \operatorname*{argmax}_{s\in\{L,U\}:\, K_s(u,t)<+\infty} \rho_{K_s(u,t)},$$

breaking ties arbitrarily, and define its contender scale by

$$K_\pm(u,t) \coloneqq K_{\sigma(u,t)}(u,t).$$

If both $K_L(u,t)$ and $K_U(u,t)$ are $+\infty$, we set $\sigma(u,t) = U$ arbitrarily and $K_\pm(u,t) = +\infty$. In this case, the contender has no unresolved root-side obligation.

A full-contender call then uses $\textsc{Resolve}(u, \sigma(u,t), K_\pm(u,t))$. For internal comparison blockers, we instead use capped side obligations. Write $\bar{L} = U$ and $\bar{U} = L$. During $\textsc{Resolve}(v,s,k)$, an internal blocker $b_s(v,k,t)$ is first called on the opposite side $\bar{s}$ at scale $K_{\bar{s}}^{\leq k}(b_s(v,k,t),t)$, whenever that capped obligation is finite. This is the endpoint that appears in the exclusion inequality. Only after the blocker has no unresolved opposite-side obligation at scales $\leq k$, while remaining active, is it called on the same side s at scale $K_s^{\leq k}(b_s(v,k,t),t)$.

The same-side blocker obligation can then discharge the blocker through $\mathsf{Comp}_s(b_s,k,t)$ or allow it to promote through the comparison-side endpoint. When evaluating a finite scale in K_s or $K_s^{\leq k}$, the EVALUATESCALE convention above initializes and latches the required scale-local flags. The fourth argument of $\textsc{Resolve}(v,s,k,B)$ is an invocation-level cost cap: the call may not perform a positive oracle action whose returned cost would exceed B. We write $\textsc{Resolve}(v,s,k)$ in the analysis when the cap is irrelevant. Each actual recursive call receives the minimum of the parent obligation's remaining recursive budget and its inherited invocation cap.

Remark B.1 (Scale-synchronous progress)**.** *Recursive descent is required to remain synchronized with the current parent obligation and its remaining recursive budget. Concretely, inside a call* $\textsc{Resolve}(v,s,k,B)$*, a child call* $\textsc{Resolve}(u,s',K_{s'}^{\leq k}(u,t),B_{\mathrm{rec}})$ *is admissible only if the scale-k certificate at v is still unresolved at the current round, the child is selected by the current scale-k rule at v, and B_{rec} is no larger than the remaining budget $\mathsf{B}_{v,k}^{\mathrm{rec}} - M_{v,k}^{\mathrm{exp}}$. In particular,*

- *in a comparison case, the algorithm may refine only the current blocking contender $b_s(v,k,t) \in A_s(v,k,t)$; the blocker is refined first on $(b_s(v,k,t), \bar{s}, K_{\bar{s}}^{\leq k}(b_s(v,k,t),t))$ whenever this obligation is finite, and otherwise on $(b_s(v,k,t), s, K_s^{\leq k}(b_s(v,k,t),t))$;*
- *in a selector case, the algorithm may refine only a child $u \in A_s(v,k,t)$ with $\neg\mathsf{Comp}_s(u,k,t)$, and it calls that child at $K_s^{\leq k}(u,t)$.*

Every invocation of $\textsc{Resolve}(v,s,k,B)$ *performs at most one positive oracle action, either a slow query or an expansion, at the current node or below it. A local query may follow a zero-cost blocked child return. After each positive action, the algorithm updates the intervals, comparison endpoints, blockers, and active child sets before descending again. Thus the coarser-scale tests that could already certify or exclude a child from the current scale-k obligation are checked before that child is refined further.*

The implementation below is split into reusable routines. Algorithm 2 contains scale and selection utilities, Algorithm 3 contains the atomic oracle actions, Algorithm 4 is the budgeted recursive resolver, and Algorithm 5 is the short top-level driver.

Algorithm 2 2FFS scale and selection utilities

procedure ENSURESCALE(v, k)
  if the counters and flags for (v, k) are undefined **then**
    Set $M^{\mathrm{loc}}_{v,k} \leftarrow 0$ and $M^{\mathrm{exp}}_{v,k} \leftarrow 0$
    Set $\mathsf{Done}_L(v, k) \leftarrow 0$ and $\mathsf{Done}_U(v, k) \leftarrow 0$
  end if
end procedure
procedure EVALUATESCALE(k)
  for all exposed non-root nodes y **do**
    ENSURESCALE(y,k)
  end for
  for all exposed non-root nodes y in bottom-up order **do**
    for all $s' \in \{L, U\}$ **do**
      if $\mathsf{Cert}_{s'}(y, k, t)$ **then**
        $\mathsf{Done}_{s'}(y, k) \leftarrow 1$
      end if
    end for
  end for
end procedure
procedure REFRESHANDLATCH(k)
  Recompute affected intervals and propagate them to the root; on any empty intersection, terminate the entire run with the fixed default action
  EVALUATESCALE(k)
end procedure
procedure CONTENDERSCALE(x)
  Compute $k_L \leftarrow K_L(x, t)$ and $k_U \leftarrow K_U(x, t)$
  if $k_L < +\infty$ and $\big(k_U = +\infty$ or $\rho_{k_L} \geq \rho_{k_U}\big)$ **then**
    return (L, k_L)
  else
    return (U, k_U)
  end if
end procedure
procedure CAPPEDSCALE(x, s, k)
  return $K_s^{\leq k}(x, t)$
end procedure
procedure RACEBUDGET(v, k)
  return $\alpha_{h(v)}\Gamma_v^{(k)}(\delta_v)$
end procedure
procedure ROOTOBLIGATION
  Compute $a \in \mathrm{argmax}_{u \in \mathrm{Ch}(r)} L_u(t)$ and $b \in \mathrm{argmax}_{u \in \mathrm{Ch}(r) \setminus \{a\}} U_u(t)$
  Compute $k_a \leftarrow K_L(a, t)$ and $(s_b, k_b) \leftarrow$ CONTENDERSCALE(b)
  if $k_a < +\infty$ and $\big(k_b = +\infty$ or $\rho_{k_a} \geq \rho_{k_b}\big)$ **then**
    return (a, L, k_a)
  else
    return (b, s_b, k_b)
  end if
end procedure
procedure CHILDOBLIGATION(v, s, k)
  if (v, s) is a comparison case **then**
    Let $b \leftarrow b_s(v, k, t)$
    $k_b^{\bar{s}} \leftarrow$ CAPPEDSCALE($b, \bar{s}, k$)
    if $k_b^{\bar{s}} < +\infty$ **then**
      return $(b, \bar{s}, k_b^{\bar{s}})$
    end if
    $k_b^{s} \leftarrow$ CAPPEDSCALE(b, s, k)
    return (b, s, k_b^{s})
  end if
  Choose $u \in A_s(v, k, t)$ with $\neg\mathsf{Comp}_s(u, k, t)$, maximizing $U_u(t) - L_u(t)$
  $k_u \leftarrow$ CAPPEDSCALE(u, s, k)
  return (u, s, k_u)
end procedure

Algorithm 3 2FFS atomic oracle actions

procedure LOCALSTEP(v, k, B)
  if $B < c$ **then**
    return $(0, \mathsf{blocked})$
  end if
  Query S once at v and update $I_v^S(t)$, $I_v^{\mathrm{loc}}(t)$, and $I_v(t)$
  $M_{v,k}^{\mathrm{loc}} \leftarrow M_{v,k}^{\mathrm{loc}} + c$
  REFRESHANDLATCH(k)
  return $(c, \mathsf{progress})$
end procedure
procedure EXPANDNODE(v, k)
  Add $\mathrm{Ch}(v)$ to $\mathcal{T}_t$ and query F on each child
  for all $u \in \mathrm{Ch}(v)$ **do**
    ENSURESCALE(u,k)
  end for
  $M_{v,k}^{\mathrm{exp}} \leftarrow M_{v,k}^{\mathrm{exp}} + |\mathrm{Ch}(v)|$
  REFRESHANDLATCH(k)
  return $(|\mathrm{Ch}(v)|, \mathsf{progress})$
end procedure

Algorithm 4 2FFS budgeted recursive resolver

procedure RESOLVE(v, s, k, B)
  ENSURESCALE(v,k)
  if $\mathsf{Comp}_s(v, k, t)$ **then**
    Set $\mathsf{Done}_s(v, k) \leftarrow 1$
    return $(0, \mathsf{cert})$
  end if
  if $h(v) = 0$ **then**
    return LOCALSTEP(v,k,B)
  end if
  $R \leftarrow$ RACEBUDGET$(v, k) - M_{v,k}^{\mathrm{exp}}$
  if $R \leq 0$ **then**
    return LOCALSTEP(v,k,B)
  end if
  $B_{\mathrm{rec}} \leftarrow \min\{B, R\}$
  if v is unexpanded **then**
    if $|\mathrm{Ch}(v)| > R$ **then**
      return LOCALSTEP(v,k,B)
    else if $|\mathrm{Ch}(v)| > B$ **then**
      return $(0, \mathsf{blocked})$
    end if
    return EXPANDNODE(v,k)
  end if
  $(u, s_u, k_u) \leftarrow$ CHILDOBLIGATION(v, s, k)
  $(q, z) \leftarrow$ RESOLVE$(u, s_u, k_u, B_{\mathrm{rec}})$
  if $z = \mathsf{blocked}$ **then**
    if $B < R$ **then**
      return $(0, \mathsf{blocked})$
    end if
    return LOCALSTEP(v,k,B)
  end if
  $M_{v,k}^{\mathrm{exp}} \leftarrow M_{v,k}^{\mathrm{exp}} + q$
  REFRESHANDLATCH(k)
  return (q, z)
end procedure

Algorithm 5 2FFS top-level driver

Input: node-wise confidence schedule (δ_v), accuracy ε, depth budgets $(\alpha_h)_{h=0}^{D}$
$\mathcal{T}_0 \leftarrow \{r\} \cup \mathrm{Ch}(r)$
for all $a \in \mathrm{Ch}(r)$ **do**
    Query F at a and initialize I_a^F, $I_a^{\mathrm{loc}}(0)$, and $I_a(0)$
end for
Propagate intervals to the root through Eq. (6)
Set $\rho_0 \leftarrow \max_{a \in \mathrm{Ch}(r)}(U_a(0) - L_a(0))$ and $\rho_k \leftarrow 2^{-k}\rho_0$ for $k \geq 0$ (evaluated as needed)
while there is no $\hat{a}_t \in \mathrm{Ch}(r)$ with $L_{\hat{a}_t}(t) \geq \max_{a \neq \hat{a}_t} U_a(t) - \varepsilon$ **do**
    $(x, s, k) \leftarrow$ RootObligation
    Resolve$(x, s, k, +\infty)$
end while
return any root child $\hat{a}_\tau$ with $L_{\hat{a}_\tau}(\tau) \geq \max_{a \neq \hat{a}_\tau} U_a(\tau) - \varepsilon$

Lazy critical children. In the selector cases, the live sets reduce to the current endpoint witness set. In the comparison cases, a child is discharged once it is endpoint-excluded by the scale-k comparison margin or same-side complete at scale k. Among the still-live children, only the endpoint-most blocker is eligible for recursive comparison work. This blocker is refined through capped side obligations only, so it may either become safe on the comparison side or be excluded through the opposite endpoint without being pushed to a finer scale than the parent.

B.4 Proof of Theorem 3.1

Proof. By Eq. (5) and the union bound in §2,

$$\mathbb{P}(\mathcal{E}_\delta) \geq 1 - \delta.$$

We prove that no ε-incorrect finite recommendation can occur on $\mathcal{E}_\delta$. Fix an outcome in $\mathcal{E}_\delta$ and suppose the algorithm stops at a finite time τ. Let $\hat{a}_\tau$ be any returned root child. Interval validity rules out the exceptional empty-intersection exit on this outcome. Hence the ordinary stopping rule gives

$$L_{\hat{a}_\tau}(\tau) \geq \max_{a \neq \hat{a}_\tau} U_a(\tau) - \varepsilon.$$

Interval validity holds simultaneously at every round, so Lemma 2.4 also applies at the finite stopping time τ. For every $a \in \mathrm{Ch}(r)$,

$$L_a(\tau) \leq V^*(r, a) \leq U_a(\tau).$$

Combining validity with the stopping inequality gives

$$V^*(r, \hat{a}_\tau) \geq L_{\hat{a}_\tau}(\tau) \geq \max_{a \neq \hat{a}_\tau} U_a(\tau) - \varepsilon \geq \max_{a \neq \hat{a}_\tau} V^*(r, a) - \varepsilon.$$

If $\hat{a}_\tau = a^*$, then the returned action is optimal. Otherwise $a^* \neq \hat{a}_\tau$, and the last display with $a = a^*$ gives

$$V^*(r, a^*) - V^*(r, \hat{a}_\tau) \leq \varepsilon.$$

Thus, every finite recommendation on $\mathcal{E}_\delta$ is ε-optimal, and the error event in the theorem is contained in $\mathcal{E}_\delta^c$, whose probability is at most δ. □

B.5 Derived Lemmas for the General-Depth Bound

We first establish soundness of the side certificates and then show that recursive calls remain at scales compatible with the effective gaps. These facts will support the cost bound in Theorem 3.6. Throughout, we work on the simultaneous-validity event $\mathcal{E}_\delta$, set $\varepsilon = 0$, and use Assumption 3.3 only in the aggregate summation and charging steps. All dyadic statements below concern $\rho_0 > 0$. If $\rho_0 = 0$, then $B(D-1) = 0$ and monotonicity of B gives $B(h(v)) = 0$ for every non-root node. Initialization already identifies the optimal root action, and every $\Gamma_v(\Delta_v^{\mathrm{eff}}, \delta_v)$ and J_v^* is zero. Hence $C_\tau = H(\boldsymbol{\delta}) = |\mathrm{Ch}(r)|$, without evaluating any dyadic cost at zero precision or requiring dyadic regularity.

The certificates control one endpoint at a time. We use the following notion of safety to express that control.

Definition B.1 (Side-specific scale safety)**.** *Fix a scale index $k \geq 0$, a round t, an explored non-root node v, and a side $s \in \{L, U\}$. We say that v is (s, k)-safe at round t if*

$$\begin{cases} V^*(v) - L_v(t) \leq \rho_k/2, & \text{if } s = L, \\ U_v(t) - V^*(v) \leq \rho_k/2, & \text{if } s = U. \end{cases}$$

On $\mathcal{E}_\delta$, both one-sided errors in Definition B.1 are nonnegative by Lemma 2.4. Moreover, if

$$U_v(t) - L_v(t) \leq \rho_k/2,$$

then v is both (L, k)-safe and (U, k)-safe.

Lemma B.2 (Nested interval updates)**.** *For every node v whose effective interval is defined at round t including the root, the effective intervals are nested over time: if $t' \geq t$, then*

$$I_v(t') \subseteq I_v(t).$$

Equivalently, $L_v(t') \geq L_v(t)$ and $U_v(t') \leq U_v(t)$. The same nesting holds for $I_v^{\mathrm{loc}}(t)$ whenever $v \neq r$ is exposed by round t.

Proof. We check the local intervals first, then propagate nesting up the explored tree. At an exposed non-root node, the fast interval is fixed, while the running intersection in Eq. (4) can only shrink as slow samples arrive. Therefore $I_v^{\mathrm{loc}}(t) = I_v^F \cap I_v^S(t)$ is nested.

For the child backups, both minimax operators preserve the required endpoint monotonicity. At a MAX node, the lower endpoint $\max_u L_u(t)$ is nondecreasing and the upper endpoint $\max_u U_u(t)$ is nonincreasing whenever every child interval is nested.

At a MIN node, the corresponding endpoints are $\min_u L_u(t)$ and $\min_u U_u(t)$, with the same monotonicity. Induction from the leaves of the explored tree therefore shows that $I_v^{\mathrm{ch}}(t)$ is nested after expansion. Before expansion it is $(-\infty, +\infty)$, so the first expansion preserves nesting as well. Intersecting the local and child-backup intervals proves the claim for $I_v(t)$ at every non-root node. At the root, the effective interval is the child backup itself. □

Interval nesting preserves an endpoint-accuracy guarantee once established. We now verify that the side certificates provide such a guarantee.

Lemma B.3 (Comparison-certificate soundness)**.** *Fix a round t, a scale index $k \geq 0$, an explored non-root node v, and a side $s \in \{L, U\}$. On $\mathcal{E}_\delta$, if $\mathsf{Comp}_s(v, k, t)$ holds, then v is (s, k)-safe at round t.*

Proof. We induct on the remaining depth $h(v)$, with the induction claim uniform over all rounds, scales, and sides: whenever $\mathsf{Comp}_s(v, k, t)$ holds, v is (s, k)-safe at round t. Fix v and assume the claim for every child of v. We first verify the current certificate $\mathsf{Cert}_s(v, k, t)$, then use interval nesting to handle certificates latched at an earlier round.

If

$$U_v(t) - L_v(t) \leq \rho_k/2,$$

then Lemma 2.4 gives $V^*(v) \in [L_v(t), U_v(t)]$, so both one-sided errors are at most $\rho_k/2$.

If

$$\mathrm{width}\big(I_v^{\mathrm{loc}}(t)\big) \leq \rho_k/2,$$

then $I_v(t) \subseteq I_v^{\mathrm{loc}}(t)$, while interval validity places $V^*(v)$ in $I_v(t)$. Thus this alternative also bounds both one-sided errors by $\rho_k/2$.

It remains to consider $\mathsf{Cert}_s(v, k, t)$ obtained through the recursive alternative. The selector cases use an endpoint witness; the comparison cases require a bound for every child.

MIN *node,* $s = L$. The selector set is

$$A_L(v, k, t) = \arg\min_{u \in \mathrm{Ch}(v)} L_u(t),$$

which is nonempty. The recursive certificate requires every $u \in A_L(v, k, t)$ to satisfy $\mathsf{Comp}_L(u, k, t)$; the induction hypothesis therefore makes each such child (L, k)-safe. Choose any $x \in A_L(v, k, t)$. Its endpoint is the selector minimum, and the parent value is no larger than its value:

$$L_x(t) = \min_{u \in \mathrm{Ch}(v)} L_u(t) \qquad \text{and} \qquad V^*(v) = \min_{u \in \mathrm{Ch}(v)} V^*(u) \leq V^*(x),$$

and containment $I_v(t) \subseteq I_v^{\mathrm{ch}}(t)$ gives

$$L_v(t) \geq \min_{u \in \mathrm{Ch}(v)} L_u(t) = L_x(t),$$

so the selected child's safety yields

$$V^*(v) - L_v(t) \leq V^*(x) - L_x(t) \leq \rho_k/2.$$

MAX *node,* $s = U$. The selector set is

$$A_U(v,k,t) = \arg\max_{u \in \mathrm{Ch}(v)} U_u(t),$$

which is again nonempty. Every $u \in A_U(v,k,t)$ satisfies $\mathsf{Comp}_U(u,k,t)$ and is therefore (U,k)-safe by the induction hypothesis. Choose any $x \in A_U(v,k,t)$. The selector maximum and the minimax definition give

$$U_x(t) = \max_{u \in \mathrm{Ch}(v)} U_u(t) \qquad \text{and} \qquad V^*(v) = \max_{u \in \mathrm{Ch}(v)} V^*(u) \geq V^*(x),$$

and containment $I_v(t) \subseteq I_v^{\mathrm{ch}}(t)$ gives

$$U_v(t) \leq \max_{u \in \mathrm{Ch}(v)} U_u(t) = U_x(t),$$

so the selected child's safety yields

$$U_v(t) - V^*(v) \leq U_x(t) - V^*(x) \leq \rho_k/2.$$

MAX *node,* $s = L$. Let $\lambda \coloneqq \lambda_L(v,t) = \max_{w \in \mathrm{Ch}(v)} L_w(t)$. Fix any child $u \in \mathrm{Ch}(v)$. If it is endpoint-excluded, then

$$V^*(u) \leq U_u(t) \leq \lambda + \rho_k/2.$$

If the child is not endpoint-excluded, the certificate requires $\mathsf{Comp}_L(u,k,t)$. The induction hypothesis then gives

$$V^*(u) \leq L_u(t) + \rho_k/2 \leq \lambda + \rho_k/2.$$

Taking the minimax backup over these child bounds gives

$$V^*(v) = \max_{u \in \mathrm{Ch}(v)} V^*(u) \leq \lambda + \rho_k/2.$$

Since $I_v(t) \subseteq I_v^{\mathrm{ch}}(t)$ and $\lambda = \max_u L_u(t)$,

$$L_v(t) \geq \max_{u \in \mathrm{Ch}(v)} L_u(t) = \lambda.$$

Combining the last two bounds yields

$$V^*(v) - L_v(t) \leq \rho_k/2.$$

MIN *node,* $s = U$. We now bound every child from below. Let $\lambda \coloneqq \lambda_U(v,t) = \min_{w \in \mathrm{Ch}(v)} U_w(t)$. Fix any child $u \in \mathrm{Ch}(v)$. If it is endpoint-excluded, then

$$V^*(u) \geq L_u(t) \geq \lambda - \rho_k/2.$$

If the child is not endpoint-excluded, the certificate requires $\mathsf{Comp}_U(u,k,t)$, so the induction hypothesis gives

$$V^*(u) \geq U_u(t) - \rho_k/2 \geq \lambda - \rho_k/2.$$

Taking the minimax backup over these child bounds gives

$$V^*(v) = \min_{u \in \mathrm{Ch}(v)} V^*(u) \geq \lambda - \rho_k/2.$$

Since $I_v(t) \subseteq I_v^{\mathrm{ch}}(t)$ and $\lambda = \min_u U_u(t)$,

$$U_v(t) \leq \min_{u \in \mathrm{Ch}(v)} U_u(t) = \lambda.$$

Combining the last two bounds yields

$$U_v(t) - V^*(v) \le \rho_k/2.$$

This establishes soundness of $\mathsf{Cert}_s(v,k,t)$, including the leaf case, where only the width alternatives apply.

We next check that latching preserves this guarantee. Suppose $\mathsf{Done}_s(v,k,t) = 1$, and let $t_0 \le t$ be the first round at which the flag changed from 0 to 1. If it was set by a bottom-up certificate evaluation, during either a scale search or a post-update sweep, then $\mathsf{Cert}_s(v,k,t_0)$ held by the update rule. Otherwise it was set by the first branch of RESOLVE(v,s,k). In this case, the flag was still zero before the assignment, so the branch condition $\mathsf{Comp}_s(v,k,t_0)$ could only hold through $\mathsf{Cert}_s(v,k,t_0)$. Hence either way, $\mathsf{Cert}_s(v,k,t_0)$ holds.

The certificate argument above applies at the earlier round t_0, so v was (s,k)-safe at round t_0. If $s = L$, Lemma B.2 gives $L_v(t) \ge L_v(t_0)$, and therefore

$$V^*(v) - L_v(t) \le V^*(v) - L_v(t_0) \le \rho_k/2.$$

If $s = U$, then $U_v(t) \le U_v(t_0)$, and thus

$$U_v(t) - V^*(v) \le U_v(t_0) - V^*(v) \le \rho_k/2.$$

Thus a latched side remains safe. Since $\mathsf{Comp}_s(v,k,t)$ is the disjunction of $\mathsf{Cert}_s(v,k,t)$ and $\mathsf{Done}_s(v,k,t)$, the lemma follows. □

We next verify that an unresolved parent always leaves a child obligation at a scale no finer than its own. This ensures that the recursive selection rule is defined.

Lemma B.4 (Capped recursive progress)**.** *Fix a round t, a finite scale $k \ge 0$, an expanded internal node v, and a side $s \in \{L,U\}$. If $\neg\mathsf{Comp}_s(v,k,t)$, then the current scale-k recursive rule has at least one finite capped child obligation whenever recursive work is attempted, i.e.,*

- *if (v,s) is a selector case, then there exists $u \in A_s(v,k,t)$ such that*

$$\neg\mathsf{Comp}_s(u,k,t) \qquad \textit{and} \qquad K_s^{\le k}(u,t) < +\infty;$$

- *if (v,s) is a comparison case, then $A_s(v,k,t) \ne \varnothing$; for $b = b_s(v,k,t)$,*

$$K_{\bar{s}}^{\le k}(b,t) < +\infty \qquad \textit{or} \qquad K_s^{\le k}(b,t) < +\infty.$$

Thus, when the budget permits, both the comparison and selector recursive branches have an admissible capped child whenever the parent side predicate $\mathsf{Comp}_s(v,k,t)$ is false.

Proof. We show that failure of the parent certificate leaves an unresolved child within the permitted range of scales. Since Comp_s includes Cert_s, $\neg\mathsf{Comp}_s(v,k,t)$ implies $\neg\mathsf{Cert}_s(v,k,t)$.

Suppose first that (v,s) is a selector case. If every child $u \in A_s(v,k,t)$ satisfied $\mathsf{Comp}_s(u,k,t)$, then the recursive selector alternative in the definition of $\mathsf{Cert}_s(v,k,t)$ would hold, contradicting $\neg\mathsf{Cert}_s(v,k,t)$. Thus there is a child $u \in A_s(v,k,t)$ with $\neg\mathsf{Comp}_s(u,k,t)$. Since the capped set defining $K_s^{\le k}(u,t)$ contains the index k, this implies

$$K_s^{\le k}(u,t) < +\infty.$$

Now suppose (v,s) is a comparison case. We first check that the active set is nonempty. If $A_s(v,k,t) = \varnothing$, then every child is discharged at scale k: in the MAX, L, case every child u satisfies

$$U_u(t) \le \lambda_L(v,t) + \rho_k/2 \qquad \text{or} \qquad \mathsf{Comp}_L(u,k,t),$$

and in the MIN, U, case every child u satisfies

$$L_u(t) \ge \lambda_U(v,t) - \rho_k/2 \qquad \text{or} \qquad \mathsf{Comp}_U(u,k,t).$$

These are the recursive discharge conditions for $\mathsf{Cert}_s(v,k,t)$, contradicting $\neg\mathsf{Cert}_s(v,k,t)$. Hence $A_s(v,k,t) \ne \varnothing$.

Let $b = b_s(v,k,t)$ be the selected blocker. Its membership $b \in A_s(v,k,t)$ gives

$$\neg \mathsf{Comp}_s(b,k,t).$$

Thus the set defining $K_s^{\le k}(b,t)$ contains k, and consequently

$$K_s^{\le k}(b,t) < +\infty.$$

If $K_s^{\le k}(b,t) < +\infty$, the algorithm refines that exclusion side. If $K_s^{\le k}(b,t) = +\infty$, the finite same-side obligation is available instead. In either case, the comparison rule has an admissible child while $\mathsf{Comp}_s(v,k,t)$ is false. □

The latching rule also prevents an obligation from incurring work after it has been certified.

Lemma B.5 (No same-scale rework)**.** *Fix a finite scale k, an explored non-root node v, and a side $s \in \{L,U\}$. If* RESOLVE(v,s,k) *begins at round t with $\mathsf{Comp}_s(v,k,t)$, then it sets $\mathsf{Done}_s(v,k) \leftarrow 1$ and returns $(0,\mathsf{cert})$ before issuing any oracle query, expanding any node, making any child call, or increasing any work counter. Consequently, once $\mathsf{Done}_s(v,k) = 1$, the node remains (s,k)-safe on $\mathcal{E}_\delta$, and every later invocation of* RESOLVE(v,s,k) *returns zero before doing positive work.*

Proof. After initializing any unused scale-local storage, RESOLVE(v,s,k) first tests $\mathsf{Comp}_s(v,k,t)$. If this predicate holds, the procedure latches $\mathsf{Done}_s(v,k)$ and immediately returns $(0,\mathsf{cert})$. The procedure reaches every oracle query, expansion, child call, and counter increment only after this guard fails.

Each flag $\mathsf{Done}_s(v,k)$ is initialized to 0 and can change only from 0 to 1. Once latched, $\mathsf{Comp}_s(v,k,t')$ holds at every later round t' by definition. Lemma B.3 then gives (s,k)-safety on $\mathcal{E}_\delta$ at every such t'. The initial guard at round t' therefore returns before any positive work, regardless of subsequent changes to the comparison endpoint, blocker, or active set. □

We next convert the first unresolved scale into an endpoint-error bound. The capped form applies to recursive child selection.

Lemma B.6 (Endpoint control at capped scales)**.** *Assume $\mathcal{E}_\delta$. Fix a finite parent cap κ, a round t, an explored non-root node x, and a side $s \in \{L,U\}$. If*

$$j = K_s^{\le \kappa}(x,t) < +\infty,$$

then

$$\begin{cases} V^*(x) - L_x(t) \le \rho_j, & s = L, \\ U_x(t) - V^*(x) \le \rho_j, & s = U. \end{cases}$$

If $K_s^{\le \kappa}(x,t) = +\infty$, then $\mathsf{Comp}_s(x,\kappa,t)$ holds, and therefore

$$\begin{cases} V^*(x) - L_x(t) \le \rho_\kappa/2, & s = L, \\ U_x(t) - V^*(x) \le \rho_\kappa/2, & s = U. \end{cases}$$

Proof. If $K_s^{\le \kappa}(x,t) = +\infty$, the capped set contains no unresolved scale: every index $0 \le i \le \kappa$ is complete on side s, in particular $i = \kappa$. Lemma B.3, applied at the cap, proves the bound.

Now let $j = K_s^{\le \kappa}(x,t) < +\infty$. If $U_x(t) = L_x(t)$, interval validity gives zero error. Otherwise, let $\ell = \underline{k}(x,t)$. We first show $j \ge \ell$, which places the selected scale within the range controlled by the current width. If $j < \ell$, then the dyadic definition of ℓ gives

$$U_x(t) - L_x(t) \le \rho_\ell \le \rho_j/2,$$

using $\rho_{i+1} = \rho_i/2$. The width alternative in $\mathsf{Cert}_s(x,j,t)$ would hold, contradicting the defining property $\neg\mathsf{Comp}_s(x,j,t)$ of j.

If $j = \ell$, the width itself controls either endpoint error, since

$$U_x(t) - L_x(t) \le \rho_\ell = \rho_j.$$

If $j > \ell$, then $j - 1 \geq 0$. Also $j \leq \kappa$, because j belongs to the capped set, so $j - 1 \leq \kappa$. The minimality of j therefore gives $\mathsf{Comp}_s(x, j-1, t)$. Lemma B.3 bounds the endpoint error by

$$\rho_{j-1}/2 = \rho_j.$$

These two cases prove the finite-j claim for either side. □

We can now bound the value gap across a selected edge. The argument uses only the current selection rule, so it also covers a child call blocked by its inherited budget.

Lemma B.7 (One-step localization under capped descent)**.** *Assume $\mathcal{E}_\delta$. Fix a round t, an expanded internal node v, a side s, and a finite scale k such that $\neg\mathsf{Comp}_s(v, k, t)$. Let (u, s_u, k_u) be the output of* ChildObligation(v, s, k) *evaluated at this state. Then $k_u < +\infty$, $k_u \leq k$, and*

$$|V^*(u) - V^*(v)| \leq \rho_{k_u}.$$

Proof. We relate the child's value gap to the endpoint that caused its selection. Lemma B.4 first ensures that the selected obligation is finite. The rule uses the capped scale $k_u = K_{s_u}^{\leq k}(u, t)$, whether or not the available budget permits an actual call. Thus $k_u \leq k$; the dyadic relation $\rho_{i+1} = \rho_i/2$ then gives $\rho_k \leq \rho_{k_u}$. Write $\bar{\rho} = \rho_k$ and $\rho = \rho_{k_u}$.

Selector cases. If v is a Min node and $s = L$, then $u \in \arg\min_x L_x(t)$. Let w be a child attaining $V^*(v) = \min_x V^*(x)$. The selected lower endpoint is accurate by Lemma B.6:

$$V^*(u) - L_u(t) \leq \rho.$$

The selector ordering gives $L_u(t) \leq L_w(t) \leq V^*(w) = V^*(v)$. Together with $V^*(u) \geq V^*(v)$, this places the value gap inside the endpoint error:

$$|V^*(u) - V^*(v)| = V^*(u) - V^*(v) \leq V^*(u) - L_u(t) \leq \rho.$$

For the Max, $s = U$, selector case, the same argument uses the upper endpoints. Now $u \in \arg\max_x U_x(t)$, so a child w attaining $V^*(v) = \max_x V^*(x)$ satisfies

$$U_u(t) \geq U_w(t) \geq V^*(w) = V^*(v).$$

Lemma B.6 gives $U_u(t) - V^*(u) \leq \rho$, and since $V^*(u) \leq V^*(v)$,

$$|V^*(u) - V^*(v)| = V^*(v) - V^*(u) \leq U_u(t) - V^*(u) \leq \rho.$$

Comparison at a Max *node.* Suppose v is Max and $s = L$. Since $\mathsf{Comp}_L(v, k, t)$ is false, Lemma B.4 gives $A_L(v, k, t) \neq \varnothing$. Let $\lambda = \lambda_L(v, t)$ and $b = b_L(v, k, t)$, so the selected child is $u = b$. We first establish $V^*(v) \leq U_b(t)$ by bounding every child value. For a child x with $x \in A_L(v, k, t)$, the definition of b gives

$$V^*(x) \leq U_x(t) \leq U_b(t).$$

If $x \notin A_L(v, k, t)$, then either

$$U_x(t) \leq \lambda + \bar{\rho}/2,$$

or $\mathsf{Comp}_L(x, k, t)$. In the latter case, Lemma B.3 gives

$$V^*(x) \leq L_x(t) + \bar{\rho}/2 \leq \lambda + \bar{\rho}/2.$$

Membership $b \in A_L(v, k, t)$ gives $U_b(t) > \lambda + \bar{\rho}/2$. The bounds for active and inactive children are therefore both at most $U_b(t)$, proving $V^*(v) \leq U_b(t)$. Since v is Max, we also have $V^*(b) \leq V^*(v)$. It remains to control the blocker's upper-endpoint error.

When the selected side is U at b, Lemma B.6 gives

$$U_b(t) - V^*(b) \leq \rho,$$

and the value ordering just proved yields

$$|V^*(b) - V^*(v)| = V^*(v) - V^*(b) \leq U_b(t) - V^*(b) \leq \rho.$$

If the rule instead selects side L at b, the opposite side has no unresolved capped scale:

$$K_U^{\leq k}(b, t) = +\infty.$$

Lemma B.6 therefore gives

$$U_b(t) - V^*(b) \le \bar{\rho}/2 \le \rho,$$

which, substituted into the preceding gap inequality, again gives $|V^*(b) - V^*(v)| \le \rho$.

Comparison at a MIN *node.* For v MIN and $s = U$, we bound every child value from below by the blocker's lower endpoint. Lemma B.4 gives $A_U(v, k, t) \neq \varnothing$. Let $\lambda = \lambda_U(v, t)$ and $b = b_U(v, k, t)$. An active child $x \in A_U(v, k, t)$ satisfies, by the definition of b,

$$V^*(x) \ge L_x(t) \ge L_b(t).$$

For $x \notin A_U(v, k, t)$, either

$$L_x(t) \ge \lambda - \bar{\rho}/2,$$

or $\mathsf{Comp}_U(x, k, t)$, in which case

$$V^*(x) \ge U_x(t) - \bar{\rho}/2 \ge \lambda - \bar{\rho}/2$$

by Lemma B.3. Since $b \in A_U(v, k, t)$, we have

$$L_b(t) < \lambda - \bar{\rho}/2.$$

The active and inactive child bounds are therefore both at least $L_b(t)$, which proves $V^*(v) \ge L_b(t)$. Since v is MIN, we also have $V^*(b) \ge V^*(v)$. We now control the blocker's lower-endpoint error.

When the selected side is L at b, Lemma B.6 gives

$$V^*(b) - L_b(t) \le \rho,$$

and the value ordering above yields

$$|V^*(b) - V^*(v)| = V^*(b) - V^*(v) \le V^*(b) - L_b(t) \le \rho.$$

If the rule instead selects side U at b, the opposite side is complete throughout the capped range:

$$K_L^{\le k}(b, t) = +\infty,$$

and Lemma B.6 gives

$$V^*(b) - L_b(t) \le \bar{\rho}/2 \le \rho.$$

Substitution into the gap inequality gives $|V^*(b) - V^*(v)| \le \rho$, completing the comparison case. □

At the root, scale selection has no parent cap. The same endpoint argument applies to the uncapped active scale.

Lemma B.8 (Endpoint control at node-local active scales)**.** *Assume $\mathcal{E}_\delta$. Fix a round t, an explored non-root node x, and a side $s \in \{L, U\}$. If*

$$k = K_s(x, t) < +\infty,$$

then

$$\begin{cases} V^*(x) - L_x(t) \le \rho_k, & s = L, \\ U_x(t) - V^*(x) \le \rho_k, & s = U. \end{cases}$$

If $K_s(x, t) = +\infty$, then the corresponding one-sided error is zero.

Proof. If $K_s(x, t) = +\infty$ because $U_x(t) = L_x(t)$, both errors are zero by interval validity. We may therefore assume positive width and write $\ell = \underline{k}(x, t)$.

For $k = K_s(x, t) < +\infty$, the definition gives $k \ge \ell$. If $k = \ell$, the width bound defining ℓ is

$$U_x(t) - L_x(t) \le \rho_\ell = \rho_k,$$

which controls either endpoint error by interval validity. If $k > \ell$, the minimality of k in the definition of $K_s(x, t)$ gives $\mathsf{Comp}_s(x, k-1, t)$. Certificate soundness at this preceding scale bounds the required error by

$$\rho_{k-1}/2 = \rho_k.$$

Finally, suppose $K_s(x, t) = +\infty$ while the width is positive. Then $\mathsf{Comp}_s(x, i, t)$ holds at every $i \ge \ell$. Lemma B.3 bounds the same endpoint error by $\rho_i/2$ for all $i \ge \ell$. Letting $i \to \infty$ and using $\rho_i \downarrow 0$ proves that this error is zero. □

The root rule compares the leader with a full contender. Its choice of the coarser unresolved scale controls the endpoint errors needed to bound the selected action's effective gap.

Lemma B.9 (Root active-call localization)**.** *Assume $\mathcal{E}_\delta$ and $\varepsilon = 0$. Consider an outer-loop round t, and let $x \in \mathrm{Ch}(r)$ be the root child selected by Algorithm 1, with selected side s and finite active scale $k < +\infty$. Then*

$$\Delta_x^{\mathrm{eff}} \le 2\rho_k.$$

Proof. We combine the endpoint bounds at the selected scale with the overlap that prevents root stopping. Let

$$a \in \operatorname*{argmax}_{y \in \mathrm{Ch}(r)} L_y(t), \qquad b \in \operatorname*{argmax}_{y \in \mathrm{Ch}(r)\setminus\{a\}} U_y(t)$$

be the current leader and challenger. The failed stopping test and $\varepsilon = 0$ give

$$L_a(t) < U_b(t).$$

For any root child y, Definition 2.4 gives

$$\Delta_y^{\mathrm{eff}} = \max\{\Delta_*,\, V^*(r,a^*) - V^*(r,y)\}.$$

Thus $\Delta_{a^*}^{\mathrm{eff}} = \Delta_*$, while for every nonoptimal root child $y \neq a^*$,

$$\Delta_y^{\mathrm{eff}} = V^*(r,a^*) - V^*(r,y),$$

because $V^*(r,a^*) - V^*(r,y) \ge \Delta_*$.

The leader is selected. If the selected child is a, then $k = K_L(a,t)$, and Lemma B.8 gives

$$V^*(r,a) - L_a(t) \le \rho_k.$$

The root selection rule also controls both endpoints of the challenger. Its full-contender scale k_b is either $+\infty$, in which case both $K_L(b,t)$ and $K_U(b,t)$ are $+\infty$, or it is finite with $\rho_{k_b} \le \rho_k$. In the latter case, the contender-side definition gives, for each $s' \in \{L,U\}$, either $K_{s'}(b,t) = +\infty$ or $\rho_{K_{s'}(b,t)} \le \rho_{k_b} \le \rho_k$. Lemma B.8 therefore yields

$$V^*(r,b) - L_b(t) \le \rho_k, \qquad U_b(t) - V^*(r,b) \le \rho_k.$$

If $a = a^*$, then $\Delta_a^{\mathrm{eff}} = \Delta_*$. Since b is a competitor, the failed stopping test and the endpoint bounds give

$$\Delta_* \le V^*(r,a) - V^*(r,b) < \big(V^*(r,a) - L_a(t)\big) + \big(U_b(t) - V^*(r,b)\big) \le 2\rho_k.$$

If $a \neq a^*$, then $a^* \neq a$, so $U_b(t) \ge U_{a^*}(t) \ge V^*(r,a^*)$. The leader ordering gives $L_a(t) \ge L_b(t)$, while the challenger bounds give $L_b(t) \ge V^*(r,b) - \rho_k$ and $U_b(t) \le V^*(r,b) + \rho_k$. Combining these inequalities yields

$$\Delta_a^{\mathrm{eff}} = V^*(r,a^*) - V^*(r,a) \le U_b(t) - L_a(t) \le U_b(t) - L_b(t) \le 2\rho_k.$$

The challenger is selected. Now suppose the selected child is b, with finite full-contender scale k. For each $s' \in \{L,U\}$, either $K_{s'}(b,t) = +\infty$ or $\rho_{K_{s'}(b,t)} \le \rho_k$, by the contender-side definition. Lemma B.8 therefore bounds both endpoint errors of b by ρ_k:

$$V^*(r,b) - L_b(t) \le \rho_k, \qquad U_b(t) - V^*(r,b) \le \rho_k.$$

For the leader a, either $K_L(a,t) = +\infty$, in which case the lower error is zero, or the algorithm's choice of the challenger implies $\rho_{K_L(a,t)} \le \rho_k$. Hence

$$V^*(r,a) - L_a(t) \le \rho_k.$$

If $b = a^*$, then a is a competitor. The leader ordering bounds their value gap by the challenger's lower-endpoint error:

$$\Delta_b^{\mathrm{eff}} = \Delta_* \le V^*(r,b) - V^*(r,a) \le V^*(r,b) - L_b(t) \le \rho_k,$$

where we used $L_a(t) \ge L_b(t)$ and $L_a(t) \le V^*(r,a)$. If $b \neq a^*$ and $a = a^*$, then the failed stop test gives

$$\Delta_b^{\mathrm{eff}} = V^*(r,a) - V^*(r,b) < \big(V^*(r,a) - L_a(t)\big) + \big(U_b(t) - V^*(r,b)\big) \le 2\rho_k.$$

In the remaining case, $b \neq a^*$ and $a \neq a^*$. Since $a^* \neq a$, the challenger choice b gives

$$V^*(r, a^*) \leq U_{a^*}(t) \leq U_b(t) \leq V^*(r, b) + \rho_k.$$

The gap is therefore bounded by the challenger's upper-endpoint error:

$$\Delta_b^{\text{eff}} = V^*(r, a^*) - V^*(r, b) \leq \rho_k.$$

This covers every choice of the selected root child. □

The root bound and the one-step bound combine without a loss at each depth, because capped descent never decreases the target tolerance.

Proposition B.10 (Active-call localization with explicit depth bound)**.** *Assume $\mathcal{E}_\delta$ and $\varepsilon = 0$. Consider any invocation* RESOLVE(v, s, k) *with finite scale k made by Algorithm 1 during an outer-loop iteration. Let m be the number of recursive edges from the initially selected root child in that outer-loop iteration to v. Then $m \leq D$ and*

$$\Delta_v^{\text{eff}} \leq C_{\text{act}}(D)\rho_k, \qquad C_{\text{act}}(D) = 2.$$

Equivalently, every active finite-scale call satisfies

$$\Delta_v^{\text{eff}} \leq 2\rho_k.$$

Proof. We propagate the root bound along the call chain. Fix one outer-loop iteration and let

$$x_0, x_1, \ldots, x_m = v$$

be the chain from the selected root child to the current invocation, with sides and finite scales

$$(s_0, k_0), (s_1, k_1), \ldots, (s_m, k_m) = (s, k).$$

Each x_i, for $i \geq 1$, is a child of x_{i-1}, so the tree depth bounds the chain length: $m \leq D$.

Let t_0 be the outer-loop round at which x_0 was selected. For $i \geq 1$, let t_i be the round, within the same nested execution, at which the call RESOLVE$(x_{i-1}, s_{i-1}, k_{i-1})$ calls RESOLVE(x_i, s_i, k_i). Each localization lemma is applied to the intervals and active set at its own call time. The true values and effective gaps remain fixed along the chain.

At round t_0, Lemma B.9 gives

$$\Delta_{x_0}^{\text{eff}} \leq 2\rho_{k_0}.$$

For each $i \geq 1$, apply Lemma B.7 at round t_i to the call from x_{i-1} to x_i. This gives

$$k_i \leq k_{i-1} \qquad \text{and} \qquad |V^*(x_i) - V^*(x_{i-1})| \leq \rho_{k_i}.$$

The cap is essential here: by $\rho_{j+1} = \rho_j/2$, the child tolerance is at least the parent tolerance,

$$\rho_{k_{i-1}} \leq \rho_{k_i}.$$

These two bounds let us carry the same constant through the maximum in the effective-gap recursion. Specifically, we induct on i to show

$$\Delta_{x_i}^{\text{eff}} \leq 2\rho_{k_i}.$$

The case $i = 0$ is the root bound above. For the induction step, Definition 2.4 gives

$$\Delta_{x_i}^{\text{eff}} = \max\left\{\Delta_{x_{i-1}}^{\text{eff}}, |V^*(x_i) - V^*(x_{i-1})|\right\}.$$

The induction hypothesis controls the inherited gap, and the one-step bound controls the new edge. Hence

$$\Delta_{x_i}^{\text{eff}} \leq \max\left\{2\rho_{k_{i-1}}, \rho_{k_i}\right\} \leq 2\rho_{k_i}.$$

Taking $i = m$ proves the claim for RESOLVE(v, s, k). Thus $C_{\text{act}}(D) = 2$: the constant is independent of D, and hence polynomial in D. □

Lemma B.11 (Local scale completion)**.** *Fix a round t, an explored non-root node v, and a scale $k \geq 0$. Assume that the local counter $M_{v,k}^{\text{loc}}(t)$ is defined. On $\mathcal{E}_\delta$, if*

$$M_{v,k}^{\text{loc}}(t) \geq \Gamma_v^{(k)}(\delta_v),$$

then

$$\text{width}\big(I_v^{\text{loc}}(t)\big) \leq \rho_k/2.$$

Consequently, $\mathsf{Cert}_L(v, k, t)$ *and* $\mathsf{Cert}_U(v, k, t)$ *both hold.*

Proof. We bound the local interval using the oracle that determines the local cost. If $B(h(v)) \le \rho_k/4$, then Eq. (7) gives $\Gamma_v^{(k)}(\delta_v) = 0$. In this case, the fast interval is already sufficient: since

$$I_v^{\mathrm{loc}}(t) = I_v^F \cap I_v^S(t) \subseteq I_v^F,$$

Eq. (2) gives

$$\mathrm{width}\big(I_v^{\mathrm{loc}}(t)\big) \le 2B(h(v)) \le \rho_k/2.$$

Suppose now that $B(h(v)) > \rho_k/4$. Let $n_{v,k}(t)$ count the slow samples issued at v while resolving scale k, through round t. The definition of the local counter gives

$$M_{v,k}^{\mathrm{loc}}(t) = c\, n_{v,k}(t).$$

The assumed spending bound and Eq. (7) therefore imply

$$n_{v,k}(t) \ge m_v(\rho_k, \delta_v).$$

The total sample count at v includes these samples, so

$$N_v^S(t) \ge n_{v,k}(t) \ge m_v(\rho_k, \delta_v).$$

Since $n \mapsto \beta_v(n, \delta_v)$ is nonincreasing,

$$\beta_v\big(N_v^S(t), \delta_v\big) \le \beta_v\big(m_v(\rho_k, \delta_v), \delta_v\big) \le \rho_k/4.$$

The running intersection in Eq. (4) includes this current confidence interval. Hence $I_v^S(t)$ satisfies

$$\mathrm{width}\big(I_v^S(t)\big) \le \rho_k/2.$$

The inclusion $I_v^{\mathrm{loc}}(t) \subseteq I_v^S(t)$ proves the required local-width bound. In either case, this is the local-width alternative in $\mathsf{Cert}_s(v, k, t)$, for both $s = L$ and $s = U$. □

Lemma B.12 (Budgeted recursive spending)**.** *Fix any execution of Algorithm 1. For every invocation* RESOLVE(v, s, k, B)*, if the invocation returns* (q, z)*, then*

$$0 \le q \le B$$

with the convention that the inequality is void when $B = +\infty$*. Moreover, if* $z = \mathsf{blocked}$*, then* $q = 0$*. Finally, for every explored non-root node* v*, scale* k*, and round* t*,*

$$M_{v,k}^{\mathrm{exp}}(t) \le \mathsf{B}_{v,k}^{\mathrm{rec}} = \alpha_{h(v)}\Gamma_v^{(k)}(\delta_v).$$

Here an undefined scale-local counter is interpreted as zero.

Proof. We first show that every return respects the input cap and that a blocked call incurs no cost, by induction on the remaining depth $h(v)$. The certificate branch returns $(0, \mathsf{cert})$. In the local branch, LOCALSTEP(v, k, B) returns $(0, \mathsf{blocked})$ if $B < c$; otherwise it takes one slow sample and returns $(c, \mathsf{progress})$, with $B \ge c$. Thus both properties hold for the local branch: its cost is at most B, and it spends nothing when blocked. This also proves the leaf case.

For an internal node, let

$$R = \mathsf{B}_{v,k}^{\mathrm{rec}} - M_{v,k}^{\mathrm{exp}}.$$

If $R \le 0$, the call uses the local branch just considered. Otherwise, its child-work cap is $B_{\mathrm{rec}} = \min\{B, R\}$. For an unexpanded v, the comparison with the expansion cost determines what happens. If $|\mathrm{Ch}(v)| > R$, expansion exceeds the node's own budget, so the call uses the local branch. If $|\mathrm{Ch}(v)| \le R$ but $|\mathrm{Ch}(v)| > B$, expansion fits that budget but exceeds the inherited cap, and the call returns $(0, \mathsf{blocked})$. In the remaining case,

$$|\mathrm{Ch}(v)| \le B_{\mathrm{rec}} \le B,$$

so expansion respects the cap.

If v is already expanded, there is at most one child call, with cap B_{rec}. The induction hypothesis bounds its returned cost q by $B_{\mathrm{rec}} \le B$, and makes that cost zero if the child reports blocked. When this happens and the inherited cap is binding $(B < R)$, the parent also returns $(0, \mathsf{blocked})$. Otherwise it uses the local branch, which respects the cap and costs zero exactly when blocked. If the

child is not blocked, the parent simply returns the child's result. This completes the induction on the return properties.

It remains to bound $M^{\exp}_{v,k}$. This counter starts at zero, with undefined counters also read as zero, and has only two possible updates. When v is expanded at scale k, the algorithm first checks

$$|\mathrm{Ch}(v)| \le B_{\mathrm{rec}} \le R = \mathsf{B}^{\mathrm{rec}}_{v,k} - M^{\exp}_{v,k},$$

so adding the expansion cost leaves $M^{\exp}_{v,k}$ at most $\mathsf{B}^{\mathrm{rec}}_{v,k}$. When a child call returns (q, z) with $z \neq \mathsf{blocked}$, the parent adds to $M^{\exp}_{v,k}$ the cost q. The return property just proved gives

$$q \le B_{\mathrm{rec}} \le R = \mathsf{B}^{\mathrm{rec}}_{v,k} - M^{\exp}_{v,k},$$

so this update also leaves the counter at most $\mathsf{B}^{\mathrm{rec}}_{v,k}$. Since no other operation increases $M^{\exp}_{v,k}$, the bound holds throughout the execution. □

Lemma B.13 (Aggregate obligation accounting)**.** *Assume $\mathcal{E}_\delta$. Fix an explored non-root node v and scale k. Undefined scale-local counters are interpreted as zero. For every round t,*

$$M^{\mathrm{loc}}_{v,k}(t) \le \Gamma^{(k)}_v(\delta_v), \qquad M^{\exp}_{v,k}(t) \le \alpha_{h(v)}\Gamma^{(k)}_v(\delta_v).$$

Consequently the total work ever charged to the scale-k obligation at v, aggregated over both sides, is bounded by

$$M^{\mathrm{loc}}_{v,k}(t) + M^{\exp}_{v,k}(t) \le \big(1 + \alpha_{h(v)}\big)\Gamma^{(k)}_v(\delta_v).$$

Moreover, once $M^{\mathrm{loc}}_{v,k}(t)$ reaches $\Gamma^{(k)}_v(\delta_v)$, both side predicates $\mathsf{Comp}_L(v,k,\cdot)$ and $\mathsf{Comp}_U(v,k,\cdot)$ remain true thereafter, and every later invocation of RESOLVE$(v,s,k,\cdot)$ *for either side returns before doing positive work.*

Proof. Lemma B.12 gives the recursive bound. For the local bound, we show that completion prevents the counter from passing its prescribed limit. The counter starts at zero, and only LOCALSTEP(v,k,B) increases it, by exactly c each time. If $\Gamma^{(k)}_v(\delta_v) = 0$, Lemma B.11 gives both side certificates before any slow query at v at scale k. Thus RESOLVE$(v,s,k,\cdot)$ returns at its first branch and $M^{\mathrm{loc}}_{v,k}$ stays zero.

If $\Gamma^{(k)}_v(\delta_v) > 0$, the identity $\Gamma^{(k)}_v(\delta_v) = \Gamma_v(\rho_k, \delta_v)$ shows that the limit is $c\, m_v(\rho_k, \delta_v)$. Thus $M^{\mathrm{loc}}_{v,k}$ cannot skip the limit: its values lie in $\{0, c, 2c, \ldots, \Gamma^{(k)}_v(\delta_v)\}$ until completion. Upon first reaching $\Gamma^{(k)}_v(\delta_v)$, Lemma B.11 gives

$$\mathsf{Cert}_L(v,k,\cdot) \qquad \text{and} \qquad \mathsf{Cert}_U(v,k,\cdot).$$

The ensuing REFRESHANDLATCH(k) in Algorithm 3 sets both Done flags. Therefore $\mathsf{Comp}_L(v,k,\cdot)$ and $\mathsf{Comp}_U(v,k,\cdot)$ remain true. By Lemma B.5, every subsequent call at this node and scale returns without positive work, on either side. The local counter therefore stays at most $\Gamma^{(k)}_v(\delta_v)$.

Adding the two counter bounds proves the aggregate estimate. Persistence follows from the same latching argument and Lemma B.5. □

Lemma B.14 (Aggregate local summation)**.** *Given* Assumption 3.3*, for a non-root node v, let*

$$\mathcal{K}^{\mathrm{act}}_v := \{k \ge 0 : \ \Delta^{\mathrm{eff}}_v \le 2\rho_k\}.$$

Then, we have

$$\sum_{k \in \mathcal{K}^{\mathrm{act}}_v} \Gamma^{(k)}_v(\delta_v) \le \Lambda_{\mathrm{loc}}\, \Gamma_v(\Delta^{\mathrm{eff}}_v, \delta_v).$$

Moreover, for every $x \ge 0$ and every finite prefix of scales considered in an execution,

$$\sum_{\substack{k \ge 0: \\ \Gamma^{(k)}_v(\delta_v) \le x}} \Gamma^{(k)}_v(\delta_v) \le \Lambda_{\mathrm{pre}}\, x,$$

where the sum is restricted to that finite prefix.

Proof. For each bound, we identify a prefix of scales and apply prefix regularity. The sequence $k \mapsto \Gamma_v^{(k)}(\delta_v)$ is nondecreasing: as ρ_k decreases, the sample requirement cannot decrease, since $m_v(\rho, \delta_v)$ is nonincreasing in the target radius ρ.

The first claim is immediate if $\mathcal{K}_v^{\mathrm{act}} = \varnothing$. Otherwise, $\mathcal{K}_v^{\mathrm{act}}$ is the prefix $\{0, \dots, K_v\}$. Its last index K_v satisfies

$$\rho_{K_v} \geq \frac{\Delta_v^{\mathrm{eff}}}{2}.$$

This lets us compare the last cost with the cost at the effective gap. Monotonicity of $\Gamma_v(\rho, \delta_v)$ in ρ and Eq. (11) give

$$\Gamma_v^{(K_v)}(\delta_v) = \Gamma_v(\rho_{K_v}, \delta_v) \leq \Gamma_v\left(\frac{\Delta_v^{\mathrm{eff}}}{2}, \delta_v\right) \leq \Lambda_{\mathrm{gap}} \Gamma_v(\Delta_v^{\mathrm{eff}}, \delta_v).$$

The prefix condition (10) at K_v now bounds the sum by this last cost:

$$\sum_{k \in \mathcal{K}_v^{\mathrm{act}}} \Gamma_v^{(k)}(\delta_v) \leq \Lambda_{\mathrm{pre}} \Gamma_v^{(K_v)}(\delta_v) \leq \Lambda_{\mathrm{loc}} \Gamma_v(\Delta_v^{\mathrm{eff}}, \delta_v).$$

For the second claim, fix the finite prefix $\{0, \dots, K_{\max}\}$ of scales in question. If no scale satisfies the cost threshold, the sum is zero. Otherwise, monotonicity makes the admissible set a prefix $\{0, \dots, K_x\}$, whose last cost satisfies $\Gamma_v^{(K_x)}(\delta_v) \leq x$. Applying prefix regularity once more,

$$\sum_{k=0}^{K_x} \Gamma_v^{(k)}(\delta_v) \leq \Lambda_{\mathrm{pre}} \Gamma_v^{(K_x)}(\delta_v) \leq \Lambda_{\mathrm{pre}} x.$$

□

We now compare the algorithm's total spending with the recursive reference cost. This comparison must also account for the cap needed to make further progress: a child may be blocked even though that cap has not yet been spent. The next definition and lemma make this distinction precise.

Definition B.2 (Aggregate subtree call cost)**.** *For a non-root node v, let $\mathfrak{C}_v(t)$ be the sum of the costs returned by all invocations at v in the first t completed outer-loop rounds. Equivalently, this is the slow cost at v, together with all fast and slow costs strictly below v; it excludes the fast query that first exposed v. Set this quantity to zero for an unexposed node. Write $L_v^{\mathrm{cost}}(t)$ for the direct slow cost at v, and $E_v^{\mathrm{cost}}(t)$ for the remaining, recursive cost. Then*

$$\mathfrak{C}_v(t) = L_v^{\mathrm{cost}}(t) + E_v^{\mathrm{cost}}(t), \qquad E_v^{\mathrm{cost}}(t) = \mathbf{1}\{v \text{ expanded by } t\}|\mathrm{Ch}(v)| + \sum_{u \in \mathrm{Ch}(v)} \mathfrak{C}_u(t).$$

The second term is zero at a leaf. Costs in a child subtree also appear in its ancestors' costs, but no sum over all nodes is used: the root-child subtrees are disjoint, and their sum counts every post-initialization oracle cost exactly once.

Lemma B.15 (Minimum cap for an unresolved obligation)**.** *Assume $\mathcal{E}_\delta$. At a fixed state of the algorithm, let (v, s, k) be a finite unresolved obligation at an explored node, so that $\neg\mathsf{Comp}_s(v, k, t)$. There is a finite positive number $d_v(s, k; t)$, called its* minimum progress cap*, such that* RESOLVE(v, s, k, B) *returns* $(0, \mathsf{blocked})$ *exactly when $B < d_v(s, k; t)$, and returns positive cost when $B \geq d_v(s, k; t)$. Fix the algorithm's tie-breaking rules when defining this quantity. For an internal node set*

$$R = \alpha_{h(v)} \Gamma_v^{(k)}(\delta_v) - M_{v,k}^{\mathrm{exp}}(t), \qquad b_v = |\mathrm{Ch}(v)|.$$

The cap has the following recursive description:

1. *If v is a leaf, $R \leq 0$, or v is unexpanded with $b_v > R$, then $d_v(s, k; t) = c$.*
2. *If v is unexpanded and $b_v \leq R$, then $d_v(s, k; t) = b_v$.*
3. *Otherwise v is expanded and $R > 0$. Let $(u, s_u, k_u) =$* CHILDOBLIGATION(v, s, k)*, and write $d_u = d_u(s_u, k_u; t)$. Then*

$$d_v(s, k; t) = \begin{cases} d_u, & d_u \leq R, \\ \max\{R, c\}, & d_u > R. \end{cases}$$

In particular, at every internal node,

$$d_v(s,k;t) \le \max\{R,c\}.$$

Moreover, if v is expanded and $\Delta_v^{\mathrm{eff}} \le 2\rho_k$, the child obligation selected by ChildObligation(v,s,k) *satisfies $\Delta_u^{\mathrm{eff}} \le 2\rho_{k_u}$, regardless of the value of R and whether the resolver actually reaches its child-call branch.*

Proof. We prove the cap formula by induction on remaining depth. The local and expansion branches of Algorithm 4 give the first two cases directly. In the third case, Lemma B.4 supplies a finite unresolved child obligation. Certificate evaluations do not affect this argument: at fixed intervals, a bottom-up induction shows that latching a true certificate leaves all current completion predicates unchanged. Hence these zero-cost evaluations preserve both the selected obligation and its cap threshold.

If $d_u \le R$, the child receives enough cap exactly when $B \ge d_u$. If $B < d_u$, then $B < R$, so the parent passes the blocked return upward. If $d_u > R$, the child is blocked for every cap $\min\{B,R\}$. The parent can then take a local step exactly when $B \ge R$ and $B \ge c$. These are the two alternatives in the claimed formula. They also show that the threshold is finite and bounded by $\max\{R,c\}$.

The child-selection rule and Lemma B.7 give $k_u \le k$ and $|V^*(u) - V^*(v)| \le \rho_{k_u}$. These bounds depend only on the current state and the child-selection rule; they do not require the child call to receive enough cap to perform work. Thus

$$\Delta_u^{\mathrm{eff}} = \max\{\Delta_v^{\mathrm{eff}}, |V^*(u) - V^*(v)|\} \le \max\{2\rho_k, \rho_{k_u}\} \le 2\rho_{k_u}.$$

□

Proposition B.16 (Aggregate subtree charging)**.** *Assume $\mathcal{E}_\delta$, $\varepsilon = 0$, and Assumption 3.3, and let P_h be defined in Eq. (12). For every finite round T and every non-root node v,*

$$\mathfrak{C}_v(T) \le P_{h(v)} J_v^*(\boldsymbol{\delta}).$$

In fact, for every currently unresolved finite obligation (v,s,k) with $\Delta_v^{\mathrm{eff}} \le 2\rho_k$, the stronger bound holds:

$$\mathfrak{C}_v(T) + d_v(s,k;T) \le P_{h(v)} J_v^*(\boldsymbol{\delta}). \tag{13}$$

Consequently,

$$C_T \le |\mathrm{Ch}(r)| + P_{D-1} \sum_{a \in \mathrm{Ch}(r)} J_a^*(\boldsymbol{\delta}) \le P_D H(\boldsymbol{\delta}).$$

For $\alpha_h = (h+1)^2$,

$$P_D \le \Lambda_{\mathrm{loc}}(1+\alpha_D) \prod_{i=1}^{D} \left(1 + \frac{\Lambda_{\mathrm{pre}}}{\alpha_i}\right),$$

so $P_D = O_{\Lambda_{\mathrm{pre}},\Lambda_{\mathrm{gap}}}(D^2)$.

Proof. We use induction on remaining depth to prove the cost bound together with (13). The stronger claim is needed when a child call is blocked: past spending alone does not control the cap required for its next action. We prove both claims for every finite execution prefix and every current unresolved obligation satisfying the localization condition. Throughout, we consider states before new positive work and interpret undefined counters as zero. A blocked call changes neither intervals nor spending counters.

The local bound. We first bound spending by the cost of local certification. Each positive cost returned at scale j increases exactly one of $M_{v,j}^{\mathrm{loc}}$ and $M_{v,j}^{\mathrm{exp}}$ by that cost. Proposition B.10 restricts every such scale to $\mathcal{K}_v^{\mathrm{act}}$. Therefore Lemma B.13 gives

$$\mathfrak{C}_v(T) \le (1+\alpha_{h(v)}) \sum_{j \in \mathcal{K}_v^{\mathrm{act}}} \Gamma_v^{(j)}(\delta_v). \tag{14}$$

To include the cap for a current unresolved obligation (v,s,k), observe that $k \in \mathcal{K}_v^{\mathrm{act}}$. Local scale completion then gives

$$\Gamma_v^{(k)}(\delta_v) > 0, \qquad M_{v,k}^{\mathrm{loc}}(T) \le \Gamma_v^{(k)}(\delta_v) - c.$$

The subtraction is justified because both quantities are multiples of c. For an internal node, write $R = \alpha_{h(v)}\Gamma_v^{(k)} - M_{v,k}^{\exp} \geq 0$. Lemma B.15 then gives

$$\begin{aligned} M_{v,k}^{\mathrm{loc}} + M_{v,k}^{\exp} + d_v(s,k;T) &\leq M_{v,k}^{\mathrm{loc}} + M_{v,k}^{\exp} + \max\{R, c\} \\ &\leq (1+\alpha_{h(v)})\Gamma_v^{(k)}. \end{aligned}$$

The unused allowance at this scale therefore covers the additional cap: the right-hand side of (14) also bounds $\mathfrak{C}_v(T) + d_v(s,k;T)$. By Lemma B.14, both are at most $(1+\alpha_{h(v)})\Lambda_{\mathrm{loc}}\Gamma_v(\Delta_v^{\mathrm{eff}}, \delta_v)$. At a leaf, there is no recursive spending, so the bound reduces to $\Lambda_{\mathrm{loc}}\Gamma_v(\Delta_v^{\mathrm{eff}}, \delta_v)$. If an unresolved obligation exists, the unused local sample covers its cap c. This proves the base case. It also proves both claims at an internal node whenever the local term attains the minimum defining J_v^*.

The recursive bound. It remains to consider a node v of remaining depth $h \geq 1$ for which the recursive term attains the minimum. Set

$$Q_v := P_{h-1}\left(|\mathrm{Ch}(v)| + \sum_{u \in \mathrm{Ch}(v)} J_u^*(\boldsymbol{\delta})\right) = P_{h-1}J_v^*(\boldsymbol{\delta}).$$

The cost induction hypothesis and $P_{h-1} \geq 1$ imply, at every earlier state $t \leq T$,

$$E_v^{\mathrm{cost}}(t) \leq Q_v. \tag{15}$$

Suppose v is expanded and (v,s,k) is a current unresolved obligation satisfying the localization condition. Let (u, s_u, k_u) be the child obligation given by Lemma B.4. It exists even if the resolver has no recursive budget left, and it satisfies the localization condition by Lemma B.15. Apply the stronger induction hypothesis to this child and the ordinary cost bounds to all other children. This gives

$$E_v^{\mathrm{cost}}(t) + d_u(s_u, k_u; t) \leq Q_v. \tag{16}$$

This is the estimate that controls a blocked child call: it includes the cap needed for progress, even though that cap has not been spent.

We next bound the direct slow cost. We show that every scale k at which a slow sample is issued at v satisfies

$$\alpha_h\Gamma_v^{(k)}(\delta_v) < Q_v. \tag{17}$$

Consider the state just before such a sample. By Proposition B.10, its obligation satisfies the localization condition. If v is unexpanded, local sampling requires $\alpha_h\Gamma_v^{(k)} < |\mathrm{Ch}(v)| \leq Q_v$, since its recursive counter is still zero. If v is expanded and its remaining budget R is zero, then

$$\alpha_h\Gamma_v^{(k)} = M_{v,k}^{\exp} \leq E_v^{\mathrm{cost}} < E_v^{\mathrm{cost}} + d_u \leq Q_v.$$

If the node is expanded and $R > 0$, the resolver can fall back to local sampling only after a blocked child call, and only when its inherited cap is at least R. The child therefore received cap R, so Lemma B.15 gives $d_u > R$. Consequently,

$$\alpha_h\Gamma_v^{(k)} = M_{v,k}^{\exp} + R < M_{v,k}^{\exp} + d_u \leq E_v^{\mathrm{cost}} + d_u \leq Q_v.$$

Thus (17) holds whenever local sampling occurs. The local cost at each such scale is bounded by its local stopping cost. Summing these bounds with Lemma B.14 yields

$$L_v^{\mathrm{cost}}(T) \leq \sum_{\{j:\, \Gamma_v^{(j)} < Q_v/\alpha_h\}} \Gamma_v^{(j)} \leq \frac{\Lambda_{\mathrm{pre}}}{\alpha_h} Q_v. \tag{18}$$

Here the sum is taken over a finite prefix containing all scales used through T. Together with (15), this proves the ordinary cost bound with factor $(1 + \Lambda_{\mathrm{pre}}/\alpha_h)P_{h-1}$.

Including the cap for further progress. We must still include the current cap $d = d_v(s,k;T)$. We show that it fits either the recursive bound or the unused local allowance. If v is unexpanded and the expansion fits its recursive budget, then $d = |\mathrm{Ch}(v)|$, $E_v^{\mathrm{cost}} = 0$, and $E_v^{\mathrm{cost}} + d \leq Q_v$. If v is expanded and its child's cap fits the remaining budget, then $d = d_u$ and (16) gives the same conclusion. Finally, if $d_u > R \geq c$, then $d = R < d_u$, so again $E_v^{\mathrm{cost}} + d \leq Q_v$. In each case, the recursive bound covers the cap, and (18) covers the local spending.

In the remaining cases, $d = c$ and the next positive step is local sampling: either expansion exceeds the unexpanded node's budget, $R = 0$, or $d_u > R$ and $R < c$. The argument for (17) applies to this scale as well, even if no sample has yet been issued there. Moreover, $M^{\mathrm{loc}}_{v,k} \le \Gamma^{(k)}_v - c$, since the current obligation is unresolved. One additional local sample therefore leaves the direct local cost at this scale at most $\Gamma^{(k)}_v$. Applying the threshold sum to the previously sampled scales and the current scale gives

$$L^{\mathrm{cost}}_v(T) + d \le \frac{\Lambda_{\mathrm{pre}}}{\alpha_h} Q_v.$$

Together with $E^{\mathrm{cost}}_v(T) \le Q_v$, this proves in every case

$$\mathfrak{C}_v(T) + d_v(s, k; T) \le \left(1 + \frac{\Lambda_{\mathrm{pre}}}{\alpha_h}\right) Q_v \le P_h J^*_v(\boldsymbol{\delta}).$$

This completes the simultaneous induction.

Summing at the root. The root's only direct cost is the initialization cost $|\mathrm{Ch}(r)|$, and every subsequent cost belongs to exactly one root-child subtree. Since $P_h \ge P_{h-1} \ge 1$ follows directly from Eq. (12), the claimed root bounds follow. Finally, unrolling that recursion and using the nondecreasing sequence α_h gives the stated product bound. For $\alpha_h = (h+1)^2$, its product is at most

$$\exp\left(\Lambda_{\mathrm{pre}} \sum_{i=1}^{\infty} \frac{1}{(i+1)^2}\right),$$

while its remaining factor is $O(D^2)$. □

Lemma B.17 (Positive progress of unresolved selected calls)**.** *Let*

$$q_{\min} := \min\{c, 1\} > 0.$$

Consider any invocation RESOLVE(v, s, k, B) *with finite scale* k *that begins at a round* t *with*

$$\neg \mathsf{Comp}_s(v, k, t).$$

Then the invocation either returns $(0, \mathsf{blocked})$*, or returns a pair* (q, z) *with* $q \ge q_{\min}$*. In particular, if* $B = +\infty$*, then the invocation returns cost at least* $q_{\min}$*.*

Proof. We argue by induction on the remaining depth $h(v)$. At a leaf, $h(v) = 0$, the first guard in RESOLVE(v, s, k, B) is false because the obligation is unresolved. The procedure therefore calls LOCALSTEP(v, k, B), which returns $(0, \mathsf{blocked})$ if $B < c$, and otherwise takes one slow sample at cost $c \ge q_{\min}$.

Suppose $h(v) \ge 1$ and the claim holds for the children of v. The first guard is again false. A nonpositive remaining recursive budget leads directly to LOCALSTEP(v, k, B), whose behavior was just established. If v is unexpanded and $|\mathrm{Ch}(v)| > R$, the same fallback LOCALSTEP(v, k, B) applies. For an unexpanded node in the remaining case, $|\mathrm{Ch}(v)| \le R$: the expansion is affordable from the node's own budget. If $|\mathrm{Ch}(v)| > B$, the inherited cap blocks it and the invocation returns $(0, \mathsf{blocked})$. If this cap also permits the expansion, the procedure expands v at cost $|\mathrm{Ch}(v)| \ge 1 \ge q_{\min}$.

Only the expanded-node case with positive recursive budget requires the induction hypothesis. Lemma B.4 ensures that $\mathsf{Comp}_s(v, k, t)$ being false provides a finite capped child obligation. In a selector case, the chosen child (u, s, k_u) satisfies

$$k_u = K^{\le k}_s(u, t) < +\infty \qquad \text{and} \qquad \neg \mathsf{Comp}_s(u, k_u, t).$$

In a comparison case, the blocker is called on the opposite side $\bar{s}$ when a finite capped obligation remains there, and otherwise on side s. The definition of $K^{\le k}_{\cdot}$ ensures that either call targets an unresolved obligation, so the induction hypothesis applies. A positive child cost is returned unchanged. After a child return of $(0, \mathsf{blocked})$, the parent either also returns $(0, \mathsf{blocked})$ because the inherited cap is binding, or calls LOCALSTEP(v, k, B). That local step is itself blocked or costs at least $q_{\min}$, completing the induction.

When $B = +\infty$, the inherited cap cannot block progress and LOCALSTEP(v, k, B) is always affordable. The invocation therefore returns cost at least $q_{\min}$. □

Lemma B.18 (Non-stopping rounds select a finite unresolved side)**.** *Assume $\mathcal{E}_\delta$ and $\varepsilon = 0$. In every outer-loop round before stopping, Algorithm 1 selects a root child x, side s, and scale $k < +\infty$ such that*

$$\neg\mathsf{Comp}_s(x, k, t).$$

Proof. Let a and b be the current leader and challenger, with respective scales $k_a = K_L(a, t)$ and $k_b = K_\pm(b, t)$. The selection rule prefers a finite scale: if $k_a < +\infty$ and $k_b = +\infty$, it selects the leader at scale k_a; if $k_a = +\infty$ and $k_b < +\infty$, it selects the challenger at scale k_b; and if both scales are finite, it selects the coarser one. Thus an infinite selected scale would require

$$K_L(a, t) = +\infty \qquad \text{and} \qquad K_\pm(b, t) = +\infty.$$

By the definition of $K_\pm$, the second equality means

$$K_L(b, t) = K_U(b, t) = +\infty.$$

These equalities are incompatible with a failed stopping test. Indeed, if $K_L(a, t) = +\infty$, Lemma B.8 gives

$$V^*(r, a) - L_a(t) = 0.$$

Likewise, $K_L(b, t) = K_U(b, t) = +\infty$ gives

$$V^*(r, b) - L_b(t) = 0, \qquad U_b(t) - V^*(r, b) = 0.$$

Failure of the stopping test means

$$L_a(t) < U_b(t).$$

Together, these relations imply $V^*(r, a) < V^*(r, b)$, and hence

$$L_b(t) = V^*(r, b) > V^*(r, a) = L_a(t),$$

contradicting the choice of a as a leader with maximal lower endpoint. The selected scale must therefore be finite. By the definitions of K_L, K_U, and $K_\pm$, every finite selected scale is unresolved, giving the required $\neg\mathsf{Comp}$ condition. □

Proposition B.19 (Finite stopping and correctness)**.** *Assume $\mathcal{E}_\delta$, $\varepsilon = 0$, and Assumption 3.3. Then Algorithm 1 halts after finitely many oracle calls. At its stopping time τ, every returned action $\hat{a}_\tau$ is the unique optimal root action a^*.*

Proof. We combine the uniform bound on finite execution prefixes with a positive cost in every non-stopping round. Consistency of the confidence radii gives $\Gamma_v(\Delta_v^{\mathrm{eff}}, \delta_v) < \infty$ at every non-root node v, since $\Delta_v^{\mathrm{eff}} \geq \Delta_* > 0$. It follows that $J_v^*(\boldsymbol{\delta}) < \infty$ for every non-root v, by induction on $h(v)$, and the tree's finiteness then gives $H(\boldsymbol{\delta}) < \infty$. Proposition B.16 bounds every finite prefix:

$$C_T \leq P_D H(\boldsymbol{\delta}) \qquad \text{for every finite round } T.$$

Suppose the algorithm never stops. In each outer-loop round, Lemma B.18 supplies a finite unresolved call $\textsc{Resolve}(x, s, k, +\infty)$. The finite scale searches and finite recursion depth ensure that each round completes. By Lemma B.17, its cost is at least $q_{\min} > 0$. After N non-stopping rounds, the total cost is therefore at least the initialization cost plus $N q_{\min}$. This diverges as $N \to \infty$, contradicting the uniform bound $P_D H(\boldsymbol{\delta})$. The algorithm must halt after finitely many oracle calls.

At the stopping time τ, the returned action $\hat{a}_\tau$ satisfies

$$L_{\hat{a}_\tau}(\tau) \geq \max_{a \neq \hat{a}_\tau} U_a(\tau).$$

Lemma 2.4 makes every root-child interval valid on $\mathcal{E}_\delta$, so

$$V^*(r, \hat{a}_\tau) \geq L_{\hat{a}_\tau}(\tau) \geq \max_{a \neq \hat{a}_\tau} U_a(\tau) \geq \max_{a \neq \hat{a}_\tau} V^*(r, a).$$

Thus $\hat{a}_\tau$ is optimal, and uniqueness gives $\hat{a}_\tau = a^*$. □

B.6 Proof of Theorem 3.6

We now assemble the bounds established in Appendix B.5.

Proof. When $\rho_0 = 0$, initialization already determines the answer, as shown at the start of Appendix B.5. The cost bound follows from $C_\tau = H(\boldsymbol{\delta})$ and $P_D \geq 1$. Assume $\rho_0 > 0$ below. On $\mathcal{E}_\delta$, finite stopping and correctness follow from Proposition B.19. Now that τ is known to be finite, Proposition B.16 applies with $T = \tau$ and gives

$$C_\tau \leq P_D H(\boldsymbol{\delta}).$$

The $\mathcal{O}(D^2)$ form follows from the final statement of Proposition B.16 for $\alpha_h = (h+1)^2$.

By Eq. (5), each feasible allocation satisfies $\mathbb{P}(\mathcal{E}_\delta) \geq 1 - \delta$. For the optimized statement, choose a near-optimal feasible allocation $\boldsymbol{\delta}^\eta$ satisfying the uniform regularity condition in Theorem 3.6, with

$$H(\boldsymbol{\delta}^\eta) \leq H^* + \eta.$$

Apply the fixed-allocation bound to the run using this allocation, on its corresponding simultaneous-validity event. This gives the claimed high-probability optimized bound. If an allocation attaining the infimum satisfies the same regularity constants, take $\eta = 0$. Otherwise, $H^* \geq |\mathrm{Ch}(r)| \geq 2$ ensures that any $\eta \leq H^*$ gives the stated H^*-order bound. □

B.7 Proofs and Qualifications for Oracle Robustness

The robustness results control the probability of an incorrect finite output. Extending the unbiased finite-stopping or cost theorem would require additional assumptions.

B.7.1 Proof of Theorem 3.7

Write $\theta_v = V^*(v) + b_v$. To account for the bias, replace Eq. (4) by

$$I_v^{S,\xi}(t) = \bigcap_{j=1}^{N_v^S(t)} [\widehat{V}_{v,j} - \beta_v(j, \delta_v) - \xi, \widehat{V}_{v,j} + \beta_v(j, \delta_v) + \xi],$$

where the intersection is $(-\infty, +\infty)$ before any samples are taken. The fast intervals, minimax backups, and root stopping test are unchanged. The assumed concentration around the biased mean is

$$\mathbb{P}\Big(\exists n \geq 1 : |\widehat{V}_{v,n} - \theta_v| > \beta_v(n, \delta_v)\Big) \leq \delta_v.$$

Proof. A union bound gives an event of probability at least $1 - \delta$ on which, simultaneously for all nodes and sample counts, $|\widehat{V}_{v,n} - \theta_v| \leq \beta_v(n, \delta_v)$. On this event,

$$|\widehat{V}_{v,n} - V^*(v)| \leq \beta_v(n, \delta_v) + \xi.$$

Every inflated slow interval therefore contains the true node value, as does every fast interval. Their intersections remain nonempty and valid, and induction through the max and min backups preserves validity without further inflation. The inconsistency exit consequently cannot occur on this event. The root stopping argument from Theorem 3.1 now proves the claim. □

Why interval inflation is required. Without inflation, the endpoint inequalities $L_v \leq V^*(v) + \xi$ and $U_v \geq V^*(v) - \xi$ would bound the error by $\varepsilon + 2\xi$ whenever the root stopping test returns a separated recommendation. But they do not ensure $L_v \leq U_v$: a fast interval containing $V^*(v)$ may be disjoint from a narrow slow interval around $V^*(v) + b_v$. They therefore cannot, by themselves, guarantee correctness of the unmodified algorithm for arbitrary biased oracles. Inflation by the known bias bound preserves valid, nonempty intervals. The inflated radius need not tend to zero, however, so the unbiased finite-stopping and cost proofs do not carry over.

B.7.2 Proof of Theorem 3.8

Proof. The enlarged fast interval contains $V^*(v)$. On $\mathcal{E}_\delta$, every slow interval also contains $V^*(v)$, so local intersections and minimax backups remain valid. The stopping-rule proof applies, and the inconsistency exit cannot occur on $\mathcal{E}_\delta$.

For the fixed-history comparison, intersecting a larger fast interval with the same slow interval gives a larger local interval. Enlarging the child intervals weakly decreases their lower endpoints and weakly increases their upper endpoints; both max and min backups preserve these inequalities. Induction up the common exposed tree, with the enlarged local intervals intersected at each node, proves the claimed containment. Actual adaptive runs may choose different queries and expansions, so this comparison does not order their realized costs. □

B.8 Qualifications for the Reference Complexity and Local Regularity

The correctness theorem bounds the unconditional probability that the algorithm stops and returns a recommendation that violates the requested accuracy.

We also note that the half-gap condition in Assumption 3.3 restricts how the hybrid cost curve may cross the deterministic fast-bias cutoff. If

$$\Delta_v^{\mathrm{eff}}/8 < B(h(v)) \le \Delta_v^{\mathrm{eff}}/4,$$

then $\Gamma_v(\Delta_v^{\mathrm{eff}}, \delta_v) = 0$ but $\Gamma_v(\Delta_v^{\mathrm{eff}}/2, \delta_v) > 0$, so the condition fails. Polynomial scaling of the slow-sample cost alone therefore does not establish regularity of the hybrid curve.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We include all the results as our contributions and paper scope.

   Guidelines:

   - The answer [NA] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [NA] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We include a detailed discussion of limitation and direction for future follow-up works in Appendix A.

   Guidelines:

   - The answer [NA] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We provide clear assumptions for the required theoretical results and provide out both proof sketch in main text along with full proofs in the Appendices.

Guidelines:

- The answer [NA] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We submited full implementation code along with environment generation code to ensure reproducibility.

   Guidelines:

   - The answer [NA] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We submitted the code for implementation and data generation.

Guidelines:

- The answer [NA] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: We specify the details for experiments in both the corresponding Experiment and Appendix section.

Guidelines:

- The answer [NA] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: For each task, we randomly generated multiple environments to report mean and worst performance.

Guidelines:

- The answer [NA] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We listed in the Appendix, although our work only require CPU-level computing.

   Guidelines:

   - The answer [NA] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We are adhering to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [NA] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No] , they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: This is a more theory-orientated work, which doesn't have societal impact.

    Guidelines:

    - The answer [NA] means that there is no societal impact of the work performed.
    - If the authors answer [NA] or [No] , they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We don't have such data for mis-use.

    Guidelines:

    - The answer [NA] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [NA]

    Justification: We didn't use any existing datasets.

    Guidelines:

    - The answer [NA] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We didn't release new assets.

    Guidelines:

    - The answer [NA] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [NA] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [NA] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [NA]

Justification: Our work is not associated with LLM research.

Guidelines:

- The answer [NA] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.