Monte-Carlo Tree Search by Best Arm Identification

Emilie Kaufmann, Wouter M. Koolen

Monte-Carlo Tree Search by Best Arm Identification

Emilie Kaufmann[1,2] and Wouter M. Koolen[3]
CNRS & CRIStAL, UMR 9189, Univ. Lille[1], Inria Lille, SequeL team[2], Centrum Wiskunde & Informatica[3]
emilie.kaufmann@univ-lille1.fr, wmkoolen@cwi.nl

June 9, 2017

Abstract

Recent advances in bandit tools and techniques for sequential learning are steadily enabling new applications and are promising the resolution of a range of challenging related problems. We study the game tree search problem, where the goal is to quickly identify the optimal move in a given game tree by sequentially sampling its stochastic payoffs. We develop new algorithms for trees of arbitrary depth, that operate by summarizing all deeper levels of the tree into confidence intervals at depth one, and applying a best arm identification procedure at the root. We prove new sample complexity guarantees with a refined dependence on the problem instance. We show experimentally that our algorithms outperform existing elimination-based algorithms and match previous special-purpose methods for depth-two trees.

1 Introduction

We consider two-player zero-sum turn-based interactions, in which the sequence of possible successive moves is represented by a maximin game tree $\mathcal{T}$. This tree models the possible actions sequences by a collection of MAX nodes, that correspond to states in the game in which player A should take action, MIN nodes, for states in the game in which player B should take action, and leaves which specify the payoff for player A. The goal is to determine the best action at the root for player A. For deterministic payoffs this search problem is primarily algorithmic, with several powerful pruning strategies available [19]. We look at problems with stochastic payoffs, which in addition present a major statistical challenge.

Sequential identification questions in game trees with stochastic payoffs arise naturally as robust versions of bandit problems. They are also a core component of Monte Carlo tree search (MCTS) approaches for solving intractably large deterministic tree search problems, where an entire sub-tree is represented by a stochastic leaf in which randomized play-out and/or evaluations are performed [4]. A play-out consists in finishing the game with some simple, typically random, policy and observing the outcome for player A.

For example, MCTS is used within the AlphaGo system [20], and the evaluation of a leaf position combines supervised learning and (smart) play-outs. While MCTS algorithms for Go have now reached expert human level, such algorithms remain very costly, in that many (expensive) leaf evaluations or play-outs are necessary to output the next action to be taken by the player. In this paper, we focus on the *sample complexity* of Monte-Carlo Tree Search methods, about which very little is known. For this purpose, we work under a simplified model for MCTS already studied by [21], and that generalizes the depth-two framework of [10].

1.1 A simple model for Monte-Carlo Tree Search

We start by fixing a game tree $\mathcal{T}$, in which the root is a MAX node. Letting $\mathcal{L}$ be the set of leaves of this tree, for each $\ell \in \mathcal{L}$ we introduce a stochastic oracle $\mathcal{O}_\ell$ that represents the leaf evaluation or play-out performed when this leaf is reached by an MCTS algorithm. In this model, we do not try to optimize the evaluation or play-out strategy, but we rather assume that the oracle $\mathcal{O}_\ell$ produces i.i.d. samples from an unknown distribution whose mean μ_ℓ is the value of the position ℓ. To ease the presentation, we focus on binary oracles (indicating the win or loss of a

play-out), in which the oracle $\mathcal{O}_\ell$ is a Bernoulli distribution with unknown mean μ_ℓ (the probability of player A winning the game in the corresponding state). Our algorithms can be used without modification in case the oracle is a distribution bounded in $[0,1]$.

For each node s in the tree, we denote by $\mathcal{C}(s)$ the set of its children and by $\mathcal{P}(s)$ its parent. The root is denoted by s_0. The *value* (for player A) of any node s is recursively defined by $V_\ell = \mu_\ell$ if $\ell \in \mathcal{L}$ and

$$V_s = \begin{cases} \max_{c \in \mathcal{C}(s)} V_c & \text{if s is a MAX node,} \\ \min_{c \in \mathcal{C}(s)} V_c & \text{if s is a MIN node.} \end{cases}$$

The best move is the action at the root with highest value,

$$s^* = \operatorname*{argmax}_{s \in \mathcal{C}(s_0)} V_s.$$

To identify s^* (or an ϵ-close move), an MCTS algorithm sequentially selects paths in the game tree and calls the corresponding leaf oracle. At round t, a leaf $L_t \in \mathcal{L}$ is chosen by this adaptive *sampling rule*, after which a sample $X_t \sim \mathcal{O}_{L_t}$ is collected. We consider here the same PAC learning framework as [21, 10], in which the strategy also requires a *stopping rule*, after which leaves are no longer evaluated, and a *recommendation rule* that outputs upon stopping a guess $\hat{s}_\tau \in \mathcal{C}(s_0)$ for the best move of player A.

Given a risk level δ and some accuracy parameter $\epsilon \geq 0$ our goal is have a recommendation $\hat{s}_\tau \in \mathcal{C}(s_0)$ whose value is within ϵ of the value of the best move, with probability larger than $1-\delta$, that is

$$\mathbb{P}\left(V(s_0) - V(\hat{s}_\tau) \leq \epsilon\right) \geq 1 - \delta.$$

An algorithm satisfying this property is called (ϵ, δ)-correct. The main challenge is *to design (ϵ, δ)-correct algorithms that use as few leaf evaluations τ as possible.*

Related work The model we introduce for Monte-Carlo Tree Search is very reminiscent of a stochastic bandit model. In those, an agent repeatedly selects one out of several probability distributions, called arms, and draws a sample from the chosen distribution. Bandits models have been studied since the 1930s [22], mostly with a focus on *regret minimization*, where the agent aims to maximize the sum of the samples collected, which are viewed as rewards [17]. In the context of MCTS, a sample corresponds to a win or a loss in one play-out, and maximizing the number of successful play-outs (that correspond to simulated games) may be at odds with identifying quickly the next best action to take at the root. In that, our best action identification problem is closer to a so-called *Best Arm Identification* (BAI) problem.

The goal in the standard BAI problem is to find quickly and accurately the arm with highest mean. The BAI problem in the fixed-confidence setting [7] is the special case of our simple model for a tree of depth one. For deeper trees, rather than finding the best arm (i.e. leaf), we are interested in finding the best action at the root. As the best root action is a function of the means of all leaves, this is a more structured problem.

Bandit algorithms, and more recently BAI algorithms have been successfully adapted to tree search. Building on the UCB algorithm [2], a regret minimizing algorithm, variants of the UCT algorithm [16] have been used for MCTS in growing trees, leading to successful AIs for games. However, there are only very weak theoretical guarantees for UCT. Moreover, observing that maximizing the number of successful play-outs is not the target, recent work rather tried to leverage tools from the BAI literature. In [18, 6] Sequential Halving [13] is used for exploring game trees. The latter algorithm is a state-of-the-art algorithm for the fixed-budget BAI problem [1], in which the goal is to identify the best arm with the smallest probability of error based on a given budget of draws. The proposed SHOT (Sequential Halving applied tO Trees) algorithm [6] is compared empirically to the UCT approach of [16], showing improvements in some cases. A hybrid approach mixing SHOT and UCT is also studied [18], still without sample complexity guarantees.

In the fixed-confidence setting, [21] develop the first sample complexity guarantees in the model we consider. The proposed algorithm, FindTopWinner is based on *uniform sampling and eliminations*, an approach that may be related to the Successive Eliminations algorithm [7] for fixed-confidence BAI in bandit models. FindTopWinner proceeds in rounds, in which the leaves that have not been eliminated are sampled repeatedly until the precision

of their estimates doubled. Then the tree is pruned of every node whose estimated value differs significantly from the estimated value of its parent, which leads to the possible elimination of several leaves. For depth-two trees, [10] propose an elimination procedure that is not round-based. In this simpler setting, an algorithm that exploits *confidence intervals* is also developed, inspired by the LUCB algorithm for fixed-confidence BAI [12]. Some variants of the proposed M-LUCB algorithm appear to perform better in simulations than elimination based algorithms. We now investigate this trend further in deeper trees, both in theory and in practice.

Our Contribution. In this paper, we propose a generic architecture, called BAI-MCTS, that builds on a Best Arm Identification (BAI) algorithm and on confidence intervals on the node values in order to solve the best action identification problem in a tree of arbitrary depth. In particular, we study two specific instances, UGapE-MCTS and LUCB-MCTS, that rely on *confidence-based* BAI algorithms [8, 12]. We prove that these are (ϵ, δ)-correct and give a high-probability upper bound on their sample complexity. Both our theoretical and empirical results improve over the elimination-based state-of-the-art algorithm, FindTopWinner [21].

2 BAI-MCTS algorithms

We present a generic class of algorithms, called BAI-MCTS, that combines a BAI algorithm with an exploration of the tree based on confidence intervals on the node values. Before introducing the algorithm and two particular instances, we first explain how to build such confidence intervals, and also introduce the central notion of *representative child* and *representative leaf*.

2.1 Confidence intervals and representative nodes

For each leaf $\ell \in \mathcal{L}$, using the past observations from this leaf we may build a confidence interval

$$\mathcal{I}_\ell(t) = [\mathrm{L}_\ell(t), \mathrm{U}_\ell(t)],$$

where $\mathrm{U}_\ell(t)$ (resp. $\mathrm{L}_\ell(t)$) is an Upper Confidence Bound (resp. a Lower Confidence Bound) on the value $V(\ell) = \mu_\ell$. The specific confidence interval we shall use will be discussed later.

These confidence intervals are then propagated upwards in the tree using the following construction. For each internal node s, we recursively define $\mathcal{I}_s(t) = [\mathrm{L}_s(t), \mathrm{U}_s(t)]$ with $\mathrm{L}_s(t) = \begin{cases} \max_{c \in \mathcal{C}(s)} \mathrm{L}_c(t) & \text{for a MAX node } s, \\ \min_{c \in \mathcal{C}(s)} \mathrm{L}_c(t) & \text{for a MIN node } s, \end{cases}$

$\mathrm{U}_s(t) = \begin{cases} \max_{c \in \mathcal{C}(s)} \mathrm{U}_c(t) & \text{for a MAX node } s, \\ \min_{c \in \mathcal{C}(s)} \mathrm{U}_c(t) & \text{for a MIN node } s. \end{cases}$ Note that these intervals are the tightest possible on the parent under the sole assumption that the child confidence intervals are all valid. A similar construction was used in the OMS algorithm of [3] in a different context. It is easy to convince oneself (or prove by induction, see Appendix A.1) that the accuracy of the confidence intervals is preserved under this construction, as stated below.

Proposition 1. *Let $t \in \mathbb{N}$. One has* $\bigcap_{\ell \in \mathcal{L}} (\mu_\ell \in \mathcal{I}_\ell(t)) \;\Rightarrow\; \bigcap_{s \in \mathcal{T}} (V_s \in \mathcal{I}_s(t))$.

We now define the *representative child* $c_s(t)$ of an internal node s as

$$c_s(t) \;=\; \begin{cases} \operatorname{argmax}_{c \in \mathcal{C}(s)} \mathrm{U}_c(t) & \text{if } s \text{ is a MAX node,} \\ \operatorname{argmin}_{c \in \mathcal{C}(s)} \mathrm{L}_c(t) & \text{if } s \text{ is a MIN node,} \end{cases}$$

and the *representative leaf* $\ell_s(t)$ of a node $s \in \mathcal{T}$, which is the leaf obtained when going down the tree by always selecting the representative child:

$$\ell_s(t) = s \;\text{ if } s \in \mathcal{L}, \qquad \ell_s(t) = \ell_{c_s(t)}(t) \;\text{ otherwise.}$$

The confidence intervals in the tree represent the statistically plausible values in each node, hence the representative child can be interpreted as an "optimistic move" in a MAX node and a "pessimistic move" in a MIN node (assuming we play against the best possible adversary). This is reminiscent of the behavior of the UCT algorithm [16].

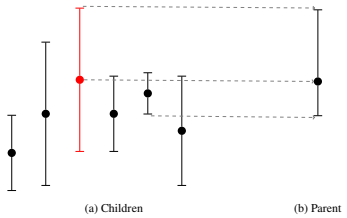

Figure 1: Construction of confidence interval and representative child (in red) for a MAX node.

Input: a BAI algorithm
Initialization: $t = 0$.
while *not* $\mathrm{BAIStop}\left(\{s \in \mathcal{C}(s_0)\}\right)$ **do**
  $R_{t+1} = \mathrm{BAIStep}\left(\{s \in \mathcal{C}(s_0)\}\right)$
  Sample the representative leaf $L_{t+1} = \ell_{R_{t+1}}(t)$
  Update the information about the arms. $t = t + 1$.
end
Output: $\mathrm{BAIReco}\left(\{s \in \mathcal{C}(s_0)\}\right)$

Figure 2: The BAI-MCTS architecture

2.2 The BAI-MCTS architecture

In this section we present the generic BAI-MCTS algorithm, whose sampling rule combines two ingredients: a best arm identification step which selects an action at the root, followed by a *confidence based exploration step*, that goes down the tree starting from this depth-one node in order to select the *representative leaf* for evaluation.

The structure of a BAI-MCTS algorithm is presented in Figure 2. The algorithm depends on a Best Arm Identification (BAI) algorithm, and uses the three components of this algorithm:

- the sampling rule $\mathrm{BAIStep}(\mathcal{S})$ selects an arm in the set $\mathcal{S}$
- the stopping rule $\mathrm{BAIStop}(\mathcal{S})$ returns `True` if the algorithm decides to stop
- the recommendation rule $\mathrm{BAIReco}(\mathcal{S})$ selects an arm as a candidate for the best arm

All the decisions are made based on the information available for the arms in $\mathcal{S}$, and possibly the number of rounds t as well. In BAI-MCTS, the arms are the depth-one nodes, and the information available will typically be summarized by their confidence intervals $\mathcal{I}_s(t)$.

2.3 UGapE-MCTS and LUCB-MCTS

Several Best Arm Identification algorithms may be used within BAI-MCTS, and we now present two variants, that are respectively based on the UGapE [8] and the LUCB [12] algorithms. These two algorithms are very similar in that they exploit confidence intervals and use the same stopping rule, however the LUCB algorithm additionally uses the empirical means of the arms, which within BAI-MCTS requires defining an estimate $\hat{V}_s(t)$ of the value of the depth-one nodes.

The generic structure of the two algorithms is similar. At round $t + 1$ two promising depth-one nodes are computed, that we denote by $\underline{a}_t$ and $\underline{b}_t$. Among these two candidates, the node whose confidence interval is the largest (that is, the most uncertain node) is selected:

$$R_{t+1} = \operatorname*{argmax}_{i \in \{\underline{a}_t, \underline{b}_t\}} \left[U_i(t) - L_i(t)\right].$$

Then, following the BAI-MCTS architecture, the representative leaf of R_{t+1} (computed by going down the tree) is sampled: $L_{t+1} = \ell_{R_{t+1}}(t)$. The algorithm stops whenever the confidence intervals of the two promising arms overlap by less than ϵ:

$$\tau = \inf\left\{t \in \mathbb{N} : U_{\underline{b}_t}(t) - L_{\underline{a}_t}(t) < \epsilon\right\},$$

and it recommends $\hat{s}_\tau = \underline{a}_\tau$.

UGapE-MCTS. In UGapE-MCTS, introducing for each depth-one node the index

$$B_s(t) = \max_{s' \in \mathcal{C}(s_0) \setminus \{s\}} U_{s'}(t) - L_s(t),$$

the promising depth-one nodes are defined as

$$\underline{a}_t = \operatorname*{argmin}_{a \in \mathcal{C}(s_0)} B_a(t) \quad \text{and} \quad \underline{b}_t = \operatorname*{argmax}_{b \in \mathcal{C}(s_0) \setminus \{\underline{a}_t\}} U_b(t).$$

LUCB-MCTS. In LUCB-MCTS, the promising depth-one nodes are defined as

$$\underline{a}_t = \operatorname*{argmax}_{a \in \mathcal{C}(s_0)} \hat{V}_a(t) \quad \text{and} \quad \underline{b}_t = \operatorname*{argmax}_{b \in \mathcal{C}(s_0) \setminus \{\underline{a}_t\}} U_b(t),$$

where $\hat{V}_s(t) = \hat{\mu}_{\ell_s(t)}(t)$ is the empirical mean of the reprentative leaf of node s. Note that several alternative definitions of $\hat{V}_s(t)$ may be proposed (such as the middle of the confidence interval $\mathcal{I}_s(t)$, or $\max_{a \in \mathcal{C}(s)} \hat{V}_a(t)$), but our choice is crucial for the analysis of LUCB-MCTS, given in Appendix C.

In both algorithm $\underline{a}_t$ represents a guess for the best depth-one node, while $\underline{b}_t$ is its "optimistic" challenger, that has the maximal possible value among the other depth-one nodes.

3 Analysis of UGapE-MCTS

In this section we first prove that UGapE-MCTS and LUCB-MCTS are both (ϵ, δ)-correct. Then we give in Theorem 3 a high-probability upper bound on the number of samples used by UGapE-MCTS. A similar upper bound is obtained for LUCB-MCTS in Theorem 9, stated in Appendix C.

3.1 Choosing the Confidence Intervals

From now on, we assume that the confidence intervals on the leaves are of the form

$$\mathrm{L}_\ell(t) = \hat{\mu}_\ell(t) - \sqrt{\frac{\beta(N_\ell(t), \delta)}{2N_\ell(t)}} \quad \text{and} \quad \mathrm{U}_\ell(t) = \hat{\mu}_\ell(t) + \sqrt{\frac{\beta(N_\ell(t), \delta)}{2N_\ell(t)}}. \tag{1}$$

$\beta(s, \delta)$ is some *exploration function*, that can be tuned to have a δ-PAC algorithm, as expressed in the following lemma, whose proof can be found in Appendix A.2

Lemma 2. *If $\delta \leq \max(0.1|\mathcal{L}|, 1)$, for the choice*

$$\beta(s, \delta) = \log(|\mathcal{L}|/\delta) + 3\log\log(|\mathcal{L}|/\delta) + (3/2)\log(\log s + 1) \tag{2}$$

both UGapE-MCTS and LUCB-MCTS satisfy $\mathbb{P}(V(s^*) - V(\hat{s}_\tau) \leq \epsilon) \geq 1 - \delta$.

An interesting practical feature of these confidence intervals is that they only depend on the local number of draws $N_\ell(t)$, whereas most of the BAI algorithms use exploration functions that depend on the number of rounds t. Hence the only confidence intervals that need to be updated at round t are those of the ancestors of the selected leaf, which can be done recursively.

Moreover, $\beta(s, \delta)$ scales with $\log(\log(s))$, and not $\log(s)$, leveraging some tools recently introduced to obtain tighter confidence intervals [11, 14]. The union bound over $\mathcal{L}$ (that may be an artifact of our current analysis) however makes the exploration function of Lemma 2 still a bit over-conservative and in practice, we recommend the use of $\beta(s, \delta) = \log(\log(es)/\delta)$.

Finally, similar correctness results (with slightly larger exploration functions) may be obtained for confidence intervals based on the Kullback-Leibler divergence (see [5]), which are known to lead to better performance in standard best arm identification problems [15] and also depth-two tree search problems [10]. However, the sample complexity analysis is much more intricate, hence we stick to the above Hoeffding-based confidence intervals for the next section.

3.2 Complexity term and sample complexity guarantees

We first introduce some notation. Recall that s^* is the optimal action at the root, identified with the depth-one node satisfying $V(s^*) = V(s_0)$, and define the second-best depth-one node as $s_2^* = \text{argmax}_{s \in \mathcal{C}(s_0) \setminus \{s^*\}} V_s$. Recall $\mathcal{P}(s)$ denotes the parent of a node s different from the root. Introducing furthermore the set $\text{Anc}(s)$ of all the ancestors of a node s, we define the complexity term by

$$H_\epsilon^*(\boldsymbol{\mu}) := \sum_{\ell \in \mathcal{L}} \frac{1}{\Delta_\ell^2 \vee \Delta_*^2 \vee \epsilon^2}, \quad \text{where} \quad \begin{array}{rcl} \Delta_* & := & V(s^*) - V(s_2^*) \\ \Delta_\ell & := & \max_{s \in \text{Anc}(\ell) \setminus \{s_0\}} |V_s - V(\mathcal{P}(s))| \end{array} \tag{3}$$

The intuition behind these squared terms in the denominator is the following. We will sample a leaf ℓ until we either prune it (by determining that it or one of its ancestors is a bad move), prune everyone else (this happens for leaves below the optimal arm) or reach the required precision ϵ.

Theorem 3. *Let $\delta \leq \min(1, 0.1|\mathcal{L}|)$. UGapE-MCTS using the exploration function (2) is such that, with probability larger than $1-\delta$, $(V(s^*) - V(\hat{s}_\tau) < \epsilon)$ and*

$$\tau \leq 20 H_\epsilon^*(\boldsymbol{\mu}) \left[\log\left(\frac{20 H_\epsilon^*(\boldsymbol{\mu}) |\mathcal{L}|}{\delta} \right) + \log\log\left(\frac{20 H_\epsilon^*(\boldsymbol{\mu}) |\mathcal{L}|}{\delta} \right) + 1 \right].$$

Remark 4. *If $\beta(N_a(t), \delta)$ is changed to $\beta(t, \delta)$, one can still prove (ϵ, δ) correctness and furthermore upper bound the expectation of τ. However the algorithm becomes less efficient to implement, since after each leaf observation, ALL the confidence intervals have to be updated. In practice, this change lowers the probability of error but does not effect significantly the number of play-outs used.*

3.3 Comparison with previous work

To the best of our knowledge, the FindTopWinner algorithm [21] is the only algorithm from the literature designed to solve the best action identification problem in any-depth trees. The number of play-outs of this algorithm is upper bounded with high probability by

$$\sum_{\ell : \Delta_\ell > 2\epsilon} \left(\frac{32}{\Delta_\ell^2} \log \frac{16|\mathcal{L}|}{\Delta_\ell \delta} + 1 \right) + \sum_{\ell : \Delta_\ell \leq 2\epsilon} \left(\frac{8}{\epsilon^2} \log \frac{8|\mathcal{L}|}{\epsilon \delta} + 1 \right)$$

Beside a (small) improvement in the constant in front of the leading term in $\log(1/\delta)$, one can note an interesting improvement in the control of the number of draws of 2ϵ-optimal leaves (such that $\Delta_\ell \leq 2\epsilon$). In UGapE-MCTS, the number of draws of such leaves is at most of order $(\epsilon \vee \Delta_*^2)^{-1} \log(1/\delta)$, which may be significantly smaller than $\epsilon^{-1} \log(1/\delta)$ if there is a gap in the best and second best value. Moreover, unlike FindTopWinner and M-LUCB [10] in the depth two case, UGapE-MCTS can also be used when $\epsilon = 0$, with provable guarantees.

Regarding the algorithms themselves, one can note that M-LUCB, an extension of LUCB suited for depth-two tree, does *not* belong to the class of BAI-MCTS algorithms. Indeed, it has a "reversed" structure, first computing the representative leaf for each depth-one node: $\forall s \in \mathcal{C}(s_0), R_{s,t} = \ell_s(t)$ and then performing a BAI step over the representative leaves: $\tilde{L}_{t+1} = \text{BAIStep}(R_{s,t}, s \in \mathcal{C}(s_0))$. This alternative architecture can also be generalized to deeper trees, and was found to have empirical performance similar to BAI-MCTS. M-LUCB, which will be used as a benchmark in Section 4, also distinguish itself from LUCB-MCTS by the fact that it uses an exploration rate that depends on the global time $\beta(t, \delta)$ and that $\underline{a}_t$ is the empirical maximin arm (which can be different from the arm maximizing $\hat{V}_s$). This alternative choice is not yet supported by theoretical guarantees in deeper trees.

Finally, the exploration step of BAI-MCTS algorithm bears some similarity with the UCT algorithm [16], as it goes down the tree choosing alternatively the move that yields the highest UCB or the lowest LCB. However, the behavior of BAI-MCTS is very different at the root, where the first move is selected using a BAI algorithm. Another key difference is that BAI-MCTS relies on *exact* confidence intervals: each interval $\mathcal{I}_s(t)$ is shown to contain with high probability the corresponding value V_s, whereas UCT uses more heuristic confidence intervals, based on the number of visits of the parent node, and aggregating all the samples from descendant nodes. Using

UCT in our setting is not obvious as it would require to define a suitable stopping rule, hence we don't include a comparison with this algorithm in Section 4. A hybrid comparison between UCT and FindTopWinner is proposed in [21], providing UCT with the random number of samples used by the the fixed-confidence algorithm. It is shown that FindTopWinner has the advantage for hard trees that require many samples. Our experiments show that our algorithms in turn always dominate FindTopWinner.

3.4 Proof of Theorem 3.

Letting $\mathcal{E}_t = \bigcap_{\ell\in\mathcal{L}} (\mu_\ell \in \mathcal{I}_\ell(t))$ and $\mathcal{E} = \bigcap_{t\in\mathbb{N}} \mathcal{E}_t$, we upper bound τ assuming the event $\mathcal{E}$ holds, using the following key result, which is proved in Appendix D:

Lemma 5. *Let* $t \in \mathbb{N}$. $\mathcal{E}_t \cap (\tau > t) \cap (L_{t+1} = \ell) \quad \Rightarrow \quad N_\ell(t) \le \frac{8\beta(t,\delta)}{\Delta_\ell^2 \vee \Delta_*^2 \vee \epsilon^2}$.

Let T be a deterministic time. Using Lemma 5 and the definition of the algorithm, one has

$$\begin{aligned}\min(\tau, T) &= \sum_{t=1}^{T} \mathbb{1}_{(\tau>t)} \le \sum_{t=1}^{T} \mathbb{1}_{\left(N_{L_{t+1}} \le \frac{8\beta(t,\delta)}{\Delta_{L_{t+1}}^2 \vee \Delta_*^2 \vee \epsilon^2}\right)} = \sum_{\ell\in\mathcal{L}} \sum_{t=1}^{T} \mathbb{1}_{(L_{t+1}=\ell)} \mathbb{1}_{\left(N_\ell \le \frac{8\beta(T,\delta)}{\Delta_\ell^2 \vee \Delta_*^2 \vee \epsilon^2}\right)} \\ &\le \sum_{\ell\in\mathcal{L}} \frac{8\beta(T,\delta)}{\Delta_\ell^2 \vee \Delta_*^2 \vee \epsilon^2} = 8H_\epsilon^*(\boldsymbol{\mu})\beta(T,\delta).\end{aligned}$$

For any T such that $8H_\epsilon^*(\boldsymbol{\mu})\beta(T,\delta) < T$, one has $\min(\tau, T) < T$, which implies $\tau < T$. Therefore

$$\tau \le \inf\{T \in \mathbb{N} : \forall t \ge T,\ 8H_\epsilon^*(\boldsymbol{\mu})\beta(t,\delta) \le t\}.$$

Then, using the expression of the exploration function (2), it can be checked (using that $\log(x) \le x/e$ for all $x > 0$) that $\beta(s,\delta) \le (5/2)\log(|\mathcal{L}|t/\delta)$ for all s, δ. Hence, if $\mathcal{E}$ holds, τ is upper bounded by

$$N^*(\boldsymbol{\mu}) = \inf\left\{T \in \mathbb{N} : \forall t \ge T, 20H_\epsilon^*(\boldsymbol{\mu})\log(t|\mathcal{L}|/\delta) \le t\right\}.$$

Simple arguments (see Lemma 8 in Appendix B) can then be used to further upper bound

$$N^*(\boldsymbol{\mu}) \le 20H_\epsilon^*(\boldsymbol{\mu})\left[\log\left(\frac{20H_\epsilon^*(\boldsymbol{\mu})|\mathcal{L}|}{\delta}\right) + \log\log\left(\frac{20H_\epsilon^*(\boldsymbol{\mu})|\mathcal{L}|}{\delta}\right) + 1\right].$$

To conclude the proof, we remark that from the proof of Lemma 2 (see Appendix A.2) it follows that on $\mathcal{E}$, $V(s^*) - V(\hat{s}_\tau) < \epsilon$ and that $\mathcal{E}$ holds with probability larger than $1 - \delta$.

4 Experimental Validation

In this section we evaluate the performance of our algorithms in three experiments. We evaluate on the depth-two benchmark tree from [10], a new depth-three tree and the random tree ensemble from [21]. We compare to the FindTopWinner algorithm from [21] in all experiments, and in the depth-two experiment we include the M-LUCB algorithm from [10]. Its relation to BAI-MCTS is discussed in Section 3.3. For our BAI-MCTS algorithms and for M-LUCB we use the exploration rate $\beta(s,\delta) = \ln\frac{|\mathcal{L}|}{\delta} + \ln(\ln(s) + 1)$ (a stylized version of Lemma 2 that works well in practice), and we use the KL refinement of the confidence intervals (1). To replicate the experiment from [21], we supply all algorithms with $\delta = 0.1$ and $\epsilon = 0.01$. For comparing with [10] we run all algorithms with $\delta = 0.1|\mathcal{L}|$ (undoing the conservative union bound over leaves) and $\epsilon = 0$. In none of our experiments the observed error rate exceeds 0.1.

Figure 3 shows the benchmark tree from [10, Section 5] and the performance of four algorithms on it. We see that the special-purpose depth-two M-LUCB performs best, very closely followed by both our new arbitrary-depth LUCB-MCTS and UGapE-MCTS methods. All three use significantly fewer samples than FindTopWinner. Figure 4 shows a full 3-way tree of depth 3 with leafs drawn uniformly from $[0, 1]$. Again our algorithms outperform

the previous state of the art by an order of magnitude. Finally, we replicate the experiment from [21, Section 4]. To make the comparison as fair as possible, we use the proven exploration rate from (2). On 10K full 10-ary trees of depth 3 with Bernoulli leaf parameters drawn uniform at random from $[0,1]$ the average numbers of samples are: LUCB-MCTS 141811, UGapE-MCTS 142953 and FindTopWinner 2254560. To closely follow the original experiment, we do apply the union bound over leaves to all algorithms, which are run with $\epsilon = 0.01$ and $\delta = 0.1$. We did not observe any error from any algorithm (even though we allow 10%). Our BAI-MCTS algorithms deliver an impressive 15-fold reduction in samples.

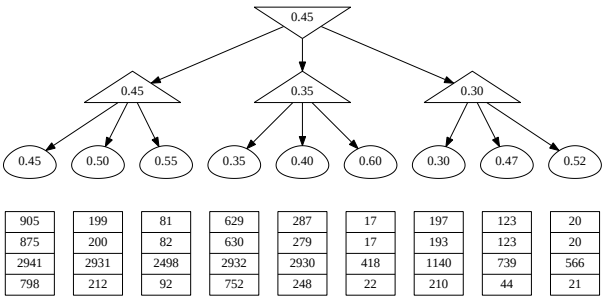

Figure 3: The 3×3 tree of depth 2 that is the benchmark in [10]. Shown below the leaves are the average numbers of pulls for 4 algorithms: LUCB-MCTS (0.89% errors, 2460 samples), UGapE-MCTS (0.94%, 2419), FindTopWinner (0%, 17097) and M-LUCB (0.14%, 2399). All counts are averages over 10K repetitions with $\epsilon = 0$ and $\delta = 0.1 \cdot 9$.

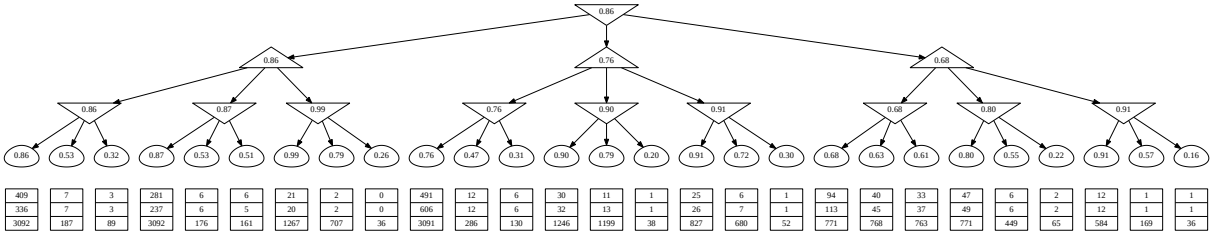

Figure 4: Our benchmark 3-way tree of depth 3. Shown below the leaves are the numbers of pulls of 3 algorithms: LUCB-MCTS (0.72% errors, 1551 samples), UGapE-MCTS (0.75%, 1584), and FindTopWinner (0%, 20730). Numbers are averages over 10K repetitions with $\epsilon = 0$ and $\delta = 0.1 \cdot 27$.

5 Lower bounds and discussion

Given a tree $\mathcal{T}$, a MCTS model is parameterized by the leaf values, $\boldsymbol{\mu} := (\mu_\ell)_{\ell \in \mathcal{L}}$, which determine the best root action: $s^* = s^*(\boldsymbol{\mu})$. For $\boldsymbol{\mu} \in [0,1]^{|\mathcal{L}|}$, We define $\mathrm{Alt}(\boldsymbol{\mu}) = \{\boldsymbol{\lambda} \in [0,1]^{|\mathcal{L}|} : s^*(\boldsymbol{\lambda}) \neq s^*(\boldsymbol{\mu})\}$. Using the same technique as [9] for the classic best arm identification problem, one can establish the following (non explicit) lower bound. The proof is given in Appendix E.

Theorem 6. *Assume $\epsilon = 0$. Any δ-correct algorithm satisfies*

$$\mathbb{E}_{\boldsymbol{\mu}}[\tau] \geq T^*(\boldsymbol{\mu}) d(\delta, 1-\delta), \quad \textit{where} \quad T^*(\boldsymbol{\mu})^{-1} := \sup_{\boldsymbol{w} \in \Sigma_{|\mathcal{L}|}} \inf_{\boldsymbol{\lambda} \in \mathrm{Alt}(\boldsymbol{\mu})} \sum_{\ell \in \mathcal{L}} w_\ell d(\mu_\ell, \lambda_\ell) \tag{4}$$

with $\Sigma_k = \{\boldsymbol{w} \in [0,1]^i : \sum_{i=1}^k w_i = 1\}$ and $d(x,y) = x \log(x/y) + (1-x)\log((1-x)/(1-y))$ is the binary Kullback-Leibler divergence.

This result is however not directly amenable for comparison with our upper bounds, as the optimization problem defined in Lemma 6 is not easy to solve. Note that $d(\delta, 1-\delta) \geq \log(1/(2.4\delta))$ [14], thus our upper bounds have the right dependency in δ. For depth-two trees with K (resp. M) actions for player A (resp. B), we can moreover prove the following result, that suggests an intriguing behavior.

Lemma 7. *Assume $\epsilon = 0$ and consider a tree of depth two with $\boldsymbol{\mu} = (\mu_{i,j})_{1\leq i\leq K, 1\leq j\leq M}$ such that $\forall (i,j), \mu_{1,1} > \mu_{i,1},\ \mu_{i,1} < \mu_{i,j}$. The supremum in the definition of $T^*(\boldsymbol{\mu})^{-1}$ can be restricted to*

$$\tilde{\Sigma}_{K,M} := \{\boldsymbol{w} \in \Sigma_{K\times M} : w_{i,j} = 0 \text{ if } i \geq 2 \text{ and } j \geq 2\}$$

and

$$T^*(\boldsymbol{\mu})^{-1} = \max_{\boldsymbol{w}\in\tilde{\Sigma}_{K,M}} \min_{\substack{i=2,\dots,K \\ a=1,\dots,M}} \left[w_{1,a} d\left(\mu_{1,a}, \frac{w_{1,a}\mu_{1,a} + w_{i,1}\mu_{i,1}}{w_{1,a}+w_{i,1}}\right) + w_{i,1} d\left(\mu_{i,1}, \frac{w_{1,a}\mu_{1,a} + w_{i,1}\mu_{i,1}}{w_{1,a}+w_{i,1}}\right)\right].$$

It can be extracted from the proof of Theorem 6 (see Appendix E) that the vector $\boldsymbol{w}^*(\boldsymbol{\mu})$ that attains the supremum in (4) represents the average proportions of selections of leaves by any algorithm matching the lower bound. Hence, the sparsity pattern of Lemma 7 suggests that matching algorithms should draw many of the leaves *much less than $O(\log(1/\delta))$ times*. This hints at the exciting prospect of optimal stochastic pruning, at least in the asymptotic regime $\delta \to 0$.

As an example, we numerically solve the lower bound optimization problem (which is a concave maximization problem) for $\boldsymbol{\mu}$ corresponding to the benchmark tree displayed in Figure 3 to obtain

$$T^*(\boldsymbol{\mu}) = 259.9 \qquad \text{and} \qquad \boldsymbol{w}^* = (0.3633, 0.1057, 0.0532), (0.3738, \mathbf{0}, \mathbf{0}), (0.1040, \mathbf{0}, \mathbf{0}).$$

With $\delta = 0.1$ we find $\mathrm{kl}(\delta, 1-\delta) = 1.76$ and the lower bound is $\mathbb{E}_{\boldsymbol{\mu}}[\tau] \geq 456.9$. We see that there is a potential improvement of at least a factor 4.

Future directions An (asymptotically) optimal algorithm for BAI called Track-and-Stop was developed by [9]. It maintains the empirical proportions of draws close to $\boldsymbol{w}^*(\hat{\boldsymbol{\mu}})$, adding forced exploration to ensure $\hat{\boldsymbol{\mu}} \to \boldsymbol{\mu}$. We believe that developing this line of ideas for MCTS would result in a major advance in the quality of tree search algorithms. The main challenge is developing efficient solvers for the general optimization problem (4). For now, even the sparsity pattern revealed by Lemma 7 for depth two does not give rise to efficient solvers. We also do not know how this sparsity pattern evolves for deeper trees, let alone how to compute $\boldsymbol{w}^*(\boldsymbol{\mu})$.

References

[1] J-Y. Audibert, S. Bubeck, and R. Munos. Best Arm Identification in Multi-armed Bandits. In *Proceedings of the 23rd Conference on Learning Theory*, 2010.

[2] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2):235–256, 2002.

[3] L. Borsoniu, R. Munos, and E. Páll. An analysis of optimistic, best-first search for minimax sequential decision making. In *ADPRL14*, 2014.

[4] C. Browne, E. Powley, D. Whitehouse, S. Lucas, P. 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,*, 4(1):1–49, 2012.

[5] O. Cappé, A. Garivier, O-A. Maillard, R. Munos, and G. Stoltz. Kullback-Leibler upper confidence bounds for optimal sequential allocation. *Annals of Statistics*, 41(3):1516–1541, 2013.

[6] T. Cazenave. Sequential halving applied to trees. *IEEE Transactions on Computational Intelligence and AI in Games*, 7(1):102–105, 2015.

[7] E. Even-Dar, S. Mannor, and Y. Mansour. Action Elimination and Stopping Conditions for the Multi-Armed Bandit and Reinforcement Learning Problems. *Journal of Machine Learning Research*, 7:1079–1105, 2006.

[8] V. Gabillon, M. Ghavamzadeh, and A. Lazaric. Best Arm Identification: A Unified Approach to Fixed Budget and Fixed Confidence. In *Advances in Neural Information Processing Systems*, 2012.

[9] A. Garivier and E. Kaufmann. Optimal best arm identification with fixed confidence. In *Proceedings of the 29th Conference On Learning Theory (COLT)*, 2016.

[10] A. Garivier, E. Kaufmann, and W.M. Koolen. Maximin action identification: A new bandit framework for games. In *Proceedings of the 29th Conference On Learning Theory*, 2016.

[11] K. Jamieson, M. Malloy, R. Nowak, and S. Bubeck. lil'UCB: an Optimal Exploration Algorithm for Multi-Armed Bandits. In *Proceedings of the 27th Conference on Learning Theory*, 2014.

[12] S. Kalyanakrishnan, A. Tewari, P. Auer, and P. Stone. PAC subset selection in stochastic multi-armed bandits. In *International Conference on Machine Learning (ICML)*, 2012.

[13] Z. Karnin, T. Koren, and O. Somekh. Almost optimal Exploration in multi-armed bandits. In *International Conference on Machine Learning (ICML)*, 2013.

[14] E. Kaufmann, O. Cappé, and A. Garivier. On the Complexity of Best Arm Identification in Multi-Armed Bandit Models. *Journal of Machine Learning Research*, 17(1):1–42, 2016.

[15] E. Kaufmann and S. Kalyanakrishnan. Information complexity in bandit subset selection. In *Proceeding of the 26th Conference On Learning Theory.*, 2013.

[16] L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. In *Proceedings of the 17th European Conference on Machine Learning*, ECML'06, pages 282–293, Berlin, Heidelberg, 2006. Springer-Verlag.

[17] T.L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22, 1985.

[18] T. Pepels, T. Cazenave, M. Winands, and M. Lanctot. Minimizing simple and cumulative regret in monte-carlo tree search. In *Computer Games Workshop, ECAI*, 2014.

[19] Aske Plaat, Jonathan Schaeffer, Wim Pijls, and Arie de Bruin. Best-first fixed-depth minimax algorithms. *Artificial Intelligence*, 87(1):255 – 293, 1996.

[20] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 2016.

[21] K. Teraoka, K. Hatano, and E. Takimoto. Efficient sampling method for monte carlo tree search problem. *IEICE Transactions on Infomation and Systems*, pages 392–398, 2014.

[22] W.R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25:285–294, 1933.

A Confidence Intervals

A.1 Proof of Proposition 1

The proof proceeds by induction. Let the inductive hypothesis be $\mathcal{H}_d$="for all the nodes s at (graph) distance d from a leaf, $V_s \in \mathcal{I}_s(t)$".

$\mathcal{H}_0$ clearly holds by definition of $\mathcal{E}_t$. Now let d such that $\mathcal{H}_d$ holds and let s be at distance $d+1$ of a leaf. Then all $s' \in \mathcal{C}(s)$ are at distance at most d from a leaf and using the inductive hypothesis,

$$\forall c \in \mathcal{C}(s), \quad \mathrm{L}_c(t) \leq V_c \leq \mathrm{U}_c(t).$$

Assume that s is a MAX node. Using that $\mathrm{U}_s(t) = \max_{c\in\mathcal{C}(s)} \mathrm{U}_c(t)$, one has $c \in \mathcal{C}(s)$, $V_c \leq \mathrm{U}_c(t) \leq \mathrm{U}_s(t)$. By definition, $V_s = \max_{c\in\mathcal{C}(s)} V_c$, thus it follows that $V_s \leq \mathrm{U}_s(t)$. Still by definition of V_s, it holds that $\forall c \in \mathcal{C}(s), \mathrm{L}_c(t) \leq V_c \leq V_s$ and finally, as $\mathrm{L}_s(t) = \max_{c\in\mathcal{C}(s)} \mathrm{L}_c(t)$, $\mathrm{L}_s(t) \leq V_s \leq \mathrm{U}_s(t)$. A similar reasoning yields the same conclusion if s is a MIN node, thus $\mathcal{H}_{d+1}$ holds.

As the tree $\mathcal{T}$ is finite, we conclude by induction that $\forall s \in \mathcal{T}, V_s \in \mathcal{I}_s(t)$.

A.2 Proof of Lemma 2

Let

$$\mathcal{E}_t = \bigcap_{\ell\in\mathcal{L}} (\mu_\ell \in \mathcal{I}_\ell(t)) \quad \text{and} \quad \mathcal{E} = \bigcap_{t\in\mathbb{N}} \mathcal{E}_t.$$

Using Proposition 1, on $\mathcal{E}_t$, for all $s \in \mathcal{T}$, $V_s \in \mathcal{I}_s(t)$. If the algorithm stops at some time t, as $\mathrm{L}_{\underline{a}_t}(t) > \mathrm{U}_{\underline{b}_t}(t) - \epsilon$, the outputted action, $\hat{s}_\tau = \underline{a}_t$, satisfies $\mathrm{L}_{\hat{s}_\tau}(t) > \mathrm{U}_{s'}(t) - \epsilon$, for all $s' \neq \hat{s}_\tau$. As $\mathcal{E}$ holds, one obtains

$$V(\hat{s}_\tau) \geq \max_{s'\neq\hat{s}_\tau} V(s') - \epsilon,$$

and $\hat{s}_\tau$ is an ϵ-maximin action. Hence, the algorithm is correct on $\mathcal{E}$. The error probability is thus upper bounded by

$$\begin{aligned}
\mathbb{P}(\mathcal{E}^c) &\leq \mathbb{P}\left(\exists \ell \in \mathcal{L}, \exists t \in \mathbb{N} : |\hat{\mu}_\ell(t) - \mu_\ell| > \sqrt{\beta(N_\ell(t),\delta)/(2N_\ell(t))}\right) \\
&\leq \sum_{\ell\in\mathcal{L}} \mathbb{P}\left(\exists s \in \mathbb{N} : 2s(\hat{\mu}_{\ell,s} - \mu_\ell)^2 > \beta(s,\delta)\right) \\
&\leq 2|\mathcal{L}|\mathbb{P}\left(\exists s \in \mathbb{N} : S_s > \sqrt{2\sigma^2 s\beta(s,\delta)}\right),
\end{aligned}$$

where $S_s = X_1 + \cdots + X_s$ is a martingale with σ^2-subgaussian increments, with $\sigma^2 = 1/4$. It was shown in [14] that for $\delta \leq 0.1$, if

$$\beta(t,\delta) = \log\left(\frac{1}{\delta}\right) + 3\log\log\left(\frac{1}{\delta}\right) + (3/2)\log(\log(s)+1),$$

one has $\mathbb{P}\left(\exists s \in \mathbb{N} : S_s > \sqrt{2\sigma^2 s\beta(s,\delta)}\right) \leq \delta$, which concludes the proof.

B A technical lemma

We state below a result that follows from simple calculus and is helpful at the final stage of our sample complexity analyzes.

Lemma 8. *Let C be a constant such that $C\log(C) \geq e-1$. Then*

$$x_0 := \log\left(\frac{e}{e-1} C\log(C)\right) \quad \Rightarrow \quad \forall x \geq x_0, \;\; e^x \geq Cx$$

Proof of Lemma B. The mapping $x \mapsto e^x/x$ is non-decreasing for $x \geq 1$. Hence, if $x_0 \geq 1$ satisfies $e^{x_0} \geq Cx_0$, it follows that for all $x \geq x_0$, the same inequality is still true. We now proceed to prove that with the above choice of x_0 satisfies $x_0 \geq 1$ and $e^{x_0} \geq Cx_0$.

The first inequality easily follows from the fact that $C\log(C) \geq e-1$. As for second, one has

$$\begin{aligned} e^{x_0} \geq Cx_0 &\Leftrightarrow \frac{e}{e-1}\log(C) \geq \log\left(\frac{e}{e-1}C\log(C)\right) \\ &\Leftrightarrow \frac{1}{e-1}\log(C) \geq \log\left(\frac{e}{e-1}\log(C)\right) \end{aligned}$$

and this inequality is true using that for all $x > 0$, $\log(x) \leq x/e$.

□

Using Lemma B in the proof of Theorem 3. Letting $\alpha = 20H_\epsilon^*(\boldsymbol{\mu})$ and $\gamma = |\mathcal{L}|/\delta$, one has

$$\begin{aligned} N^*(\boldsymbol{\mu}) &\leq \alpha \times \inf\{u \in \mathbb{R} : \forall x \geq u, \log(x\alpha\gamma) \leq x\} \\ &\leq \alpha \times \inf\{u \in \mathbb{R} : \forall x \geq u, (\alpha\gamma)x \leq \exp(x)\} \\ &\leq \alpha\log\left(\frac{e}{e-1}(\alpha\gamma)\log(\alpha\gamma)\right) \\ &\leq \alpha\left[\log(\alpha\gamma) + \log\log(\alpha\gamma) + 1\right], \end{aligned}$$

where the third inequality follows from Lemma 8, assuming that $C = \alpha\gamma$ satisfies $C\log(C) \geq e-1$. A sufficient condition is $C \geq e$ which translates to

$$\frac{20H_\epsilon^*(\boldsymbol{\mu})|\mathcal{L}|}{\delta} \geq e \quad \Leftrightarrow \quad \delta \leq \left(\frac{20}{e}\right)H_\epsilon^*(\boldsymbol{\mu})|\mathcal{L}|.$$

Given that $H_\epsilon^*(\boldsymbol{\mu}) \geq |\mathcal{L}| \geq 1$, and $\delta \leq 1$, this last inequality is always true.

C Sample complexity analysis of LUCB-MCTS

We provide an analysis of a slight variant of LUCB-MCTS that may stop at even rounds only and for $t \in 2\mathbb{N}$ draws the representative leaf of the two promising depth-one nodes:

$$L_{t+1} = \ell_{\underline{a}_t}(t) \quad \text{and} \quad L_{t+2} = \ell_{\underline{b}_t}(t). \tag{5}$$

The stopping rule is then $\tau = \inf\left\{t \in 2\mathbb{N}^* : U_{\underline{b}_t}(t) - L_{\underline{a}_t}(t) < \epsilon\right\}$. For this algorithm, one can prove the following

Theorem 9. *Let $\delta \leq \min(1, 0.1|\mathcal{L}|)$. LUCB-MCTS using the exploration function (2) and selecting the two promising leaves at each round is such that, with probability larger than $1-\delta$, $(V(s^*) - V(\hat{s}_\tau) < \epsilon)$ and*

$$\tau \leq 40H_\epsilon^*(\boldsymbol{\mu})\left[\log\left(\frac{40H_\epsilon^*(\boldsymbol{\mu})|\mathcal{L}|}{\delta}\right) + \log\log\left(\frac{40H_\epsilon^*(\boldsymbol{\mu})|\mathcal{L}|}{\delta}\right) + 1\right].$$

Proof. The analysis follows the same lines as that of UGapE-MCTS, yet it relies on a slightly different key result, proved in the next section. Letting $\mathcal{E}_t = \cap_{\ell\in\mathcal{L}}(\mu_\ell \in \mathcal{I}_\ell(t))$ as in the proof of Theorem 3 and defining $\mathcal{E} = \cap_{t\in 2\mathbb{N}^*}\mathcal{E}_t$, one can state the following.

Lemma 10. *Let $t \in 2\mathbb{N}$.*

$$\mathcal{E}_t \cap (\tau > t) \quad \Rightarrow \quad \exists \ell \in \{L_{t+1}, L_{t+2}\} : \quad N_\ell(t) \leq \frac{8\beta(t,\delta)}{\Delta_\ell^2 \vee \epsilon^2}.$$

Let T be a deterministic time. We upper bound τ assuming the event $\mathcal{E}$ holds. Using Lemma 10 and the fact that for every even t, $(\tau_\delta > t) = (\tau_\delta > t+1)$ by definition of the algorithm, one has

$$\begin{aligned}
\min(\tau, T) &= \sum_{t=1}^{T} \mathbb{1}_{(\tau > t)} = 2 \sum_{\substack{t \in 2\mathbb{N} \\ t \leq T}} \mathbb{1}_{(\tau_\delta > t)} = 2 \sum_{\substack{t \in 2\mathbb{N} \\ t \leq T}} \mathbb{1}_{\left(\exists \ell \in \{L_{t+1}, L_{t+2}\} : N_\ell(t) \leq 2\beta(t,\delta)/(\Delta_\ell^2 \vee \Delta_*^2 \epsilon^2)\right)} \\
&\leq 2 \sum_{\substack{t \in 2\mathbb{N} \\ t \leq T}} \sum_{\ell \in \mathcal{L}} \mathbb{1}_{(L_{t+1}=\ell) \cup (L_{t+2}=\ell)} \mathbb{1}_{(N_\ell(t) \leq 8\beta(T,\delta)/(\Delta_\ell^2 \vee \Delta_*^2 \epsilon^2))} \\
&\leq 16 \sum_{\ell \in \mathcal{L}} \frac{1}{\Delta_\ell^2 \vee \Delta_*^2 \vee \epsilon^2} \beta(T,\delta) = 16 H_\epsilon^*(\boldsymbol{\mu}) \beta(T,\delta).
\end{aligned}$$

For any T such that $16 H_\epsilon^*(\boldsymbol{\mu})\beta(T,\delta) < T$, one has $\min(\tau, T) < T$, which implies $\tau < T$. Therefore

$$\tau \leq \inf\{T \in \mathbb{N} : \forall t \geq T, 16 H_\epsilon^*(\boldsymbol{\mu})\beta(t,\delta) \leq t\}.$$

Using the same elements as in the proof of Theorem 3, notably Lemma 8, we can now further upper bound τ by

$$40 H_\epsilon^*(\boldsymbol{\mu}) \left[\log\left(\frac{40 H_\epsilon^*(\boldsymbol{\mu})|\mathcal{L}|}{\delta} \right) + \log\log\left(\frac{40 H_\epsilon^*(\boldsymbol{\mu})|\mathcal{L}|}{\delta} \right) + 1 \right].$$

Just like for Theorem 3, using that $\mathcal{P}(\mathcal{E}) \geq 1 - \delta$ and that the algorithm is correct on $\mathcal{E}$ yields the conclusion.

D Proof of Lemma 5 and Lemma 10

We first state Lemma 12, that holds for both UGapE and LUCB-MCTS and is a consequence of the definition of the exploration procedure. This result builds on the following lemma, that expresses the fact that along a path from the root to a representative leaf, the confidence intervals are nested.

Lemma 11. *Let $t \in \mathbb{N}$ and $s_0, s_1, \ldots, s_D$ be a path from the root down to a leaf $\ell = s_D$.*

$$(\ell_{s_1}(t) = s_D) \Rightarrow (\forall i = 2, \ldots, D, \quad \mathcal{I}_{s_{k-1}}(t) \subseteq \mathcal{I}_{s_k}(t))$$

Lemma 12. *Let $t \in \mathbb{N}$ and $s_0, s_1, \ldots, s_D$ be a path from the root down to a leaf $\ell = s_D$. If $\mathcal{E}_t$ holds and ℓ is selected at round $t+1$ (UGapE) or if t is even and $\ell \in \{L_{t+1}, L_{t+2}\}$ (LUCB), then*

$$\sqrt{\frac{2\beta(t,\delta)}{N_\ell(t)}} \geq \max_{k=2\ldots D} |V(s_k) - V(s_{k-1})|.$$

UGapE-MCTS: proof of Lemma 5. The following lemma is specific to UGapE-MCTS. We let $s_0, s_1, \ldots, s_D$ be a path down to a leaf $\ell = s_D$.

Lemma 13. *Let $t \in \mathbb{N}$. If $\mathcal{E}_t$ holds and UGapE-MCTS has not stopped after t observations, that is $(\tau > t)$,*

$$(L_{t+1} = \ell) \Rightarrow \left(\sqrt{\frac{8\beta(t,\delta)}{N_\ell(t)}} \geq \max\left(\Delta_*, V(s_0) - V(s_1), \epsilon\right) \right)$$

Putting together Lemma 12 and Lemma 13 and using that

$$\Delta_\ell = \max\left(V(s_0) - V(s_1), \max_{k=2\ldots D} |V(s_k) - V(s_{k-1})| \right)$$

one obtains

$$\mathcal{E}_t \cap (\tau > t) \cap (L_{t+1} = \ell) \Rightarrow \left(\sqrt{\frac{8\beta(t,\delta)}{N_\ell(t)}} \geq \max\left(\Delta_\ell, \Delta_*, \epsilon\right) \right),$$

which yields the proof of Lemma 5 by inverting the bound.

LUCB-MCTS: proof of Lemma 10. The following lemma is specific to the LUCB-MCTS algorithm. It can be viewed as a generalization of Lemma 2 in [12].

Lemma 14. *Let $t \in 2\mathbb{N}$ and let $\gamma \in [V(s_2^*), V(s^*)]$. If $\mathcal{E}_t$ holds and LUCB-MCTS has not stopped after t observations, that is $(\tau > t)$, then*

$$\exists \ell \in \{L_{t+1}, L_{t+2}\} : (\gamma \in \mathcal{I}_\ell(t)) \cap \left(\sqrt{\frac{2\beta(t,\delta)}{N_\ell(t)}} \geq \epsilon \right).$$

Choosing $\gamma = \frac{V(s^*)+V(s_2^*)}{2}$ and letting s_ℓ be the depth-one ancestor of ℓ, on $\mathcal{E}_t$ it holds that $V(s_\ell) \in \mathcal{I}_\ell(t)$ (by Lemma 11) and

$$\begin{aligned} (\gamma \in \mathcal{I}_\ell(t)) \quad &\Rightarrow \quad \left(\sqrt{\frac{2\beta(t,\delta)}{N_\ell(t)}} \geq |V(s_\ell) - \gamma| \right) \\ &\Rightarrow \quad \left(\sqrt{\frac{2\beta(t,\delta)}{N_\ell(t)}} \geq \frac{V(s^*) - V(s_2^*)}{2} \right). \end{aligned}$$

Recall $\Delta_* = V(s^*) - V(s_2^*)$. Then by Lemma 14, on $\mathcal{E}_t \cap (\tau > t)$, there exists $\ell \in \{L_{t+1}, L_{t+2}\}$ such that

$$\sqrt{\frac{8\beta(t,\delta)}{N_\ell(t)}} \geq \max(\Delta_*, \epsilon).$$

Combining this with Lemma 12 and using the expression of Δ_ℓ given in (3) yields that

$$\mathcal{E}_t \cap (\tau > t) \;\Rightarrow\; \exists \ell \in \{L_{t+1}, L_{t+2}\} : \; \sqrt{\frac{8\beta(t,\delta)}{N_\ell(t)}} \geq \max\left(\Delta_\ell, \Delta_*, \epsilon\right),$$

which yields the proof of Lemma 10 by inverting the bound.

D.1 Proof of Lemma 11.

The leaf ℓ is the representative of the depth 1 node s_1, therefore the path $s_1, \dots, s_D$ is such that $c_{s_{k-1}}(t) = s_k$ for all $k = 2, \dots, D$. Using the way the representative are build, we now show that

$$\forall k \in \{2, \dots, D\}, \quad \mathcal{I}_{s_{k-1}}(t) \subseteq \mathcal{I}_{s_k}(t).$$

If s_{k-1} is a MAX node, $\mathrm{U}_{s_{k-1}}(t) = \mathrm{U}_{s_k}(t)$ by definition and $\mathrm{L}_{s_{k-1}}(t) = \max_{s \in \mathcal{C}(s_{k-1})} \mathrm{L}_s(t) \geq \mathrm{L}_{s_k}(t)$. Similarly, if s_{k-1} is a MIN node, $\mathrm{L}_{s_{k-1}}(t) = \mathrm{L}_{s_k}(t)$ by definition and $\mathrm{U}_{s_{k-1}}(t) = \min_{s \in \mathcal{C}(s_{k-1})} \mathrm{U}_s(t) \leq \mathrm{U}_{s_k}(t)$, so that in both cases $\mathcal{I}_{s_{k-1}}(t) \subseteq \mathcal{I}_{s_k}(t)$.

D.2 Proof of Lemma 12.

Let $\ell \in \mathcal{L}$ be a leaf that is sampled based on the information available at round t. In particular, as ℓ is a representative leaf of the depth 1 node s_1, the path $s_1, \dots, s_D$ is such that $c_{s_{k-1}}(t) = s_k$ for all $k = 2, \dots, D$. Let $k = 2, \dots, D$. If $s_{k-1} \in \{2, \dots, D\}$ is a MAX node, it holds by definition of the representative children that, for all $s' \in \mathcal{C}(s_{k-1})$,

$$\mathrm{U}_{s_k}(t) \geq \mathrm{U}_{s'}(t).$$

Now, from Lemma 11 one has $\mathrm{U}_\ell(t) \geq \mathrm{U}_{s_k}(t)$ and from Proposition 1 as $\mathcal{E}_t$ holds, one has

$$\forall s \in \mathcal{T}, \;\; V_s \in \mathcal{I}_s(t). \tag{6}$$

Using these two ingredients yields

$$\begin{aligned} \mathrm{U}_\ell(t) &\geq V(s') \\ \mathrm{L}_\ell(t) + 2\sqrt{\frac{\beta(N_\ell(t),\delta)}{2N_\ell(t)}} &\geq V(s') \\ \mathrm{L}_{s_k}(t) + 2\sqrt{\frac{\beta(t,\delta)}{2N_\ell(t)}} &\geq V(s') \\ V(s_k) + 2\sqrt{\frac{\beta(t,\delta)}{2N_\ell(t)}} &\geq V(s'). \end{aligned}$$

Thus

$$\sqrt{\frac{2\beta(t,\delta)}{N_\ell(t)}} \geq \max_{s'\in\mathcal{C}(s_{k-1})} V(s') - V(s_k) = V(s_{k-1}) - V(s_k) \geq 0.$$

If s_{k-1} is a MIN node, a similar reasoning show that

$$\sqrt{\frac{2\beta(t,\delta)}{N_\ell(t)}} \geq V(s_k) - V(s_{k-1}) \geq 0.$$

Putting everything together yields

$$\sqrt{\frac{2\beta(t,\delta)}{N_\ell(t)}} \geq \max_{k=2,\dots,D} |V(s_k) - V(s_{k-1})|.$$

D.3 Proof of Lemma 13.

We first prove the following intermediate result, that generalizes Lemma 4 in [8].

Lemma 15. *For all $t \in \mathbb{N}^*$, the following holds*

$$\begin{aligned} &\text{if } R_{t+1} = \underline{a}_t \quad \text{then} \quad U_{\underline{b}_t}(t) \leq U_{\underline{a}_t}(t) \\ &\text{if } R_{t+1} = \underline{b}_t \quad \text{then} \quad L_{\underline{b}_t}(t) \leq L_{\underline{a}_t}(t) \end{aligned}$$

Proof. Assume $\underline{b}_t$ is selected (i.e. $R_{t+1} = \underline{b}_t$) and $\mathrm{L}_{\underline{b}_t}(t) > \mathrm{L}_{\underline{a}_t}(t)$. As the confidence interval on $V(\underline{b}_t)$ is larger than the confidence intervals on $V(\underline{a}_t)$ ($\underline{b}_t$ is selected), this also yields $\mathrm{U}_{\underline{b}_t}(t) > \mathrm{U}_{\underline{a}_t}(t)$. Hence

$$B_{\underline{a}_t}(t) = \mathrm{U}_{\underline{b}_t}(t) - \mathrm{L}_{\underline{a}_t}(t) > \mathrm{U}_{\underline{a}_t}(t) - \mathrm{L}_{\underline{b}_t}(t).$$

Also, by definition of $\underline{b}_t$, $\mathrm{U}_{\underline{b}_t}(t) \geq \mathrm{U}_b(t)$. Hence

$$B_{\underline{a}_t}(t) > \max_{b\neq\underline{b}_t} \mathrm{U}_b(t) - \mathrm{L}_{\underline{b}_t}(t) = B_{\underline{b}_t}(t),$$

which contradicts the definition of $\underline{a}_t$. Thus, we proved by contradiction that $\mathrm{L}_{\underline{b}_t}(t) \geq \mathrm{L}_{\underline{a}_t}(t)$.

A similar reasoning can be used to prove that $R_{t+1} = \underline{a}_t \Rightarrow U_{\underline{b}_t}(t) \leq U_{\underline{a}_t}(t)$.

□

A simple consequence of Lemma 15 is the fact that, on $\mathcal{E}_t \cap (\tau > t)$,

$$(L_{t+1} = \ell) \Rightarrow \left(\sqrt{\frac{2\beta(t,\delta)}{N_\ell(t)}} > \epsilon \right). \tag{7}$$

Indeed, as the algorithm doesn't stop after t rounds, it holds that $U_{\underline{b}_t}(t) - L_{\underline{a}_t}(t) > \epsilon$. If ℓ is the arm selected at round $t+1$, $\ell = \ell_{R_{t+1}}(t)$ and one can prove using Lemma 15 that $U_{R_{t+1}}(t) - L_{R_{t+1}}(t) > \epsilon$ (by distinguishing two cases). Finally, as $\mathcal{E}_t$ holds, by Lemma 11, $\mathcal{I}_{R_{t+1}}(t) \subseteq \mathcal{I}_\ell(t)$. Hence $U_\ell(t) - L_\ell(t) > \epsilon$, and (7) follows using the particular form of the confidence intervals.

To complete the proof, we now show that

$$(L_{t+1} = \ell) \Rightarrow \left(\sqrt{\frac{8\beta(t,\delta)}{N_\ell(t)}} > \max(\Delta_*, V(s_0) - V(s_1) \right). \tag{8}$$

by distinguishing several cases.

Case 1: $s^* \notin \mathrm{Anc}(\ell)$ **and** $R_{t+1} = \underline{b}_t$**.** Using that the algorithm doesn't stop yields

$$\mathrm{L}_{\underline{b}_t}(t) - \mathrm{U}_{\underline{a}_t}(t) + 2(\mathrm{U}_{\underline{b}_t}(t) - L_{\underline{b}_t}(t)) > \epsilon.$$

As $\mathcal{E}_t$ holds, $L_{\underline{b}_t}(t) \le V(\underline{b}_t) = V(s_1)$ and $U_{\underline{a}_t}(t) \ge V(\underline{a}_t)$.

Therefore, if $\underline{a}_t = s^*$ it holds that

$$2(\mathrm{U}_{\underline{b}_t}(t) - \mathrm{L}_{\underline{b}_t}(t)) > V(s^*) - V(s_1) + \epsilon.$$

If $\underline{a}_t \neq s^*$, by definition of $\underline{b}_t$ one has

$$\mathrm{U}_{\underline{b}_t}(t) \ge \mathrm{U}_{s^*}(t) \ge V(s^*),$$

hence

$$\begin{aligned} \mathrm{L}_{\underline{b}_t}(t) + (\mathrm{U}_{\underline{b}_t}(t) - \mathrm{L}_{\underline{b}_t}(t)) &\ge V(s^*) \\ V(s_1) + (\mathrm{U}_{\underline{b}_t}(t) - \mathrm{L}_{\underline{b}_t}(t)) &\ge V(s^*). \end{aligned}$$

Thus, recalling that $V(s_0) = V(s^*)$, whatever the value of $\underline{a}_t$, one obtains

$$2(\mathrm{U}_{\underline{b}_t}(t) - \mathrm{L}_{\underline{b}_t}(t)) \ge V(s_0) - V(s_1).$$

From Lemma 11 the width of $\mathcal{I}_{\underline{b}_t}(t)$ is upper bounded by the width of $\mathcal{I}_\ell(t)$, hence

$$2\left(\mathrm{U}_\ell(t) - \mathrm{L}_\ell(t)\right) \ge V(s_0) - V(s_1). \tag{9}$$

Case 2: $s^* \notin \mathrm{Anc}(\ell)$ **and** $R_{t+1} = \underline{a}_t$**.** As $s^* \neq \underline{a}_t$, by definition of $\underline{b}_t$ one has

$$\mathrm{U}_{\underline{b}_t}(t) \ge \mathrm{U}_{s^*}(t) \ge V(s^*).$$

Hence, using Lemma 15,

$$\mathrm{U}_{\underline{a}_t}(t) - \mathrm{L}_{\underline{a}_t}(t) \ge \mathrm{U}_{\underline{b}_t}(t) - \mathrm{L}_{\underline{a}_t}(t) \ge V(s^*) - V(s_1),$$

as $\mathcal{E}_t$ holds. Finally, by Lemma 11,

$$\mathrm{U}_\ell(t) - \mathrm{L}_\ell(t) \ge V(s_0) - V(s_1). \tag{10}$$

Case 3: $s^* \in \mathrm{Anc}(\ell)$ **and** $R_{t+1} = \underline{a}_t$**.** One has $\underline{a}_t = s^*$. Using that the algorithm doesn't stop yields

$$\mathrm{L}_{\underline{b}_t}(t) - \mathrm{U}_{\underline{a}_t}(t) + 2(\mathrm{U}_{\underline{a}_t}(t) - \mathrm{L}_{\underline{a}_t}(t)) > \epsilon.$$

As $\mathcal{E}_t$ holds, $\mathrm{U}_{\underline{a}_t}(t) \le V(s^*)$ and $\mathrm{L}_{\underline{b}_t}(t) \le V(\underline{b}_t) \le V(s_2^*)$. Therefore, if $\underline{a}_t = s^*$ it holds that

$$2(U_{\underline{a}_t}(t) - L_{\underline{a}_t}(t)) > V(s^*) - V(s_2^*) + \epsilon.$$

and by Lemma 11

$$2\left(\mathrm{U}_\ell(t) - \mathrm{L}_\ell(t)\right) \ge V(s^*) - V(s_2^*). \tag{11}$$

Case 4: $s^* \in \mathrm{Anc}(\ell)$ **and** $R_{t+1} = \underline{b}_t$**.** One has $\underline{b}_t = s^*$. Using Lemma 15 yields

$$\mathrm{U}_{\underline{b}_t}(t) - \mathrm{L}_{\underline{b}_t}(t) \ge \mathrm{U}_{\underline{b}_t}(t) - \mathrm{L}_{\underline{a}_t}(t) \ge V(s^*) - V(s_2^*),$$

as $\mathcal{E}_t$ holds and $V(\underline{a}_t) \le V(s_2^*)$. Finally, by Lemma 11,

$$\mathrm{U}_\ell(t) - \mathrm{L}_\ell(t) \ge V(s^*) - V(s_2^*). \tag{12}$$

Combining (9)-(12), we see that in all four cases

$$2(\mathrm{U}_\ell(t) - \mathrm{L}_\ell(t)) \ge \max(V(s^*) - V(s_2^*), V(s_0) - V(s_1)),$$

as for $s^* \notin \mathrm{Anc}(\ell)$, $V(s_0) - V(s_1) = V(s^*) - V(s_1) \ge V(s^*) - V(s_2^*)$, and for $s^* \in \mathrm{Anc}(\ell)$, $V(s_0) - V(s_1) = 0$. Using the expression of the confidence intervals and recalling that $\Delta_* = V(s^*) - V(s_2^*)$, one obtains

$$4\sqrt{\frac{\beta(t,\delta)}{2N_\ell(t)}} \ge \max(\Delta_*, V(s_0) - V(s_1))$$

which proves (8).

D.4 Proof of Lemma 14.

Fix $\gamma \in [V(s_2^*), V(s^*)]$ and assume $\mathcal{E}_t \cap (\tau > t)$ holds. We assume (by contradiction) that γ doesn't belong to $\mathcal{I}_{L_{t+1}}(t)$ nor to $\mathcal{I}_{L_{t+2}}(t)$. There are four possibilities:

- $\mathrm{L}_{L_{t+1}}(t) > \gamma$ and $\mathrm{L}_{L_{t+2}}(t) > \gamma$. As $\mathcal{E}_t$ holds and L_{t+1} and L_{t+2} are representative, it yields that there exists two nodes $s \in \mathcal{C}(s_0)$ such that $V_s > \gamma$, which contradicts the definition of γ.
- $\mathrm{U}_{L_{t+1}}(t) < \gamma$ and $\mathrm{U}_{L_{t+2}}(t) < \gamma$. From the definition of $\underline{b}_{t+1}$, it yields that for all $s \in \mathcal{C}(s_0)$, $\mathrm{U}_s(t) < \gamma$ and as $\mathcal{E}_t$ holds one obtains $V_s < \gamma$ for all $s \in \mathcal{C}(s_0)$, which contradicts the definition of γ.
- $\mathrm{L}_{L_{t+1}}(t) > \gamma$ and $\gamma > \mathrm{U}_{L_{t+2}}(t)$. This implies that $\mathrm{L}_{L_{t+1}}(t) > \mathrm{U}_{L_{t+2}}(t)$ and that $\mathrm{L}_{\underline{a}_t}(t) > \mathrm{U}_{\underline{b}_t}(t)$ (by Lemma 11 and the fact that L_{t+1} and L_{t+2} are representative leaves). This yields $(\tau \le t)$ and a contradiction.
- $\mathrm{U}_{L_{t+1}}(t) < \gamma$ and $\gamma < \mathrm{L}_{L_{t+2}}(t)$. This implies in particular that $\hat{\mu}_{L_{t+1}}(t) < \hat{\mu}_{L_{t+2}}(t)$. Thus $\hat{V}(\underline{a}_t, t) < \hat{V}(\underline{b}_t, t)$, which contradicts the definition of $\underline{a}_t$.

Hence, we just proved by contradiction that there exists $\ell \in \{L_{t+1}, L_{t+2}\}$ such that $\gamma \in \mathcal{I}_\ell(t)$. To prove Lemma 14, it remains to establish the following three statements.

1. $(\gamma \in \mathcal{I}_{L_{t+1}}(t)) \cap (\gamma \in \mathcal{I}_{L_{t+2}}(t)) \;\Rightarrow\; \left(\exists \ell \in \{L_{t+1}, L_{t+2}\} : \sqrt{\frac{2\beta(t,\delta)}{N_\ell(t)}} > \epsilon\right)$
2. $(\gamma \in \mathcal{I}_{L_{t+1}}(t)) \cap (\gamma \notin \mathcal{I}_{L_{t+2}}(t)) \;\Rightarrow\; \left(\sqrt{\frac{2\beta(t,\delta)}{N_{L_{t+1}}(t)}} > \epsilon\right)$

3. $(\gamma \notin \mathcal{I}_{L_{t+1}}(t)) \cap (\gamma \in \mathcal{I}_{L_{t+2}}(t)) \;\Rightarrow\; \left(\sqrt{\frac{2\beta(t,\delta)}{N_{L_{t+2}}(t)}} > \epsilon\right)$

Statement 1. As the algorithm doesn't stop, $\mathrm{U}_{\underline{b}_t}(t) - \mathrm{L}_{\underline{a}_t}(t) > \epsilon$. Hence

$$\begin{aligned} \mathrm{U}_{L_{t+2}}(t) - \mathrm{L}_{L_{t+1}}(t) &> \epsilon \\ \hat{\mu}_{L_{t+2}}(t) + \sqrt{\frac{\beta(N_{L_{t+2}}(t),\delta)}{2N_{L_{t+2}}(t)}} - \hat{\mu}_{L_{t+1}}(t) + \sqrt{\frac{\beta(N_{L_{t+1}}(t),\delta)}{2N_{L_{t+1}}(t)}} &> \epsilon \\ \sqrt{\frac{\beta(N_{L_{t+2}}(t),\delta)}{2N_{L_{t+2}}(t)}} + \sqrt{\frac{\beta(N_{L_{t+1}}(t),\delta)}{2N_{L_{t+1}}(t)}} &> \epsilon \end{aligned}$$

using that by definition of $\underline{a}_t$, $\hat{\mu}_{L_{t+2}}(t) < \hat{\mu}_{L_{t+1}}(t)$. Hence, there exists $\ell \in \{L_{t+1}, L_{t+2}\}$ such that

$$\sqrt{\frac{\beta(N_\ell(t),\delta)}{2N_\ell(t)}} > \frac{\epsilon}{2} \;\Rightarrow\; \sqrt{\frac{2\beta(t,\delta)}{N_\ell(t)}} > \epsilon.$$

Statement 2. We consider two cases and first assume that $\gamma \in \mathcal{I}_{L_{t+1}}(t)$ and $\gamma \geq \mathrm{U}_{L_{t+2}}(t)$. Using the fact that the algorithm doesn't stop at round t, the following events hold

$$\begin{aligned} & (\mathrm{U}_{L_{t+2}}(t) - \mathrm{L}_{L_{t+1}}(t) > \epsilon) \cap (\mathrm{U}_{L_{t+1}}(t) > \gamma) \cap (\mathrm{U}_{L_{t+2}}(t) \leq \gamma) \\ \Rightarrow\; & (\mathrm{U}_{L_{t+2}}(t) - \mathrm{L}_{L_{t+1}}(t) > \epsilon) \cap (\mathrm{U}_{L_{t+1}}(t) > \gamma) \cap (\mathrm{U}_{L_{t+2}}(t) - \mathrm{L}_{L_{t+1}}(t) + \mathrm{L}_{L_{t+1}}(t) \leq \gamma) \\ \Rightarrow\; & (\mathrm{U}_{L_{t+1}}(t) > \gamma) \cap (\mathrm{L}_{L_{t+1}}(t) \leq \gamma - \epsilon) \\ \Rightarrow\; & (\mathrm{U}_{L_{t+1}}(t) - \mathrm{L}_{L_{t+1}}(t) > \epsilon) \\ \Rightarrow\; & \left(\sqrt{\frac{2\beta(t,\delta)}{N_{L_{t+1}}(t)}} > \epsilon\right). \end{aligned}$$

The second case is $\gamma \in \mathcal{I}_{L_{t+1}}(t)$ and $\gamma \leq \mathrm{L}_{L_{t+2}}(t)$. Then the following holds

$$\begin{aligned} & (\mathrm{U}_{L_{t+2}}(t) - \mathrm{L}_{L_{t+1}}(t) > \epsilon) \cap (\mathrm{L}_{L_{t+1}}(t) \leq \gamma) \cap (\mathrm{L}_{L_{t+2}}(t) \geq \gamma) \\ \Rightarrow\; & (\mathrm{L}_{L_{t+1}}(t) \leq \gamma) \cap (\mathrm{U}_{L_{t+2}}(t) + \mathrm{L}_{L_{t+2}}(t) - \mathrm{L}_{L_{t+1}}(t) > \gamma + \epsilon) \\ \Rightarrow\; & (\mathrm{L}_{L_{t+1}}(t) \leq \gamma) \cap \left(2\hat{\mu}_{L_{t+2}}(t) - \hat{\mu}_{L_{t+1}}(t) + \sqrt{\frac{2\beta(N_{L_{t+1}}(t),\delta)}{2N_{L_{t+1}}(t)}} > \gamma + \epsilon\right) \\ \Rightarrow\; & (\mathrm{L}_{L_{t+1}}(t) \leq \gamma) \cap \left(\hat{\mu}_{L_{t+1}}(t) + \sqrt{\frac{2\beta(N_{L_{t+1}}(t),\delta)}{2N_{L_{t+1}}(t)}} > \gamma + \epsilon\right) \\ \Rightarrow\; & (\mathrm{U}_{L_{t+1}}(t) - \mathrm{L}_{L_{t+1}}(t) > \epsilon), \end{aligned}$$

where the third implication uses the fact that $\hat{\mu}_{L_{t+2}}(t) \leq \hat{\mu}_{L_{t+1}}(t)$.

Statement 3. We consider two cases and first assume that $\gamma \in \mathcal{I}_{L_{t+2}}(t)$ and $\gamma \leq \mathrm{L}_{L_{t+1}}(t)$. Using the fact that the algorithm doesn't stop at round t, the following events hold

$$\begin{aligned} & (\mathrm{U}_{L_{t+2}}(t) - \mathrm{L}_{L_{t+1}}(t) > \epsilon) \cap (\mathrm{L}_{L_{t+1}}(t) \geq \gamma) \cap (\mathrm{L}_{L_{t+2}}(t) \leq \gamma) \\ \Rightarrow\; & (\mathrm{U}_{L_{t+2}}(t) > \gamma + \epsilon) \cap (\mathrm{L}_{L_{t+2}}(t) \leq \gamma) \\ \Rightarrow\; & (\mathrm{U}_{L_{t+2}}(t) - \mathrm{L}_{L_{t+2}}(t) > \epsilon) \\ \Rightarrow\; & \left(\sqrt{\frac{2\beta(t,\delta)}{N_{L_{t+2}}(t)}} > \epsilon\right). \end{aligned}$$

The second case is $\gamma \in \mathcal{I}_{L_{t+2}}(t)$ and $\gamma \geq \mathrm{U}_{L_{t+1}}(t)$. Then the following holds

$$\begin{aligned}
& (\mathrm{U}_{L_{t+2}}(t) - \mathrm{L}_{L_{t+1}}(t) > \epsilon) \cap (\mathrm{U}_{L_{t+2}}(t) \geq \gamma) \cap (\gamma \geq \mathrm{U}_{L_{t+1}}(t)) \\
\Rightarrow \quad & (\mathrm{U}_{L_{t+2}}(t) - (\mathrm{L}_{L_{t+1}}(t) + \mathrm{U}_{L_{t+1}}(t)) > \epsilon - \gamma) \cap (\mathrm{U}_{L_{t+2}}(t) \geq \gamma) \\
\Rightarrow \quad & \left(\hat{\mu}_{L_{t+2}}(t) + \sqrt{\frac{2\beta(N_{L_{t+2}}(t), \delta)}{2N_{L_{t+2}}(t)}} - 2\hat{\mu}_{L_{t+1}}(t) > \epsilon - \gamma \right) \cap (\mathrm{U}_{L_{t+2}}(t) \geq \gamma) \\
\Rightarrow \quad & \left(-\hat{\mu}_{L_{t+2}}(t) + \sqrt{\frac{2\beta(N_{L_{t+2}}(t), \delta)}{2N_{L_{t+2}}(t)}} > \epsilon - \gamma \right) \cap (\mathrm{U}_{L_{t+2}}(t) \geq \gamma) \\
\Rightarrow \quad & (\mathrm{U}_{L_{t+2}}(t) - \mathrm{L}_{L_{t+2}}(t) > \epsilon),
\end{aligned}$$

where the third implication uses the fact that $\hat{\mu}_{L_{t+2}}(t) \leq \hat{\mu}_{L_{t+1}}(t)$.

E Proof of the lower bounds

E.1 Proof of Theorem 6

Theorem 6 follows from considering the best possible change of distribution $\boldsymbol{\lambda} \in \mathrm{Alt}(\boldsymbol{\mu})$. The expected log-likelihood ratio of the observations until τ under a model parameterized by $\boldsymbol{\mu}$ and a model parameterized by $\boldsymbol{\lambda}$ is

$$\mathbb{E}_{\boldsymbol{\mu}}[L_\tau(\boldsymbol{\mu}, \boldsymbol{\lambda})] = \sum_{\ell \in \mathcal{L}} \mathbb{E}_{\boldsymbol{\mu}}[N_\ell(\tau)] d(\mu_\ell, \lambda_\ell),$$

where $N_\ell(t)$ is the number of draws of the leaf ℓ until round t. Using Lemma 1 of [14], for any event $\mathcal{E}$ in the filtration generated by τ,

$$\mathbb{E}_{\boldsymbol{\mu}}[L_\tau(\boldsymbol{\mu}, \boldsymbol{\lambda})] \geq d(\mathbb{P}_{\boldsymbol{\mu}}(\mathcal{E}), \mathbb{P}_{\boldsymbol{\lambda}}(\mathcal{E})).$$

As the strategy is δ-correct, letting $\mathcal{E} = (\hat{s}_\tau = s^*(\boldsymbol{\mu}))$ one has $\mathbb{P}_{\boldsymbol{\mu}}(\mathcal{E}) \geq 1 - \delta$ and $\mathbb{P}_{\boldsymbol{\lambda}}(\mathcal{E}) \leq \delta$ (under this model, $s^*(\boldsymbol{\mu})$ is not the best action at the root under the model parameterized by $\boldsymbol{\lambda}$). Using monotonicity properties of the Bernoulli KL-divergence, one obtains, for any $\boldsymbol{\lambda} \in \mathrm{Alt}(\boldsymbol{\mu})$,

$$\sum_{\ell \in \mathcal{L}} \mathbb{E}_{\boldsymbol{\mu}}[N_\ell(\tau)] d(\mu_\ell, \lambda_\ell) \geq d(1-\delta, \delta).$$

Then, one can write

$$\begin{aligned}
\inf_{\boldsymbol{\lambda} \in \mathrm{Alt}(\boldsymbol{\mu})} \sum_{\ell \in \mathcal{L}} \mathbb{E}_{\boldsymbol{\mu}}[N_\ell(\tau)] d(\mu_\ell, \lambda_\ell) \quad &\geq \quad d(1-\delta, \delta) \\
\mathbb{E}_{\boldsymbol{\mu}}[\tau] \inf_{\boldsymbol{\lambda} \in \mathrm{Alt}(\boldsymbol{\mu})} \sum_{\ell \in \mathcal{L}} \frac{\mathbb{E}_{\boldsymbol{\mu}}[N_\ell(\tau)]}{\mathbb{E}_{\boldsymbol{\mu}}[\tau]} d(\mu_\ell, \lambda_\ell) \quad &\geq \quad d(1-\delta, \delta) \\
\mathbb{E}_{\boldsymbol{\mu}}[\tau] \left(\sup_{\boldsymbol{w} \in \Sigma_{|\mathcal{L}|}} \inf_{\boldsymbol{\lambda} \in \mathrm{Alt}(\boldsymbol{\mu})} \sum_{\ell \in \mathcal{L}} \frac{\mathbb{E}_{\boldsymbol{\mu}}[N_\ell(\tau)]}{\mathbb{E}_{\boldsymbol{\mu}}[\tau]} d(\mu_\ell, \lambda_\ell) \right) \quad &\geq \quad d(1-\delta, \delta),
\end{aligned}$$

using that $\sum_{\ell \in \mathcal{L}} \frac{\mathbb{E}_{\boldsymbol{\mu}}[N_\ell(\tau)]}{\mathbb{E}_{\boldsymbol{\mu}}[\tau]} = 1$. This concludes the proof.

One can also note that for an algorithm to match the lower bound, all the inequalities above should be equalities. In particular one would need $w^*_\ell(\boldsymbol{\mu}) \simeq \frac{\mathbb{E}_{\boldsymbol{\mu}}[N_\ell(\tau)]}{\mathbb{E}_{\boldsymbol{\mu}}[\tau]}$, where $w^*_\ell(\boldsymbol{\mu})$ is a maximizer in the definition of $T^*(\boldsymbol{\mu})^{-1}$ in (4).

E.2 Proof of Lemma 7

In the particular case of a depth-two tree with K actions for player A and M actions for player B,

$$T^*(\boldsymbol{\mu})^{-1} = \sup_{\boldsymbol{w}\in\Sigma_{K\times M}} \inf_{\boldsymbol{\lambda}\in \mathrm{Alt}(\boldsymbol{\mu})} \sum_{k=1}^{K}\sum_{m=1}^{K} w_{k,m} d(\mu_{k,m}, \lambda_{k,m}).$$

From the particular structure of $\boldsymbol{\mu}$, the best action at the root is action $i = 1$. Hence

$$\mathrm{Alt}(\boldsymbol{\mu}) = \{\boldsymbol{\lambda} : \exists a \in \{1,\dots,M\}, \exists i \in \{2,\dots,K\} : \forall j \in \{1,\dots,M\}, \lambda_{1,a} < \lambda_{i,j}\}.$$

It follows that

$$\begin{aligned} T^*(\boldsymbol{\mu})^{-1} &= \sup_{\boldsymbol{w}\in\Sigma_{K\times M}} \min_{\substack{a\in\{1,\dots,M\}\\ i\in\{2,\dots,K\}}} \inf_{\boldsymbol{\lambda}:\forall j, \lambda_{1,a}<\lambda_{i,j}} \sum_{k=1}^{K}\sum_{m=1}^{M} w_{k,m} d(\mu_{k,m}, \lambda_{k,m}) \\ &= \sup_{\boldsymbol{w}\in\Sigma_{K\times M}} \min_{\substack{a\in\{1,\dots,M\}\\ i\in\{2,\dots,K\}}} \underbrace{\inf_{\boldsymbol{\lambda}:\forall j, \lambda_{1,a}<\lambda_{i,j}} \left[w_{1,a} d(\mu_{1,a}, \lambda_{1,a}) + \sum_{j=1}^{M} w_{i,j} d(\mu_{i,j}, \lambda_{i,j}) \right]}_{:= F_{a,i}(\boldsymbol{\mu}, \boldsymbol{w})}. \end{aligned} \tag{13}$$

Indeed, in the rightmost constrained minimization problem, all the $\lambda_{k,m}$ on which no constraint lie can be set to $\mu_{k,m}$ to minimize the corresponding term in the sum. Using tools from constrained optimization, one can prove that

$$F_{a,i}(\boldsymbol{\mu}, \boldsymbol{w}) = \inf_{(\lambda_{1,a}, (\lambda_{i,j})_j)\in\mathcal{C}} \left[w_{1,a} d(\mu_{1,a}, \lambda_{1,a}) + \sum_{j=1}^{M} w_{i,j} d(\mu_{i,j}, \lambda_{i,j}) \right],$$

where $\mathcal{C}$ is the set

$$\mathcal{C} = \{(\mu', \boldsymbol{\mu}') \in [0,1]^{M+1} : \exists j_0 \in \{1,\dots,K\}, \exists c \in [\mu_{i,j_0}, \mu_{1,a}] : \mu' = \boldsymbol{\mu}'_1 = \cdots = \boldsymbol{\mu}'_{j_0} = c \text{ and } \boldsymbol{\mu}'_j = \mu_{i,j} \text{ for } j > j_0\}.$$

Letting $H(\mu', \boldsymbol{\mu}', w, \boldsymbol{w}) = w_{1,a} d(\mu_{1,a}, \mu') + \sum_{j=1}^{M} w_{i,j} d(\mu_{i,j}, \boldsymbol{\mu}'_j)$ one can easily show that for all $(\mu', \boldsymbol{\mu}') \in \mathcal{C}$,

$$H(\mu', \boldsymbol{\mu}', w, \boldsymbol{w}) \le H(\mu', \boldsymbol{\mu}', w, \tilde{\boldsymbol{w}}),$$

where $\tilde{\boldsymbol{w}}$ is constructed from $\boldsymbol{w}$ by putting all the weight on the smallest arm:

$$\tilde{w}_{i,1} = \sum_{j\ge 1} w_{1,j} \text{ and } \tilde{w}_{i,j} = 0 \text{ for } j \ge 2.$$

This is because the largest $d(\mu_{i,j}, c)$ is $d(\mu_{i,1}, c)$ as $\mu_{i,1} \le \mu_{i,j} \le c$ for $j \le j_0$. Hence taking the infimum, one obtains

$$F_{a,i}(\boldsymbol{\mu}, \boldsymbol{w}) \le F_{a,i}(\boldsymbol{\mu}, \tilde{\boldsymbol{w}}).$$

Repeating this argument for all i, one can construct $\tilde{\boldsymbol{w}}$ such that

$$\forall i \ge 2, \tilde{w}_{i,1} = \sum_{j\ge 1} w_{i,j} \text{ and } \tilde{w}_{i,j} = 0 \text{ for } j \ge 2$$

and $F_{a,i}(\boldsymbol{\mu}, \boldsymbol{w}) \le F_{a,i}(\boldsymbol{\mu}, \tilde{\boldsymbol{w}})$ for all a, i. Thus, the supremum in 13 is necessarily attained for $\boldsymbol{w}$ in the set $\tilde{\Sigma}_{K\times M} = \{\boldsymbol{w} \in \Sigma_{K,M} : w_{i,j} = 0 \text{ for } i \ge 2, j \ge 2\}$. It follows that

$$\begin{aligned} T^*(\boldsymbol{\mu})^{-1} &= \sup_{\boldsymbol{w}\in\tilde{\Sigma}_{M+K-1}} \min_{\substack{a\in\{1,\dots,M\}\\ i\in\{2,\dots,K\}}} \inf_{\boldsymbol{\lambda}:\forall j, \lambda_{1,a}<\lambda_{i,1}} \left[w_{1,a} d(\mu_{1,a}, \lambda_{1,a}) + w_{i,1} d(\mu_{i,1}, \lambda_{i,1}) \right] \\ &= \sup_{\boldsymbol{w}\in\tilde{\Sigma}_{K+M-1}} \min_{\substack{a=1,\dots,M\\ i=2,\dots,K}} \left[w_{1,a} d\left(\mu_{1,a}, \frac{w_{1,a}\mu_{1,a} + w_{i,1}\mu_{i,1}}{w_{1,a} + w_{i,1}}\right) + w_{i,1} d\left(\mu_{i,1}, \frac{w_{1,a}\mu_{1,a} + w_{i,1}\mu_{i,1}}{w_{1,a} + w_{i,1}}\right) \right], \end{aligned}$$

which concludes the proof.