Black-box optimization of noisy functions with unknown smoothness

Jean-Bastien Grill, Michal Valko, Rémi Munos

Black-box optimization of noisy functions with unknown smoothness

Jean-Bastien Grill **Michal Valko** **Rémi Munos**
SequeL team, INRIA Lille - Nord Europe, France Google DeepMind, UK*
jean-bastien.grill@inria.fr michal.valko@inria.fr munos@google.com

Abstract

We study the problem of black-box optimization of a function f of any dimension, given function evaluations perturbed by noise. The function is assumed to be locally smooth around one of its global optima, but this *smoothness is unknown*. Our contribution is an adaptive optimization algorithm, POO or *parallel optimistic optimization*, that is able to deal with this setting. POO performs almost as well as the best known algorithms requiring the knowledge of the smoothness. Furthermore, POO works for a larger class of functions than what was previously considered, especially for functions that are *difficult to optimize*, in a very precise sense. We provide a *finite-time* analysis of POO's performance, which shows that its error after n evaluations is at most a factor of $\sqrt{\ln n}$ away from the error of the best known optimization algorithms using the knowledge of the smoothness.

1 Introduction

We treat the problem of optimizing a function $f : \mathcal{X} \to \mathbb{R}$ given a finite budget of n noisy evaluations. We consider that the cost of any of these *function evaluations* is high. That means, we care about assessing the optimization performance in terms of the sample complexity, i.e., the number of n function evaluations. This is typically the case when one needs to tune parameters for a complex system seen as a black-box, which performance can only be evaluated by a costly simulation. One such example, is the *hyper-parameter tuning* where the sensitivity to perturbations is large and the derivatives of the objective function with respect to these parameters do not exist or are unknown.

Such setting fits the sequential decision-making setting under *bandit feedback*. In this setting, the actions are the points that lie in a domain $\mathcal{X}$. At each step t, an algorithm selects an action $x_t \in \mathcal{X}$ and receives a reward r_t, which is a noisy function evaluation such that $r_t = f(x_t) + \varepsilon_t$, where ε_t is a bounded noise with $\mathbb{E}\left[\varepsilon_t \,|x_t\right] = 0$. After n evaluations, the algorithm outputs its best guess $x(n)$, which can be different from x_n. The performance measure we want to minimize is the value of the function at the returned point compared to the optimum, also referred to as *simple regret*,

$$R_n \stackrel{\text{def}}{=} \sup_{x \in \mathcal{X}} f(x) - f\left(x\left(n\right)\right).$$

We assume there exists at least one point $x^\star \in \mathcal{X}$ such that $f(x^\star) = \sup_{x \in \mathcal{X}} f(x)$.

The relationship with bandit settings motivated UCT [9, 7], an empirically successful heuristic that hierarchically partitions domain $\mathcal{X}$ and selects the next point $x_t \in \mathcal{X}$ using upper confidence bounds [1]. The empirical success of UCT on one side but the absence of performance guarantees for it on the other, incited research on similar but theoretically founded algorithms [4, 8, 11, 2, 6].

As the global optimization of the unknown function without absolutely any assumptions would be a daunting needle-in-a-haystack problem, most of the algorithms assume at least a very weak

*on the leave from SequeL team, INRIA Lille - Nord Europe, France

assumption that the function does *not decrease faster than a known rate* around *one* of its global optima. In other words, they assume a certain *local smoothness* property of f. This smoothness is often expressed in the form of a semi-metric ℓ that quantifies this regularity [4]. Naturally, this regularity also influences the guarantees that these algorithms are able to furnish. Many of them define a *near-optimality dimension* d or a *zooming dimension*. These are ℓ-dependent quantities used to bound the simple regret R_n or a related notion called *cumulative regret*.

Our work focuses on a notion of such near-optimality dimension d that does not directly relate the smoothness property of f to a specific metric ℓ but *directly* to the *hierarchical partitioning* $\mathcal{P} = \{\mathcal{P}_{h,i}\}$, a *tree-based representation* of the space used by the algorithm. Indeed, an interesting fundamental question is to determine a good characterization of the difficulty of the optimization for an algorithm that uses a given hierarchical partitioning of the space $\mathcal{X}$ as its input. The kind of hierarchical partitioning $\{\mathcal{P}_{h,i}\}$ we consider is similar to the ones introduced in prior work: for any depth $h \geq 0$ in the tree representation, the set of *cells* $\{\mathcal{P}_{h,i}\}_{1 \leq i \leq I_h}$ form a partition of $\mathcal{X}$, where I_h is the number of cells at depth h. At depth 0, the root of the tree, there is a single cell $\mathcal{P}_{0,1} = \mathcal{X}$. A cell $\mathcal{P}_{h,i}$ of depth h is split into several children subcells $\{\mathcal{P}_{h+1,j}\}_j$ of depth $h+1$. We refer to the standard partitioning as to one where each cell is split into regular same-sized subcells [12].

An important insight, detailed in Section 2, is that a near-optimality dimension d that is independent from the partitioning used by an algorithm (as defined in prior work [4, 8, 2]) *does not embody the optimization difficulty perfectly*. This is easy to see, as for any f we could define a partitioning, perfectly suited for f. An example is a partitioning, that at the root splits $\mathcal{X}$ into $\{x^\star\}$ and $\mathcal{X} \setminus x^\star$, which makes the optimization trivial, whatever d is. This insight was already observed by Slivkins [13] and Bull [6], whose *zooming dimension* depends both on the function and the partitioning.

In this paper, we define a notion of near-optimality dimension d which measures the complexity of the optimization problem *directly in terms of the partitioning* used by an algorithm. First, we make the following local smoothness assumption about the function, expressed in terms of the partitioning and *not any metric*: For a given partitioning $\mathcal{P}$, we assume that there exist $\nu > 0$ and $\rho \in (0, 1)$, s.t.,

$$\forall h \geq 0, \forall x \in \mathcal{P}_{h,i_h^\star}, \quad f(x) \geq f(x^\star) - \nu\rho^h$$

where $(h, i_h^\star)$ is the (unique) cell of depth h containing $x^\star$. Then, we define the near-optimality dimension $d(\nu, \rho)$ as

$$d(\nu, \rho) \stackrel{\text{def}}{=} \inf \left\{ d' \in \mathbb{R}^+ : \exists C > 0, \forall h \geq 0, \mathcal{N}_h(2\nu\rho^h) \leq C\rho^{-d'h} \right\},$$

where for all $\varepsilon > 0$, $\mathcal{N}_h(\varepsilon)$ is the number of cells $\mathcal{P}_{h,i}$ of depth h s.t. $\sup_{x \in \mathcal{P}_{h,i}} f(x) \geq f(x^\star) - \varepsilon$. Intuitively, functions with smaller d are easier to optimize and we denote (ν, ρ), for which $d(\nu, \rho)$ is the smallest, as $(\nu_\star, \rho_\star)$. Obviously, $d(\nu, \rho)$ depends on $\mathcal{P}$ and f, but *does not depend* on any choice of a specific metric. In Section 2, we argue that this definition of d[1] encompasses the optimization complexity *better*. We stress this is not an artifact of our analysis and previous algorithms, such as `HOO` [4], `Zooming` [8], or `HCT` [2], can be shown to scale with this new notion of d.

Most of the prior bandit-based algorithms proposed for function optimization, for either deterministic or stochastic setting, assume that the smoothness of the optimized function is *known*. This is the case of known *semi-metric* [4, 2] and *pseudo-metric* [8]. This assumption limits the application of these algorithms and opens a very compelling question of whether this knowledge is necessary.

Prior work has addressed this question by designing algorithms that do not require the knowledge of this smoothness. Bubeck et al. [5] provided an algorithm for optimization of Lipschitz functions without the knowledge of the Lipschitz constant. However, they have to assume that f is twice differentiable and a bound on the second order derivative is known. Slivkins [13] considered a general optimization problem embedded in a *taxonomy*[2] and provided performance as a function of the *quality* of the taxonomy. The quality refers to the probability of reaching two cells belonging to the same branch that can have values that differ by more that half of the diameter (expressed by the true metric) of the branch. The problem is that the algorithm requires the knowledge of a lower bound on this quality (which can be tiny) and the performance depends inversely on this quantity. Also it assumes that the quality is strictly positive. In this paper, we do not rely on the knowledge of this quality and also consider a more general class of functions for which the quality can be 0.

[1] we use the simplified notation d instead of $d(\nu, \rho)$ for clarity when no confusion is possible

[2] which is similar to the hierarchical partitioning previously defined

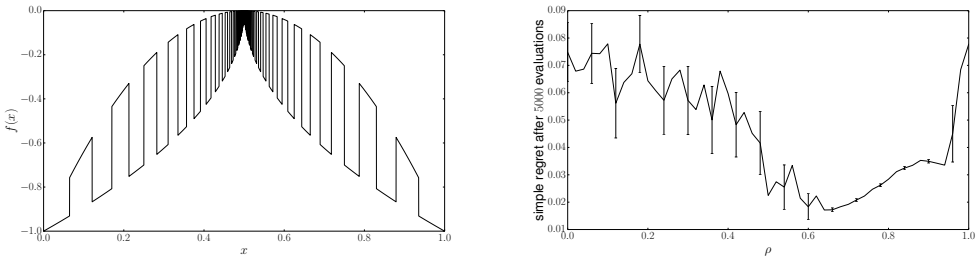

Figure 1: *Left:* Oscillation between two envelopes of different smoothness leading to a nonzero d for a standard partitioning. *Right:* Regret of HOO after 5000 evaluations for different values of ρ.

Another direction has been followed by Munos [10], where in the deterministic case (the function evaluations are not perturbed by noise), their SOO algorithm performs almost as well as the best known algorithms without the knowledge of the function smoothness. SOO was later extended to StoSOO [14] for the stochastic case. However StoSOO only extends SOO for a limited case of *easy instances* of functions for which there exists a semi-metric under which $d = 0$. Also, Bull [6] provided a similar regret bound for the ATB algorithm for a class of functions, called *zooming continuous functions*, which is related to the class of functions for which there exists a semi-metric under which the near-optimality dimension is $d = 0$. But none of the prior work considers a more general class of functions where there is no semi-metric adapted to the standard partitioning for which $d = 0$.

To give an example of a difficult function, consider the function in Figure 1. It possesses a lower and upper envelope around its global optimum that are equivalent to x^2 and $\sqrt{x}$; and therefore have different smoothness. Thus, for a standard partitioning, there is no semi-metric of the form $\ell(x, y) = ||x - y||^\alpha$ for which the near-optimality dimension is $d = 0$, as shown by Valko et al. [14]. Other examples of nonzero near-optimality dimension are the functions that for a standard partitioning behave differently depending on the direction, for instance $f : (x, y) \mapsto 1 - |x| - y^2$.

Using a bad value for the ρ parameter can have dramatic consequences on the simple regret. In Figure 1, we show the simple regret after 5000 function evaluations for different values of ρ. For the values of ρ that are too low, the algorithm does not explore enough and is stuck in a local maximum while for values of ρ too high the algorithm wastes evaluations by exploring too much.

In this paper, we provide a new algorithm, POO, *parallel optimistic optimization*, which competes with the best algorithms that assume the knowledge of the function smoothness, for a larger class of functions than was previously done. Indeed, POO handles a panoply of functions, including *hard instances*, i.e., such that $d > 0$, like the function illustrated above. We also recover the result of StoSOO and ATB for functions with $d = 0$. In particular, we bound the POO's simple regret as

$$\mathbb{E}[R_n] \leq \mathcal{O}\left(\left(\ln^2 n\right)/n\right)^{1/(2+d(\nu_\star,\rho_\star))}\Big).$$

This result should be compared to the simple regret of the best known algorithm that uses the knowledge of the metric under which the function is smooth, or equivalently (ν, ρ), which is of the order of $\mathcal{O}((\ln n/n)^{1/(2+d)})$. Thus POO's performance is at most a factor of $(\ln n)^{1/(2+d)}$ away from that of the best known optimization algorithms that require the knowledge of the function smoothness. Interestingly, this factor decreases with the complexity measure d: the harder the function to optimize, the less important it is to know its precise smoothness.

2 Background and assumptions

2.1 Hierarchical optimistic optimization

POO optimizes functions *without* the knowledge of their smoothness using a subroutine, an anytime algorithm optimizing functions *using* the knowledge of their smoothness. In this paper, we use a modified version of HOO [4] as such subroutine. Therefore, we embark with a quick review of HOO.

HOO follows an optimistic strategy close to UCT [9], but unlike UCT, it uses proper confidence bounds to provide theoretical guarantees. HOO refines a partition of the space based on a hierarchical partitioning, where at each step, a yet unexplored cell (a leaf of the corresponding tree) is selected, and

the function is evaluated at a point within this cell. The selected path (from the root to the leaf) is the one that maximizes the minimum value $U_{h,i}(t)$ among all cells of each depth, where the value $U_{h,i}(t)$ of any cell $\mathcal{P}_{h,i}$ is defined as

$$U_{h,i}(t) = \widehat{\mu}_{h,i}(t) + \sqrt{\frac{2\ln(t)}{N_{h,i}(t)}} + \nu\rho^h,$$

where t is the number of evaluations done so far, $\widehat{\mu}_{h,i}(t)$ is the empirical average of all evaluations done within $\mathcal{P}_{h,i}$, and $N_{h,i}(t)$ is the number of them. The second term in the definition of $U_{h,i}(t)$ is a Chernoff-Hoeffding type confidence interval, measuring the estimation error induced by the noise. The third term, $\nu\rho^h$ with $\rho \in (0,1)$ is, by assumption, a bound on the difference $f(x^\star) - f(x)$ for any $x \in \mathcal{P}_{h,i^\star_h}$, a cell containing $x^\star$. Is it this bound, where HOO relies on the knowledge of the smoothness, because the algorithm requires the values of ν and ρ. In the next sections, we clarify the assumptions made by HOO vs. related algorithms and point out the differences with POO.

2.2 Assumptions made in prior work

Most of previous work relies on the knowledge of a semi-metric on $\mathcal{X}$ such that the function is either locally smooth near to one of its maxima with respect to this metric [10, 14, 2] or require a stronger, weakly-Lipschitz assumption [4, 11, 2]. Furthermore, Kleinberg et al. [8] assume the full metric. Note, that the semi-metric does not require the triangular inequality to hold. For instance, consider the semi-metric $\ell(x,y) = ||x-y||^\alpha$ on $\mathbb{R}^p$ with $||\cdot||$ being the euclidean metric. When $\alpha < 1$ then this semi-metric does not satisfy the triangular inequality. However, it is a metric for $\alpha \geq 1$. Therefore, using only semi-metric allows us to consider a larger class of functions.

Prior work typically requires two assumptions. The first one is on semi-metric ℓ and the function. An example is the *weakly-Lipschitz* assumption needed by Bubeck et al. [4] which requires that

$$\forall x, y \in \mathcal{X}, \quad f(x^\star) - f(y) \leq f(x^\star) - f(x) + \max\left\{f(x^\star) - f(x), \ell(x,y)\right\}.$$

It is a weak version of a Lipschitz condition, restricting f in particular for the values close to $f(x^\star)$.

More recent results [10, 14, 2] assume only a *local smoothness* around one of the function maxima,

$$x \in \mathcal{X} \quad f(x^\star) - f(x) \leq \ell(x^\star, x).$$

The second common assumption *links* the hierarchical partitioning with the semi-metric. It requires the partitioning to be *adapted* to the (semi) metric. More precisely the well-shaped assumption states that there exist $\rho < 1$ and $\nu_1 \geq \nu_2 > 0$, such that for any depth $h \geq 0$ and index $i = 1, \ldots, I_h$, the subset $\mathcal{P}_{h,i}$ is *contained by* and *contains* two open balls of radius $\nu_1\rho^h$ and $\nu_2\rho^h$ respectively, where the balls are w.r.t. the same semi-metric used in the definition of the function smoothness.

'Local smoothness' is weaker than 'weakly Lipschitz' and therefore preferable. Algorithms requiring the local-smoothness assumption always sample a cell $\mathcal{P}_{h,i}$ in a special *representative point* and, in the stochastic case, collect several function evaluations from the same point before splitting the cell. This is not the case of HOO, which allows to sample *any* point inside the selected cell and to expand each cell after one sample. This additional flexibility comes at the price of requiring the stronger weakly-Lipschitzness assumption. Nevertheless, although HOO does not wait before expanding a cell, it does something similar by selecting a path from the root to this leaf that maximizes the minimum of the U-value over the cells of the path, as mentioned in Section 2.1. The fact that HOO follows an optimistic strategy even after reaching the cell that possesses the minimal U-value along the path is not used in the analysis of the HOO algorithm.

Furthermore, a reason for better dependency on the smoothness in other algorithms, e.g., HCT [2], is not only algorithmic: HCT needs to assume a slightly stronger condition on the cell, i.e., that the single center of the two balls (one that covers and the other one that contains the cell) is actually the same point that HCT uses for sampling. This is stronger than just assuming that there simply exist such centers of the two balls, which are not necessarily the same points where we sample (which is the HOO assumption). Therefore, this is *in contrast with HOO* that samples *any point* from the cell. In fact, it is straightforward to modify HOO to only sample at a representative point in each cell and only require the local-smoothness assumption. In our analysis and the algorithm, we use this modified version of HOO, thereby profiting from this weaker assumption.

Prior work [8, 4, 10, 2, 11] often defined some 'dimension' d of the near-optimal space of f measured according to the (semi-) metric ℓ. For example, the so-called *near-optimality dimension* [4] measures the size of the near-optimal space $\mathcal{X}_\varepsilon = \{x \in \mathcal{X} : f(x) > f(x^\star) - \varepsilon\}$ in terms of *packing numbers*: For any $c > 0, \varepsilon_0 > 0$, the (c, ε_0)-near-optimality dimension d of f with respect to ℓ is defined as

$$\inf \left\{ d \in [0, \infty) : \exists C \text{ s.t. } \forall \varepsilon \leq \varepsilon_0, \mathcal{N}(\mathcal{X}_{c\varepsilon}, \ell, \varepsilon) \leq C\varepsilon^{-d} \right\}, \tag{1}$$

where for any subset $A \subseteq \mathcal{X}$, the packing number $\mathcal{N}(A, \ell, \varepsilon)$ is the maximum number of disjoint balls of radius ε contained in A.

2.3 Our assumption

Contrary to the previous approaches, we need *only a single assumption*. We do not introduce any (semi)-metric and instead directly relate f to the hierarchical partitioning $\mathcal{P}$, defined in Section 1. Let K be the maximum number of children cells $(\mathcal{P}_{h+1,j_k})_{1 \leq k \leq K}$ per cell $\mathcal{P}_{h,i}$. We remind the reader that given a global maximum $x^\star$ of f, $i_h^\star$ denotes the index of the unique cell of depth h containing $x^\star$, i.e., such that $x^\star \in \mathcal{P}_{h,i_h^\star}$. With this notation we can state our sole assumption on both the partitioning $(\mathcal{P}_{h,i})$ and the function f.

Assumption 1. *There exists $\nu > 0$ and $\rho \in (0, 1)$ such that*

$$\forall h \geq 0, \forall x \in \mathcal{P}_{h,i_h^\star}, \quad f(x) \geq f(x^\star) - \nu\rho^h.$$

The values (ν, ρ) defines a lower bound on the possible drop of f near the optimum $x^\star$ according to the partitioning. The choice of the exponential rate $\nu\rho^h$ is made to cover a very large class of functions, as well as to relate to results from prior work. In particular, for a standard partitioning on $\mathbb{R}^p$ and any $\alpha, \beta > 0$, any function f such that $f(x) \sim_{x \to x^\star} \beta||x - x^\star||^\alpha$ fits this assumption. This is also the case for more complicated functions such as the one illustrated in Figure 1. An example of a function and a partitioning that does not satisfy this assumption is the function $f : x \mapsto 1/\ln x$ and a standard partitioning of $[0, 1)$ because the function decreases too fast around $x^\star = 0$. As observed by Valko [14], this assumption can be weaken to hold only for values of f that are η-close to $f(x^\star)$ up to an η-dependent constant in the regret.

Let us note that the set of assumptions made by prior work (Section 2.2) can be reformulated using solely Assumption 1. For example, for any $f(x) \sim_{x \to x^\star} \beta||x - x^\star||^\alpha$, one could consider the semi-metric $\ell(x, y) = \beta||x - y||^\alpha$ for which the corresponding near-optimality dimension defined by Equation 1 is $d = 0$. Yet we argue that our setting provides a more natural way to describe the complexity of the optimization problem for a given hierarchical partitioning.

Indeed, existing algorithms, that use a hierarchical partitioning of $\mathcal{X}$, like `HOO`, do not use the full metric information but instead only use the values ν and ρ, paired up with the partitioning. Hence, the precise value of the metric does not impact the algorithms't decisions, neither their performance. What really matters, is how the hierarchical partitioning of $\mathcal{X}$ fits f. Indeed, this fit is what we measure. To reinforce this argument, notice again that any function can be trivially optimized given a perfectly adapted partitioning, for instance the one that associates $x^\star$ to one child of the root.

Also, the previous analyses tried to provide performance guaranties based only on the metric and f. However, since the metric is assumed to be such that *the cells of the partitioning are well shaped*, the large diversity of possible metrics vanishes. Choosing such metric then comes down to choosing only ν, ρ, and a hierarchical decomposition of $\mathcal{X}$. Another way of seeing this is to remark that previous works make an assumption on both the function and the metric, and an other on both the metric and the partitioning. We underline that the metric is actually there just to create a link between the function and the partitioning. By discarding the metric, we merge the two assumptions into a single one and convert a topological problem into a combinatorial one, leading to easier analysis.

To proceed, we define a *new near-optimality dimension*. For any $\nu > 0$ and $\rho \in (0, 1)$, the near-optimality dimension $d(\nu, \rho)$ of f with respect to the partitioning $\mathcal{P}$ is defined as follows.

Definition 1. *Near-optimality dimension of f is*

$$d(\rho) \stackrel{\text{def}}{=} \inf \left\{ d' \in \mathbb{R}^+ : \exists C > 0, \ \forall h \geq 0, \ \ \mathcal{N}_h(2\nu\rho^h) \leq C\rho^{-d'h} \right\}$$

where $\mathcal{N}_h(\varepsilon)$ is the number of cells $\mathcal{P}_{h,i}$ of depth h such that $\sup_{x \in \mathcal{P}_{h,i}} f(x) \geq f(x^\star) - \varepsilon$.

The hierarchical decomposition of the space $\mathcal{X}$ is the only prior information available to the algorithm. The (new) near-optimality dimension is a measure of how well is this partitioning adapted to f. More precisely, it is a measure of the size of the near-optimal set, i.e., the cells which are such that $\sup_{x \in \mathcal{P}_{h,i}} f(x) \geq f(x^\star) - \varepsilon$. Intuitively, this corresponds to the set of cells that any algorithm would have to sample in order to discover the optimum.

As an example, any f such that $f(x) \sim_{x \to x^\star} ||x - x^\star||^\alpha$, for any $\alpha > 0$, has a zero near-optimality dimension with respect to the standard partitioning of $\mathbb{R}^d$ and an appropriate choice of ρ. As discussed by Valko et al. [14], any function such that the upper and lower envelopes of f near its maximum are of the same order has a near-optimality dimension of zero for a standard partitioning of $[0, 1]$. An example of a function with $d > 0$ for the standard partitioning is the one in Figure 1. Functions that behave differently in different dimensions have also $d > 0$. Nonetheless, for a specifically handcrafted partitioning, it is possible to have $d = 0$ even for those troublesome functions.

Under our new assumption and our new definition of near-optimality dimension, one can prove the same regret bound for HOO as Bubeck et al. [4] and the same can be done for other related algorithms.

3 The POO algorithm

3.1 Description of POO

The POO algorithm uses, as a subroutine, an optimizing algorithm that *requires the knowledge* of the function smoothness. We use HOO [4] as the base algorithm, but other algorithms, such as HCT [2], could be used as well. POO, with pseudocode in Algorithm 1, runs several HOO instances in parallel, hence the name ***p**arallel **o**ptimistic **o**ptimization*. The number of base HOO instances and other parameters are adapted to the budget of evaluations and are automatically decided on the fly.

Each instance of HOO requires two real numbers ν and ρ. Running HOO parametrized with (ρ, ν) that are far from the optimal one $(\nu_\star, \rho_\star)$[3] would cause HOO to underperform. Surprisingly, our analysis of this *suboptimality gap* reveals that it does not decrease too fast as we stray away from $(\nu_\star, \rho_\star)$. This motivates the following observation. If we *simultaneously* run a slew of HOOs with different (ν, ρ)s, one of them is going to perform decently well.

In fact, we show that to achieve good performance, we only require $(\ln n)$ HOO instances, where n is the current number of function evaluations. Notice, that we do not require to know the total number of rounds in advance which hints that we can hope for a *naturally anytime* algorithm.

The strategy of POO is quite simple: It consists of running N instances of HOO in parallel, that are all launched with different (ν, ρ)s. At the end of the whole process, POO selects the instance $s^\star$ which performed the best and returns one of the points selected by this instance, chosen uniformly at random. Note that just using a doubling trick in HOO with increasing values of ρ and ν is not enough to guarantee a good performance. Indeed, it is important to keep track of all HOO instances. Otherwise, the regret rate would suffer way too much from using the value of ρ that is too far from the optimal one.

Algorithm 1 POO

Parameters: K, $\mathcal{P} = \{\mathcal{P}_{h,i}\}$
  Optional parameters: $\rho_{\max}, \nu_{\max}$
Initialization:
  $D_{\max} \leftarrow \ln K / \ln(1/\rho_{\max})$
  $n \leftarrow 0$ {number of evaluation performed}
  $N \leftarrow 1$ {number of HOO instances}
  $\mathcal{S} \leftarrow \{(\nu_{\max}, \rho_{\max})\}$ {set of HOO instances}
while computational budget is available **do**
  while $N \geq \frac{1}{2} D_{\max} \ln(n/(\ln n))$ **do**
    for $i \leftarrow 1, \ldots, N$ **do** {start new HOOs}
      $s \leftarrow \left(\nu_{\max}, {\rho_{\max}}^{2N/(2i+1)}\right)$
      $\mathcal{S} \leftarrow \mathcal{S} \cup \{s\}$
      Perform $\frac{n}{N}$ function evaluation with HOO(s)
      Update the average reward $\widehat{\mu}[s]$ of HOO(s)
    end for
    $n \leftarrow 2n$
    $N \leftarrow 2N$
  end while{ensure there is enough HOOs}
  for $s \in \mathcal{S}$ **do**
    Perform a function evaluation with HOO(s)
    Update the average reward $\widehat{\mu}[s]$ of HOO(s)
  end for
  $n \leftarrow n + N$
end while
$s^\star \leftarrow \text{argmax}_{s \in \mathcal{S}} \widehat{\mu}[s]$
Output: A random point evaluated by HOO($s^\star$)

[3] the parameters (ν, ρ) satisfying Assumption 1 for which $d(\nu, \rho)$ is the smallest

For clarity, the pseudo-code of Algorithm 1 takes $\rho_{\max}$ and $\nu_{\max}$ as parameters but in Appendix C we show how to set $\rho_{\max}$ and $\nu_{\max}$ *automatically* as functions of the number of evaluations, i.e., $\rho_{\max}(n)$, $\nu_{\max}(n)$. Furthermore, in Appendix D, we explain how to share information between the HOO instances which makes the empirical performance *light-years better*.

Since POO is anytime, the number of instances $N(n)$ is time-dependent and does not need to be known in advance. In fact, $N(n)$ is increased alongside the execution of the algorithm. More precisely, we want to ensure that

$$N(n) \geq \tfrac{1}{2} D_{\max} \ln(n/\ln n), \quad \text{where} \quad D_{\max} \stackrel{\text{def}}{=} (\ln K)/\ln(1/\rho_{\max}).$$

To keep the set of different (ν, ρ)s well distributed, the number of HOOs is not increased one by one but instead is doubled when needed. Moreover, we also require that HOOs run in parallel, perform the same number of function evaluations. Consequently, when we start running new instances, we first ensure to make these instances on par with already existing ones in terms of number of evaluations.

Finally, as our analysis reveals, a good choice of parameters (ρ_i) is not a uniform grid on $[0, 1]$. Instead, as suggested by our analysis, we require that $1/\ln(1/\rho_i)$ is a uniform grid on $[0, 1/(\ln 1/\rho_{\max})]$. As a consequence, we add HOO instances in batches such that $\rho_i = {\rho_{\max}}^{N/i}$.

3.2 Upper bound on POO's regret

POO does not require the knowledge of a (ν, ρ) verifying Assumption 1 and[4] yet we prove that it achieves a performance close[5] to the one obtained by HOO using the best parameters $(\nu_\star, \rho_\star)$. This results solves the open question of Valko et al. [14], whether the stochastic optimization of f with unknown parameters (ν, ρ) when $d > 0$ for the standard partitioning is possible.

Theorem 1. *Let R_n be the simple regret of POO at step n. For any (ν, ρ) verifying Assumption 1 such that $\nu \leq \nu_{\max}$ and $\rho \leq \rho_{\max}$ there exists α such that for all n*

$$\mathbb{E}[R_n] \leq \kappa \cdot \left(\ln^2 n\right)/n\Big)^{1/(d(\nu,\rho)+2)}$$

Moreover, $\kappa = \alpha \cdot D_{\max}(\nu_{\max}/\nu_\star)^{D_{\max}}$, where α is a constant independent of ρ_{max} and ν_{max}.

We prove Theorem 1 in the Appendix A and B. Notice that Theorem 1 holds for any $\nu \leq \nu_{\max}$ and $\rho \leq \rho_{\max}$ and in particular for the parameters $(\nu_\star, \rho_\star)$ for which $d(\nu, \rho)$ is minimal as long as $\nu_\star \leq \nu_{\max}$ and $\rho_\star \leq \rho_{\max}$. In Appendix C, we show how to make $\rho_{\max}$ and $\nu_{\max}$ *optional*.

Note that given our definition of $D_{\max}$, we can always increase K, while decreasing ρ, and keep $\ln(K)/\ln(1/\rho)$ unchanged. Therefore, with a finer partitioning, smaller values of ρ are possible

The POO's performance should be compared to the simple regret of HOO run with the best parameters $\nu_\star$ and $\rho_\star$, which is of order

$$\mathcal{O}\left(\left((\ln n)/n\right)^{1/(d(\nu_\star,\rho_\star)+2)}\right).$$

Thus POO's performance is only a factor of $\mathcal{O}((\ln n)^{1/(d(\nu_\star,\rho_\star)+2)})$ away from the optimally fitted HOO. Furthermore, we our regret bound for POO is slightly better than the known regret bound for StoSOO [14] in the case when $d(\nu, \rho) = 0$ for the same partitioning, i.e., $\mathbb{E}[R_n] = \mathcal{O}(\ln n/\sqrt{n})$. With our algorithm and analysis, we generalize this bound for any value of $d \geq 0$.

Note that we only give a simple regret bound for POO whereas HOO ensures a bound on both the cumulative and simple regret.[6] Notice that since POO runs several HOOs with non-optimal values of the (ν, ρ) parameters, this algorithm explores much more than optimally fitted HOO, which dramatically impacts the cumulative regret. As a consequence, our result applies to the simple regret only.

[4] note that several possible values of those parameters are possible for the same function

[5] up to a logarithmic term $\sqrt{\ln n}$ in the simple regret

[6] in fact, the bound on the simple regret is a direct consequence of the bound on the cumulative regret [3]

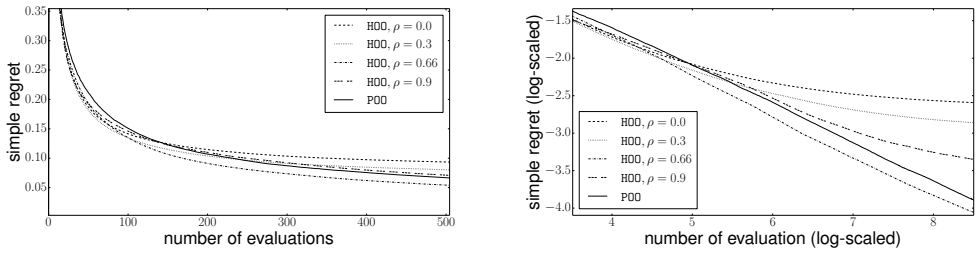

Figure 2: Regret of POO and HOO run for different values of ρ.

4 Experiments

We ran experiments on the function plotted in Figure 1 for HOO algorithms with different values of ρ and the POO[7] algorithm for $\rho_{\max} = 0.9$. This function, as described in Section 1, has an upper and lower envelope that are not of the same order and therefore has $d > 0$ for a standard partitioning.

In Figure 2, we show the simple regret of the algorithms as function of the number of evaluations. In the figure on the left, we plot the simple regret after 500 evaluations. In the right one, we plot the regret after 5000 evaluations in the log-log scale, in order to see the trend better. The HOO algorithms return a random point chosen uniformly among those evaluated. POO does the same for the best empirical instance of HOO. We compare the algorithms according to the expected simple regret, which is the difference between the optimum and the expected value of function value at the point they return. We compute it as the average of the value of the function for all evaluated points. While we did not investigate possibly different heuristics, we believe that returning the deepest evaluated point would give a better empirical performance.

As expected, the HOO algorithms using values of ρ that are too low, do not explore enough and become quickly stuck in a local optimum. This is the case for both UCT (HOO run for $\rho = 0$) and HOO run for $\rho = 0.3$. The HOO algorithm using ρ that is too high waste their budget on exploring too much. This way, we empirically confirmed that the performance of the HOO algorithm is greatly impacted by the choice of this ρ parameter for the function we considered. In particular, at $T = 500$, the empirical regret of HOO with $\rho = 0.66$ was a half of the regret of UCT.

In our experiments, HOO with $\rho = 0.66$ performed the best which is a bit lower than what the theory would suggest, since $\rho_\star = 1/\sqrt{2} \approx 0.7$. The performance of HOO using this parameter is almost matched by POO. This is surprising, considering the fact the POO was simultaneously running 100 different HOOs. It shows that carefully sharing information between the instances of HOO, as described and justified in Appendix D, has a major impact on empirical performance. Indeed, among the 100 HOO instances, only two (on average) actually needed a fresh function evaluation, the 98 could reuse the ones performed by another HOO instance.

5 Conclusion

We introduced POO for global optimization of stochastic functions with unknown smoothness and showed that it competes with the best known optimization algorithms that know this smoothness. This results extends the previous work of Valko et al. [14], which is only able to deal with a near-optimality dimension $d = 0$. POO is provable able to deal with a trove of functions for which $d \geq 0$ for a standard partitioning. Furthermore, we gave a new insight on several assumptions required by prior work and provided a more natural measure of the complexity of optimizing a function given a hierarchical partitioning of the space, without relying on any (semi-)metric.

Acknowledgements The research presented in this paper was supported by French Ministry of Higher Education and Research, Nord-Pas-de-Calais Regional Council, a doctoral grant of École Normale Supérieure in Paris, Inria and Carnegie Mellon University associated-team project EduBand, and French National Research Agency project ExTra-Learn (n.ANR-14-CE24-0010-01).

[7] code available at https://sequel.lille.inria.fr/Software/POO

References

[1] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning*, 47(2-3):235–256, 2002.

[2] Mohammad Gheshlaghi Azar, Alessandro Lazaric, and Emma Brunskill. Online Stochastic Optimization under Correlated Bandit Feedback. In *International Conference on Machine Learning*, 2014.

[3] Sébastien Bubeck, Rémi Munos, and Gilles Stoltz. Pure Exploration in Finitely-Armed and Continuously-Armed Bandits. *Theoretical Computer Science*, 412:1832–1852, 2011.

[4] Sébastien Bubeck, Rémi Munos, Gilles Stoltz, and Csaba Szepesvári. $\mathcal{X}$-armed Bandits. *Journal of Machine Learning Research*, 12:1587–1627, 2011.

[5] Sébastien Bubeck, Gilles Stoltz, and Jia Yuan Yu. Lipschitz Bandits without the Lipschitz Constant. In *Algorithmic Learning Theory*, 2011.

[6] Adam D. Bull. Adaptive-treed bandits. *Bernoulli*, 21(4):2289–2307, 2015.

[7] Pierre-Arnaud Coquelin and Rémi Munos. Bandit Algorithms for Tree Search. In *Uncertainty in Artificial Intelligence*, 2007.

[8] Robert Kleinberg, Alexander Slivkins, and Eli Upfal. Multi-armed Bandit Problems in Metric Spaces. In *Symposium on Theory Of Computing*, 2008.

[9] Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo Planning. In *European Conference on Machine Learning*, 2006.

[10] Rémi Munos. Optimistic Optimization of Deterministic Functions without the Knowledge of its Smoothness. In *Neural Information Processing Systems*, 2011.

[11] Rémi Munos. From Bandits to Monte-Carlo Tree Search: The Optimistic Principle Applied to Optimization and Planning. *Foundations and Trends in Machine Learning*, 7(1):1–130, 2014.

[12] Philippe Preux, Rémi Munos, and Michal Valko. Bandits Attack Function Optimization. In *Congress on Evolutionary Computation*, 2014.

[13] Aleksandrs Slivkins. Multi-armed Bandits on Implicit Metric Spaces. In *Neural Information Processing Systems*, 2011.

[14] Michal Valko, Alexandra Carpentier, and Rémi Munos. Stochastic Simultaneous Optimistic Optimization. In *International Conference on Machine Learning*, 2013.

A Proof sketch of Theorem 1

In this part we give the roadmap of the proof. The full proof is in Appendix B.

First step For any choice of $\rho_\star$ verifying Assumption 1 and any suboptimal ρ such that

$$0 < \rho_\star \leq \rho < 1,$$

we bound the difference of near-optimality dimension

$$d(\rho) - d(\rho_\star) \leq \ln K \left(\frac{1}{\ln(1/\rho)} - \frac{1}{\ln(1/\rho_\star)} \right).$$

And deduce that

$$\min_{i:\rho_i \geq \rho_\star} [d(\rho_i) - d(\rho_\star)] \leq \frac{D_{\max}}{N},$$

Second step By simultaneously running a large number of HOO instances, we ensure that for all $\rho_\star \leq \rho_{\max}$, one of them uses a ρ close to $\rho_\star$ and therefore suffers a low regret. On the other hand, simultaneously running a large number of HOOs has a cost, as more evaluations need to be done at each step, one for each HOO. We optimize this tradeoff to deduce the following good choice of δ, which is the maximum distance $|d(\rho_i) - d(\rho_j)|$, where i and j are two consecutive HOOs.

$$\delta = \mathcal{O}(\ln(t/\ln t)).$$

Third step Using the result of the second step, we can compute the simple regret R_n^ρ of the HOO instance running with the parameter $\bar{\rho} > \rho_\star$, which is the closest to $\rho_\star$. Note that, as POO is running, the instance it choose may change over time and so $\bar{\rho}$ depends on n.

We prove that there exists a constant $\alpha > 0$ such that for all n, $\nu_{\max} > 0$, and $\rho_{\max} < 1$,

$$R_n^\rho \leq \alpha \cdot D_{\max}(\nu_{\max}/\nu_\star)^{D_{\max}} \left(\left(\ln^2 n \right) / n \right)^{1/(d(\bar{\rho})+2)}.$$

Fourth step At the end of the algorithm, we empirically determine which HOO performed the best. However, this best empirical instance may not be the instance running with ρ closest to the optimal unknown $\rho_\star$. Nonetheless, we prove that this error is small enough such that it only impacts the simple regret by a constant factor.

B Full proof of Theorem 1

B.1 First step

We show that for any choice of $\rho_\star$ verifying Assumption 1 and any ρ such that $0 < \rho_\star \leq \rho < 1$,

$$d(\rho) - d(\rho_\star) \leq \ln K \left(\frac{1}{\ln(1/\rho)} - \frac{1}{\ln(1/\rho_\star)} \right).$$

We start by defining $\mathcal{I}_h(\varepsilon)$ as the set of cells of depth h which are ε-near-optimal,

$$\mathcal{I}_h(\varepsilon) \stackrel{\text{def}}{=} \left\{ i : \sup_{x \in \mathcal{P}_{h,i}} f(x) \geq f(x_\star) - \varepsilon \right\}.$$

$\mathcal{N}_h(\varepsilon)$, defined in Section 1, is then equal to the cardinality of $\mathcal{I}_h(\varepsilon)$. Notice that if a cell (h,i) is ε-near-optimal then all of its antecedents are also ε-near-optimal. Therefore, for any ε and $h' > h$, the cells in $\mathcal{I}_{h'}(\varepsilon)$ are descendants of the cells in $\mathcal{I}_h(\varepsilon)$.

Since the number of descendants at depth h' of a cell at depth $h' > h$ is bounded by $K^{h'-h}$ we bound the cardinality $\mathcal{N}_h(\varepsilon)$ of $\mathcal{I}_{h'}(\varepsilon)$,

$$\forall \varepsilon, \forall h' > h, \quad \mathcal{N}_{h'}(\varepsilon) \leq K^{h'-h} \mathcal{N}_h(\varepsilon).$$

By definition of the near-optimality dimension, we know that for any $\nu > 0$ and $\rho_\star \in (0,1)$, there exists C such that for all h,

$$\mathcal{N}_h\left(2\nu\rho^h\right) \leq C\rho^{-d(\rho)h}.$$

We define $C(\nu,\rho)$ as the smallest C verifying the above condition.

For any $0 < \nu_\star < \nu$, $0 < \rho_\star < \rho < 1$ and any integer $h \geq h_{\min} \stackrel{\text{def}}{=} \ln(\nu/\nu_\star)/\ln(1/\rho)$ let us define $h_\star$ as the greatest integer such that $\nu\rho^h < \nu_\star\rho_\star^{h_\star}$. From this definition, we get $\nu\rho^h \geq \nu_\star\rho_\star^{h_\star+1}$ from which we deduce that

$$h_\star \geq h \cdot \frac{\ln\rho}{\ln\rho_\star} + \frac{\ln\nu - \ln\nu_\star}{\ln\rho_\star} - 1,$$

and then

$$h - h_\star \leq h_\star \ln\rho_\star \left(\frac{1}{\ln\rho} - \frac{1}{\ln\rho_\star}\right) + \frac{\ln\rho_\star + \ln\nu_\star - \ln\nu}{\ln\rho}.$$

Since $\mathcal{N}_h(\varepsilon)$ is not increasing in ε, $\nu\rho^h < \nu_\star\rho_\star^{h_\star}$ implies

$$\mathcal{N}_h(2\nu\rho^h) \leq \mathcal{N}_h(2\nu_\star\rho_\star^{h_\star}).$$

We now put everything together to obtain

$$\begin{aligned}
\mathcal{N}_h(2\nu\rho^h) &\leq \mathcal{N}_h(2\nu_\star\rho_\star^{h_\star}) \\
&\leq K^{h-h_\star}\mathcal{N}_{h_\star}(2\nu_\star\rho_\star^{h_\star}) \\
&\leq K^{(\ln\rho_\star+\ln\nu_\star-\ln\nu)/\ln\rho + h_\star\ln\rho_\star(1/\ln\rho - 1/\ln\rho_\star)} C(\nu_\star,\rho_\star)\rho_\star^{-d(\rho_\star)h_\star} \\
&\leq C(\nu_\star,\rho_\star)K^{(\ln\rho_\star+\ln\nu_\star-\ln\nu)/\ln\rho}\rho_\star^{-h_\star[d(\rho_\star)+\ln K(1/\ln(1/\rho)-1/\ln(1/\rho_\star))]}.
\end{aligned}$$

From $\nu\rho^h < \nu_\star\rho_\star^{h_\star}$ and $\nu_\star < \nu$ we get $\rho^{-h} > \rho_\star^{-h_\star}$ and therefore

$$\mathcal{N}_h(2\nu\rho^h) \leq C(\nu_\star,\rho_\star)K^{(\ln\rho_\star+\ln\nu_\star-\ln\nu)/\ln\rho}\rho^{-h[d(\rho_\star)+\ln K(1/\ln(1/\rho)-1/\ln(1/\rho_\star))]}.$$

We just proved that there exists C such that for all $h > 0$

$$\mathcal{N}_h(2\nu\rho^h) \leq C\rho^{-h[d(\rho_\star)+\ln K(1/\ln(1/\rho)-1/\ln(1/\rho_\star))]}.$$

By taking

$$C \stackrel{\text{def}}{=} \max\left(C(\nu_\star,\rho_\star)K^{(\ln\rho_\star+\ln\nu_\star-\ln\nu)/\ln\rho}, K^{h_{\min}}\right),$$

we deduce by the definition of the near-optimality dimension the following bound

$$d(\rho) \leq d(\rho_\star) + \ln K\left(\frac{1}{\ln(1/\rho)} - \frac{1}{\ln(1/\rho_\star)}\right).$$

We can now deduce that POO should use ρ_i parameters that satisfy

$$\frac{1}{\ln(1/\rho_i)} = \frac{i}{N}\frac{1}{\ln(1/\rho_{\max})},$$

where N is the total number of HOO instances run and $i \in \{1,\dots,N\}$.

We now define $\bar{\rho}$ as the closest ρ_i to $\rho_\star$ used by an existing HOO instance, such that $\rho_i > \rho_\star$.

$$\bar{\rho} \stackrel{\text{def}}{=} \arg\min_{i:\rho_i \geq \rho_\star} \left[d(\rho_i) - d(\rho_\star)\right].$$

Since we assumed that $\rho_\star < \rho_{\max}$, we know that

$$d(\bar{\rho}) - d(\rho_\star) \leq \frac{D_{\max}}{N}, \quad \text{with} \quad D_{\max} \stackrel{\text{def}}{=} (\ln K)/\ln(1/\rho_{\max}).$$

B.2 Second step

Let us now compute the optimal number of N instances to run in parallel. We bound the logarithm of the cumulative regret $R_t^{\nu,\rho}$ of a single HOO instance using parameters ν and ρ after this particular instance performed t function evaluations. In particular, we bound the regret by a linear approximation for $\rho \sim \rho_\star$. In the following, β is a numerical constant coming from the analysis of HOO [4]. For all $t > 0$, we have

$$\begin{aligned}
\ln R_t^{\nu,\rho} &\leq \ln \beta + \frac{\ln C(\nu,\rho)}{2+d(\rho)} - \frac{\ln\left(t/\ln t\right)}{2+d(\rho)} \\
&= \ln \beta + \frac{\ln C(\nu,\rho)}{2+d(\rho)} - \frac{\ln\left(t/\ln t\right)}{2+d(\rho_\star)} \cdot \frac{1}{1+\left(d\left(\rho\right)-d\left(\rho_\star\right)\right)/\left(2+d\left(\rho_\star\right)\right)} \\
&\leq \ln \beta + \frac{\ln C(\nu,\rho)}{2+d(\rho)} - \frac{\ln\left(t/\ln t\right)}{2+d(\rho_\star)} \cdot \left(1 - \frac{d(\rho)-d(\rho_\star)}{2+d(\rho_\star)}\right).
\end{aligned}$$

After n function evaluations by POO, each instance performed at least $t = \lfloor n/N \rfloor$ function evaluations. We can now bound the commutative regret $R_n^{\texttt{POO},\nu,\bar{\rho}}$ of the HOO instance using ν and $\bar{\rho}$ after n evaluations performed by all the instances

$$\ln R_n^{\texttt{POO},\nu,\bar{\rho}} \leq \ln \beta + \frac{\ln C(\nu,\bar{\rho})}{2+d(\bar{\rho})} + \ln\left(\frac{\ln\lfloor n/N\rfloor}{\lfloor n/N\rfloor}\right)\left(\frac{1}{2+d(\rho_\star)} - \frac{D_{\max}/N}{(2+d(\rho_\star))^2}\right). \tag{2}$$

Optimizing this upper bound for N leads to the following choice of N,

$$N \sim \tfrac{1}{2} D_{\max} \ln\left(n/\ln n\right).$$

Therefore, in POO we choose to ensure $N \geq \frac{1}{2} D_{\max} \ln\left(n/\ln n\right)$.

If the time horizon was known in advance, N could be any integer. Nevertheless, since the algorithm is anytime, all the previous HOO instances have to be kept and new instances need to be added in between. Therefore, we restrict N to be of the form 2^i, for $i \in \mathbb{N}$.

As a consequence of this choice, N can be at most 2 times its lower bound and therefore

$$\tfrac{1}{2} D_{\max} \ln\left(n/\ln n\right) \leq N \leq D_{\max} \ln\left(n/\ln n\right).$$

B.3 Third step

Using our choice of N, we can bound the regret of the HOO instance using $\bar{\rho}$. We proceed by separately bounding each of the terms in Equation 2.

$$\begin{aligned}
\frac{\ln C(\nu,\bar{\rho})}{2+d(\bar{\rho})} &\leq \frac{1}{2+d(\rho_\star)} \ln C(\nu,\bar{\rho}) \\
&\leq \frac{1}{2+d(\rho_\star)} \ln \max\left(C(\nu_\star,\rho_\star) K^{(\ln\rho_\star + \ln\nu_\star - \ln\nu)/\ln\bar{\rho}}, K^{h_{\min}}\right) \\
&\leq \frac{1}{2+d(\rho_\star)} \max\left(\ln C(\nu_\star\rho_\star) + \ln K\left(\frac{\ln 1/\rho_\star}{\ln 1/\bar{\rho}} + \frac{\ln\left(\nu/\nu_\star\right)}{\ln 1/\rho}\right), \ln\left[K^{\ln(\nu/\nu_\star)/\ln(1/\rho)}\right]\right) \\
&\leq \frac{1}{2+d(\rho_\star)} \max\left(\ln C(\nu_\star\rho_\star) + \max\left(\frac{\ln K \ln \rho_\star D_{\max}}{N}, 2\right) + \frac{\ln K \ln \frac{\nu_{\max}}{\nu_\star}}{\ln 1/\rho}, D_{\max} \ln \frac{\nu}{\nu_\star}\right) \\
&\leq \gamma + \frac{D_{\max}}{2+d(\rho_\star)} \ln\left(\nu_{\max}/\nu_\star\right)
\end{aligned}$$

In the last expression, γ is independent of $\nu_{\max}$, $\rho_{\max}$, and N.

We now use $N \leq D_{\max} \ln\left(n/\ln n\right)$ to get

$$\ln\left(\frac{\ln\lfloor n/N\rfloor}{\lfloor n/N\rfloor}\right) \leq \ln\left(D_{\max} \ln n \ln\left(n/\ln n\right)/n\right).$$

To bound the last term, we use $\frac{1}{2} D_{\max} \ln(n/\ln n) \leq N$ to get

$$-\ln\left(\frac{\ln\lfloor n/N\rfloor}{\lfloor n/N\rfloor}\right)\frac{D_{\max}/N}{(2+d(\rho_\star))^2} \leq \ln\left(\frac{1}{D_{\max}}\cdot\frac{n}{\ln n}\cdot\frac{1}{\ln(n/\ln n)}\right)\frac{1}{2\ln(n/\ln n)} \leq 2.$$

We can finally bound the regret $R_n^{\texttt{POO},\bar{\rho}}$ of the HOO instance using $\bar{\rho}$ after n function evaluations overall. Combining the results above, we know that for all n, $\nu_{\max}$, and $\rho_{\max}$,

$$R_n^{\texttt{POO},\bar{\rho}} \leq \beta\exp(\gamma+2)\left(D_{\max}\left(\nu_{\max}/\nu_\star\right)^{D_{\max}}(\ln n)\ln(n/\ln n)/n\right)^{1/(2+d(\bar{\rho}))}.$$

We bound $\ln(n/\ln n)$ by $\ln n$ to get the following bound, there exists α independent of $\rho_{\max}$ and $\nu_{\max}$ such that

$$R_n^{\texttt{POO},\bar{\rho}} \leq \alpha\cdot D_{\max}(\nu_{\max}/\nu_\star)^{D_{\max}}\left(\left(\ln^2 n\right)/n\right)^{1/(d(\bar{\rho})+2)}.$$

B.4 Fourth step

Let $(X_{i,j})_{i\leq n, j\leq N}$ be a family of points in $\mathcal{X}$ evaluated by POO. We denote $\widehat{f}(X_{i,j})$ the noisy evaluation at $X_{i,j}$ and $f(X_{i,j}) = \mathbb{E}[\widehat{f}(X_{i,j})]$. We also define:

$$\mu_j \overset{\text{def}}{=\!=} \frac{1}{n}\sum_{i=1}^{n} f(X_{i,j}) \qquad \widehat{\mu}_j \overset{\text{def}}{=\!=} \frac{1}{n}\sum_{i=1}^{n}\widehat{f}(X_{i,j})$$

$$j \overset{\text{def}}{=\!=} \underset{1\leq j\leq N}{\arg\max}\,\mu_j \qquad \widehat{j} \overset{\text{def}}{=\!=} \underset{1\leq j\leq N}{\arg\max}\,\widehat{\mu}_j$$

By Hoeffding-Azuma inequality for martingale differences, for any $\Delta > 0$,

$$\mathbb{P}\left[\sum_{i=1}^{n}\widehat{f}(X_{i,j}) - f(X_{i,j}) > n\Delta\right] \leq \exp\left(-\frac{2(n\Delta)^2}{n}\right).$$

Therefore

$$\mathbb{P}[\widehat{\mu}_j - \mu_j > \Delta] \leq \exp\left(-2n\Delta^2\right).$$

As we have

$$\forall x \geq 0, x\cdot\exp\left(-2nx^2\right) \leq \frac{e^{-2}}{2\sqrt{n}},$$

we can now integrate $\exp\left(-2n\Delta^2\right)$ over $\Delta \in [0,1]$ to get

$$\mathbb{E}[\widehat{\mu}_j - \mu_j] \leq \frac{e^{-2}}{2\sqrt{n}}.$$

Now consider

$$\mathbb{E}\left[\mu_j - \mu_{\widehat{j}}\right] = \mathbb{E}\left[\mu_j - \widehat{\mu}_j\right] + \mathbb{E}\left[\widehat{\mu}_j - \widehat{\mu}_{\widehat{j}}\right] + \mathbb{E}\left[\widehat{\mu}_{\widehat{j}} - \mu_{\widehat{j}}\right].$$

Notice that the first and last term are both bounded by $e^{-2}/(2\sqrt{n})$ and the middle term is negative. Finally, taking a union bound over the N variables μ_j we get

$$\mathbb{E}\left[\mu_{j_\star} - \mu_{\widehat{j}_\star}\right] \leq \frac{e^{-2}N}{\sqrt{n}}.$$

As $N = o(\ln n)$, we conclude that this additional term is negligible with respect to

$$\left(\ln n\ln(n/\ln n)/n\right)^{1/(2+d(\rho_\star))}.$$

C Increasing sequence for $\rho_{\max}$ and $\nu_{\max}$

Besides the number K of children for each cell, POO needs two parameters, $\rho_{\max} \in (0, 1)$ and $\nu_{\max} > 0$. Theorem 1 states that POO run with those parameters performs almost as well as the best instance of HOO run with $\nu \leq \nu_{\max}$ and $\rho \leq \rho_{\max}$, i.e., corresponding to the near-optimality dimension $\min\{d(\nu, \rho), \nu \leq \nu_{\max}, \rho \leq \rho_{\max}\}$.

Therefore, the larger the values $\rho_{\max}$ and $\nu_{\max}$ used by POO, the wider the set of HOO instances that we can compete with. Nevertheless, large values of $\rho_{\max}$ and $\nu_{\max}$ impact the performance by a multiplicative constant of order $D_{\max}\nu_{\max}{}^{D_{\max}}$. This tradeoff between performance and size of our comparison class is unfortunate but unavoidable.

In practice, as we strive for an algorithm that does not require the knowledge of the smoothness we may increase the values of $\rho_{\max}(n)$ and $\nu_{\max}(n)$ with the number of evaluations n, so that the class of functions covered by POO gets bigger with the numerical budget. Nevertheless, the increase should be slow enough so that we do not compromise the performance. In particular, we will require that $\nu_{\max}(n)^{D_{\max}(n)}$ does not increase too fast. In fact, any sequence $\rho_{\max}(n)$ converging to 1 and $\nu_{\max}(n)$ diverging to infinity impacts the regret by an additive term which is the smallest time n such that $\rho^\star < \rho_{\max}(n)$ and $\nu^\star < \nu_{\max}(n)$, i.e., the first time the assumptions are verified. A slowly increasing sequence means a smaller impact on the regret rate but a higher additive term (a constant independent of n). Any sensible choice of increasing sequence $\rho_{\max}(n)$ and $\nu_{\max}(n)$, impacting the rate by only a subpolynomial factor, is a valid choice.

Algorithm 1 is described using constant $\rho_{\max}$ and $\nu_{\max}$ for clarity, but its implementation is easily modifiable to deal with increasing values of these two parameters while preserving the anytime property of the algorithm, as follows. At any time, all the HOO instances must use the same $\nu_{\max}$ parameter. On the other hand, considering $\rho_{\max}$, the value of $D_{\max}$ has to be increased such that the already running HOO instances stay relevant. One way to do that is to increase $D_{\max}$ as $D_{\max}(N+1)/N$ and run an additional HOO instance. An alternative solution is to perform, each time when needed, the following increment $\rho_{\max} \leftarrow \sqrt{\rho_{\max}}$ and run N additional HOO instances with parameters $\rho_{\max}{}^{2N/i}$, for $i \in \{1, \ldots, N\}$.

D Information sharing among parallel runs

Since we run several instances of HOO on the same partitioning of $\mathcal{X}$, we may think of *sharing the samples* among them, in order to decrease the estimation error. However, this needs to be done carefully in order to avoid adding unwanted bias in the estimation of the U values in the HOO instances. Ideally, each HOO instance would reuse all function evaluations acquired by all other instances. Unfortunately, this solution would not easily come with theoretical guaranties, as this would reduce artificially the confidence intervals at some cells and introduce *search bias*.

Instead, whenever a HOO instance requires a function evaluation, we perform a *look-up* to find out whether another HOO instance has already evaluated f at this point. In affirmative, then instead of evaluating the function at this point again, we simply *reuse* the sample. This way, HOO instances are *not given access to samples they never asked for*. However, the empirical regrets of HOOs becomes correlated with each other. This is not a problem because in B.4, we do not assume the independence between empirical means of HOOs, only the independence of rewards within each instance—which still holds. Therefore with this modification, our theoretical guaranties continue to apply. Note that if all the instances share all their rewards, then they are all equivalent and there is no mistake possible. Then one can show, that the worst case is when no rewards are shared and the error due to choosing the wrong instance actually decreases when the information is shared.

Finally, we want to stress that sharing information is extremely important in practice, as our experiments reveal. Since the number of HOO instances can be very large[8] one could expect the performance of POO to be pitiful. However, as the vast majority of the function evaluations are in practice shared, POO performs almost as well as HOO fitted with the best parameters. Summing up, although the performance bound on the simple regret with this modification is the same, empirical performance *improves tremendously*.

[8] even though it scales only as $\ln n$ with the number of evaluations n, it does not scale well with $\rho_{\max}$