



INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

## *Bandit Algorithms for Tree Search*

Pierre-Arnaud Coquelin — Rémi Munos

**N° 6141**

March 2007

Thème COG

A large, light gray, stylized letter 'R' that serves as a background for the text 'Rapport de recherche'.

*Rapport  
de recherche*





## Bandit Algorithms for Tree Search

Pierre-Arnaud Coquelin <sup>\*</sup>, Rémi Munos <sup>†</sup>

Thème COG — Systèmes cognitifs  
Projet SequeL

Rapport de recherche n° 6141 — March 2007 — 19 pages

**Abstract:** Bandit based methods for tree search have recently gained popularity when applied to huge trees, e.g. in the game of go [GWMT06]. The *UCT algorithm* [KS06], a tree search method based on Upper Confidence Bounds (UCB) [ACBF02], is believed to adapt locally to the effective smoothness of the tree. However, we show that UCT is too “optimistic” in some cases, leading to a regret  $\Omega(\exp(\exp(D)))$  where  $D$  is the depth of the tree. We propose alternative bandit algorithms for tree search. First, a modification of UCT using a confidence sequence that scales exponentially with the horizon depth is proven to have a regret  $O(2^D \sqrt{n})$ , but does not adapt to possible smoothness in the tree. We then analyze *Flat-UCB* performed on the leaves and provide a finite regret bound with high probability. Then, we introduce a UCB-based *Bandit Algorithm for Smooth Trees* which takes into account actual smoothness of the rewards for performing efficient “cuts” of sub-optimal branches with high confidence. Finally, we present an incremental tree search version which applies when the full tree is too big (possibly infinite) to be entirely represented and show that with high probability, essentially only the optimal branches is indefinitely developed. We illustrate these methods on a global optimization problem of a Lipschitz function, given noisy data.

**Key-words:** Bandit algorithms, tree search, exploration-exploitation tradeoff, upper confidence bounds, minimax game, reinforcement learning

<sup>\*</sup> CMAP, Ecole Polytechnique, [coquelin@cmappx.polytechnique.fr](mailto:coquelin@cmappx.polytechnique.fr)

<sup>†</sup> SequeL, INRIA, [remi.munos@inria.fr](mailto:remi.munos@inria.fr)

## Bandit Algorithms for Tree Search

**Résumé :** Les méthodes de recherche arborescentes utilisant des algorithmes de bandit ont récemment connu une forte popularité, pour leur capacité de traiter des grands arbres, par exemple pour le jeu de go [GWMT06]. Il est connu que l’algorithme UCT [KS06], une méthode de recherche arborescente basée sur des intervalles de confiance (algorithme *Upper Confidence Bounds* (UCB) de [ACBF02]), s’adapte localement à la profondeur effective de l’arbre. Cependant, nous montrons ici que UCT peut être trop “optimiste” dans certains cas, menant à un regret  $\Omega(\exp(\exp(D)))$  où  $D$  est la profondeur de l’arbre. Nous proposons plusieurs alternatives d’algorithmes de bandit pour la recherche arborescente. Tout d’abord, nous proposons une modification d’UCT utilisant un intervalle de confiance qui croît exponentiellement avec la profondeur de l’horizon de l’arbre, et montrons qu’il mène à un regret  $O(2^D \sqrt{n})$  mais ne s’adapte pas à la régularité de l’arbre. Puis nous analysons un algorithme *Flat-UCB* de bandit de type UCB directement sur les feuilles et prouvons une borne finie (indépendante de  $n$ ) sur le regret avec forte probabilité. Ensuite, nous introduisons un algorithme *Bandit Algorithm for Smooth Trees* qui prend en compte d’éventuelles régularités dans l’arbre pour réaliser des “coupes” efficaces de branches sous-optimales avec grande confiance. Enfin, nous présentons une version incrémentale de recherche arborescente qui s’applique lorsque l’arbre est trop grand (voire infini) pour pouvoir être représenté entièrement, et montrons qu’essentiellement, et avec forte probabilité, seule la branche optimale est indéfiniment développée. Nous illustrons ces méthodes sur un problème d’optimisation d’une fonction Lipschitzienne, à partir de données bruitées.

**Mots-clés :** Algorithmes de bandit, recherche arborescente, compromis exploration-exploitation, bornes supérieures d’intervalles de confiance, jeux minimax, apprentissage par renforcement

## 1 Introduction

Bandit algorithms have been used recently for tree search, because of their efficient trade-off between exploration of the most uncertain branches and exploitation of the most promising ones, leading to very promising results in dealing with huge trees (e.g. the go program MoGo, see [GWMT06]). In this paper we focus on Upper Confidence Bound (UCB) bandit algorithms [ACBF02] applied to tree search, such as UCT (Upper Confidence Bounds applied to Trees) [KS06]. The general procedure is described by Algorithm 1 and depends on the way the upper-bounds  $B_{i,p,n_i}$  for each node  $i$  are maintained.

---

**Algorithm 1** Bandit Algorithm for Tree Search
 

---

```

for  $n \geq 1$  do
  Run  $n$ -th trajectory from the root to a leaf:
  Set the current node  $i_0$  to the root
  for  $d = 1$  to  $D$  do
    Select node  $i_d$  as the children  $j$  of node  $i_{d-1}$  that maximizes  $B_{j,n_{i_{d-1}},n_j}$ 
  end for
  Receive reward  $x_n \stackrel{iid}{\sim} X_{i_D}$ 
  Update the nodes visited by this trajectory:
  for  $d = D$  to  $0$  do
    Update the number of visits:  $n_{i_d} = n_{i_d} + 1$ 
    Update the bound  $B_{i_d,n_{i_{d-1}},n_{i_d}}$ 
  end for
end for

```

---

A trajectory is a sequence of nodes from the root to a leaf, where at each node, the next node is chosen as the one maximizing its  $B$  value among the children. A reward is received at the leaf. After a trajectory is run, the  $B$  values of each node in the trajectory are updated. In the case of UCT, the upper-bound  $B_{i,p,n_i}$  of a node  $i$ , given that the node has already been visited  $n_i$  times and its parent's node  $p$  times, is the average of the rewards  $\{x_t\}_{1 \leq t \leq n_i}$  obtained from that node  $X_{i,n_i} = \frac{1}{n_i} \sum_{t=1}^{n_i} x_t$  plus a confidence interval, derived from a Chernoff-Hoeffding bound (see e.g. [GDL96]):

$$B_{i,p,n_i} \stackrel{\text{def}}{=} X_{i,n_i} + \sqrt{\frac{2 \log(p)}{n_i}} \quad (1)$$

In this paper we consider a max search (the minimax problem is a direct generalization of the results presented here) in binary trees (i.e. there are 2 actions in each node), although the extension to more actions is straightforward. Let a binary tree of depth  $D$  where at each leaf  $i$  is assigned a random variable  $X_i$ , with bounded support  $[0, 1]$ , whose law is unknown. Successive visits of a leaf  $i$  yield a sequence of independent and identically distributed (i.i.d.) samples  $x_{i,t} \sim X_i$ , called rewards, or payoff. The value of a leaf  $i$  is its expected reward:

$\mu_i \stackrel{\text{def}}{=} \mathbb{E}X_i$ . Now we define the value of any node  $i$  as the maximal value of the leaves in the branch starting from node  $i$ . Our goal is to compute the value  $\mu^*$  of the root.

An optimal leaf is a leaf having the largest expected reward. We will denote by  $*$  quantities related to an optimal node. For example  $\mu^*$  denote  $\max_i \mu_i$ . An optimal branch is a sequence of nodes from the root to a leaf, having the  $\mu^*$  value. We define the *regret* up to time  $n$  as the difference between the optimal expected payoff and the sum of obtained rewards:

$$R_n \stackrel{\text{def}}{=} \mu^* - \sum_{t=1}^n x_{i_t, t},$$

where  $i_t$  is the chosen leaf at round  $t$ . We also define the *pseudo-regret* up to time  $n$ :

$$\bar{R}_n \stackrel{\text{def}}{=} \mu^* - \sum_{t=1}^n \mu_{i_t} = \sum_{j \in \mathcal{L}} n_j \Delta_j,$$

where  $\mathcal{L}$  is the set of leaves,  $\Delta_j \stackrel{\text{def}}{=} \mu^* - \mu_j$ , and  $n_j$  is the random variable that counts the number of times leaf  $j$  has been visited up to time  $n$ . The pseudo-regret may thus be analyzed by estimating the number of times each sub-optimal leaf is visited.

In tree search, our goal is thus to find an exploration policy of the branches such as to minimize the regret, in order to select an optimal leaf as fast as possible. Now, thanks to a simple contraction of measure phenomenon, the regret per bound  $R_n/n$  turns out to be very close to the pseudo regret per round  $\bar{R}_n/n$ . Indeed, using Azuma's inequality for martingale difference sequences (see Proposition 1), with probability at least  $1 - \beta$ , we have at time  $n$ ,

$$\frac{1}{n} |R_n - \bar{R}_n| \leq \sqrt{\frac{2 \log(2/\beta)}{n}}.$$

The fact that  $R(n) - \bar{R}_n$  is a martingale difference sequence comes from the property that, given the filtration  $\mathcal{F}_{t-1}$  defined by the random samples up to time  $t - 1$ , the expectation of the next reward  $\mathbb{E}x_t$  is conditioned to the leaf  $i_t$  chosen by the algorithm:  $\mathbb{E}[x_t | \mathcal{F}_{t-1}] = \mu_{i_t}$ . Thus  $R_n - \bar{R}_n = \sum_{t=1}^n x_t - \mu_{i_t}$  with  $\mathbb{E}[x_t - \mu_{i_t} | \mathcal{F}_{t-1}] = 0$ . Hence, we will only focus on providing high probability bounds on the pseudo-regret.

First, we analyze the UCT algorithm defined by the upper confidence bound (1). We show that its behavior is risky and may lead to a regret as bad as  $\Omega(\exp(\dots \exp(D) \dots))$  ( $D - 1$  composed exponential functions). We modify the algorithm by increasing the exploration sequence, defining:

$$B_{i,p,n_i} \stackrel{\text{def}}{=} X_{i,n_i} + \sqrt{\frac{\sqrt{p}}{n_i}}. \quad (2)$$

This yields an improved worst-case behavior over regular UCT, but the regret may still be as bad as  $\Omega(\exp(\exp(D)))$  (see Section 2). We then propose in Section 3 a *modified UCT* based on the bound (2), where the confidence interval is multiplied by a factor that scales exponentially with the horizon depth. We derive a worst-case regret  $O(2^D / \sqrt{n})$  with high

probability. However this algorithm does not adapt to the effective smoothness of the tree, if any.

Next we analyze the *Flat-UCB* algorithm, which simply performs UCB directly on the leaves. With a slight modification of the usual confidence sequence, we show in Section 4 that this algorithm has a finite regret  $O(2^D/\Delta)$  (where  $\Delta = \min_{i, \Delta_i > 0} \Delta_i$ ) with high probability.

In Section 5, we introduce a UCB-based algorithm, called *Bandit Algorithm for Smooth Trees*, which takes into account actual smoothness of the rewards for performing efficient “cuts” of sub-optimal branches based on concentration inequality. We give a numerical experiment for the problem of optimizing a Lipschitz function given noisy observations.

Finally, in Section 6 we present and analyze a growing tree search, which builds incrementally the tree by expanding, at each iteration, the most promising node. This method is memory efficient and well adapted to search in large (possibly infinite) trees.

**Additional notations:** Let  $\mathcal{L}$  denotes the set of leaves and  $\mathcal{S}$  the set of sub-optimal leaves. For any node  $i$ , we write  $\mathcal{L}(i)$  the set of leaves in the branch starting from node  $i$ . For any node  $i$ , we write  $n_i$  the number of times node  $i$  has been visited up to round  $n$ , and we define the cumulative rewards:

$$X_{i,n_i} = \frac{1}{n_i} \sum_{j \in \mathcal{L}(i)} n_j X_{j,n_j},$$

the cumulative expected rewards:

$$\bar{X}_{i,n_i} = \frac{1}{n_i} \sum_{j \in \mathcal{L}(i)} n_j \mu_j,$$

and the pseudo-regret:

$$\bar{R}_{i,n_i} = \sum_{j \in \mathcal{L}(i)} n_j (\mu_i - \mu_j).$$

## 2 Lower regret bound for UCT

The UCT algorithm introduced in [KS06] is believed to adapt automatically to the effective (and a priori unknown) smoothness of the tree: If the tree possesses an effective depth  $d < D$  (i.e. if all leaves of a branch starting from a node of depth  $d$  have the same value) then its regret will be equal to the regret of a tree of depth  $d$ . First, we notice that the bound (1) is not a true upper confidence bound on the value  $\mu_i$  of a node  $i$  since the rewards received at node  $i$  are not identically distributed (because the chosen leaves depend on a non-stationary node selection process). However, due to the increasing confidence term  $\log(p)$  when a node is not chosen, all nodes will be infinitely visited, which guarantees an asymptotic regret of  $O(\log(n))$ . However the transitory phase may last very long.

Indeed, consider the example illustrated in Figure 1. The rewards are deterministic and for a node of depth  $d$  in the optimal branch (obtained after choosing  $d$  times action 1), if action 2 is chosen, then a reward of  $\frac{D-d}{D}$  is received (all leaves in this branch have the same reward). If action 1 is chosen, then this moves to the next node in the optimal branch. At depth  $D-1$ , action 1 yields reward 1 and action 2, reward 0. We assume that when a node is visited for the first time, the algorithm starts by choosing action 2 before choosing action 1.

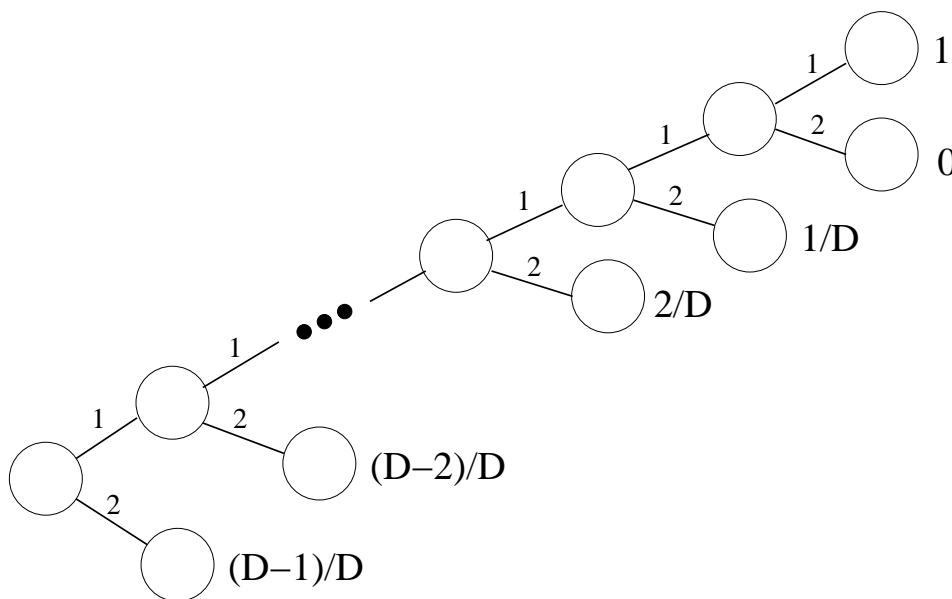


Figure 1: A bad example for UCT. From the root (left node), action 2 leads to a node from which all leaves yield reward  $\frac{D-1}{D}$ . The optimal branch consists in choosing always action 1, which yields reward 1. In the beginning, the algorithm believes the arm 2 is the best, spending most of its times exploring this branch (as well as all other sub-optimal branches). It takes  $\Omega(\exp(\exp(D)))$  rounds to get the 1 reward!

We now establish a lower bound on the number of times suboptimal rewards are received before getting the optimal 1 reward for the first time. Write  $n$  the first instant when the optimal leaf is reached. Write  $n_d$  the number of times the node (also written  $d$  making a slight abuse of notation) of depth  $d$  in the optimal branch is reached. Thus  $n = n_0$  and  $n_D = 1$ . At depth  $D-1$ , we have  $n_{D-1} = 2$  (since action 2 has been chosen once in node  $D-1$ ).

We consider both the logarithmic confidence sequence used in (1) and the square root sequence in (2). Let us start with the square root confidence sequence (2). At depth  $d-1$ , since the optimal branch is followed by the  $n$ -th trajectory, we have (writing  $d'$  the node



resulting from action 2 in the node  $d - 1$ ):

$$X_{d',n_{d'}} + \sqrt{\frac{\sqrt{n_{d-1}}}{n_{d'}}} \leq X_{d,n_d} + \sqrt{\frac{\sqrt{n_{d-1}}}{n_d}}.$$

But  $X_{d',n_{d'}} = (D - d)/D$  and  $X_{d,n_d} \leq (D - (d + 1))/D$  since the 1 reward has not been received before. We deduce that

$$\frac{1}{D} \leq \sqrt{\frac{\sqrt{n_{d-1}}}{n_d}}.$$

Thus for the square root confidence sequence, we have  $n_{d-1} \geq n_d^2/D^4$ . Now, by induction,

$$n \geq \frac{n_1^2}{D^4} \geq \frac{n_2^2}{D^{4(1+2)}} \geq \frac{n_3^2}{D^{4(1+2+3)}} \geq \dots \geq \frac{n_{D-1}^2}{D^{2D(D-1)}}$$

Since  $n_{D-1} = 2$ , we obtain  $n \geq \frac{2^{2^{D-1}}}{D^{2D(D-1)}}$ . This is a double exponential dependency w.r.t.  $D$ . For example, for  $D = 20$ , we have  $n \geq 10^{156837}$ . Consequently, the regret is also  $\Omega(\exp(\exp(D)))$ .

Now, the usual logarithmic confidence sequence defined by (1) yields an even worst lower bound on the regret since we may show similarly that  $n_{d-1} \geq \exp(n_d/(2D^2))$  thus  $n \geq \exp(\exp(\dots \exp(2) \dots))$  (composition of  $D - 1$  exponential functions).

Thus, although UCT algorithm has asymptotically regret  $O(\log(n))$  in  $n$ , (or  $O(\sqrt{n})$  for the square root sequence), the transitory regret is  $\Omega(\exp(\exp(\dots \exp(2) \dots)))$  (or  $\Omega(\exp(\exp(D)))$  in the square root sequence).

The reason for this bad behavior is that the algorithm is too optimistic (it does not explore enough and may take a very long time to discover good branches that looked initially bad) since the bounds (1) and (2) are not true upper bounds.

### 3 Modified UCT

We modify the confidence sequence to explore more the nodes close to the root than the leaves, taking into account the fact that the time needed to decrease the bias ( $\mu_i - \mathbb{E}[X_{i,n_i}]$ ) at a node  $i$  of depth  $d$  increases with the depth horizon  $(D - d)$ . For such a node  $i$  of depth  $d$ , we define the upper confidence bound:

$$B_{i,n_i} \stackrel{\text{def}}{=} X_{i,n_i} + (k_d + 1) \sqrt{\frac{2 \log(\beta_{n_i}^{-1})}{n_i}} + \frac{k'_d}{n_i}, \quad (3)$$

where  $\beta_n \stackrel{\text{def}}{=} \frac{\beta}{2Nn(n+1)}$  with  $N = 2^{D+1} - 1$  the number of nodes in the tree, and the coefficients:

$$\begin{aligned} k_d &\stackrel{\text{def}}{=} \frac{1 + \sqrt{2}}{\sqrt{2}} [(1 + \sqrt{2})^{D-d} - 1] \\ k'_d &\stackrel{\text{def}}{=} (3^{D-d} - 1)/2 \end{aligned} \quad (4)$$

Notice that we used a simplified notation, writing  $B_{i,n_i}$  instead of  $B_{i,p,n_i}$  since the bound does not depend on the number of visits of the parent's node.

**Theorem 1.** *Let  $\beta > 0$ . Consider Algorithm 1 with the upper confidence bound (3). Then, with probability at least  $1 - \beta$ , for all  $n \geq 1$ , the pseudo-regret is bounded by*

$$\bar{R}_n \leq \frac{1 + \sqrt{2}}{\sqrt{2}} [(1 + \sqrt{2})^D - 1] \sqrt{2 \log(\beta_n^{-1})n} + \frac{3^D - 1}{2}$$

*Proof.* We first remind Azuma's inequality (see [GDL96]):

**Proposition 1.** *Let  $f$  be a Lipschitz function of  $n$  independent random variables such that  $f - \mathbb{E}f = \sum_{i=1}^n d_i$  where  $(d_i)_{1 \leq i \leq n}$  is a martingale difference sequence, i.e.:  $\mathbb{E}[d_i | \mathcal{F}_{i-1}] = 0$ ,  $1 \leq i \leq n$ , such that  $\|d_i\|_\infty \leq 1/n$ . Then for every  $\epsilon > 0$ ,*

$$\mathbb{P}(|f - \mathbb{E}f| > \epsilon) \leq 2 \exp(-n\epsilon^2/2).$$

We apply Azuma's inequality to the random variables  $Y_{i,n_i}$  and  $Z_{i,n_i}$ , defined respectively, for all nodes  $i$ , by  $Y_{i,n_i} \stackrel{\text{def}}{=} X_{i,n_i} - \bar{X}_{i,n_i}$ , and for all non-leaf nodes  $i$ , by

$$Z_{i,n_i} \stackrel{\text{def}}{=} \frac{1}{n_i} (n_{i_1} Y_{i_1,n_{i_1}} - n_{i_2} Y_{i_2,n_{i_2}}),$$

where  $i_1$  and  $i_2$  denote the children of  $i$ .

Since at each round  $t \leq n_i$ , the choice of the next leaf only depends on the random samples drawn at previous times  $s < t$ , we have  $\mathbb{E}[Y_{i,t} | \mathcal{F}_{t-1}] = 0$  and  $\mathbb{E}[Z_{i,t} | \mathcal{F}_{t-1}] = 0$ , (i.e..  $Y$  and  $Z$  are martingale difference sequences), and Azuma's inequality gives that, for any node  $i$ , for any  $n_i$ , for any  $\epsilon > 0$ ,  $\mathbb{P}(|Y_{i,n_i}| > \epsilon) \leq 2 \exp(-n_i \epsilon^2/2)$  and  $\mathbb{P}(|Z_{i,n_i}| > \epsilon) \leq 2 \exp(-n_i \epsilon^2/2)$ .

We now define a confidence level  $c_{n_i}$  such that with probability at least  $1 - \beta$ , the random variables  $Y_{i,n_i}$  and  $Z_{i,n_i}$  belong to their confidence intervals for all nodes and for all times. More precisely, let  $\mathcal{E}$  be the event under which, for all  $n_i \geq 1$ , for all nodes  $i$ ,  $|Y_{i,n_i}| \leq c_{n_i}$  and for all non-leaf nodes  $i$ ,  $|Z_{i,n_i}| \leq c_{n_i}$ . Then, by defining

$$c_n \stackrel{\text{def}}{=} \sqrt{\frac{2 \log(\beta_n^{-1})}{n}}, \text{ with } \beta_n \stackrel{\text{def}}{=} \frac{\beta}{2Nn(n+1)},$$

the event  $\mathcal{E}$  holds with probability at least  $1 - \beta$ .

Indeed, from an union bound argument, there are at most  $2N$  inequalities (for each node, one for  $Y$ , one for  $Z$ ) of the form:

$$\mathbb{P}(|Y_{i,n_i}| > c_{n_i}, \forall n_i \geq 1) \leq \sum_{n_i \geq 1} \frac{\beta}{2Nn_i(n_i+1)} = \frac{\beta}{2N}.$$

We now prove Theorem 1 by bounding the pseudo-regret under the event  $\mathcal{E}$ . We show by induction that the pseudo-regret at any node  $j$  of depth  $d$  satisfies:

$$\bar{R}_{j,n_j} \leq k_d n_j c_{n_j} + k'_d, \quad (5)$$

This is obviously true for  $d = D$ , since the pseudo-regret is zero at the leaves. Now, let a node  $i$  of depth  $d - 1$ . Assume that the regret at the children's nodes satisfies (5) (for depth  $d$ ). For simplicity, write 1 the optimal child and 2 the sub-optimal one. Write

$$c_n^d = (k_d + 1)c_n + k'_d/n$$

the confidence interval defined by the choice of the bound (3) at a node of depth  $d$ .

If at round  $n$ , the node 2 is chosen, this means that  $X_{1,n_1} + c_{n_1}^d \leq X_{2,n_2} + c_{n_2}^d$ . Now, since  $|Z_{i,n_i}| \leq c_{n_i}$ , we have  $n_2(X_{2,n_2} - \bar{X}_{2,n_2}) \leq n_1 Y_{1,n_1} + n_i c_{n_i}$ , thus:

$$n_2(X_{1,n_1} + c_{n_1}^d) \leq n_1 Y_{1,n_1} + n_2(\bar{X}_{2,n_2} + c_{n_2}^d) + n_i c_{n_i}.$$

Now, since,  $|Y_{i,n_i}| \leq c_{n_i}$ , we deduce that:

$$n_2(\bar{X}_{1,n_1} - \bar{X}_{2,n_2} - c_{n_1} + c_{n_1}^d) \leq n_1 c_{n_1} + n_2 c_{n_2}^d + n_i c_{n_i}.$$

Now, from the definitions of  $\bar{X}$  and  $\bar{R}$ , we have:

$$\bar{X}_{1,n_1} - \bar{X}_{2,n_2} = \mu_1 - \frac{\bar{R}_{1,n_1}}{n_1} - \left( \mu_2 - \frac{\bar{R}_{2,n_2}}{n_2} \right) = \Delta_i - \frac{\bar{R}_{1,n_1}}{n_1} + \frac{\bar{R}_{2,n_2}}{n_2},$$

where  $\Delta_i \stackrel{\text{def}}{=} \mu_1 - \mu_2$ . Thus, if action 2 is chosen, we have:

$$n_2 \left( \Delta_i - \frac{\bar{R}_{1,n_1}}{n_1} + \frac{\bar{R}_{2,n_2}}{n_2} - c_{n_1} + c_{n_1}^d \right) \leq n_1 c_{n_1} + n_i c_{n_i} + n_2 c_{n_2}^d.$$

From the definition of  $c_{n_1}^d$  and the assumption (5) on  $\bar{R}_{1,n_1}$ , we have  $-\bar{R}_{1,n_1}/n_1 - c_{n_1} + c_{n_1}^d \geq 0$ , thus

$$\begin{aligned} n_2 &\leq [n_1 c_{n_1} + n_i c_{n_i} + n_2(k_d + 1)c_{n_2} + k'_d] / \Delta_i \\ &\leq [(1 + \sqrt{2} + k_d)n_i c_{n_i} + k'_d] / \Delta_i. \end{aligned}$$

Thus if  $n_2 > [(1 + \sqrt{2} + k_d)n_i c_{n_i} + k'_d] / \Delta_i$ , the arm 2 will never be chosen any more. We deduce that for all  $n \geq 0$ ,  $n_2 \leq [(1 + \sqrt{2} + k_d)n_i c_{n_i} + k'_d] / \Delta_i + 1$ .

Now, the pseudo-regret at node  $i$  satisfies:

$$\begin{aligned} \bar{R}_{i,n_i} &\leq \bar{R}_{1,n_1} + \bar{R}_{2,n_2} + n_2 \Delta_i \\ &\leq (1 + \sqrt{2})(1 + k_d)n_i c_{n_i} + 3k'_d + \Delta_i, \end{aligned}$$

which is of the same form as (5) with

$$\begin{aligned} k_{d-1} &= (1 + \sqrt{2})(1 + k_d) \\ k'_{d-1} &= 3k'_d + 1. \end{aligned}$$

Now, by induction, given that  $k_D = 0$  and  $k'_D = 0$ , we deduce the general form (4) of  $k_d$  and  $k'_d$  and the bound on the pseudo-regret at the root for  $d = 0$ .  $\square$

Notice that the  $B$  values defined by (3) are true upper bounds on the nodes value: under the event  $\mathcal{E}$ , for all node  $i$ , for all  $n_i \geq 1$ ,  $\mu_i \leq B_{i,n_i}$ . Thus this procedure is safe, which prevents from having bad behaviors for which the regret could be disastrous, like in regular UCT. However, contrarily to regular UCT, in good cases, the procedure does not adapt to the effective smoothness in the tree. For example, at the root level, the confidence sequence is  $O(\exp(D)/\sqrt{n})$  which lead to almost uniform sampling of both actions during a time  $O(\exp(D))$ . Thus, if the tree were to contain 2 branches, one only with zeros, one only with ones, this smoothness would not be taken into account, and the regret would be comparable to the worst-case regret. Modified UCT is less optimistic than regular UCT but safer in a worst-case scenario.

## 4 Flat UCB

A method that would combine both the safety of modified UCT and the adaptivity of regular UCT is to consider a regular UCB algorithm on the leaves. Such a *flat UCB* could naturally be implemented in the tree structure by defining the upper confidence bound of a non-leaf node as the maximal value of the children's bound:

$$B_{i,n_i} \stackrel{\text{def}}{=} \begin{cases} X_{i,n_i} + \sqrt{\frac{2 \log(\beta_{n_i}^{-1})}{n_i}} & \text{if } i \text{ is a leaf,} \\ \max[B_{i_1,n_{i_1}}, B_{i_2,n_{i_2}}] & \text{otherwise.} \end{cases} \quad (6)$$

where we use

$$\beta_n \stackrel{\text{def}}{=} \frac{\beta}{2^D n(n+1)}.$$

We deduce:

**Theorem 2.** *Consider the flat UCB defined by Algorithm 1 and (6). Then, with probability at least  $1 - \beta$ , the pseudo-regret is bounded by a constant:*

$$\bar{R}_n \leq 40 \sum_{i \in \mathcal{S}} \frac{1}{\Delta_i} \log\left(\frac{2^{D+1}}{\Delta_i^2 \beta}\right) \leq 40 \frac{2^D}{\Delta} \log\left(\frac{2^{D+1}}{\Delta^2 \beta}\right),$$

where  $\mathcal{S}$  is the set of sub-optimal leaves, i.e..  $\mathcal{S} = \{i \in \mathcal{L}, \Delta_i > 0\}$ , and  $\Delta = \min_{i \in \mathcal{S}} \Delta_i$ .

*Proof.* Consider the event  $\mathcal{E}$  under which, for all leaves  $i$ , for all  $n \geq 1$ , we have  $|X_{i,n} - \mu_i| \leq c_n$ , with the confidence interval  $c_n = \sqrt{\frac{2 \log(\beta_n^{-1})}{n}}$ . Then, the event  $\mathcal{E}$  holds with probability at least  $1 - \beta$ . Indeed, as before, using an union bound argument, there are at most  $2^D$  inequalities (one for each leaf) of the form:

$$\mathbb{P}(|X_{i,n} - \mu_i| > c_n, \forall n \geq 1) \leq \sum_{n \geq 1} \frac{\beta}{2^D n(n+1)} = \frac{\beta}{2^D}.$$

Under the event  $\mathcal{E}$ , we now provide a regret bound by bounding the number of times each sub-optimal leaf is visited. Let  $i \in \mathcal{S}$  be a sub-optimal leaf. Write  $*$  an optimal leaf. If at some round  $n$ , the leaf  $i$  is chosen, this means that  $X_{*,n_*} + c_{n_*} \leq X_{i,n_i} + c_{n_i}$ . Using the (lower and upper) confidence interval bounds for leaves  $i$  and  $*$ , we deduce that  $\mu^* \leq \mu_i + 2c_{n_i}$ . Thus  $(\frac{\Delta_i}{2})^2 \leq \frac{2 \log(\beta_{n_i}^{-1})}{n_i}$ . Hence, for all  $n \geq 1$ ,  $n_i$  is bounded by the smallest integer  $m$  such that  $\frac{m}{\log(\beta_m^{-1})} > 8/\Delta_i^2$ . Thus  $\frac{n_i - 1}{\log(2^D n_i (n_i - 1) \beta^{-1})} \leq w$ , writing  $w = 8/\Delta_i^2$ . This implies

$$n_i \leq 1 + w \log(2^D n_i^2 \beta^{-1}) \quad (7)$$

A first rough bound yields  $n_i \leq w^2 2^{D-2} \beta^{-1}$ , which can be used to derive a tighter upper bound on  $n_i$ . After two recursive uses of (7) we obtain:

$$n_i \leq 5w \log(w 2^{D-2} \beta^{-1}).$$

Thus, for all  $n \geq 1$ , the number of times leaf  $i$  is chosen is at most  $40 \log(2^{D+1} \beta^{-1} / \Delta_i^2) / \Delta_i^2$ . The bound on the regret follows immediately from the property that  $\bar{R}_n = \sum_{i \in \mathcal{S}} n_i \Delta_i$ .  $\square$

This algorithm is safe in the same sense as previously define, i.e. with high probability, the bounds defined by (6) are true upper bounds on the value on the leaves. However, since there are  $2^D$  leaves, the regret still depends exponentially on the depth  $D$ .

**Remark 1.** *Modified UCT has a regret  $O(2^D \sqrt{n})$  whereas Flat UCB has a regret  $O(2^D / \Delta)$ . The non dependency w.r.t.  $\Delta$  in Modified UCT, obtained at a price of an additional  $\sqrt{n}$  factor, comes from the application of Azuma's inequality also to  $Z$ , i.e. the difference between the children's deviations  $X - \bar{X}$ . An similar analysis in Flat UCB would yield a regret  $O(2^D \sqrt{n})$ .*

In the next section, we consider another UCB-based algorithm that takes into account possible smoothness of the rewards to process effective ‘‘cuts’’ of sub-optimal branches with high confidence.

## 5 Bandit Algorithm for Smooth Trees

We want to exploit the fact that if the leaves of a branch have similar values, then a confidence interval on that branch may be made much tighter than the maximal confidence

interval of its leaves (as processed in the Flat UCB). Indeed, assume that from a node  $i$ , all leaves  $j \in \mathcal{L}(i)$  in the branch  $i$  have values  $\mu_j$ , such that  $\mu_i - \mu_j \leq \delta$ . Then,

$$\mu_i \leq \frac{1}{n_i} \sum_{j \in \mathcal{L}(i)} n_j (\mu_j + \delta) \leq X_{i,n_i} + \delta + \bar{X}_{i,n_i} - X_{i,n_i},$$

and thanks to Azuma's inequality, the term  $\bar{X}_{i,n_i} - X_{i,n_i}$  is bounded with probability  $1 - \beta$  by a confidence interval  $\sqrt{\frac{2 \log(\beta^{-1})}{n_i}}$  which depends only on  $n_i$  (and not on  $n_j$  for  $j \in \mathcal{L}(i)$ ). We now make the following assumption on the rewards:

**Smoothness assumption:** Assume that for all depth  $d < D$ , there exists  $\delta_d > 0$ , such that for any node  $i$  of depth  $d$ , for all leaves  $j \in \mathcal{L}(i)$  in the branch  $i$ , we have  $\mu_i - \mu_j \leq \delta_d$ .

Typical choices of the smoothness coefficients  $\delta_d$  are exponential  $\delta_d \stackrel{\text{def}}{=} \delta \gamma^d$  (with  $\delta > 0$  and  $\gamma < 1$ ), polynomial  $\delta_d \stackrel{\text{def}}{=} \delta d^\alpha$  (with  $\alpha < 0$ ), or linear  $\delta_d \stackrel{\text{def}}{=} \delta(D - d)$  (Lipschitz in the tree distance) sequences.

We define the *Bandit Algorithm for Smooth Trees* (BAST) by Algorithm 1 with the upper confidence bounds defined, for any leaf  $i$ , by  $B_{i,n_i} \stackrel{\text{def}}{=} X_{i,n_i} + c_{n_i}$ , and for any non-leaf node  $i$  of depth  $d$ , by

$$B_{i,n_i} \stackrel{\text{def}}{=} \min \left\{ \max [B_{i_1,n_{i_1}}, B_{i_2,n_{i_2}}], X_{i,n_i} + \delta_d + c_{n_i} \right\} \quad (8)$$

with the confidence interval

$$c_n \stackrel{\text{def}}{=} \sqrt{\frac{2 \log(Nn(n+1)\beta^{-1})}{n}}.$$

We now provide high confidence bounds on the number of times each sub-optimal node is visited.

**Theorem 3.** *Let  $I$  denotes the set of nodes  $i$  such that  $\Delta_i > \delta_{d_i}$ , where  $d_i$  is the depth of node  $i$ . Define recursively the values  $N_i$  associated to each node  $i$  of a sub-optimal branch (i.e. for which  $\Delta_i > 0$ ):*

- If  $i$  is a leaf, then

$$N_i \stackrel{\text{def}}{=} \frac{40 \log(2N\beta^{-1}/\Delta_i^2)}{\Delta_i^2},$$

- If  $i$  is not a leaf, then

$$N_i \stackrel{\text{def}}{=} \begin{cases} N_{i_1} + N_{i_2}, & \text{if } i \notin I \\ \min(N_{i_1} + N_{i_2}, \frac{40 \log(2N\beta^{-1}/(\Delta_i - \delta_{d_i})^2)}{(\Delta_i - \delta_{d_i})^2}), & \text{if } i \in I \end{cases}$$

where  $i_1$  and  $i_2$  are the children nodes of  $i$ . Then, with probability  $1 - \beta$ , for all  $n \geq 1$ , for all sub-optimal nodes  $i$ ,  $n_i \leq N_i$ .

*Proof.* We consider the event  $\mathcal{E}$  under which  $|X_{i,n} - \bar{X}_{i,n}| \leq c_n$  for all nodes  $i$  and all times  $n \geq 1$ . The confidence interval  $c_n = \sqrt{\frac{2 \log(Nn(n+1)\beta^{-1})}{n}}$  is chosen such that  $\mathbb{P}(\mathcal{E}) \geq 1 - \beta$ . Under  $\mathcal{E}$ , using the same analysis as in the Flat UCB, we deduce the bound  $n_i \leq N_i$  for any sub-optimal leaf  $i$ .

Now, by backward induction on the depth, assume that  $n_i \leq N_i$  for all sub-optimal nodes of depth  $d + 1$ . Let  $i$  be a node of depth  $d$ . Then  $n_i \leq n_{i_1} + n_{i_2} \leq N_{i_1} + N_{i_2}$ .

Now consider a sub-optimal node  $i \in I$ . If the node  $i$  is chosen at round  $n$ , the form of the bound (8) implies that for any optimal node  $*$ , we have  $B_{*,n_*} \leq B_{i,n_i}$ . Under  $\mathcal{E}$ ,  $\mu^* \leq B_{*,n_*}$  and  $B_{i,n_i} \leq X_{i,n_i} + \delta_d + c_{n_i} \leq \mu_i + \delta_d + 2c_{n_i}$ . Thus  $\mu^* \leq \mu_i + \delta_d + 2c_{n_i}$ , which rewrites  $\Delta_i - \delta_d \leq 2c_{n_i}$ . Using the same argument as in the proof of Flat UCB, we deduce that for all  $n \geq 1$ , we have  $n_i \leq \frac{40 \log(2N\beta^{-1}/(\Delta_i - \delta_{d_i})^2)}{(\Delta_i - \delta_{d_i})^2}$ . Thus  $n_i \leq N_i$  at depth  $d$ , which finishes the inductive proof.  $\square$

Now we would like to compare the regret of BAST to that of Flat UCB. First, we expect a direct gain for nodes  $i \in I$ . Indeed, from the previous result, whenever a node  $i$  of depth  $d$  is such that  $\Delta_i > \delta_d$ , then this node will be visited, with high probability, at most  $O(1/(\Delta_i - \delta_d)^2)$  times (neglecting log factors). But we also expect an upper bound on  $n_i$  whenever  $\Delta_i > 0$  if at a certain depth  $h \in [d, D]$ , all nodes  $j$  of depth  $h$  in the branch  $i$  satisfy  $\Delta_j > \delta_h$ .

The next result enables to further analyze the expected improvement over Flat UCB.

**Theorem 4.** *Consider the exponential assumption on the smoothness coefficients :  $\delta_d \leq \delta\gamma^d$ . For any  $\eta \geq 0$  define the set of leaves  $I_\eta \stackrel{\text{def}}{=} \{i \in \mathcal{S}, \Delta_i \leq \eta\}$ . Then, with probability at least  $1 - \beta$ , the pseudo regret satisfies, for all  $n \geq 1$ ,*

$$\begin{aligned} \bar{R}_n &\leq \sum_{i \in I_\eta, \Delta_i > 0} \frac{40}{\Delta_i} \log\left(\frac{2N\beta^{-1}}{\Delta_i^2}\right) + |I_\eta| \frac{320(2\delta)^c}{\eta^{2+c}} \log\left(\frac{2N}{\eta^2\beta}\right) \\ &\leq 40|I_\eta| \left( \frac{1}{\Delta} \log\left(\frac{2N}{\Delta^2\beta}\right) + \frac{8(2\delta)^c}{\eta^{2+c}} \log\left(\frac{2N}{\eta^2\beta}\right) \right) \end{aligned} \quad (9)$$

where

$$c \stackrel{\text{def}}{=} \log(2)/\log(1/\gamma).$$

Note that this bound (9) does not depend explicitly on the depth  $D$ . Thus we expect this method to scale nicely in big trees (large  $D$ ). The first term in the bound is the same as in Flat UCB, but the sum is performed only on leaves  $i \in I_\eta$  whose value is  $\eta$ -close to optimality. Thus, BAST is expected to improve over Flat UCB (at least as expressed by the bounds) whenever the number  $|I_\eta|$  of  $\eta$ -optimal leaves is small compared to the total number of leaves  $2^D$ . In particular, with  $\eta < \Delta$ ,  $|I_\eta|$  equals the number of optimal leaves, so taking  $\eta \rightarrow \Delta$  we deduce a regret  $O(1/\Delta^{2+c})$ .

*Proof.* We consider the same event  $\mathcal{E}$  as in the proof of Theorem 3. Call “ $\eta$ -optimal” a branch that contains at least a leaf in  $I_\eta$ . Let  $i$  be a node, of depth  $d$ , that does not belong

to an  $\eta$ -optimal branch. Let  $h$  be the smallest integer such that  $\delta_h \leq \eta/2$ , where  $\delta_h = \delta\gamma^h$ . We have  $h \leq \frac{\log(2\delta/\eta)}{\log(1/\gamma)} + 1$ . Let  $j$  be a node of depth  $h$  in the branch  $i$ . Using similar arguments as in Theorem 2, the number of times  $n_j$  the node  $j$  is visited is at most

$$n_j \leq \frac{40 \log(2N\beta^{-1}/(\Delta_j - \delta_h)^2)}{(\Delta_j - \delta_h)^2},$$

but since  $\Delta_j - \delta_h \geq \Delta_i - \delta_h \geq \Delta_i - \eta/2 \geq \eta/2$ , we have:  $n_j \leq l/\eta^2$ , writing  $l = 160 \log(8N\beta^{-1}/\eta^2)$ .

Now the number of such nodes  $j$  is at most  $2^{h-d}$ , thus:

$$n_i \leq 2^{h-d} \frac{l}{\eta^2} \leq 2 \left( \frac{2\delta}{\eta} \right)^{\frac{\log(2)}{\log(1/\gamma)}} 2^{-d} \frac{l}{\eta^2} = \frac{2l(2\delta)^c}{\eta^{c+2}} 2^{-d}$$

with  $c = \log(2)/\log(1/\gamma)$ . Thus, the number of times  $\eta$ -optimal branches are not followed until the  $\eta$ -optimal leaves is at most

$$|I_\eta| \sum_{d=1}^D \frac{2l(2\delta)^c}{\eta^{c+2}} 2^{-d} \leq |I_\eta| \frac{2l(2\delta)^c}{\eta^{c+2}}.$$

Now for the leaves  $i \in I_\eta$ , we derive similarly to the Flat UCB that  $n_i \leq 40 \log(2N\beta^{-1}/\Delta_i^2)/\Delta_i^2$ . Thus, the pseudo regret is bounded by the sum for all sub-optimal leaves  $i \in I_\eta$  of  $n_i \Delta_i$  plus the sum of all trajectories that do not follow  $\eta$ -optimal branches until  $\eta$ -optimal leaves  $|I_\eta| \frac{2l(2\delta)^c}{\eta^{c+2}}$ . This implies (9).  $\square$

**Remark 2.** Notice that if we choose  $\delta = 0$ , then BAST algorithm reduces to regular UCT (with a slightly different confidence interval), whereas if  $\delta = \infty$ , then this is simply Flat UCB. Thus BAST may be seen as a generic UCB-based bandit algorithm for tree search, that allows to take into account actual smoothness of the tree, if available.

**Numerical experiments: global optimization of a noisy function.** We search the global optimum of an  $[0, 1]$ -valued function, given noisy data. The domain  $[0, 1]$  is uniformly discretized by  $2^D$  points  $\{y_j\}$ , each one related to a leaf  $j$  of a tree of depth  $D$ . The tree implements a recursive binary splitting of the domain. At time  $t$ , if the algorithm selects a leaf  $j$ , then the (binary) reward  $x_t \stackrel{i.i.d.}{\sim} \mathcal{B}(f(y_j))$ , a Bernoulli random variable with parameter  $f(y_j)$  (i.e.  $\mathbb{P}(x_t = 1) = f(y_j)$ ,  $\mathbb{P}(x_t = 0) = 1 - f(y_j)$ ).

We assume that  $f$  is Lipschitz. Thus the exponential smoothness assumption  $\delta_d = \delta 2^{-d}$  on the rewards holds with  $\delta$  being the Lipschitz constant of  $f$  and  $\gamma = 1/2$  (thus  $c = 1$ ). In the experiments, we used the function

$$f(x) \stackrel{\text{def}}{=} |\sin(4\pi x) + \cos(x)|/2$$

plotted in Figure 2. Note that an immediate upper bound on the Lipschitz constant of  $f$  is  $(4\pi + 1)/2 < 7$ .



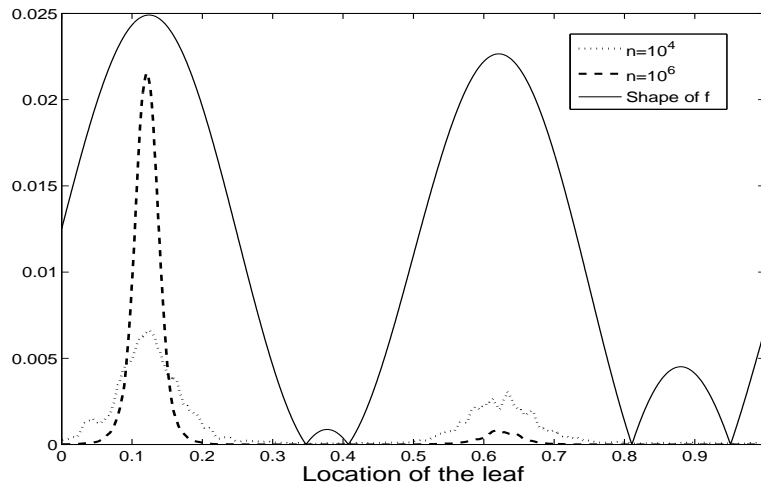


Figure 2: Function  $f$  rescaled (in plain) and proportion  $n_j/n$  of leaves visitation for BAST with  $\delta = 7$ , depth  $D = 10$ , after  $n = 10^4$  and  $n = 10^6$  rounds.

We compare Flat UCB and BAST algorithm for different values of  $\delta$ . Figure 3 show the respective pseudo-regret per round  $\bar{R}_n/n$  for BAST used with a good evaluation of the Lipschitz constant ( $\delta = 7$ ), BAST used with a poor evaluation of the Lipschitz constant ( $\delta = 20$ ), and Flat UCB ( $\delta = \infty$ ). As expected, we observe that BAST outperforms Flat UCB, and that the performance of BAST is less dependent of the size of the tree than Flat UCB. BAST with a poor evaluation of  $\delta$  still performs better than Flat UCB. BAST concentrates its resources on the good leaves: In Figure 2 we show the proportion  $n_j/n$  of visits of each leaf  $j$ . We observe that, when  $n$  increases, the proportion of visits concentrates on the leaves with highest  $f$  value.

**Remark 3.** *If we know in advance that the function is smooth (e.g. of class  $\mathcal{C}^2$  with bounded second order derivative), then one could use Taylor's expansion to derive much tighter upper bounds, which would cut more efficiently sub-optimal branches and yield improved performance. Thus any a priori knowledge about the tree smoothness could be taken into account in the BAST bound (8).*

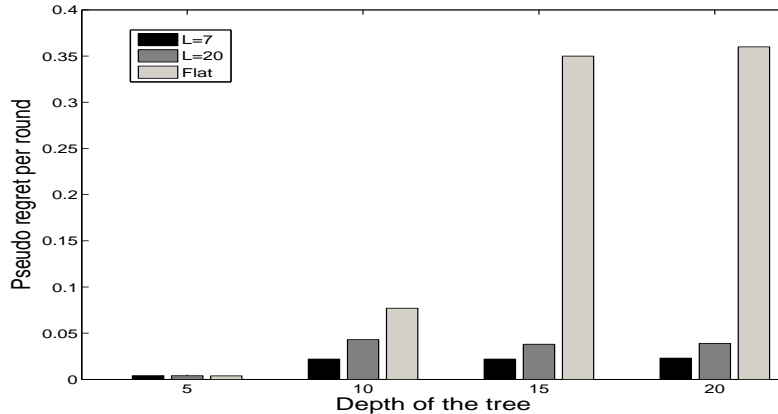


Figure 3: Pseudo regret per round  $\bar{R}_n/n$  for  $n = 10^6$ , as a function of the depth  $D \in \{5, 10, 15, 20\}$ , for BAST with  $\delta \in \{7, 20\}$  and Flat UCB.

## 6 Growing trees

If the tree is too big (possibly infinite) to be represented, one may wish to discover it iteratively, exploring it as the same time as searching for an optimal value. We propose an incremental algorithm similar to the method described in [Cou06] and [GWMT06]: The algorithm starts with only the root node. Then, at each stage  $n$  it chooses which leaf, call it  $i$ , to expand next. Expanding a leaf means turning it into a node  $i$ , and adding in our current tree representation its children leaves  $i_1$  and  $i_2$ , from which a reward (one for each child) is received. The process is then repeated in the new tree.

We make an assumption on the rewards: from any leaf  $j$ , the received reward  $x$  is a random variable whose expected value satisfies:  $\mu_j - \mathbb{E}[x] \leq \delta_d$ , where  $d$  is the depth of  $j$ , and  $\mu_j$  the value of  $j$  (defined as previously by the maximum of its children values).

Such an iterative growing tree requires a amount of memory  $O(n)$  directly related to the number of exploration rounds in the tree.

We now apply BAST algorithm and expect the tree to grow in an asymmetric way, expanding in depth the most promising branches first, leaving mainly unbuilt the sub-optimal branches.

**Theorem 5.** Consider this incremental tree method using Algorithm 1 defined by the bound (8) with the confidence interval

$$c_{n_i} \stackrel{\text{def}}{=} \sqrt{\frac{2 \log(n(n+1)n_i(n_i+1)\beta^{-1})}{n_i}},$$

where  $n$  is the current number of expanded nodes. Then, with probability  $1 - \beta$ , for any sub-optimal node  $i$  (of depth  $d$ ), i.e. s.t.  $\Delta_i > 0$ ,

$$n_i \leq 10 \left(\frac{\delta}{c}\right)^c \left(\frac{2+c}{\Delta_i}\right)^{c+2} \log\left(\frac{n(n+1)(2+c)^2}{\beta 2\Delta_i^2}\right) 2^{-d}. \quad (10)$$

Thus this algorithm essentially develops the optimal branch, i.e. except for  $O(\log(n))$  samples at each depth, all computational resources are devoted to further explore the optimal branch.

*Proof.* Consider the event  $\mathcal{E}$  under which  $|X_{i,n_i} - \bar{X}_{i,n_i}| \leq c_{n_i}$  for all expanded nodes  $1 \leq i \leq n$ , all times  $n_i \geq 1$ , and all rounds  $n \geq 1$ . The confidence interval  $c_{n_i}$  are such that  $\mathbb{P}(\mathcal{E}) \geq 1 - \beta$ . At round  $n$ , let  $i$  be a node of depth  $d$ . Let  $h$  be a depth such that  $\delta_h < \Delta_i$ . This is satisfied for all integer  $h \geq \frac{\log(\delta/\Delta_i)}{\log(1/\gamma)}$ . Similarly to Flat UCB, we deduce that the number of times  $n_j$  a node  $j$  of depth  $h$  has been visited is bounded by  $40 \log(2n(n+1)\beta^{-1}/(\Delta_i - \delta_h)^2)/(\Delta_i - \delta_h)^2$ . Thus  $i$  has been visited at most

$$n_i \leq \min_{h \geq \frac{\log(\delta/\Delta_i)}{\log(1/\gamma)}} 2^{h-d} \frac{40 \log(2n(n+1)\beta^{-1}/(\Delta_i - \delta_h)^2)}{(\Delta_i - \delta_h)^2}.$$

This function is minimized (neglecting the log term) for  $h = \log(\frac{\delta(2+c)}{c\Delta_i})/\log(1/\gamma)$ , which leads to (10).  $\square$

For illustration, Figure 4 shows the tree obtained applied to the function optimization problem of previous section, after  $n = 300$  rounds. The most in-depth explored branches are those with highest value.

## 7 Conclusion

We analyzed several UCB-based bandit algorithms for tree search. The good exploration exploitation tradeoff of these methods enables to return rapidly a good value, and improve precision if more time is provided. BAST enables to take into account possible smoothness in the tree to perform efficient “cuts”<sup>1</sup> of sub-optimal branches, with high probability.

If additional smoothness information is provided, the  $\delta$  term in the bound (8) may be refined, leading to improved performance. Empirical information, such as variance estimate, could improve knowledge about local smoothness which may be very helpful to refine the bounds. However, it seems important to use true upper confidence bounds, in order to avoid bad cases as illustrated in regular UCT. Applications include minimax search for games in large trees, and global optimization under uncertainty.

<sup>1</sup>Note that this term may be misleading here since the UCB-based methods described here never explicitly delete branches

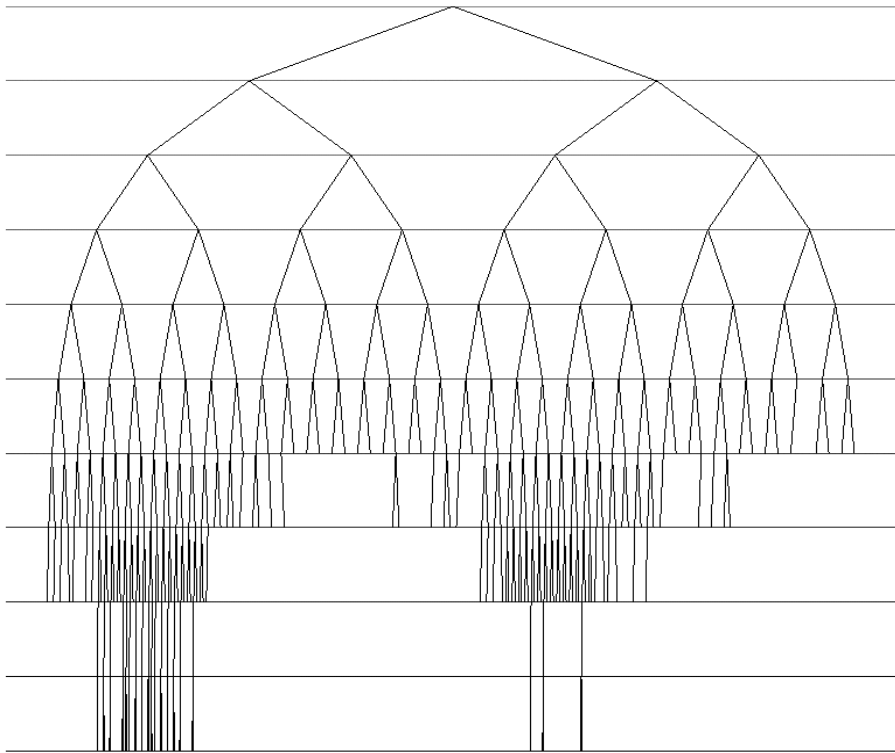


Figure 4: Tree resulting from the iterative growing BAST algorithm, after  $n = 300$  rounds, with  $\delta = 7$ .

## Acknowledgements

We wish to thank Jean-François Hren for running the numerical experiments.

## References

- [ACBF02] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning Journal*, 47(2-3):235–256, 2002.
- [Cou06] R. Coulom. Efficient selectivity and backup operators in Monte-Carlo tree search. *5th International Conference on Computer and Games*, 2006.
- [GDL96] L. Györfi, L. Devroye, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer-Verlag, 1996.
- [GWMT06] S. Gelly, Y. Wang, R. Munos, and O. Teytaud. Modication of UCT with patterns in Monte-Carlo go. *Technical Report INRIA RR-6062*, 2006.
- [KS06] L. Kocsis and Cs. Szepesvari. Bandit based monte-carlo planning. *European Conference on Machine Learning*, pages 282–293, 2006.



---

Unité de recherche INRIA Futurs  
Parc Club Orsay Université - ZAC des Vignes  
4, rue Jacques Monod - 91893 ORSAY Cedex (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique  
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)

Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

---

Éditeur  
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)  
<http://www.inria.fr>  
ISSN 0249-6399