Search-Tree Scaling in Parallel Monte Carlo Tree Search

Scott Cheng[1] **Meng-Yu Tsai**[2] **Ding-Yong Hong**[3] **Mahmut Taylan Kandemir**[1]

[1]The Pennsylvania State University, USA [2]Independent
[3]Institute of Information Science, Academia Sinica, Taiwan

[1]{ypc5394,mtk2}@psu.edu [2]adamatuno@gmail.com [3]dyhong@iis.sinica.edu.tw

Abstract

The exploration-exploitation tradeoff is a fundamental challenge in reinforcement learning. In Monte Carlo Tree Search (MCTS), this tradeoff is balanced explicitly by the pUCT exploration coefficient and implicitly by parallelism when the search is combined with neural network evaluations. However, the exploration coefficient is typically tuned as an algorithmic hyperparameter, while the degree of parallelism is often treated as a system detail. To this end, we characterize how these two factors shape the search-tree structure and jointly affect performance scaling. We prove that, under regularity conditions, the depth of the optimal path grows as $\Theta(\sqrt{n}/\log n)$ for n simulations under the MuZero exploration coefficient formula. For tree-parallel MCTS, we prove that the search depth preserves the same asymptotic order when $T = o(\sqrt{n})$, while sufficiently large parallelism introduces contention and reduces the depth growth rate to $\Theta(n/T)$ for T threads. Our empirical results indicate that, in the Go experiment, each doubling of the thread count reduces the best-performing exploration coefficient by approximately 0.19. Overall, our characterization connects the search-tree structure with the exploration-exploitation behavior in pUCT, providing insight into how exploration coefficients and parallelism can be jointly tuned.

1 Introduction

Reinforcement learning has shown remarkable success in complex decision-making problems such as game playing, planning, and control [1–3]. At the heart of these problems lies the "exploration-exploitation tradeoff" [1, 4]: how to balance limited computation between exploiting actions that currently appear most promising and exploring actions whose value remains uncertain but may turn out to be higher. In Monte Carlo Tree Search (MCTS) [5, 6], the UCT formulation [6, 7] makes this tradeoff explicit through an exploration coefficient that controls the relative strength of an action-value term and an exploration term. Recent successes of MCTS, as demonstrated by AlphaZero [8] and MuZero [3], rely on the pUCT formulation [2, 9], which is a variant of UCT that preserves the exploration coefficient while incorporating a neural network-based policy prior to guide the exploration term.

In pUCT, the exploration-exploitation balance is mainly determined by two factors, as shown in Figure 1. The first is the exploration coefficient c, which directly controls the exploration term in the pUCT selection formula. Larger values of c place more weight on less-visited or high-prior actions, leading to a wider branching at each depth. The second factor is parallelism. In tree-parallel MCTS [10], multiple search threads traverse the same tree concurrently, and mechanisms such as virtual loss [10] steer different threads toward different parts of the tree instead of repeatedly following the same path. This produces wider branching near the root, while still allowing deeper subtrees to

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

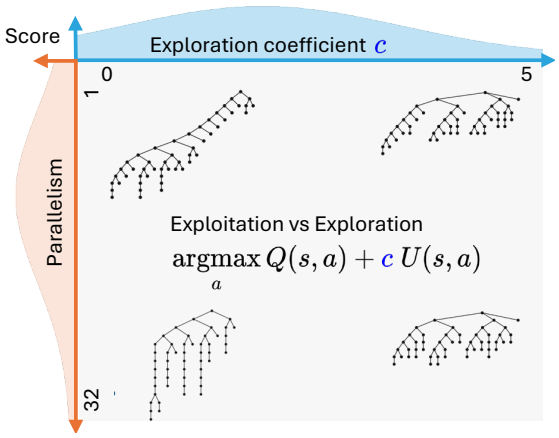


Figure 1: The exploration-exploitation tradeoff is a fundamental challenge in reinforcement learning and, in MCTS, gives rise to various search dynamics and performance profiles. In MCTS with the pUCT formula, actions are selected by combining an action-value term $Q(s,a)$ with an exploration term $U(s,a)$, scaled by the exploration coefficient c. The coefficient c changes the search explicitly by changing the relative strength of the exploration term, while tree parallelism changes the search implicitly through concurrent traversal. Our work characterizes how these two factors jointly shape search tree structure and performance.[1]

form along different parallel selections. Moreover, changes in the node distribution of the search tree also implicitly change the effective exploration-exploitation behavior. Consequently, different combinations of c and parallelism can lead to different shapes of the search tree, shift the balance between exploration and exploitation, and lead to different search results.

Unfortunately, despite the central role of the interplay between these two factors, they are often studied separately. Previous work has analyzed MCTS through convergence properties and regret bounds [11–14], which provide the theoretical basis for our formulation of how the pUCT formula shapes the search tree. Prior studies on parallel MCTS have focused mainly on improving search throughput and scalability [10, 15–17]. However, to our knowledge, the interaction between parallelism and exploration has not been quantified through the search tree structure.

Motivated by this observation, in this work, we characterize how the exploration coefficient and parallelism shape the search tree structure in MCTS. We first analyze the asymptotic depth under the pUCT formula. Under regularity assumptions, we prove that, for a constant exploration coefficient, the depth of the optimal path[2] grows as $\Theta(\sqrt{n})$ with the number of simulations n, until capped by the terminal depth. We then extend this result to general exploration schedules $c(\cdot)$. In particular, for the MuZero exploration schedule, the growth rate of the optimal path is of order $\Theta(\sqrt{n}/\log n)$. We further prove that, in tree-parallel MCTS, a fixed number of threads preserves the sequential asymptotic depth order, while sufficiently large parallelism introduces contention and reduces the depth growth rate to $\Theta(n/T)$ for T threads. Moreover, we include a **Lean formalization** [18] of the algebraic derivations used in the proofs in Appendix B, available at `https://github.com/chengscott/search-tree-scaling`. Our experiments, conducted across Go, Atari, and large language model (LLM) planning, reveal that the empirical scaling of search depth closely matches the theoretical estimate with RMSE of 4.10 in 44 Atari games. We then characterize the exploration-exploitation balance through the distribution of nodes over depth based on our theoretical results, and further discuss the performance scaling with domain-specific factors. Finally, the log-linear fit shows that each doubling of the number of threads decreases the best-performing exploration coefficient by approximately 0.19, with $R^2 = 0.96$ in Go. Overall, our theoretical results on search-tree scaling reflect the exploration-exploitation balance observed in practice.

[1] Each example search tree contains 50 simulations on Atari Freeway, with $(\text{threads}, c) = (1,0), (1,5), (32,0), (32,5)$.
[2] See Definition 2.

2 Background

2.1 pUCT Formula

The pUCT formula is an action-selection rule in MCTS, adapted in AlphaGo-style search to combine action-value estimates with policy priors from neural networks [2, 9]. As shown in Equation (1), it consists of an exploitation term, given by the current action-value estimate, and a prior-weighted exploration term. This exploration term favors actions with high policy prior and relatively low state-action visit count. For a fixed parent visit count, the term decreases as the same action is repeatedly selected. The exploration coefficient may be chosen as a constant or as a function of the visit count.

Definition 1. (pUCT) At each node s, pUCT selects

$$\arg\max_a \left(Q(s,a) + cP(s,a)\frac{\sqrt{N(s)}}{1+N(s,a)} \right) \tag{1}$$

where $N(s,a)$ is the number of times that action a has been selected from s, $N(s) = \sum_b N(s,b)$ is the total number of visits to node s, $Q(s,a)$ is the estimate of the action-value, $P(s,a)$ is the policy prior and $c > 0$ is the exploration coefficient.

2.2 Parallel MCTS

In this paper, we focus on parallelism within a single MCTS search tree, since our analysis studies properties of the search tree itself. Other common paradigms and implementation variants are discussed separately in Section 3. In particular, tree-parallel MCTS [10] runs multiple simulations concurrently on a shared search tree. Similar to sequential MCTS, each thread repeatedly selects actions from the root according to the pUCT formula in Equation (1), using the currently shared visit counts and value estimates. After reaching a new state, the thread expands and evaluates the new node, and then updates the return along the selected path. Because threads run asynchronously, multiple threads can reach the same state s and select an action using the current shared statistics before the updates from any of these selections are applied in s. We refer to this period of overlapping selections as a *stale selection window* in s. Within such a window, the shared statistics in s do not yet include the true updates generated by the selections in the same window. Our theoretical analysis of parallelism in Section 4.2 is formulated using these asynchronous selections and stale selection windows in the shared tree search.

In practice, *virtual loss* [10] is often introduced to reduce repeated selection of the same action under shared statistics. When a thread selects an action a from state s, the shared statistics are temporarily modified before the simulation finishes, for example, by temporarily increasing $N(s,a)$ or decreasing $Q(s,a)$. This makes the selected action less attractive to other concurrent threads under the selection rule, encouraging them to explore different branches. Once the simulation completes, the temporary modification is removed, and the true return is updated normally. Thus, virtual loss changes the temporary statistics observed during a stale selection window, while the persistent shared visit counts and value estimates are updated from completed simulations.

3 Related Work

Most prior theoretical analyses of MCTS have focused on convergence properties [11, 12, 19], regret bounds derived from multi-armed bandit frameworks [13, 14, 20], and finite-budget action selection via best-arm identification or sequential halving methods [21–23]. Early work introduced UCT by adapting upper-confidence-bound (UCB) principles from multi-armed bandits to guide node selection in tree search [6, 7, 24]. Subsequent studies extended the research by providing non-asymptotic guarantees [25], convergence results for continuous-action planning [26, 27], and uncertainty-aware analyses [28]. In particular, the pUCT formulation [2, 9] is central to AlphaGo, AlphaZero, MuZero, and their variants [3, 8, 29, 30]. Recent work has also interpreted MCTS as a form of regularized policy improvement [31–33], but these approaches study a different decision rule, while our analysis focuses on the search-tree scaling behavior of pUCT.

Moreover, parallel MCTS has been studied from several aspects, including root parallelization [34, 35], leaf and tree parallelization with virtual loss [10], master-slave coordination [15], large-scale

implementations [16, 17, 36], and lock-free multi-threaded updates [37, 38]. Root parallelism uses independent search trees, whereas our analysis concerns tree parallelism within a shared tree and the resulting contention. While most prior works focus on the computing paradigm and runtime speedup, we study the exploration and exploitation implications of parallel MCTS. In particular, we characterize the joint effect of the exploration coefficient and threads theoretically in Section 4 and empirically in Section 5.

On the other hand, the asymptotic structure of random trees has been studied through a variety of approaches, including branching-process models [39], complexity analyses of branch-and-bound search [40], limit theorems for normalized tree profiles [41], and Brownian-excursion characterizations of contour processes for binary trees [42]. Similarly, local contour properties of random binary search trees have been analyzed using limit laws [43]. Most of this prior work examines tree growth under predefined random processes. In contrast, our work studies tree properties guided by the pUCT formula used in MCTS, and is therefore complementary to these existing lines of research.

4 Characterization of the Asymptotic Tree Structure

In this section, we characterize the asymptotic depth of the MCTS tree obtained by the pUCT formula, with regard to two factors: the exploration coefficient in Section 4.1 and parallelism in Section 4.2. Detailed proofs and derivations are provided in Appendix B.

4.1 Exploration Coefficient

Assumption 1 (Regularity)**.** Consider an MCTS search tree with bounded branching factor b, obtained from the pUCT formula. The following regularity conditions hold. First, every nonterminal state s has some suboptimal action $a \neq a^*(s)$ whose prior probability $P(s,a)$ is uniformly bounded away from 0. Second, for every state s and suboptimal action $a \neq a^*(s)$, the value gap

$$\Delta(s,a) := Q(s,a^*(s)) - Q(s,a) \tag{2}$$

is uniformly bounded away from both 0 and ∞; that is, $\Delta_{\min} = \inf_{s,a\neq a^*(s)} \Delta(s,a) > 0$ and $\Delta_{\max} = \sup_{s,a\neq a^*(s)} \Delta(s,a) < \infty$.

Definition 2. (Optimal Path) The *optimal path* is defined by repeatedly choosing an action with maximal asymptotic value on the path. A path is *suboptimal* if it is not an optimal path.

Theorem 1 (Constant exploration coefficient c)**.** *Consider an MCTS search tree of n simulations with a constant exploration coefficient $c > 0$, and suppose Assumption 1 holds. Then, as $n \to \infty$,*

(1) For each state s with $N(s) \to \infty$, and every suboptimal action $a \neq a^*(s)$ with $P(s,a) > 0$, the visit count satisfies $N(s,a) = \Theta(\sqrt{N(s)})$.

(2) The maximum depth reached along the optimal path satisfies $H_{\text{opt}} = \Theta(\sqrt{n})$ until capped by a finite terminal depth.

(3) The maximum depth of the search tree satisfies $H_{\max} = O(\sqrt{n})$.

Remark 1. Theorem 1 gives the main asymptotic result under a constant exploration coefficient. Part (1) shows that suboptimal actions are sampled at the $\Theta(\sqrt{N(s)})$ rate, and Parts (2) and (3) translate these local dynamics into global depth bounds.

Corollary 1.1. *Under the assumptions of Theorem 1, consider a path $s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots \xrightarrow{a_{d-1}} s_d$ formed by consecutively selecting suboptimal actions. Suppose that the priors are uniformly small along the path, satisfying $c \cdot P(s_i,a_i) < \Delta_{\min}/4$ for $i \in \{0,\dots,d-1\}$. Then, the path length satisfies $d = O(\log\log n)$.*

Remark 2. The corollary implies that, while the search tree may contain suboptimal branches off the optimal path, any consecutive chain of suboptimal actions with uniformly small priors is strongly suppressed. Moreover, when $c = 0$, the search reduces to pure exploitation, and this also shows highly suppressed suboptimal paths.

Theorem 2 (General exploration coefficient $c(\cdot)$)**.** *Consider an MCTS search tree of n simulations satisfying Assumption 1. Suppose that the exploration coefficient function $c(\cdot)$ of the parent visit*

count is positive and continuous, and $c(m)\sqrt{m}$ is eventually nondecreasing. If $c(m) = O(\sqrt{m})$, the maximum depth along the optimal path satisfies

$$H_{\text{opt}}(n) = \Theta\left(\int_1^n \frac{1}{c(m)\sqrt{m}}\, dm\right). \tag{3}$$

If $c(m) = \Omega(\sqrt{m})$, then $H_{\text{opt}}(n) = \Theta(\log n)$. Both hold until capped by a finite terminal depth.

Remark 3. Theorem 2 generalizes Theorem 1 to the case of a nonconstant exploration coefficient $c(\cdot)$. In practice, c is typically chosen from a particular growth class rather than an arbitrary growth rate. Hence, Corollary 2.1 specializes this result to functions of the form $c(m) = m^\alpha L(m)$, where L is slowly varying.

Definition 3 (Slowly varying function)**.** A positive measurable function L is said to be *slowly varying* if, for every fixed $\lambda > 0$,

$$\lim_{x\to\infty} \frac{L(\lambda x)}{L(x)} = 1. \tag{4}$$

Example 1. Constant functions and $\log x$ are slowly varying. In contrast, polynomials $L(x)$ of degree $d \geq 1$ are not slowly varying, since $L(\lambda x)/L(x) \to \lambda^d$, which is not 1 in general.

Corollary 2.1. *Under the first case of Theorem 2, suppose further that $c(m) = m^\alpha L(m)$, where L is slowly varying. Then, $\alpha \leq \frac{1}{2}$ since $c(m) = O(\sqrt{m})$. Moreover,*

(1) If $\alpha < \frac{1}{2}$,

$$H_{\text{opt}}(n) = \Theta\left(\frac{\sqrt{n}}{c(n)}\right). \tag{5}$$

(2) If $\alpha = \frac{1}{2}$,

$$H_{\text{opt}}(n) = \Theta\left(\int_1^n \frac{1}{m\, L(m)}\, dm\right). \tag{6}$$

In both cases, the estimate holds until capped by a finite terminal depth.

Example 2 (Square-root exploration)**.** Suppose $c(m) = \sqrt{m}$, which is the form $c(m) = m^\alpha L(m)$ with $\alpha = \frac{1}{2}$ and $L \equiv 1$. Then, by Corollary 2.1(2),

$$H_{\text{opt}}(n) = \Theta\left(\int_1^n \frac{1}{m}\, dm\right) = \Theta(\log n). \tag{7}$$

Example 3 (MuZero)**.** In MuZero, the exploration coefficient is given by

$$c_{\text{puct}}(m) = \log\left(\frac{m + \texttt{pb_c_base} + 1}{\texttt{pb_c_base}}\right) + \texttt{pb_c_init}, \tag{8}$$

where `pb_c_base` and `pb_c_init` are constants. Thus, $c(m) = \Theta(\log m)$, corresponding to the form $c(m) = m^\alpha L(m)$ with $\alpha = 0$ and $L(m) = \Theta(\log m)$. Then, by Part (1), we have

$$H_{\text{opt}}(n) = \Theta\left(\frac{\sqrt{n}}{c(n)}\right) = \Theta\left(\frac{\sqrt{n}}{\log n}\right). \tag{9}$$

Remark 4. Theorem 1 and Corollary 2.1 match the intuition that larger exploration schedules lead to slower growth of the optimal depth. In particular, as $c(m)$ becomes larger, the order of $H_{\text{opt}}(n)$ decreases.

4.2 Parallelism

Theorem 3 (Constant parallelism)**.** *Consider tree-parallel MCTS with a constant exploration coefficient $c > 0$, a fixed number T of concurrent threads, and n simulations, under Assumption 1. Then, as $n \to \infty$, $H_{\text{opt}}(n, T) = \Theta(\sqrt{n})$, until capped by a finite terminal depth.*

Remark 5. Since T does not grow with n, the asymptotic depth remains the same as in Theorem 1. Similarly, under the assumptions of Theorem 2, the corresponding asymptotic depth estimates are unchanged. Although this result is immediate, it covers the most common parallelization setting, where the number of threads is moderate and much smaller than the number of simulations.

Assumption 2 (Thread contention)**.** Consider tree-parallel MCTS with $T = T(n)$ concurrent threads and n simulations, where $1 \leq T(n) \leq n$. Assume that there exist constants $\rho > 0$ and $K < n$ such that, for every nonterminal state s on the optimal path and for sufficiently large $N(s)$, the following hold: (1) there is a *stale selection window* at s, during which at least ρT selections reach s before any backup generated by selections in that window is updated to s; (2) among these overlapping selections, at most K select the optimal action $a^*(s)$.

Remark 6. Assumption 2 models contention among threads in tree-parallel MCTS during the stale selection window defined in Section 2.2. The first condition naturally reflects the setting in which the number of active threads $T(n)$ scales with the simulation count n, and thus, the number of selections made during a stale selection window also scales with $T(n)$. The second condition states that action selection is not fully concentrated on the optimal action within such a window since, in practice, mechanisms such as virtual loss discourage multiple threads from following the same action.

Theorem 4 (General parallelism)**.** *Consider tree-parallel MCTS with a constant exploration coefficient $c > 0$, $T = T(n)$ concurrent threads, and n simulations, under Assumptions 1 and 2. Then, as $n \to \infty$, we have*

$$H_{\mathrm{opt}}(n, T) = \Theta\left(\int_1^n \frac{dm}{\sqrt{m} + T}\right), \tag{10}$$

until capped by a finite terminal depth. In particular,

(1) If $T = o(\sqrt{n})$,

$$H_{\mathrm{opt}}(n, T) = \Theta(\sqrt{n}). \tag{11}$$

(2) If $T \gg \sqrt{n}$,

$$H_{\mathrm{opt}}(n, T) = \Theta(n/T). \tag{12}$$

Remark 7. Theorem 4 generalizes the previous theorem by allowing the number of threads T to vary with n, thus covering massively parallel settings. The additional assumptions model contention among threads, where many selections may reach the same high-traffic states before their current search backups become visible to other threads. These assumptions are also reasonable since, when the number of threads T is small, the asymptotic depth estimate is the same as in the sequential case, suggesting that contention is asymptotically negligible. In contrast, when T is greater than the turning point, $\sqrt{n}$, contention reduces the depth growth rate.

5 Characterization of Performance Scaling

We first describe the experimental setup in Section 5.1, and then empirically evaluate the scaling of the search depth in Section 5.2. We further study how the distribution of nodes over depth shifts when varying search settings in Section 5.3, analyze performance scaling across domains in Section 5.4, and investigate the joint effect of parallelism and the exploration coefficient in Section 5.5.

5.1 Experimental Setting

We evaluate the scaling behavior of MCTS search trees across three domains: 19x19 Go, Atari games, and large language model (LLM) planning. For these experiments, we use the KataGo implementation [30] for Go, the MuZero framework [44] for Atari, and the Reasoning via Planning (RAP) framework [45, 46] for LLM planning on the Blocksworld task [47, 48]. Moreover, we use c_{puct} to denote the initial exploration constant, `pb_c_init`, in Equation (8), while the actual exploration coefficient still follows the equation. Detailed experimental settings are provided in Appendix A.1.

5.2 Empirical Scaling of Search Depth

To evaluate the asymptotic depth growth discussed in Section 4.1, we compare the estimated scaling with the empirical depths measured in Atari games. Since the exploration coefficient follows MuZero's schedule in Equation (8), the estimated depth is given by Equation (9). Moreover, the empirical optimal path is obtained by recursively selecting the action with the largest visit count after a complete search. For each simulation count, we measure the depth of this path over 120,000 game

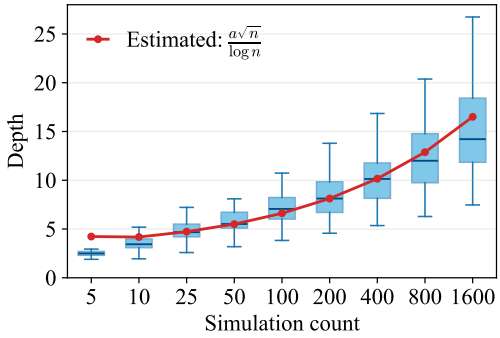
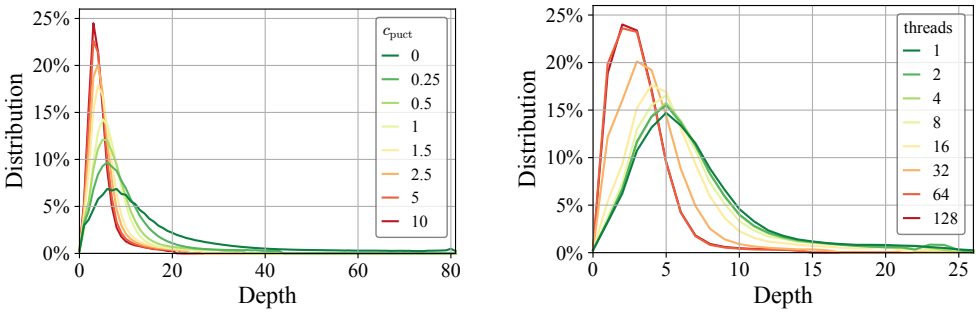


Figure 2: Empirical depth of the optimal path $\hat{H}_{\text{opt}}$ across different simulation counts on Atari games. Blue boxes show the empirical distributions, with the median and interquartile range indicated by the box, and whiskers denoting the minimum and maximum, while the red curve shows the fitted depth estimate based on the simulation count.

states for each of the 44 Atari games. We then fit the coefficient of the estimated curve using ordinary least squares.

Figure 2 shows that the estimated curve aligns well with the empirical trend across the simulations, with a fitted coefficient of $a \approx 3.004$, an RMSE of 4.10, and R^2 of 0.61. The fitting residuals mainly come from the increasing dispersion of empirical depths at larger simulation counts, since different Atari games have different branching factors. These game-dependent differences become more prominent as the simulation count increases. Overall, the fitted curve tracks the main empirical trend, suggesting that Equation (9) captures the dominant scaling behavior of the search depth with respect to the number of simulations. Additional evaluations are included in Appendix A.2, where we also observe a close fit in Go and study alternative schedules for the exploration coefficient.

5.3 Distribution of Nodes over Depth



(a) Varying c_{puct}.

(b) Varying the number of threads.

Figure 3: Distribution of nodes over search depth in Go with 400 simulations under different search settings. **(a)** When varying c_{puct} with a single search thread, smaller c_{puct} encourages deeper search with a heavier tail, whereas larger c_{puct} concentrates nodes near the root and yields shallower search. **(b)** When varying the number of search threads with $c_{\text{puct}} = 1$, small thread counts leave the depth distribution largely unchanged, while larger thread counts increase contention and shift the search toward shallower depths.

To further examine the scaling behavior of search depth, we study the distribution of nodes over depth, defined as the fraction of nodes at depth d among all nodes in the search tree. In Figure 3, each distribution is averaged over 110,000 games per setting, and the corresponding playing-strength measurements have a median 95% confidence interval (CI) half-width of 39.07 Elo ratings. Intuitively, c_{puct} provides explicit control over the exploration-exploitation balance. As shown in Figure 3a, when c_{puct} is small, the depth distribution is wider and has a heavier tail, indicating that the search assigns more nodes to deeper levels. As c_{puct} increases, the distribution becomes more concentrated

at small depths, suggesting that the tree expands wider near the root while becoming shallower overall. In particular, when $c_{\mathrm{puct}} = 0$, the search is purely driven by the action-value estimates without the exploration term, and thus, with pure exploitation, the tree depth can reach up to 80. In general, a larger c_{puct} leads to a wider expansion at each node and therefore reduces the total depth of the tree, which matches the trend shown in Equation (9).

The degree of parallelism is another factor in the scaling behavior of the search depth. Figure 3b shows that the distributions for 1 to 8 threads largely overlap, with only a slight shift toward smaller depths as the number of threads increases. This observation matches the discussion in Section 4.2, where Equation (11) shows that the asymptotic optimal depth H_{opt} remains unchanged when the number of threads is relatively small. Since the depth remains nearly unchanged in this case, increasing parallelism mainly broadens exploration within the same depth range rather than pushing the search deeper. In contrast, for a larger number of threads, the empirical distribution shifts more clearly toward shallow depths. The shift begins to appear clearly around 16 threads and becomes most prominent at 64 and 128 threads. Importantly, the trend also aligns with the previous discussion of Equation (12), which indicates that when the number of threads becomes large, contention at shared nodes changes the search dynamics and shifts the distribution toward smaller depths. In particular, since the total number of simulations in this experiment is 400, the clear shift starting around 16 threads occurs near the scale $\sqrt{400} = 20$ threads. Together, these results indicate that c_{puct} and parallelism influence the depth distribution in distinct ways. Accordingly, we examine their joint interactions in Section 5.5.

5.4 Performance Scaling

The previous section shows that the depth distribution reflects the exploration-exploitation balance. Although better balance typically leads to stronger performance scaling, Figure 4 shows that performance scaling also depends on domain-specific factors, and each result is averaged over 1,000 games per setting, with median 95% CI half-widths of 0.0160, 0.0191, and 0.0482 for Go, LLM, and Atari, respectively. In Go, performance increases almost monotonically as the number of simulations increases. Moreover, varying the exploration coefficient leads to a consistent performance shift, which aligns with the depth-distribution behavior shown earlier in Figure 3a. In the LLM setting, the search yields greater gains at low simulation counts, but the improvement quickly saturates, indicating diminishing returns from additional simulations. This is because the episode length in Blocksworld is limited to 8 steps, which constrains the effective lookahead horizon of MCTS. As a result, the overall scaling is limited by the underlying model's capability rather than by the additional lookahead provided by tree search. In contrast, Atari exhibits a qualitatively different pattern, in which performance improves at low simulation counts, reaches a peak at an intermediate simulation count, and then degrades as the simulation count increases further. Overall, we observe that the scaling trend is roughly preserved under different exploration coefficients.

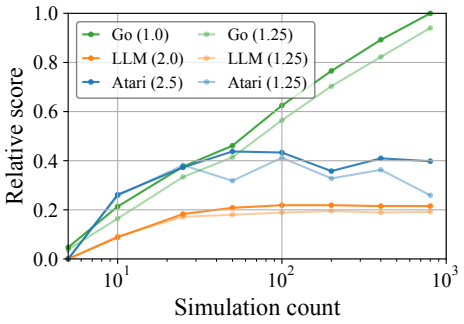
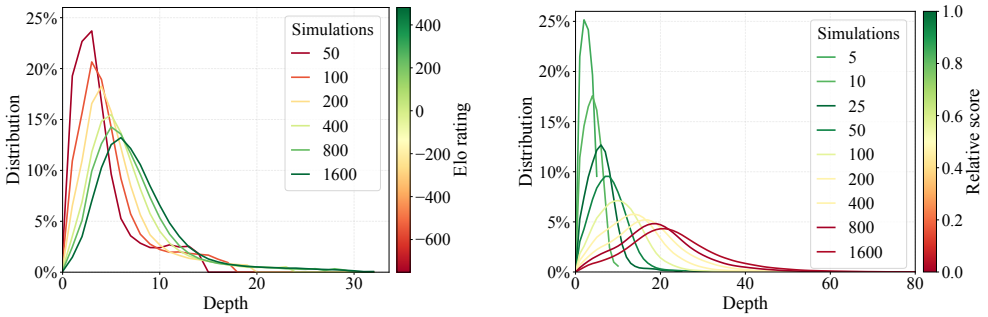


Figure 4: Performance scaling across Go, LLM planning (Blocksworld), and Atari (Amidar), with exploration coefficients shown in parentheses. The lighter lines correspond to the baseline setting, $c_{\mathrm{puct}} = 1.25$, while the solid lines indicate the best-performing value of c_{puct} in each domain.

To further explain these scaling behaviors, we relate performance scaling to the distribution of nodes over depth discussed in the previous section. Figure 5a shows that, in Go, increasing the simulation count gradually shifts the node distribution toward larger depths. The overall distribution profile remains similar across simulations, indicating that additional simulations do not abruptly reshape the search tree. This suggests that additional simulations continue to support effective exploration and exploitation at deeper levels, leading to a steady improvement in playing strength.

In contrast, Figure 5b shows a different scaling behavior in Atari. As the simulation count increases, the node distribution also moves toward larger depths, but the shift is substantially more rapid than in Go. This rapid shift suggests that, beyond a certain simulation count, deeper search becomes less



(a) Go (b) Atari (Bank Heist)

Figure 5: Distribution of nodes over depth for different simulation counts in Go and Atari. Increasing the number of simulations shifts the distribution toward larger depths. Curve color reflects performance, measured by Elo rating [49] compared to the 400-simulation baseline in Go, and by Atari game score rescaled between the 5 to 1600-simulation results. **(a)** In Go, increasing the number of simulations gradually shifts the distribution toward greater depth and consistently improves playing strength. Simulation counts start from 50 due to the branching factor. **(b)** In Atari, the distribution shifts toward greater depth more rapidly, but performance improves only up to 25 simulations and then decreases as search becomes deeper.

reliable for guiding node selection and expansion. Since MuZero relies on a learned dynamics model, prediction errors may accumulate with depth, making deeper latent states less reliable for guiding the search. Similar degradation at high simulation counts has been observed in prior evaluations [3]. As a result, additional simulations no longer support effective selection and expansion at deeper levels. Instead, the search is pushed deeper without a corresponding gain in decision quality, and the node distribution shifts sharply toward larger depths. Thus, performance improves only at small to moderate simulation counts and then declines, even as the search continues to expand deeper.

5.5 Interaction between Parallelism and Exploration Coefficient

Motivated by the observation made in Section 5.3, we study the joint effect of threads and c_{puct} through a grid sweep in Go, as shown in Table 1. Each setting is evaluated over 1,000 games against the same KataGo baseline configuration ($c_{\text{puct}} = 1.15$, $T = 4$), with the reported Elo ratings relative to this baseline and a standard deviation of 4.5 Elo ratings across repeated runs. The best overall performance is achieved with a single thread, and the best result for each thread count decreases monotonically as the number of threads increases. However, moderate parallelism remains useful in practice, as it reduces search latency with only a slight drop in strength. Moreover, we report results only up to 10 threads, as larger numbers of threads lead to larger strength drops and are of less practical interest. The interaction between parallelism and exploration also aligns with observations in the previous sections. For each thread count, performance is best at an intermediate value of c_{puct}, but this best-performing value shifts toward smaller values as the number of threads increases. This trend is consistent with the depth-distribution analysis in Section 5.3, where increasing the number of threads pushes the search toward broader and shallower expansions, and thus, larger thread counts tend to have more exploration in the multi-threading scenario. As a result, reducing c_{puct} helps to recover a better balance between exploration and exploitation. Extrapolating this trend, under sufficiently high parallelism, the tree can become highly contended, effectively introducing too much exploration at each depth; in this case, the optimal c_{puct} could further decrease, potentially approaching zero, meaning pure exploitation, to reach an exploration-exploitation balance.

Table 1: Playing strength in Go with 400 simulations, measured in Elo ratings, under different numbers of threads and exploration coefficients (c_{puct}). Cell shading indicates the relative strength and the best result for each thread count is highlighted in blue.

threads	c_{puct}				
	0.8	1.0	1.1	1.3	1.5
1	−18.1	11.8	22.3	27.9	**30.0**
2	−9.0	−0.7	1.4	**21.6**	−6.9
4	−24.4	−18.1	**6.3**	−12.5	−22.3
8	−38.4	**−11.8**	−19.5	−12.5	−34.2
10	**−46.1**	−53.2	−57.5	−46.2	−51.1

To quantify the interaction between parallelism and the best-performing exploration coefficient, we fit the best-performing c_{puct} values in Table 1 as a function of thread count using a log-linear model. The least-squares fit is

$$c_{\mathrm{puct}}(T) = 1.498 - 0.192 \log_2 T. \tag{13}$$

The fit achieves $R^2 = 0.966$ with an RMSE of 0.045, indicating that the best-performing exploration coefficient decreases approximately linearly with the logarithm of the thread count. Equivalently, each doubling of the number of threads reduces the best-performing exploration coefficient by approximately 0.192.

6 Concluding Remarks

We characterize search-tree scaling in pUCT-based MCTS by analyzing how the exploration-coefficient schedule and tree parallelism shape the depth growth, thereby influencing the balance between exploration and exploitation. Our main theorem provides a classification of how different growth rates of the exploration-coefficient schedule and the degree of parallelism determine the growth rate of the depth of the optimal path. One limitation of our work is that the regularity assumption requires at least two actions at each nonterminal state to have nonzero prior probability, ensuring that the search can explore at least two children. Otherwise, the search may degenerate into a single-path expansion. For instance, if only one action has nonzero prior probability, the optimal path can grow linearly with n. Furthermore, in hard-exploration domains where the performance of MuZero-style search does not scale with additional simulations, parallel scaling is also ineffective, limiting the interpretation of its interaction with the exploration coefficient. Future work could apply similar analysis to broader UCT-based search dynamics or other parallelization schemes. Another direction is to develop a quantitative characterization of the joint effect between the exploration coefficient and parallelism, complementing the empirical findings in this work. Finally, our released `Lean` formalization provides a basis for future extensions of the analysis to more general search settings.

Acknowledgments and Disclosure of Funding

This research is supported in part by DOE grant DE-SC0023186, NSF grants 2453653 and 2211018, and National Science and Technology Council of Taiwan project number NSTC114-2221-E-001-019-MY3. The opinions expressed in this paper are those of the authors and not necessarily the views of the NSF or the DOE. We thank the Taiwan National Center for High-performance Computing (NCHC) for providing computational and storage resources. The authors would also like to thank the anonymous reviewers for their valuable comments.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[3] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, *et al.*, "Mastering Atari, Go, chess and shogi by planning with a learned model," *Nature*, vol. 588, no. 7839, pp. 604–609, 2020.

[4] J. G. March, "Exploration and exploitation in organizational learning," *Organization Science*, vol. 2, no. 1, pp. 71–87, 1991.

[5] L. Kocsis and C. Szepesvári, "Bandit based Monte-Carlo planning," in *European Conference on Machine Learning*, pp. 282–293, Springer, 2006.

[6] R. Coulom, "Efficient selectivity and backup operators in Monte-Carlo tree search," in *International Conference on Computers and Games*, pp. 72–83, Springer, 2006.

[7] L. Kocsis, C. Szepesvári, and J. Willemson, "Improved Monte-Carlo search," tech. rep., University of Tartu, Tartu, Estonia, 2006.

[8] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[9] C. D. Rosin, "Multi-armed bandits with episode context," *Annals of Mathematics and Artificial Intelligence*, vol. 61, no. 3, pp. 203–230, 2011.

[10] G. M. J.-B. Chaslot, M. H. M. Winands, and H. J. van den Herik, "Parallel Monte-Carlo tree search," in *International Conference on Computers and Games*, pp. 60–71, Springer, 2008.

[11] G. Tesauro, V. T. Rajan, and R. B. Segal, "Bayesian inference in Monte-Carlo tree search," in *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, pp. 580–588, AUAI Press, 2010.

[12] V. Lisý, V. Kovařík, M. Lanctot, and B. Bošanský, "Convergence of Monte Carlo tree search in simultaneous move games," *Advances in Neural Information Processing Systems*, vol. 26, 2013.

[13] P.-A. Coquelin and R. Munos, "Bandit algorithms for tree search," in *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*, pp. 67–74, AUAI Press, 2007.

[14] R. Munos, "From bandits to Monte-Carlo tree search: The optimistic principle applied to optimization and planning," *Foundations and Trends® in Machine Learning*, vol. 7, no. 1, pp. 1–129, 2014.

[15] T. Cazenave and N. Jouandeau, "A parallel Monte-Carlo tree search algorithm," in *International Conference on Computers and Games*, pp. 72–80, Springer, 2008.

[16] K. Rocki and R. Suda, "Large-scale parallel Monte Carlo tree search on GPU," in *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD forum*, pp. 2034–2037, IEEE, 2011.

[17] K. Yoshizoe, A. Kishimoto, T. Kaneko, H. Yoshimoto, and Y. Ishikawa, "Scalable distributed Monte-Carlo tree search," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 2, pp. 180–187, 2011.

[18] L. de Moura and S. Ullrich, "The Lean 4 theorem prover and programming language," in *International Conference on Automated Deduction*, pp. 625–635, Springer, 2021.

[19] H. S. Chang, "On the convergence rate of MCTS for the optimal value estimation in Markov decision processes," *IEEE Transactions on Automatic Control*, vol. 70, no. 7, pp. 4788–4793, 2025.

[20] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, no. 2–3, pp. 235–256, 2002.

[21] D. Tolpin and S. Shimony, "MCTS based on simple regret," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 26, pp. 570–576, 2012.

[22] E. Kaufmann and W. M. Koolen, "Monte-Carlo tree search by best arm identification," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[23] T. Cazenave, "Sequential halving applied to trees," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 7, no. 1, pp. 102–105, 2015.

[24] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of Monte Carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, pp. 1–43, 2012.

[25] D. Shah, Q. Xie, and Z. Xu, "Non-asymptotic analysis of Monte Carlo tree search," in *Abstracts of the 2020 SIGMETRICS/Performance Joint International Conference on Measurement and Modeling of Computer Systems*, pp. 31–32, 2020.

[26] B. Kim, K. Lee, S. Lim, L. Kaelbling, and T. Lozano-Pérez, "Monte Carlo tree search in continuous spaces using Voronoi optimistic optimization with regret bounds," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 9916–9924, 2020.

[27] W. Mao, K. Zhang, Q. Xie, and T. Başar, "POLY-HOOT: Monte-Carlo planning in continuous space MDPs with non-asymptotic analysis," *Advances in Neural Information Processing Systems*, vol. 33, pp. 4549–4559, 2020.

[28] Y. Oren, M. T. J. Spaan, and W. Böhmer, "E-MCTS: Deep exploration in model-based reinforcement learning by planning with epistemic uncertainty," in *Sixteenth European Workshop on Reinforcement Learning*, 2023.

[29] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[30] D. J. Wu, "Accelerating self-play learning in Go," *arXiv preprint arXiv:1902.10565*, 2019.

[31] J.-B. Grill, F. Altché, Y. Tang, T. Hubert, M. Valko, I. Antonoglou, and R. Munos, "Monte-Carlo tree search as regularized policy optimization," in *International Conference on Machine Learning*, pp. 3769–3778, PMLR, 2020.

[32] T. Q. Dam, C. D'Eramo, J. Peters, and J. Pajarinen, "Convex regularization in Monte-Carlo tree search," in *International Conference on Machine Learning*, pp. 2365–2375, PMLR, 2021.

[33] I. Danihelka, A. Guez, J. Schrittwieser, and D. Silver, "Policy improvement by planning with Gumbel," in *International Conference on Learning Representations*, 2022.

[34] Y. Soejima, A. Kishimoto, and O. Watanabe, "Evaluating root parallelization in Go," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 2, no. 4, pp. 278–287, 2010.

[35] E. Steinmetz and M. Gini, "More trees or larger trees: Parallelizing Monte Carlo tree search," *IEEE Transactions on Games*, vol. 13, no. 3, pp. 315–320, 2021.

[36] X. Yang, T. Aasawat, and K. Yoshizoe, "Practical massively parallel Monte-Carlo tree search applied to molecular design," in *International Conference on Learning Representations*, 2021.

[37] M. Enzenberger and M. Müller, "A lock-free multithreaded Monte-Carlo tree search algorithm," in *Advances in Computer Games*, pp. 14–20, Springer, 2009.

[38] S. A. Mirsoleimani, H. J. van den Herik, A. Plaat, and J. Vermaseren, "A lock-free algorithm for parallel MCTS," in *Proceedings of the 10th International Conference on Agents and Artificial Intelligence (ICAART), Volume 2*, pp. 589–598, 2018.

[39] L. Devroye, "Branching processes and their applications in the analysis of tree structures and tree algorithms," in *Probabilistic methods for algorithmic discrete mathematics*, pp. 249–314, Springer, 1998.

[40] L. Devroye and C. Zamora-Cura, "On the complexity of branch-and-bound search for random trees," *Random Structures & Algorithms*, vol. 14, no. 4, pp. 309–327, 1999.

[41] M. Drmota, S. Janson, and R. Neininger, "A functional limit theorem for the profile of search trees," *Annals of Applied Probability*, vol. 18, no. 1, pp. 288–333, 2008.

[42] W. J. Gutjahr and G. C. Pflug, "The asymptotic contour process of a binary tree is a Brownian excursion," *Stochastic Processes and their Applications*, vol. 41, no. 1, pp. 69–89, 1992.

[43] L. Devroye, "Limit laws for local counters in random binary search trees," *Random Structures & Algorithms*, vol. 2, no. 3, pp. 303–315, 1991.

[44] T.-R. Wu, H. Guei, P.-C. Peng, P.-W. Huang, T. H. Wei, C.-C. Shih, and Y.-J. Tsai, "MiniZero: Comparative analysis of AlphaZero and MuZero on Go, Othello, and Atari games," *IEEE Transactions on Games*, vol. 17, no. 1, pp. 125–137, 2025.

[45] S. Hao, Y. Gu, H. Luo, T. Liu, X. Shao, X. Wang, S. Xie, H. Ma, A. Samavedhi, Q. Gao, Z. Wang, and Z. Hu, "LLM reasoners: New evaluation, library, and analysis of step-by-step reasoning with large language models," in *First Conference on Language Modeling*, 2024.

[46] S. Hao, Y. Gu, H. Ma, J. Hong, Z. Wang, D. Wang, and Z. Hu, "Reasoning with language model is planning with world model," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 8154–8173, 2023.

[47] K. Valmeekam, M. Marquez, S. Sreedharan, and S. Kambhampati, "On the planning abilities of large language models: a critical investigation," *Advances in Neural Information Processing Systems*, vol. 36, pp. 75993–76005, 2023.

[48] K. Valmeekam, M. Marquez, A. Olmo, S. Sreedharan, and S. Kambhampati, "PlanBench: An extensible benchmark for evaluating large language models on planning and reasoning about change," *Advances in Neural Information Processing Systems*, vol. 36, pp. 38975–38987, 2023.

[49] R. Coulom, "Computing Elo ratings of move patterns in the game of Go," *ICGA Journal*, vol. 30, no. 4, pp. 198–208, 2007.

[50] D. J. Wu, "Networks for KataGo." `https://katagotraining.org/networks/`, 2020.

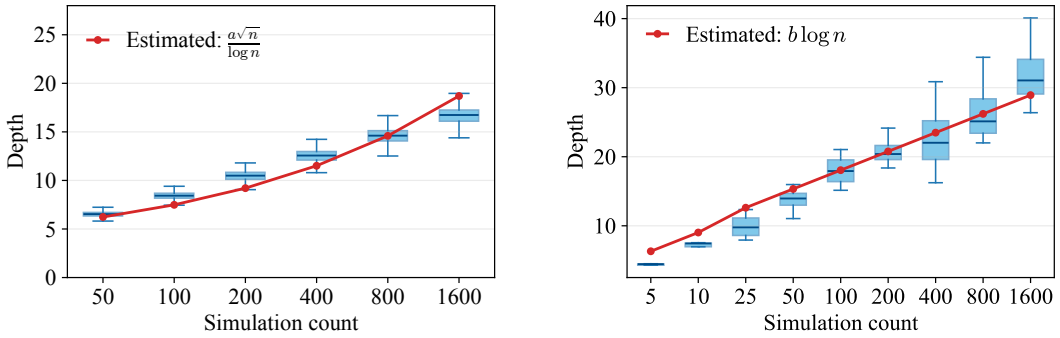
A Supplemental Evaluation

A.1 Experimental Setting

Our experiments are primarily conducted on two NVIDIA A100 GPUs with 768 GB of system memory. The compute requirements are as follows. For Figure 2, each Atari game requires 72 GPU-hours, for a total of 3,168 GPU-hours across 44 games. Figure 3 requires approximately 60 GPU-hours per subfigure, or 120 GPU-hours in total. For Figure 4, the scaling experiments require 32 GPU-hours for Go, 48 GPU-hours for Atari, and 40 GPU-hours for LLM planning. For Figure 5, the depth-distribution experiments require 48 GPU-hours for Go and 72 GPU-hours for Atari. The experiments for Table 1 require 200 GPU-hours.

Moreover, we use the `b6c96-s175395328-d26788732` model from the KataGo repository [50], the MuZero models released with the framework, and the `TheBloke/LLaMA-Pro-8B-Instruct-GPTQ` model for the LLM planning task with `num_step=8`. In Atari, we exclude hard-exploration games and games with episode lengths greater than 10,000: `asterix`, `battle_zone`, `berzerk`, `boxing`, `double_dunk`, `fishing_derby`, `montezuma_revenge`, `pitfall`, `robotank`, `solaris`, `tennis`, `venture`, `video_pinball`. In addition, the relative scores in Figure 4 are computed separately within each domain and therefore cannot be compared across domains. For Go, we normalize the Elo rating, whereas for Atari and LLM planning, we normalize the task scores. The KataGo Elo ratings in Figure 5 for 50, 100, 200, 400, 800, and 1,600 simulations are -752.2, -552.1, -263.4, -1.4, 231.2, and 481.1. The corresponding Atari relative scores for 5, 10, 25, 50, 100, 200, 400, 800, and 1,600 simulations are 0.849, 0.819, 1.000, 0.940, 0.569, 0.464, 0.464, 0.000, and 0.020. The median 95% CI half-widths are 36.78 and 0.1840 for Go and Atari, respectively.

A.2 Empirical Scaling of Search Depth



(a) Go under the MuZero-style exploration schedule.

(b) Atari under a modified square-root exploration schedule.

Figure 6: Empirical depth of the optimal path $\hat{H}_{\text{opt}}$ across different simulation counts. Boxes show the empirical distributions, while red curves show the corresponding fitted depth estimates based on the simulation count. In Go, the MuZero-style schedule yields the estimate $a\sqrt{n}/\log n$. In Atari, the modified schedule uses $c(m) = \Theta(\sqrt{m})$, yielding the estimate $b \log n$.

To further demonstrate the asymptotic depth estimate beyond Atari, Figure 6a shows the empirical scaling of the optimal-path depth in Go. Since the exploration coefficient in this Go experiment follows the MuZero-style schedule in Equation (8), the fitted curve follows the estimate $a\sqrt{n}/\log n$, with a fitted coefficient $a = 3.448$, yielding $R^2 = 0.8645$. Moreover, to reduce the impact of endgame states, whose search depth may be capped by episode termination, we repeat the analysis after excluding states near the end of each game. The R^2 fitting score increases from 0.8645 without filtering to 0.8776, 0.8886, and 0.8921 when excluding the final 10%, 20%, and 30% of states, respectively, and remains approximately unchanged for larger filtering ratios.

Moreover, Figure 6b evaluates an alternative exploration schedule in Atari. Based on Equation (8), we modify the exploration coefficient as

$$c_{\mathrm{puct}}(m) = \sqrt{\left(\frac{m + \texttt{pb_c_base} + 1}{\texttt{pb_c_base}}\right)} + \texttt{pb_c_init}, \tag{14}$$

to obtain $c(m) = \Theta(\sqrt{m})$. In this case, Corollary 2.1 gives the depth growth of order $\Theta(\log n)$, in contrast to the $\Theta(\sqrt{n}/\log n)$ scaling under the MuZero schedule shown in Figure 2. Across individual Atari games, the fitted curves achieve R^2 scores between 0.85 and 0.97, indicating that the modified coefficient schedule also yields the predicted asymptotic depth growth.

A.3 Interaction of Parallelism and Exploration Coefficient

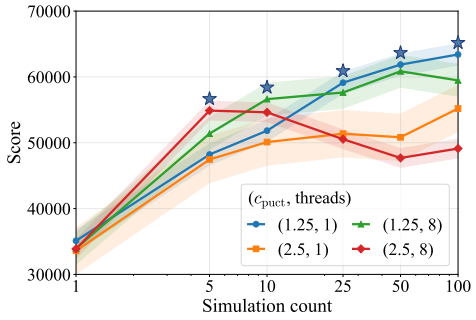


Figure 7: Atari (Kung-Fu Master) score across simulation counts for different combinations of thread count and exploration coefficient c_{puct}. Stars indicate the best-performing configuration at each simulation count.

Figure 7 further shows the joint effect of c_{puct} and thread count across different simulation counts. Across all four settings, performance first improves as the number of simulations increases and, in most cases, drops after reaching its peak, aligning with the Atari scaling trend in Figure 4. This result also reflects the interaction between parallelism and exploration discussed in Table 1. When c_{puct} is fixed, using fewer threads typically leads to better performance, consistent with the observation that increasing the number of threads pushes the search toward stronger exploration. In particular, at 25, 50, and 100 simulations, the blue curve is higher than the green curve, and the orange curve is higher than the red curve. Meanwhile, with more threads, reducing c_{puct} decreases the amount of exploration caused by parallelism and leads to better performance, as shown by the green curve outperforming the orange curve. Overall, these results indicate that thread count and c_{puct} can be tuned jointly to maintain a better exploration-exploitation balance under multithreaded settings.

A.4 Sensitivity to Exploration Coefficient

Figure 8 extends Figure 4 to a wider range of c_{puct}. Similar to the observation in Section 5.4, the overall scaling follows a similar trend across different exploration coefficients in each domain. In particular, in Go, pure exploitation ($c_{\mathrm{puct}} = 0$) performs much worse than settings with even a small amount of exploration, showing that exploration is important for Go.

In Atari, the performance is less sensitive to the exploration coefficient with a small number of simulations. However, with a larger number of simulations, pure exploitation can sometimes perform better than settings with exploration. We attribute this to the search relying on a learned dynamics model rather than the true environment dynamics. Broader exploration tends to lead the search into states and actions that the model has learned less accurately, where prediction errors may accumulate along trajectories, while pure exploitation keeps the search on trajectories that the model is more likely to have learned well.

A.5 Distribution of Nodes over Depth across Simulations

Figure 9 extends Figure 3b by examining the effect of parallelism on the depth distribution across different simulation counts. As the simulation count increases, the distributions shift toward greater

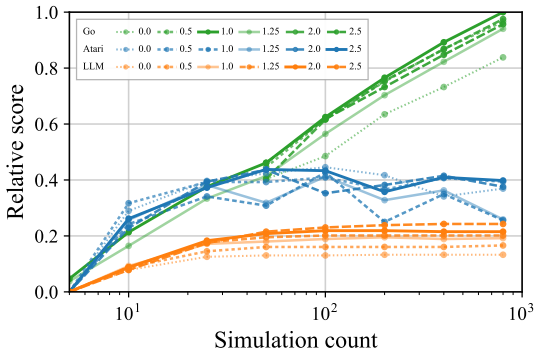
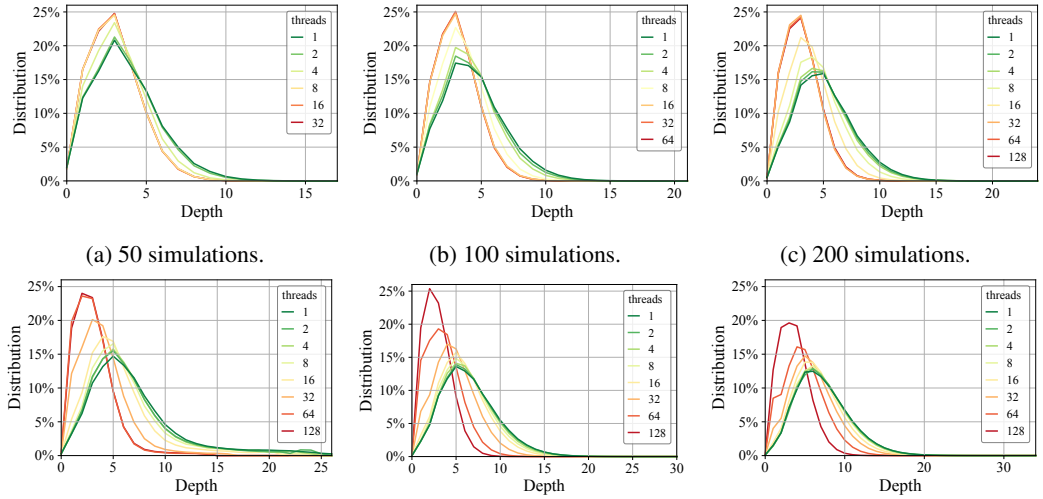


Figure 8: Performance scaling across Go, Atari, and LLM planning under different exploration coefficients c_{puct}. The overall scaling trends remain similar, while the best coefficient varies across domains and simulation counts.



(a) 50 simulations. (b) 100 simulations. (c) 200 simulations.

(d) 400 simulations. (e) 800 simulations. (f) 1600 simulations.

Figure 9: Distribution of nodes over search depth in Go under different thread counts and simulation counts, ranging from 50 to 1,600 simulations. The distributions remain largely similar for thread counts below $\sqrt{n}$, while larger thread counts shift the search toward shallower depths.

depths. However, the effect of parallelism depends on the thread count relative to the transition scale $\sqrt{n}$. When the number of threads is relatively small, the distributions largely overlap, indicating that parallelism has a limited effect on the search-tree structure. In contrast, as the thread count approaches or exceeds $\sqrt{n}$, the distributions become increasingly concentrated at shallower depths.

B Proof

We begin by establishing several preparatory lemmas.

Lemma 1. *For a state s, every child with $P(s,a) > 0$ is selected infinitely often as $N(s) \to \infty$.*

Proof. Suppose a is selected only finitely many times. Then, after some point, $N(s,a) = n_0$ becomes a constant when $m := N(s) \to \infty$. Its pUCT score is

$$\underbrace{\hat{Q}(s,a)}_{O(1)} + \underbrace{cP(s,a)\frac{\sqrt{m}}{1+n_0}}_{\Theta(\sqrt{m})\to\infty}. \tag{15}$$

Let S be the set of children selected only finitely often. For each child $b \in S$ with positive prior, its pUCT score divided by $\sqrt{m}$ converges to a positive constant $cP(s,b)/(1+n_b) > 0$, while for every child outside S, the normalized score converges to zero. Since there are finitely many children, pUCT must eventually select a child in S, contradicting that no child in S is selected after its last selection. Thus, every child with $P(s,a) > 0$ is selected infinitely often. □

Lemma 2. *For a state s, every child with $P(s,a) < 1$ is infinitely often not selected as $N(s) \to \infty$.*

Proof. Since $P(s,a) < 1$, there exists a child $b \neq a$ with $P(s,b) > 0$. By Lemma 1, b is selected infinitely often as $N(s) \to \infty$. Hence, a is infinitely often not selected. □

Remark 8. Lemmas 1 and 2 formalize the intuitive exploration property that actions with positive prior are explored indefinitely, while no single child can permanently dominate the selection process. We state this explicitly since it will be used in the subsequent proofs.

Lemma 3. *For a state s, let $m = N(s)$ and $n_* = N(s,a^*)$. If $m - n_* = O(\sqrt{m})$, then as $m \to \infty$:*

$$cP(s,a^*)\frac{\sqrt{m}}{1+n_*} = o(1). \tag{16}$$

Proof. Since $m - n_* = O\left(\sqrt{m}\right)$, there exists some constant C such that $m - n_* \leq C\sqrt{m}$; namely, $m - C\sqrt{m} \leq n_*$. Therefore,

$$\frac{\sqrt{m}}{1+n_*} \leq \frac{\sqrt{m}}{1+m-C\sqrt{m}} = O(m^{-1/2}) = o(1). \tag{17}$$

□

Remark 9. Lemma 3 implies that, under the same assumption, if the value estimate $\hat{Q}(s,a^*)$ converges to $Q(s,a^*)$, then the pUCT score for a^* converges to the limiting value $Q(s,a^*)$:

$$\hat{Q}(s,a^*) + cP(s,a^*)\frac{\sqrt{m}}{1+n_*} \to Q(s,a^*). \tag{18}$$

The condition $m - n_* = O(\sqrt{m})$ will be established in the following theorem. Throughout the following proofs, we assume the consistency property of MCTS value estimates, where $\hat{Q}(s,a)$ converges to $Q(s,a)$ uniformly over states and actions as the parent visit count $N(s) \to \infty$. All asymptotic statements are taken along subsequences for which the relevant state visit counts tend to infinity.

B.1 Exploration Coefficient

Theorem 1 (Constant exploration coefficient c)**.** *Consider an MCTS search tree of n simulations with a constant exploration coefficient $c > 0$, and suppose Assumption 1 holds. Then, as $n \to \infty$,*

(1) For each state s with $N(s) \to \infty$, and every suboptimal action $a \neq a^*(s)$ with $P(s,a) > 0$, the visit count satisfies $N(s,a) = \Theta(\sqrt{N(s)})$.

(2) The maximum depth reached along the optimal path satisfies $H_{\text{opt}} = \Theta(\sqrt{n})$ until capped by a finite terminal depth.

(3) The maximum depth of the search tree satisfies $H_{\max} = O(\sqrt{n})$.

Proof. Part (1). We first establish the upper bound and then the lower bound. By the convergence properties of MCTS, there exist $\varepsilon \in (0, \Delta_{\min}/4)$ and an integer $m_0 \geq 1$ such that, for every state s and action a, whenever $N(s) \geq m_0$,

$$|\hat{Q}(s,a) - Q(s,a)| \leq \varepsilon. \tag{19}$$

(a) (Upper bound)

If a suboptimal child a_i is selected when s has $m = N(s) \geq m_0$ visits, then

$$\hat{Q}_i + cP_i \frac{\sqrt{m}}{1+n_i} \geq \hat{Q}_* + cP_* \frac{\sqrt{m}}{1+n_*} \geq \hat{Q}_*, \tag{20}$$

where $P_i = P(s,a_i), n_i = N(s,a_i), \hat{Q}_i = \hat{Q}(s,a_i)$, $\hat{Q}_* = \hat{Q}(s,a^*)$, and $Q_* = Q(s,a^*)$. Since $\hat{Q}_i \leq Q_i + \varepsilon$ and $\hat{Q}_* \geq Q_* - \varepsilon$ by Equation (19), we have

$$Q_i + \varepsilon + cP_i \frac{\sqrt{m}}{1+n_i} \geq Q_* - \varepsilon. \tag{21}$$

Let $\Delta_i = \Delta(s,a_i)$. Rearranging yields

$$cP_i \frac{\sqrt{m}}{1+n_i} \geq \Delta_i - 2\varepsilon. \tag{22}$$

Solving for n_i gives

$$n_i \leq \frac{cP_i}{\Delta_i - 2\varepsilon}\sqrt{m} - 1. \tag{23}$$

If a_i was last selected when $N(s) \geq m_0$, its current count is one more than the count bounded in Equation (23). The parent count at that selection was at most m. Otherwise, $n_i \leq m_0$. Since $P_i \leq 1$ and $\Delta_i - 2\varepsilon > \Delta_{\min}/2$, for $m \geq 1$ we obtain

$$n_i \leq \alpha\sqrt{m}, \tag{24}$$

where $\alpha = \max\{m_0, 2c/\Delta_{\min}\} > 0$ is independent of the state and action. Thus, $n_i = O(\sqrt{m})$.

On the other hand, summing over suboptimal actions in Equation (24) yields

$$\sum_{i \neq *} n_i \leq C\sqrt{m}, \tag{25}$$

where the constant $C = (b-1)\alpha$. Thus, the visits to the optimal action satisfy

$$n_* = m - \sum_{i \neq *} n_i \geq m - C\sqrt{m}, \tag{26}$$

which implies

$$m - n_* = O(\sqrt{m}). \tag{27}$$

(b) (Lower bound)

Fix a large m, and let $\tilde{m} \leq m$ be the largest visit count such that a^* is selected when s has $\tilde{m}$ visits. Such a selection exists for sufficiently large m by Lemma 1 if $P(s,a^*) > 0$, and by Equation (26) otherwise. By Equation (26), among the first m visits to s, the action a^* is selected at least $m - C\sqrt{m}$ times. Since $\tilde{m}$ counts visits before the last such selection,

$$\tilde{m} \geq n_* - 1 \geq m - C\sqrt{m} - 1. \tag{28}$$

In particular, for all sufficiently large m, we have $\tilde{m} \geq m/2$.

Let a_i be a suboptimal action with positive prior. At the visit $\tilde{m}$, since a^* is selected, its score is at least that of a_i, yielding

$$\hat{Q}_i + cP_i \frac{\sqrt{\tilde{m}}}{1+n_i} \leq \hat{Q}_* + cP_* \frac{\sqrt{\tilde{m}}}{1+n_*}, \tag{29}$$

where $P_* = P(s,a^*)$. Since Equation (27) satisfies the assumption in Lemma 3,

$$\frac{\sqrt{\tilde{m}}}{1+n_*} = o(1). \tag{30}$$

Thus, for all sufficiently large m,

$$cP_* \frac{\sqrt{\tilde{m}}}{1+n_*} \leq \varepsilon. \tag{31}$$

Moreover, since $\hat{Q}_i \geq Q_i - \varepsilon$ and $\hat{Q}_* \leq Q_* + \varepsilon$ by Equation (19), we obtain

$$Q_i - \varepsilon + cP_i \frac{\sqrt{\tilde{m}}}{1+n_i} \leq Q_* + 2\varepsilon. \tag{32}$$

Rearranging yields

$$cP_i \frac{\sqrt{\tilde{m}}}{1+n_i} \leq \Delta_i + 3\varepsilon. \tag{33}$$

Solving for n_i provides

$$n_i(\tilde{m}) \geq \frac{cP_i}{\Delta_i + 3\varepsilon}\sqrt{\tilde{m}} - 1, \tag{34}$$

which implies

$$n_i(\tilde{m}) \geq \beta\sqrt{\tilde{m}} - 1 \tag{35}$$

for some $\beta > 0$ such that $\frac{cP_i}{\Delta_i+3\varepsilon} \geq \beta$. Since $n_i(\cdot)$ is nondecreasing and $\tilde{m} \geq m/2$ for large m,

$$n_i(m) \geq n_i(\tilde{m}) \geq \beta\sqrt{\tilde{m}} - 1 \geq \frac{\beta}{\sqrt{2}}\sqrt{m} - 1. \tag{36}$$

Hence, $n_i(m) = \Omega(\sqrt{m})$.

Together, these bounds establish that $N(s,a) = \Theta(\sqrt{N(s)})$ for all suboptimal actions with positive prior.

Part (2). In the following, we first bound the difference in visit counts between consecutive nodes, then prove the asymptotic upper and lower bounds. Let s_x be the node on the optimal path at depth x and let $N_x := N(s_x)$, with $N_0 = n$.

Continuing from Equation (36), Assumption 1 gives a uniform choice of $\beta > 0$ for a suboptimal action at each nonterminal state. Since $N_x - N_{x+1} \geq \sum_{i \neq *} n_i$, for sufficiently large N_x we have

$$N_x - N_{x+1} \geq \frac{\beta}{\sqrt{2}}\sqrt{N_x} - 1. \tag{37}$$

For sufficiently large N_x, we have $\frac{\beta}{\sqrt{2}}\sqrt{N_x} - 1 \geq \frac{\beta}{2\sqrt{2}}\sqrt{N_x}$. So,

$$\frac{N_x - N_{x+1}}{\sqrt{N_x}} \geq \frac{\beta}{2\sqrt{2}}. \tag{38}$$

Similarly, Equation (25) yields

$$C_3\sqrt{N_x} \geq 1 + \sum_{i \neq *} n_i = N_x - N_{x+1}, \tag{39}$$

where $C_3 = C + 1$. This also implies

$$N_x - N_{x+1} = O(\sqrt{N_x}). \tag{40}$$

To estimate the depth, we sum only over the levels where the bounds on $N_x - N_{x+1}$ apply, then bound the remaining depth separately. Choose a fixed threshold $M > C^2$, independent of n, large enough that Equations (38) and (39) hold and $N_{x+1} \geq N_x/2$ whenever $N_x \geq M$.

Let H be the first depth with $N_H < M$. Then $N_x \geq M$ for $x < H$, so these inequalities apply throughout the sum. Before a terminal node is reached, the optimal action at the last visited node $s_{H_{\text{opt}}}$ has not been selected. Setting $n_* = 0$ in Equation (26) gives $N_{H_{\text{opt}}} \leq C^2 < M$, which ensures that H exists.

The depth H may grow with n, but the number of levels after it is bounded. Each further edge reduces the visit count by at least one, so

$$0 \leq H_{\text{opt}} - H \leq N_H < M. \tag{41}$$

Since M is fixed, $H_{\text{opt}} = H + O(1)$. Thus, the asymptotic bounds proved for H also hold for H_{opt}.

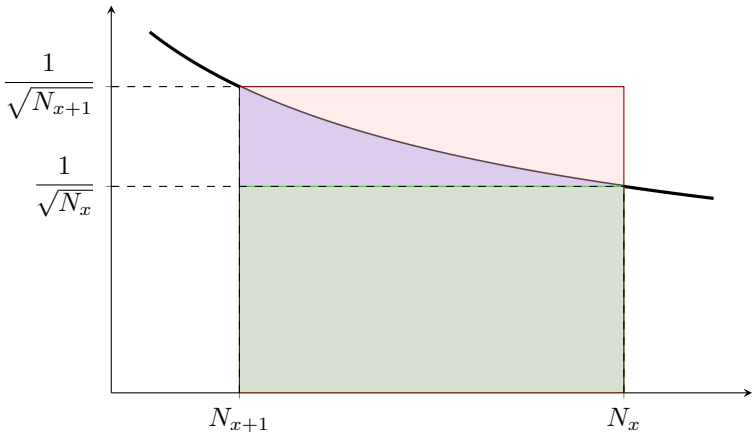


Figure 10: Two rectangles that bound the area of $f(y) = \frac{1}{\sqrt{y}}$ on $[N_{x+1}, N_x]$. The green area is $\frac{N_x - N_{x+1}}{\sqrt{N_x}}$, corresponding to the left-hand side of Equation (43), and the area under the curve corresponds to the integral $\int_{N_{x+1}}^{N_x} \frac{dy}{\sqrt{y}}$. The area of the red rectangle is $\frac{N_x - N_{x+1}}{\sqrt{N_{x+1}}}$, corresponding to the right-hand side of Equation (45).

(a) (Upper bound)

Then, summing over the optimal path from Equation (38) yields

$$\frac{\beta}{2\sqrt{2}} H = \sum_{x=0}^{H-1} \frac{\beta}{2\sqrt{2}} \leq \sum_{x=0}^{H-1} \frac{N_x - N_{x+1}}{\sqrt{N_x}}. \tag{42}$$

Moreover, since $f(y) = \frac{1}{\sqrt{y}}$ is strictly decreasing for all $y > 0$, as illustrated in Figure 10, the area of a rectangle with width $(N_x - N_{x+1})$ and height determined by the lowest boundary $\frac{1}{\sqrt{N_x}}$ is strictly less than the Riemann integral of the curve over the interval, yielding

$$\frac{N_x - N_{x+1}}{\sqrt{N_x}} \leq \int_{N_{x+1}}^{N_x} \frac{dy}{\sqrt{y}}. \tag{43}$$

Summing these integrals along the path gives

$$\frac{\beta}{2\sqrt{2}} H \leq \sum_{x=0}^{H-1} \frac{N_x - N_{x+1}}{\sqrt{N_x}} \leq \sum_{x=0}^{H-1} \int_{N_{x+1}}^{N_x} \frac{dy}{\sqrt{y}} = \int_{N_H}^{n} \frac{dy}{\sqrt{y}} \leq 2\sqrt{n}. \tag{44}$$

Thus, $H_{\text{opt}} \leq \frac{4\sqrt{2n}}{\beta} + O(1)$, implying that $H_{\text{opt}} = O(\sqrt{n})$.

(b) (Lower bound)

Since $N_{x+1} \geq N_x/2 > 0$ for every $x < H$, the maximum height of $\frac{1}{\sqrt{y}}$ on $[N_{x+1}, N_x]$ then occurs at the left endpoint, N_{x+1}. Therefore,

$$\int_{N_{x+1}}^{N_x} \frac{dy}{\sqrt{y}} \leq \frac{N_x - N_{x+1}}{\sqrt{N_{x+1}}} = \frac{N_x - N_{x+1}}{\sqrt{N_x}} \frac{\sqrt{N_x}}{\sqrt{N_{x+1}}} \leq C_3 \frac{\sqrt{N_x}}{\sqrt{N_{x+1}}}. \tag{45}$$

Hence, $\sqrt{N_x/N_{x+1}} \leq \sqrt{2}$ for $x < H$. Summing over the path yields

$$\sum_{x=0}^{H-1} \int_{N_{x+1}}^{N_x} \frac{dy}{\sqrt{y}} = \int_{N_H}^{n} \frac{dy}{\sqrt{y}} \leq \sum_{x=0}^{H-1} C_3\sqrt{2} = C_3\sqrt{2}H. \tag{46}$$

Rearranging gives

$$H_{\text{opt}} \geq \frac{\sqrt{2}}{C_3}(\sqrt{n} - \sqrt{N_H}), \tag{47}$$

which implies $H_{\text{opt}} = \Omega(\sqrt{n})$.

Together, these bounds establish that $H_{\text{opt}} = \Theta(\sqrt{n})$.

If the optimal path terminates at a fixed depth L, then $H_{\text{opt}}(n) = L$ for all sufficiently large n. In fact, Equation (26) shows that the optimal child receives a number of visits tending to infinity whenever its parent does. Starting from the root, whose visit count tends to infinity, and applying this observation successively along the finite optimal path shows that the terminal node is eventually reached. Since the path cannot extend beyond depth L, its reached depth then equals L.

Part (3).

Consider any root-to-leaf path, and let M_x denote the visit count at depth x. At each transition, the edge taken is either optimal or suboptimal:

(a) (Optimal edge)

By Equation (38), we have

$$M_x - M_{x+1} \geq C\sqrt{M_x} \tag{48}$$

for some constant $C > 0$ for all sufficiently large M_x.

(b) (Suboptimal edge)

Equation (24) shows $M_{x+1} \leq \alpha\sqrt{M_x}$ for some constant $\alpha > 0$. Moreover, for sufficiently large M_x, $M_x - \alpha\sqrt{M_x} \geq \sqrt{M_x}$, and thus,

$$M_x - M_{x+1} \geq M_x - \alpha\sqrt{M_x} \geq \sqrt{M_x}. \tag{49}$$

In both cases, for all sufficiently large M_x, there exists a constant $\lambda > 0$ such that

$$M_{x+1} \leq M_x - \lambda\sqrt{M_x}. \tag{50}$$

By rearranging and applying the same Riemann integral argument used previously,

$$\lambda \leq \frac{M_x - M_{x+1}}{\sqrt{M_x}} \leq \int_{M_{x+1}}^{M_x} \frac{dy}{\sqrt{y}}. \tag{51}$$

Summing over the path above a fixed count threshold yields

$$\lambda H_{\max} \leq \sum_{x=0}^{H_{\max}-1} \int_{M_{x+1}}^{M_x} \frac{dy}{\sqrt{y}} + O(1) \leq \int_0^n \frac{dy}{\sqrt{y}} + O(1) = 2\sqrt{n} + O(1). \tag{52}$$

Rearranging gives $H_{\max} \leq \frac{2}{\lambda}\sqrt{n} + O(1)$, which implies $H_{\max} = O(\sqrt{n})$.

□

Corollary 1.1. *Under the assumptions of Theorem 1, consider a path $s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots \xrightarrow{a_{d-1}} s_d$ formed by consecutively selecting suboptimal actions. Suppose that the priors are uniformly small along the path, satisfying $c \cdot P(s_i, a_i) < \Delta_{\min}/4$ for $i \in \{0, \ldots, d-1\}$. Then, the path length satisfies $d = O(\log \log n)$.*

Proof. Let $N_x := N(s_x) = \sum_b N(s_x, b)$ denote the visit count of s_x, as defined after Equation (1), so that $N_0 \leq n$. Since the visit that expands s_{x+1} selects no action, $N_{x+1} \leq N(s_x, a_x) - 1 \leq N_x - 1$ along the path. In particular, the endpoint may have $N_d = 0$.

By the proof of Theorem 1 Part (1), there exist $\varepsilon \in (0, \Delta_{\min}/4)$ and a fixed threshold $m_0 \geq 1$ for the value estimates. By the small-prior assumption, the coefficient in Equation (23) satisfies, for every $x < d$,

$$0 \leq \frac{c\,P(s_x, a_x)}{\Delta(s_x, a_x) - 2\varepsilon} < \frac{\Delta_{\min}/4}{\Delta_{\min}/2} = \frac{1}{2}. \tag{53}$$

Let $m_1 := 4m_0^2$ be the fixed threshold independent of the path and simulation count. If $N_x \geq m_1$, then $\frac{1}{2}\sqrt{N_x} \geq \frac{1}{2}\sqrt{4m_0^2} = m_0$. In this case, Equation (23) and the proof of Theorem 1 Part (1) give

$$N_{x+1} \leq N(s_x, a_x) \leq \tfrac{1}{2}\sqrt{N_x}. \tag{54}$$

For any depth $x \geq 1$ with $N_x \geq m_1$, all preceding counts are also at least m_1. Iterating the preceding inequality gives

$$m_1 \leq N_x \leq \left(\frac{1}{2}\right)^{1+\frac{1}{2}+\cdots+\frac{1}{2^{x-1}}} N_0^{2^{-x}} \leq n^{2^{-x}}. \tag{55}$$

Once the visit count falls below m_1, at most m_1 further edges remain, since the count decreases by at least one along each edge. Since $m_1 > 1$, taking logarithms twice and adding these edges and the edge that crosses the threshold gives, for sufficiently large n,

$$d \leq \log_2 \left(\frac{\log n}{\log m_1}\right) + m_1 + 1 = O(\log \log n). \tag{56}$$

□

Theorem 2 (General exploration coefficient $c(\cdot)$)**.** *Consider an MCTS search tree of n simulations satisfying Assumption 1. Suppose that the exploration coefficient function $c(\cdot)$ of the parent visit count is positive and continuous, and $c(m)\sqrt{m}$ is eventually nondecreasing. If $c(m) = O(\sqrt{m})$, the maximum depth along the optimal path satisfies*

$$H_{\text{opt}}(n) = \Theta\left(\int_1^n \frac{1}{c(m)\sqrt{m}}\, dm\right). \tag{3}$$

If $c(m) = \Omega(\sqrt{m})$, then $H_{\text{opt}}(n) = \Theta(\log n)$. Both hold until capped by a finite terminal depth.

Proof. Let $m = N(s)$, $n_* = N(s, a^*)$, $D = m - n_*$, and $F(m) = c(m)\sqrt{m}$. We first prove the integral comparison without imposing $c(m) = O(\sqrt{m})$.

By the convergence properties of MCTS, the value estimates have the correct ordering at sufficiently large parent counts. Thus, for a suboptimal selection, we have

$$P(s, a^*)\big(1 + N(s, a)\big) \leq P(s, a)\big(1 + N(s, a^*)\big). \tag{57}$$

Applying this at each suboptimal action's last selection gives $N(s, a) \leq \frac{P(s,a)}{P(s,a^*)}(1 + n_*) + O(1)$, where the constant accounts for selections before convergence. Summing over actions and using $\sum_a P(s, a) = 1$ gives

$$m = n_* + \sum_{a \neq a^*} N(s, a) \leq n_* + \frac{1 - P(s, a^*)}{P(s, a^*)}(1 + n_*) + O(1) = \frac{n_*}{P(s, a^*)} + O(1). \tag{58}$$

Since $P(s, a^*)$ is uniformly bounded away from zero, the constant term remains uniform and the last bound gives $n_* \geq \eta m$ for some uniform $\eta > 0$ and all sufficiently large m.

Let $g(m) = \min\{F(m), m\}$. This function is eventually nondecreasing and bounded below by a positive constant. The upper-bound argument of Theorem 1 gives $D \leq K_0 + AF(m)$ for uniform constants K_0, A, while $D \leq m$. Hence, $D \leq Bg(m)$ for a uniform $B > 0$ and all sufficiently large m.

For the lower bound, fix $\varepsilon > 0$ and suppose that the optimal action is selected at a sufficiently large parent count u. Choose a suboptimal action with prior at least $p_0 > 0$. Its score comparison gives

$$\frac{p_0 F(u)}{1 + n_i(u)} \leq \Delta_{\max} + 2\varepsilon + \frac{F(u)}{1 + n_*(u)} \leq \Delta_{\max} + 2\varepsilon + \frac{F(u)}{\eta u}. \tag{59}$$

Consequently, for a uniform $\kappa > 0$ and all sufficiently large u,

$$1 + D(u) \geq 1 + n_i(u) \geq \kappa g(u). \tag{60}$$

We transfer these bounds to the integral along each optimal edge. Write $m = N_x$, $v = N_{x+1} = n_*(m) - 1$, and $f(y) = g(y - 1)$. At the last optimal selection before count m, the parent count u satisfies $n_*(u) = v$ and $u + 1 \leq m$. By monotonicity,

$$\int_v^m \frac{dy}{f(y)} \geq \int_v^{u+1} \frac{dy}{g(u)} = \frac{1 + D(u)}{g(u)} \geq \kappa. \tag{61}$$

For the upper bound, take $k = \lfloor y - 1 \rfloor$ for $v \le y \le m$. Since $n_*(k) \le n_*(m) = v + 1$, the bound on $D(k)$ gives

$$y - v \le D(k) + 3 \le B' f(y) \tag{62}$$

for a uniform $B' > 0$. Also, $f(y) \ge f(v)$. Integrating the bounds $1/f(y) \le 1/f(v)$ and $1/f(y) \le B'/(y - v)$ for $y > v$, splitting at $v + B'f(v)$ if this lies below m, yields

$$\int_v^m \frac{dy}{f(y)} \le B' \left(1 + \log \frac{f(m)}{f(v)}\right), \tag{63}$$

where we used $m - v \le B' f(m)$.

To sum these inequalities, choose a sufficiently large fixed threshold M and let H be the first depth with $N_H < M$. As in Theorem 1, this depth exists before a terminal node is reached, and $H_{\text{opt}} = H + O(1)$. The bound $n_* \ge \eta m$ gives $N_{x+1} \ge \theta N_x$ above the threshold for some fixed $0 < \theta < 1$. In particular, $N_H \ge \theta M$ when $n \ge M$, so M can be chosen large enough that all preceding estimates apply for $x < H$. Summing gives

$$\kappa H \le \int_{N_H}^n \frac{dy}{f(y)} \le B' H + B' \log \frac{f(n)}{f(N_H)} = O(H + \log n). \tag{64}$$

Since $N_H \ge \theta^H n$ and $N_H < M$, we have $\log n = O(H)$. Thus, the integral is $\Theta(H)$. Shifting the integration variable by one and replacing the bounded lower endpoint by 1 contribute only $O(1)$, giving

$$H_{\text{opt}}(n) = \Theta\left(\int_1^n \frac{dm}{g(m)}\right) = \Theta\left(\log n + \int_1^n \frac{dm}{c(m)\sqrt{m}}\right). \tag{65}$$

The second equality follows because $1/g(m) = \max\{1/F(m), 1/m\}$ is within a factor of two of $1/F(m) + 1/m$. If $c(m) = O(\sqrt{m})$, the exploration integral dominates the logarithmic contribution, proving the first case. If $c(m) = \Omega(\sqrt{m})$, it is $O(\log n)$, proving the second case. For a fixed terminal depth L, the bound $n_* \ge \eta m$ implies that the optimal-child count tends to infinity whenever its parent count does. Applying this successively along the finite optimal path shows that $H_{\text{opt}}(n) = L$ for all sufficiently large n. □

Corollary 2.1. *Under the first case of Theorem 2, suppose further that $c(m) = m^\alpha L(m)$, where L is slowly varying. Then, $\alpha \le \frac{1}{2}$ since $c(m) = O(\sqrt{m})$. Moreover,*

(1) If $\alpha < \frac{1}{2}$,

$$H_{\text{opt}}(n) = \Theta\left(\frac{\sqrt{n}}{c(n)}\right). \tag{5}$$

(2) If $\alpha = \frac{1}{2}$,

$$H_{\text{opt}}(n) = \Theta\left(\int_1^n \frac{1}{m\, L(m)}\, dm\right). \tag{6}$$

In both cases, the estimate holds until capped by a finite terminal depth.

Proof. The condition $c(m) = O(\sqrt{m})$ in the first case of Theorem 2 gives $m^{\alpha - 1/2} L(m) = O(1)$. Positivity and slow variation of L therefore imply $\alpha \le \frac{1}{2}$.

If $\alpha < \frac{1}{2}$, then $1/L$ is slowly varying, and Karamata's theorem gives

$$\int_1^n \frac{dm}{c(m)\sqrt{m}} = \int_1^n \frac{dm}{m^{\alpha + 1/2} L(m)} = \Theta\left(\frac{n^{1/2 - \alpha}}{L(n)}\right) = \Theta\left(\frac{\sqrt{n}}{c(n)}\right). \tag{66}$$

Applying Theorem 2 proves Part (1).

If $\alpha = \frac{1}{2}$, substitution into Theorem 2 gives

$$H_{\text{opt}}(n) = \Theta\left(\int_1^n \frac{dm}{mL(m)}\right). \tag{67}$$

In particular, $L(m) = \Theta(1)$ yields $H_{\text{opt}}(n) = \Theta(\log n)$. Both cases inherit the finite terminal cap. □

Corollary 2.2. *Under the common assumptions of Theorem 2, suppose further that $c(m) = m^\alpha L(m)$, where L is positive and slowly varying. Then, $\alpha \geq -\frac{1}{2}$, and*

$$H_{\mathrm{opt}}(n) = \begin{cases} \Theta\left(\dfrac{\sqrt{n}}{c(n)}\right), & -\frac{1}{2} \leq \alpha < \frac{1}{2}, \\ \Theta\left(\log n + \displaystyle\int_1^n \frac{dm}{mL(m)}\right), & \alpha = \frac{1}{2}, \\ \Theta(\log n), & \alpha > \frac{1}{2}. \end{cases} \tag{68}$$

All estimates hold until capped by a finite terminal depth.

Proof. Since $c(m)\sqrt{m}$ is eventually nondecreasing,

$$1 \leq \frac{c(2m)\sqrt{2m}}{c(m)\sqrt{m}} = 2^{\alpha+1/2}\frac{L(2m)}{L(m)} \longrightarrow 2^{\alpha+1/2}. \tag{69}$$

Thus, $\alpha \geq -\frac{1}{2}$.

If $\alpha < \frac{1}{2}$, slow variation gives $c(m)/\sqrt{m} = m^{\alpha-1/2}L(m) \to 0$, so the first case of Corollary 2.1 applies. If $\alpha > \frac{1}{2}$, the same ratio tends to infinity, so the logarithmic case of Theorem 2 applies.

For $\alpha = \frac{1}{2}$, substituting $c(m) = \sqrt{m}\, L(m)$ into Equation (65) gives

$$H_{\mathrm{opt}}(n) = \Theta\left(\log n + \int_1^n \frac{dm}{c(m)\sqrt{m}}\right) = \Theta\left(\log n + \int_1^n \frac{dm}{mL(m)}\right). \tag{70}$$

The finite terminal cap is inherited from the same depth estimate. □

For $\alpha > -\frac{1}{2}$, the following corollary extends Corollary 2.2 by removing the continuity requirement on c and the eventual monotonicity requirement on $c(m)\sqrt{m}$.

Corollary 2.3. *Consider an MCTS search tree of n simulations satisfying Assumption 1. Assume also that the optimal action priors are uniformly bounded away from zero at nonterminal states and that the value estimates converge uniformly as in the proof of Theorem 2. Suppose that the exploration coefficient has the form $c(m) = m^\alpha L(m)$, where $\alpha > -\frac{1}{2}$ and L is slowly varying. Assume further that c and $1/c$ are locally bounded on $[1, \infty)$. Then, the maximum depth along the optimal path satisfies*

$$H_{\mathrm{opt}}(n) = \begin{cases} \Theta\left(\dfrac{\sqrt{n}}{c(n)}\right), & -\frac{1}{2} < \alpha < \frac{1}{2}, \\ \Theta\left(\log n + \displaystyle\int_1^n \frac{dm}{mL(m)}\right), & \alpha = \frac{1}{2}, \\ \Theta(\log n), & \alpha > \frac{1}{2}. \end{cases} \tag{71}$$

The integral is understood in the Lebesgue sense. All estimates hold until capped by a finite terminal depth.

Proof. The proof follows the same argument as in Theorem 2, with the following comparisons in place of monotonicity. Let $F(m) = c(m)\sqrt{m} = m^\rho L(m)$, where $\rho = \alpha + \frac{1}{2} > 0$. For any fixed finite $0 < a < b$, the ratio $L(tm)/L(m)$ approaches 1 uniformly for $a \leq t \leq b$. Hence, $F(2m) \geq 2^{\rho/2}F(m)$ for sufficiently large m, and $F(tm) = \Theta(F(m))$ uniformly for $1 \leq t \leq 2$. Iterating these comparisons over successive doublings gives $F(m) \to \infty$ and constants $C \geq 1$ and $M \geq 1$ such that

$$\sup_{M \leq u \leq m} F(u) \leq CF(m). \tag{72}$$

The same uniformity gives $F(u) = \Theta(F(m))$ uniformly over $u \in [\eta m, m]$ for every fixed $\eta \in (0, 1)$ and sufficiently large m.

Let $m = N(s)$, $n_* = N(s, a^*)$, and $D = m - n_*$ at a nonterminal state s. By the same score-comparison argument as in Theorem 2, the positive lower bound on the optimal-action prior gives $n_* \geq \eta m$ for a uniform $\eta > 0$ and sufficiently large m. Applying the upper-bound argument at each

suboptimal action's last selection and using $\sup_{M\le u\le m} F(u) \le CF(m)$ gives $D \le K_0 + K_1 F(m)$, while $D \le m$. The constant K_0 accounts for counts accumulated below the fixed threshold.

For the lower bound, consider the last optimal selection, at parent count u. As in Theorem 2, we have $u \ge \eta m/2$ and $1 + n_*(u) \ge \eta m$ for sufficiently large m. Comparing its score with that of a suboptimal action whose prior is at least $p_0 > 0$ gives

$$1 + D \ge K_2 \frac{F(u)}{1 + F(u)/m} = \Omega\left(\min\{F(m), m\}\right). \tag{73}$$

Since $F(m) \to \infty$, the additive constants and the one expansion visit are absorbed, giving uniformly for sufficiently large N_x,

$$N_x - N_{x+1} = \Theta(g(N_x)), \qquad g(m) := \min\{F(m), m\}. \tag{74}$$

Moreover, the bound $n_* \ge \eta m$ gives $N_{x+1} \ge \eta' N_x$ for a fixed $\eta' > 0$, after accounting for the expansion visit. Applying the comparison for F with η' gives $g(m) = \Theta(g(N_x))$ uniformly on $[N_{x+1}, N_x]$. Therefore,

$$\int_{N_{x+1}}^{N_x} \frac{dm}{g(m)} = \Theta\left(\frac{N_x - N_{x+1}}{g(N_x)}\right) = \Theta(1). \tag{75}$$

Since c is measurable and $1/c$ is locally bounded, these integrals and the contribution below the fixed threshold are finite. By the same argument as in Theorem 2, the remaining depth below that threshold is bounded, and the deepest reached nonterminal node on the optimal path has count below the threshold. Hence, summing along the optimal path, before the terminal cap, yields

$$H_{\mathrm{opt}}(n) = \Theta\left(\int_1^n \frac{dm}{g(m)}\right) = \Theta\left(\log n + \int_1^n \frac{dm}{c(m)\sqrt{m}}\right), \tag{76}$$

since $1/g(m) = \max\{1/F(m), 1/m\}$ is within a factor of two of $1/F(m) + 1/m$.

If $-\frac{1}{2} < \alpha < \frac{1}{2}$, Karamata's theorem gives

$$\int_1^n \frac{dm}{c(m)\sqrt{m}} = \Theta\left(\frac{n^{1/2-\alpha}}{L(n)}\right). \tag{77}$$

Since this quantity dominates $\log n$, the first case follows. If $\alpha > \frac{1}{2}$, slow variation gives $c(m)/\sqrt{m} \to \infty$. Thus, the exploration integral is $O(\log n)$, and $H_{\mathrm{opt}}(n) = \Theta(\log n)$.

If $\alpha = \frac{1}{2}$, substituting $c(m) = \sqrt{m}\, L(m)$ into the depth estimate above gives

$$H_{\mathrm{opt}}(n) = \Theta\left(\log n + \int_1^n \frac{dm}{mL(m)}\right). \tag{78}$$

All estimates hold until capped by a finite terminal depth, by the same argument as in Theorem 2. □

B.2 Parallelism

Lemma 4 (One-step estimates under parallelism)**.** *Let s_x be the node at depth x on the optimal path, let $N_x := N(s_x)$, and let $N_{x+1} := N(s_{x+1})$. For tree-parallel MCTS with T concurrent threads, there exists a constant $C_2 < \infty$ such that, for all sufficiently large N_x,*

$$N_x - N_{x+1} \le C_2\left(\sqrt{N_x} + T\right). \tag{79}$$

Moreover, there exist constants $c_0 > 0$ and $\eta > 0$ such that, whenever $T \le \eta\sqrt{N_x}$ and N_x is sufficiently large,

$$N_x - N_{x+1} \ge c_0\sqrt{N_x}. \tag{80}$$

If, in addition, the contention assumptions in Theorem 4 hold, then there exists a constant $c_1 > 0$ such that, for all sufficiently large N_x and all sufficiently large T,

$$N_x - N_{x+1} \ge c_1 T. \tag{81}$$

Consequently, under the assumptions of Theorem 4, there exist constants $0 < \kappa_1 \le \kappa_2 < \infty$ such that, for all sufficiently large N_x,

$$\kappa_1\left(\sqrt{N_x} + T\right) \le N_x - N_{x+1} \le \kappa_2\left(\sqrt{N_x} + T\right). \tag{82}$$

Proof. Throughout the parallel proofs, we measure the tree after all n simulations have completed their backups. Moreover, at each selection, the visible parent and action counts lag the corresponding issued counts by at most T. Similarly, the value estimates used for selection converge uniformly as the parent visit count seen by the thread grows.

Let $a_i \neq a^*(s_x)$ be a suboptimal action and write $n_i = N(s_x, a_i)$. An action never selected has $n_i = 0$. Otherwise, consider its last issued selection before the final observation, by which time all backups have completed. Let $\bar{m}$ and $\bar{n}_i$ denote the parent and child counts visible immediately before this selection. Choose a common visible-count threshold $m_0 \geq 1$ for value stabilization. If $\bar{m} \geq m_0$, the same pUCT comparison as in Theorem 1, Part (1), gives

$$\bar{n}_i \leq A\sqrt{\bar{m}} \tag{83}$$

for a uniform constant $A < \infty$. The issued child count before this selection is at most $\bar{n}_i + T$, so $n_i \leq \bar{n}_i + T + 1$. If $\bar{m} < m_0$, the issued parent count is at most $\bar{m} + T$, giving $n_i \leq \bar{m} + T + 1 \leq m_0 + T$. Since $\bar{m} \leq N_x$, both cases imply

$$n_i \leq m_0 + A\sqrt{N_x} + T + 1. \tag{84}$$

Summing over the at most $b - 1$ suboptimal actions gives

$$N_x - N_{x+1} = 1 + \sum_{a_i \neq a^*(s_x)} N(s_x, a_i) \leq C_2 \left(\sqrt{N_x} + T\right), \tag{85}$$

where C_2 absorbs the fixed constants since $T \geq 1$. This proves Equation (79). The same calculation gives $1 + N(s) - N(s, a^*) \leq C_2(\sqrt{N(s)} + T)$ at every nonterminal state, even when the optimal child has not been reached.

We next prove the sequential lower part. Suppose $T \leq \eta\sqrt{N_x}$ with $0 < \eta \leq 1$, and choose a suboptimal action a_i whose prior is uniformly bounded below, as guaranteed by Assumption 1. The upper bound gives $n_* = N_x - O(\sqrt{N_x})$. At the last issued optimal selection, the issued parent and optimal child counts are at least $n_* - 1$, so their visible counts are $\Theta(N_x)$ by the lag bound. The optimal exploration bonus is therefore $O(N_x^{-1/2})$. Comparing scores at this selection, as in Theorem 1, Part (1), gives a visible count of $\Omega(\sqrt{N_x})$ for a_i. Since all backups have completed by the final observation,

$$N(s_x, a_i) \geq c_0\sqrt{N_x} \tag{86}$$

for some uniform $c_0 > 0$ and all sufficiently large N_x. Since $N_x - N_{x+1}$ is one plus the sum of visits to suboptimal children, this proves Equation (80).

Finally, assume the contention conditions in Theorem 4. In the stale selection window of Assumption 2, at least ρT selections reach s_x, and at most K of these selections choose $a^*(s_x)$. These selections belong to the n simulations and have completed their backups by the final observation. Hence, at least $\rho T - K$ of these selections choose suboptimal actions and contribute to $N_x - N_{x+1}$. For $T \geq 2K/\rho$,

$$N_x - N_{x+1} \geq \rho T - K \geq \frac{\rho}{2} T. \tag{87}$$

Thus, Equation (81) holds with $c_1 = \rho/2$.

Combining the lower bounds gives Equation (82). If $T \leq \eta\sqrt{N_x}$, then Equation (80) implies $N_x - N_{x+1} = \Omega(\sqrt{N_x} + T)$. If $T > \eta\sqrt{N_x}$ and T is sufficiently large, then Equation (81) implies $N_x - N_{x+1} = \Omega(T) = \Omega(\sqrt{N_x} + T)$. The remaining case has bounded T and bounded N_x, and is excluded by increasing the fixed count threshold. □

Theorem 3 (Constant parallelism)**.** *Consider tree-parallel MCTS with a constant exploration coefficient $c > 0$, a fixed number T of concurrent threads, and n simulations, under Assumption 1. Then, as $n \to \infty$, $H_{\mathrm{opt}}(n, T) = \Theta(\sqrt{n})$, until capped by a finite terminal depth.*

Proof. Let s_x denote the node at depth x on the optimal path and let $N_x := N(s_x)$. Since T is fixed, the staleness error caused by in-flight simulations is $O(1)$.

Since T is fixed, $T \leq \eta\sqrt{N_x}$ for sufficiently large N_x. The upper and lower bounds in Lemma 4 therefore give constants $0 < c_1 < c_2 < \infty$ such that, for all sufficiently large N_x,

$$c_1\sqrt{N_x} \leq N_x - N_{x+1} \leq c_2\sqrt{N_x}. \tag{88}$$

Before the terminal cap is reached, the frontier has optimal-child count zero. The node-count bound in the proof of Lemma 4 therefore gives $N_{H_{\mathrm{opt}}} \le C_2(\sqrt{N_{H_{\mathrm{opt}}}} + T)$, so its count is $O(1)$ for fixed T. Counts decrease by at least one along reached edges; thus, the levels below a sufficiently large fixed threshold contribute only $O(1)$ depth and $O(1)$ integral mass.

We now apply the same Riemann-sum comparison used in Theorem 1. Since $m \mapsto m^{-1/2}$ is decreasing,

$$\frac{N_x - N_{x+1}}{\sqrt{N_x}} \le \int_{N_{x+1}}^{N_x} \frac{dm}{\sqrt{m}}. \tag{89}$$

Using the lower bound in Equation (88) and summing along the optimal path gives

$$c_1 H_{\mathrm{opt}}(n,T) \le \sum_{x=0}^{H_{\mathrm{opt}}-1} \int_{N_{x+1}}^{N_x} \frac{dm}{\sqrt{m}} + O(1) \le \int_1^n \frac{dm}{\sqrt{m}} + O(1) = O(\sqrt{n}). \tag{90}$$

Therefore, $H_{\mathrm{opt}}(n,T) = O(\sqrt{n})$.

For the reverse inequality, the upper bound in Equation (88) implies, again by the same Riemann comparison as in Theorem 1, that each step contributes at most a constant amount to the integral. Hence,

$$\int_1^n \frac{dm}{\sqrt{m}} \le C' H_{\mathrm{opt}}(n,T) + O(1), \tag{91}$$

where $C' = \sqrt{2}\, c_2$. Since the integral is $\Theta(\sqrt{n})$, this gives

$$H_{\mathrm{opt}}(n,T) = \Omega(\sqrt{n}). \tag{92}$$

Combining the two bounds yields

$$H_{\mathrm{opt}}(n,T) = \Theta(\sqrt{n}). \tag{93}$$

For a fixed terminal depth L, the same node-count upper bound gives $N(s,a^*) \ge N(s) - O(\sqrt{N(s)} + T)$. Since T is fixed, the optimal-child count tends to infinity whenever its parent count does. Induction along the finite optimal path gives $H_{\mathrm{opt}}(n,T) = L$ eventually. □

Theorem 4 (General parallelism)**.** *Consider tree-parallel MCTS with a constant exploration coefficient $c > 0$, $T = T(n)$ concurrent threads, and n simulations, under Assumptions 1 and 2. Then, as $n \to \infty$, we have*

$$H_{\mathrm{opt}}(n,T) = \Theta\left(\int_1^n \frac{dm}{\sqrt{m} + T}\right), \tag{10}$$

until capped by a finite terminal depth. In particular,

(1) If $T = o(\sqrt{n})$,

$$H_{\mathrm{opt}}(n,T) = \Theta(\sqrt{n}). \tag{11}$$

(2) If $T \gg \sqrt{n}$,

$$H_{\mathrm{opt}}(n,T) = \Theta(n/T). \tag{12}$$

Proof. If the root is terminal, then $H_{\mathrm{opt}}(n,T) = 0$ and the terminal depth is $L = 0$, so the capped conclusion holds immediately. Thus, we may assume that the root is nonterminal.

Let s_x be the node at depth x on the optimal path, let $N_x := N(s_x)$, and write

$$D_x := N_x - N_{x+1}. \tag{94}$$

By Lemma 4, under the assumptions of the theorem there exist constants $0 < \kappa_1 \le \kappa_2 < \infty$ such that, for all sufficiently large N_x,

$$\kappa_1(\sqrt{N_x} + T) \le D_x \le \kappa_2(\sqrt{N_x} + T). \tag{95}$$

Define

$$I(n,T) := \int_1^n \frac{dm}{\sqrt{m} + T}. \tag{96}$$

Since $T = o(n)$, we have $I(n,T) \geq (n-1)/(\sqrt{n}+T) \to \infty$. Since $m \mapsto (\sqrt{m}+T)^{-1}$ is decreasing, the lower bound in Equation (95) gives

$$\kappa_1 \leq \frac{D_x}{\sqrt{N_x}+T} \leq \int_{N_{x+1}}^{N_x} \frac{dm}{\sqrt{m}+T}. \tag{97}$$

Summing over x along the optimal path yields

$$\kappa_1 H_{\mathrm{opt}}(n,T) \leq \sum_{x=0}^{H_{\mathrm{opt}}-1} \int_{N_{x+1}}^{N_x} \frac{dm}{\sqrt{m}+T} + O(1) \leq I(n,T) + O(1). \tag{98}$$

The additive constants cover the bounded number of levels below the count threshold and the integral below 1, uniformly for $T \geq 1$. Since $I(n,T) \to \infty$, they can be absorbed, giving

$$H_{\mathrm{opt}}(n,T) = O\left(\int_1^n \frac{dm}{\sqrt{m}+T}\right). \tag{99}$$

It remains to prove the matching lower bound. We use the upper estimate in Equation (95). The following elementary bound will be used: for every $u \geq 1$, every $T \geq 1$, and every $0 \leq d \leq u$ satisfying

$$d \leq \kappa_2(\sqrt{u}+T), \tag{100}$$

there exists a constant C_*, depending only on κ_2, such that

$$\int_{u-d}^{u} \frac{dm}{\sqrt{m}+T} \leq C_*. \tag{101}$$

Indeed, substituting $m = z^2$ and using that $z/(z+T)$ is increasing gives

$$\begin{aligned} \int_{u-d}^{u} \frac{dm}{\sqrt{m}+T} &= 2\int_{\sqrt{u-d}}^{\sqrt{u}} \frac{z}{z+T}\,dz \\ &\leq \frac{2\sqrt{u}}{\sqrt{u}+T}\left(\sqrt{u}-\sqrt{u-d}\right) \\ &= \frac{2\sqrt{u}}{\sqrt{u}+\sqrt{u-d}}\frac{d}{\sqrt{u}+T} \leq \frac{2d}{\sqrt{u}+T} \leq 2\kappa_2. \end{aligned} \tag{102}$$

Thus, Equation (101) holds with $C_* = 2\kappa_2$.

Before the terminal cap is reached, the optimal child at the frontier has zero visits. The node-count bound in the proof of Lemma 4 gives $N_{H_{\mathrm{opt}}} \leq C_2(\sqrt{N_{H_{\mathrm{opt}}}}+T)$, hence $N_{H_{\mathrm{opt}}} = O(T)$ since $T \geq 1$. Then, setting $u = \max\{1, N_{H_{\mathrm{opt}}}\}$, the omitted integral satisfies $\int_1^u (\sqrt{m}+T)^{-1}\,dm \leq u/T = O(1)$.

Applying Equation (101) with $u = N_x$ and $d = D_x$, and summing over the optimal path, gives

$$\int_1^n \frac{dm}{\sqrt{m}+T} \leq C_* H_{\mathrm{opt}}(n,T) + O(1). \tag{103}$$

Since $I(n,T) \to \infty$, the additive constant can be absorbed. Therefore,

$$H_{\mathrm{opt}}(n,T) = \Omega\left(\int_1^n \frac{dm}{\sqrt{m}+T}\right). \tag{104}$$

Combining Equations (99) and (104) yields

$$H_{\mathrm{opt}}(n,T) = \Theta\left(\int_1^n \frac{dm}{\sqrt{m}+T}\right), \tag{105}$$

until capped by a finite terminal depth. To justify the cap, suppose the terminal depth is a fixed L and $H_{\mathrm{opt}} < L$. Each reached edge loses at most $C_2(\sqrt{n}+T)$ visits, and the frontier count is $O(T)$. Summing over fewer than L edges gives $n = O(L(\sqrt{n}+T)+T)$, which is impossible for sufficiently large n because $T = o(n)$. Thus, $H_{\mathrm{opt}} = L$ eventually.

It remains to evaluate $I(n,T)$ in the two stated regimes. If $T = o(\sqrt{n})$, then

$$I(n,T) \leq \int_1^n \frac{dm}{\sqrt{m}} = O(\sqrt{n}), \tag{106}$$

while

$$I(n,T) \geq \int_{T^2}^n \frac{dm}{2\sqrt{m}} = \sqrt{n} - O(T) = \Theta(\sqrt{n}). \tag{107}$$

Therefore,

$$H_{\text{opt}}(n,T) = \Theta(\sqrt{n}). \tag{108}$$

If $T \gg \sqrt{n}$, meaning $\sqrt{n} = o(T)$ with $T \leq n$, then

$$\sup_{1 \leq m \leq n} \frac{\sqrt{m}}{T} \leq \frac{\sqrt{n}}{T} = o(1). \tag{109}$$

Thus, uniformly for $1 \leq m \leq n$,

$$\sqrt{m} + T = (1 + o(1))T. \tag{110}$$

Hence,

$$I(n,T) = \int_1^n \frac{dm}{(1+o(1))T} = \Theta\left(\frac{n}{T}\right), \tag{111}$$

and therefore

$$H_{\text{opt}}(n,T) = \Theta\left(\frac{n}{T}\right). \tag{112}$$

□

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: The claims in the abstract and introduction are aligned with the paper's contributions and evaluation results.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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 discuss the limitations of the work in Section 6.

   Guidelines:

   - The answer [N/A] 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: Theoretical assumptions and complete proofs are provided in Section 4 and Appendix B.

Guidelines:

- The answer [N/A] 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: Experimental settings are provided in Section 5.1 and Appendix A.1, and all reported results are averaged over multiple runs.

   Guidelines:

   - The answer [N/A] 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: The Lean proof corresponding to Appendix B, with instructions for running and verifying it, is available at `https://github.com/chengscott/search-tree-scaling`.

Guidelines:

- The answer [N/A] 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: Experimental settings are provided in Section 5.1 and Appendix A.1.

Guidelines:

- The answer [N/A] 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: Statistical significance is reported for each experiment in Section 5.

Guidelines:

- The answer [N/A] 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: The compute resources required for the experiments are described in Appendix A.1.

   Guidelines:

   - The answer [N/A] 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: The research described in the paper conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] 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: [N/A]

    Justification: The paper conducts foundational research.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] 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: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] 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: [Yes]

    Justification: All assets are properly credited and cited in the paper.

    Guidelines:

    - The answer [N/A] 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: [Yes]

    Justification: The `Lean` formalization is released at `https://github.com/chengscott/search-tree-scaling`, with a README describing instructions for running and verifying the proof.

    Guidelines:

    - The answer [N/A] 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: [N/A]

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

    Guidelines:

    - The answer [N/A] 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: [N/A]

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

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: The core method development in this research does not involve LLMs.

    Guidelines:

    - The answer [N/A] 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.