Noisy Blackbox Optimization using Multi-fidelity Queries: A Tree Search Approach

Rajat Sen
University of Texas at Austin

Kirthevasan Kandasamy
Carnegie Mellon University

Sanjay Shakkottai
University of Texas at Austin

Abstract

We study the problem of black-box optimization of a noisy function in the presence of low-cost approximations or fidelities, which is motivated by problems like hyper-parameter tuning. In hyper-parameter tuning evaluating the black-box function at a point involves training a learning algorithm on a large dataset at a particular hyper-parameter and evaluating the validation error. Even a single such evaluation can be prohibitively expensive. Therefore, it is beneficial to use low-cost approximations, like training the learning algorithm on a sub-sampled version of the whole data-set. These low-cost approximations/fidelities can however provide a *biased* and *noisy* estimate of the function value. In this work, we combine structured state-space exploration through hierarchical partitioning with querying these partitions at multiple fidelities, and develop a multi-fidelity bandit based tree-search algorithm for noisy black-box optimization. We derive simple regret guarantees for our algorithm and validate its performance on real and synthetic datasets.

1 Introduction

Several important problems in the fields of physical simulations [29], industrial engineering [31] and model selection [36] in machine learning can be cast as sequential optimization of a function $f(.)$ over a domain $\mathcal{X}$, with black-box access. A black-box optimization algorithm evaluates the function at a set of sequentially chosen points $x_1, ..., x_n$ obtaining the function values $f(x_1), ..., f(x_n)$ in that order, and out-

puts a point $\hat{x}(n)$ at the end of the sequence. The performance of such algorithms are commonly evaluated in terms of *simple regret* defined as $S(n) = \sup_{x \in \mathcal{X}} f(x) - f(\hat{x}(n))$.

As the driving example, we consider the problem of tuning hyper-parameters of machine learning algorithms over large datasets. Here, the space of hyper-parameters constitutes the domain $\mathcal{X}$, while the function $f(x)$ represents the validation error after training the machine learning algorithm with the hyper-parameter setting $x \in \mathcal{X}$. Even a single evaluation of the function can be expensive and therefore conventional methods of black-box optimizations are infeasible (e.g. training a deep network over a large data-set could take days). This motivates our setting where access to lower-cost but biased estimates of the function is assumed [18, 12, 7] through a multi-fidelity query framework. In the context of hyper-parameter tuning, a possible low-cost approximation/fidelity can be training and validating the machine learning algorithm over a much smaller sub-sampled version of the dataset. However, the resulting validation error can be a *biased* and *noisy* estimate of the validation error on the whole dataset, and the bias depends on the size of the sub-sampled dataset used.

A well-studied approach for such problems is through bayesian optimization formulations [19, 17, 13, 20]. In our work, instead, we approach this problem using *tree-search* methods that have received much recent attention [22, 21, 3, 30, 39, 34]. Our study combines an exploration of the state-space using hierarchical binary partitions (a binary tree, with nodes representing subsets of the function domain $\mathcal{X}$ [3, 30, 11]), and queries at these nodes at different fidelities (represented through a continuous parameter over $[0, 1]$), that correspond to lower-cost evaluations of the function, but with both bias and noise.

The **main contributions** of this paper are as follows:

(i) We model multiple fidelities in the framework of black-box optimization of a noisy function, with hierarchical partitions in Section 3. We demonstrate that

bandit-based algorithms using hierarchical partitions can be naturally adapted to effectively use a continuous range of low-cost approximations.

(ii) We propose MFHOO (Algorithm 1 in Section 4) which is a natural adaptation of HOO [3] to the multi-fidelity setup. We analyze our algorithm under smoothness assumptions on the function and the partitioning, which are similar to the assumptions adopted in [11]. We show that our simple regret guarantees can be much stronger than that of HOO [3] (which operates only at the highest fidelity) under some natural conditions on the bias and cost functions. Our simple regret guarantees are presented in Theorem 1 and Corollary 1. MFHOO however needs the optimal smoothness parameters (ν^*, ρ^*) as inputs (similar to HOO [3], see Section 3 for details). In Section 4, we propose a second algorithm MFPOO (Algorithm 2) that can achieve a simple regret bound close to that of MFHOO, even without access to the parameters (ν^*, ρ^*). MFPOO is inspired by the recent techniques proposed in [11]. Theorem 2 provides simple regret guarantees for MFPOO.

(ii) Finally, in Section 6 we empirically compare the performance of our algorithm on real and synthetic data, against state of the art algorithms for black-box optimization. First, we perform simulated experiments on well-known benchmark functions. Then, we compare our algorithm to others in the context of hyper-parameter tuning for regression and classification tasks.

2   Related Work

This work builds on a rich literature on black-box optimization using hierarchical partitions [30, 39, 21, 3], which in turn build on optimistic algorithms in a bandit setting [2]. Our work is closely related to [3] which introduces HOO, a tree-search based algorithm for noisy black-box optimization. The regret guarantees in [3] are provided under some local Lipschitz assumption on the function with respect to a semi-metric and also some assumptions on the diameter of the nodes as a function of height in the tree. Later these assumptions were combined into a single combinatorial assumption in [11]. We follow similar assumptions. In a recent related work [34], the multi-fidelity tree-search problem has been studied in a setting where the evaluations of the black-box function are *not noisy*. In contrast we study the problem of multi-fidelity black box optimization in the presence of noise, which makes the setting much more challenging. In particular in the deterministic setting, one can simply descend through the tree without back-tracking. However in our setting, back-tracking in a tree occurs as time progresses

because additional samples improves estimates at various nodes in the tree, thus introducing bandit explore-exploit trade-offs, and hence can change their relative ordering over time. This distinction leads to a considerably different algorithm and analysis in our paper as compared to [34].

Multi-fidelity optimization is also relevant in several application [10, 25, 12, 32, 20], but theoretical guarantees are generally lacking in these studies. Recently, the multi-fidelity setting has been theoretically studied in online problems [40, 1, 33, 18]. In some recent works [19, 16, 17], UCB like algorithms with Bayesian Gaussian process assumptions on f have been analyzed in a multi-fidelity black-box optimization setting.

Also relevant to this work are the bandit based techniques for hyper-parameter optimization such as [28, 14]. However, these methods rely on iterative loss sequences and therefore are not directly applicable to the general multi-fidelity setup.

3   Problem Setting

The problem setting in this work is that of optimizing a function $f : \mathcal{X} \to \mathbb{R}$ with *noisy and biased* black-box access. Given a finite cost budget Λ, and access to fidelity-dependent queries (with higher cost for higher fidelity and lower bias), we need to determine $\hat{x} \in \mathcal{X}$ such that $|\sup_{x \in \mathcal{X}} f(x) - f(\hat{x})|$ is small. A similar problem setting has been considered in a recent work [34], however there the function evaluations at different fidelities are *not noisy*, that is the cheap approximations only add *bias* to the function values. In this work, we consider a setting where a function evaluation at a cheaper fidelity incurs both bias and sub-Gaussian noise around the function value. This makes the problem setting different and significantly more challenging.

3.1   Details about Function Evaluations

We build on the notation in [34], but modify it to permit noisy evaluations. The fidelity is modeled as a continuous parameter $z \in \mathcal{Z} \triangleq [0, 1]$. An evaluation of a function consists of two inputs (x, z) corresponding to the query point $x \in \mathcal{X}$ and fidelity $z \in \mathcal{Z}$, and results in a random variable Y as the output, with $Y = f_z(x) + \epsilon$. Here, ϵ is a sub-Gaussian noise random variable [4] with parameter σ^2 i.e $\mathbb{E}[\epsilon] = 0$ and $\mathbb{E}[\exp(s\epsilon)] \leq \exp(s^2\sigma^2/2)$ all $s \geq 0$. Such random variables will be denoted as subG(σ).

In our model, the mean of the query, $f_z(x)$, is biased and progressively smaller bias can be obtained but with higher costs. This is formalized through the monotone decreasing *bias function* (assumed to be

known in our analysis[1]) $\zeta : \mathcal{Z} \to \mathbb{R}^+$ and a monotone increasing *cost function* $\lambda : \mathcal{Z} \to \mathbb{R}^+$, with $\lambda(z) \geq 1$. A query at (x, z) results in an output mean with $|f_z(x) - f(x)| \leq \zeta(z)$ and at cost $\lambda(z)$. We assume that at the highest fidelity ($z = 1$), there is no bias, i.e., $f_1(x) = f(x)$. Finally, we assume that the global optimum is unique, i.e. $\exists x^* = \arg \max_{x \in \mathcal{X}} f(x)$.

Our problem setting is a natural model for the driving example of hyper-parameter tuning. For instance, the domain $\mathcal{X}$ represents the range of hyper-parameters. In the case of deep networks this may include kernel sizes and number of channels in different layers (rounded off to the nearest integers), learning rate of optimizers, dropout level and weight decay levels etc. While working with large datasets, training and validating on sub-sampled versions of the datasets serve as cheap approximations which can be modeled by our continuous range of fidelities $\mathcal{Z}$. For instance on Cifar10 [23], using the whole training set with $N_s > 50k$ samples corresponds to $z = 1$ while $z = 0$ can correspond to using $n_s \sim 1000$ samples. The range [1k,50k] can be linearly mapped to $[0, 1]$ (rounding off to the nearest interger number of samples). It is also clear that function evaluations obtained at the cheaper fidelities are both biased (validation accuracy is lower while using lesser number of samples) and noisy (randomness in sub-sampling and SGD). This falls under the purview of our evaluation model. The cost function simply corresponds to the time required to train a model over a sub-sampled version of the data-set and it increases with the number of samples (fidelity).

Putting all the pieces together, the problem setting is a sequential process as follows: Given a cost budget Λ at each time-slot $t = 1, 2, ..$, (i) Select a point X_t and evaluate it at a fidelity Z_t, (ii) Observe a noisy feedback Y_t, (iii) Incur a cost $\lambda(Z_t)$. This process in continued till the cost budget is exhausted. The goal is to find a point $x \in \mathcal{X}$ such that $f(x)$ is as close as possible to $f^* \triangleq f(x^*)$. We assume that x^* is a unique point in $\mathcal{X}$ where the supremum is achieved. The performance of a policy/optimization algorithm shall be characterized by *simple regret*.

3.2 Definition of Regret

Let $X_1, X_2, ...$ be the points queried by an algorithm/policy at fidelities $Z_1, Z_2, ...$ respectively. Let $\hat{X}_\Lambda$ be the point returned by the policy after $N(\Lambda)$ queries, where $N(\Lambda)$ is a random quantity such that $N(\Lambda) = \max\{n \geq 1 : \sum_{i=1}^n \lambda(Z_i) \leq \Lambda\}$ and Λ is the

---

total cost budget. Then the *simple regret* is given by $S(\Lambda) = f^* - \mathbb{E}[f(\hat{X}_\Lambda)]$. Here, the expectation is over the randomness in the observations and the policy. Note that the regret is measured only at the highest fidelity that is $z = 1$, as we are interested in optimizing the original function at $z = 1$. A similar definition of regret has been used before in the multi-fidelity literature [18, 19]. In the course of our analysis, we are also interested in the *cumulative regret* as an intermediate quantity. Given that a policy performs n evaluations or queries, the cumulative regret is given by $R_n = \mathbb{E}\left[\sum_{t=1}^n [f^* - f(X_t)]\right]$.

The problem of black-box optimization cannot be solved efficiently without assuming any structure or regularity assumptions on the function being evaluated. In the next sub-section, we assume access to a hierarchical partitioning of the domain $\mathcal{X}$ and impose some regularity assumption jointly on the function and the hierarchical partitions. Similar assumptions have been used in the theoretical analysis of several prior works [30, 11, 3, 21], that work with hierarchical partitioning of the domain.

3.3 Tree-like Partitions and Structural Assumptions

We first define the tree-like hierarchical partitions of the domain $\mathcal{X}$ and then provide some technical assumptions that impose regularity conditions jointly on the function and the partitions.

Hierarchical Partitions: We assume that the domain $\mathcal{X}$ is partitioned hierarchically according to an infinite binary tree. We denote this partitioning as $\mathcal{P} = \{(h, i)\}$, where h is a depth parameter and i is an index. For any depth $h \geq 0$, the cells $\{(h, i)\}_{1 \leq i \leq I_h}$ denote a partitioning of the space $\mathcal{X}$. Note that we use the notation (h, i) to denote both the index of a cell/node and the part of the domain it represents. For example, $x \in (h, i)$ refers to a point x within the part of the domain represented by the cell indexed at (h, i). At depth 0 there is a single cell $(0, 1) = \mathcal{X}$. A cell (h, i) can be split into two child nodes at depth $h + 1$, which are indexed as $(h + 1, 2i - 1)$ and $(h + 1, 2i)$. A cell (h, i) is said to be *queried*, when a fixed representative point $x_{h,i} \in (h, i)$ (ideally *centrally* located) is evaluated at any fidelity. Let $\mathcal{C}(h, i)$ denote all the descendant cells of (h, i) in the infinite tree. The unique cell at height h that contains the optima x^* is indexed as (h, i^*). For all sub-optimal cells (h, i) (those that do not contain x^*) the sub-optimality gap is denoted by $\Delta_{h,i} = f^* - \sup_{x \in (h,i)} f(x)$. In other words, a sub-optimal cell (h, i) is $\Delta_{h,i}$-optimal. Let $\mathcal{X}_\epsilon = \{x \in \mathcal{X} : f(x) \geq f^* - \epsilon\}$ for all $\epsilon > 0$.

For instance, consider the domain $\mathcal{X} = [0, 1] \times [0, 1] \subset$

$\mathbb{R}^2$. In this example, we only consider cells that intervals $\mathcal{C} = \{x \triangleq [x_1, x_2] \in \mathcal{X} : b_{1,l} < x_1 \le b_{1,u}, b_{2,l} < x_2 \le b_{2,u}\} \triangleq [[b_{1,l}, b_{1,u}], [b_{2,l}, b_{2,u}]]$. In this case, the hierarchical partitioning has a root $(0,1) = [[0,1],[0,1]]$ (the entire domain). At $h = 1$, the two children of the root resulting form a split along x_1 would be $(1,1) = [[0,0.5],[0,1]]$ and $(1,2) = [[0.5,1],[0,1]]$. The cell $(1,1)$ can be split into two children along the x_2 coordinate to result in $[[0.5,1],[0,0.5]]$ and $[[0.5,1],[0.5,1]]$, with the coordinate-wise midpoint of the cell used as the representative point $x_{h,i}$. In our example, $x_{1,1}$ can be chosen as the point $[0.25, 0.5]$.

We impose the following joint assumptions on the hierarchical partition $\mathcal{P}$ and the black-box function f, similar to the recent works [11, 34].

Assumption 1. *There exists ν and $\rho \in (0,1)$ such that for all cells (h,i) such that $\Delta_{h,i} \le c\nu\rho^h$ (for a constant $c \ge 0$) we have that, $f(x) \ge f^* - \max\{2c, c + 1\}\nu\rho^h$, for all $x \in (h,i)$.*

Finally, the following definition of *near-optimality-dimension* with parameters (ν, ρ) is borrowed from [11].

Definition 1. *The near-optimality dimension of f with respect to parameters (ν, ρ) is given by,*

$$d(\nu, \rho) \triangleq \inf \big\{ d' \in \mathbb{R}^+ : \exists C(\nu, \rho), \forall h \ge 0,$$
$$\mathcal{N}_h(2\nu\rho^h) \le C(\nu, \rho)\rho^{-d'h} \big\} \qquad (1)$$

where $\mathcal{N}_h(\epsilon)$ is the number of cells (h,i) such that $\sup_{x \in (h,i)} f(x) \ge f(x^*) - \epsilon$.

We denote the parameters associated with the minimum near optimality dimension $d(\nu, \rho)$ to be (ν^*, ρ^*). The optimal near-optimality dimension $d(\nu^*, \rho^*)$ controls the hardness of optimizing the function, given access to the particular hierarchical partition.

Our assumptions are closely related to the ones in the seminal paper [3]. Bubeck et al. [3] consider a similar noisy tree-search based black-box optimization problem. In their work, it was assumed that there is a dissimilarity metric $\ell(x,y)$ over the domain and the function satisfies a weak-Lipschitz condition around the optima with respect to the dissimilarity. These assumptions have been progressively refined [30, 39], with [11] providing a succinct assumption using the framework of hierarchical partitions. As in [34], we adopt this assumption in our paper. Assumption 1 is a slightly stronger version of Assumption 1 in [11] i.e., in [11] it has been assumed that Assumption 1 is satisfied with only $c = 0$. It has been recently observed [35] that it is highly non-trivial to prove the regret guarantees of HOO [3] under the assumptions in [11] and this stronger version may be indeed necessary. As-

sumption 1 is akin to ensuring that the conditions of Lemma 3 in [3] are satisfied.

4 Algorithms

We first propose MFHOO (Multi-Fidelity Hierarchical Optimistic Optimization) which is a noisy tree-search based multi-fidelity black-box optimization policy that requires the optimal smoothness parameters as input. Then we propose another algorithm MFPOO (Multi-Fidelity Parallel Optimistic Optimization) that can recover regret guarantees similar to that of MFHOO without the exact knowledge of smoothness parameters.

When (ν^*, ρ^*) are known: Our first algorithm MFHOO is inspired by the HOO strategy in [3]. We essentially show that the tree-search based technique in [3] can be naturally adapted to a multi-fidelity setting, with some modifications. In certain settings, our algorithm can achieve a much stronger simple regret scaling when compared to HOO which queries only at fidelity $z = 1$. The detailed pseudo-code of the algorithm is provided as Algorithm 1. We first establish some notation specific to our algorithm.

For any black-box optimization policy, let X_t be the random variable denoting the point queried at time t which is part of the cell (H_t, I_t), while Z_t is the fidelity at which the query is made. Let Y_t be the observation at the corresponding time-step such that $Y_t = f_{Z_t}(X_t) + \epsilon_t$, where $\epsilon_t \sim \text{subG}(\sigma)$. Let $T_{h,i}(t)$ be the number of times nodes in $\mathcal{C}(h,i)$ have been queried i.e, $T_{h,i}(t) = \sum_{s=1}^{t} \mathbb{1}\{(H_s, I_s) \in \mathcal{C}(h,i)\}$. Let $\mathcal{T}_t$ denote the finite subtree visited by the algorithm at the end of round t. The tree is initialized at $\mathcal{T}_0 = \{(0,1)\}$. Now we are at a position to introduce Algorithm 1.

The notable difference from HOO [3] is that all queries at height h are performed at a fidelity z_h such that $\zeta(z_h) = \nu\rho^h$. The intuition is that in 'near optimal' cells at height h, the function values of all points inside a cell are at most $O(\nu\rho^h)$ apart from each other. Therefore, if x^* belongs to a cell (h, i^*) at height h, then all points in that cell are $O(\nu\rho^h)$ optimal. Thus in the absence of noise, ideally beyond this point we would only like to expand nodes/cells that are at least $O(\nu\rho^h)$ optimal, which is only possible if the error due to the fidelities is $O(\nu\rho^h)$.

Remark 1. *Note that in the pseudo-code of Algorithm 1, the final point returned is randomly chosen from the points evaluated in the course of the algorithm. This is sufficient in theoretically bounding the simple regret as in Theorem 1. However, in practice several optimizations can be performed to return the most promising point among the ones evalu-*

Algorithm 1 MFHOO: Multi-Fidelity Hierarchical Optimistic Optimization

1: **Inputs** - Cost Budget: Λ, Sub-Gaussian Parameter: σ, Partitioning Structure: (h, i), Bias function: $\zeta(.)$, Cost function: $\lambda(.)$, Smoothness Parameters: (ν, ρ).
2: **Initialization** - $\mathcal{T} = \{(0, 1)\}$, $B_{1,2} = B_{2,2} = \infty$, $C = 0$ and $n = 0$.
3: **while** $C \leq \Lambda$ **do**
4: $\quad (h, i) \leftarrow (0, 1)$.
5: $\quad P \leftarrow \{(h, i)\}$.
6: $\quad$ **while** $(h, i) \in \mathcal{T}$ **do**
7: $\quad\quad$ Update (h, i) to $(h+1, 2i-1)$ if $B_{h,2i-1} > B_{h,2i}$ or to $(h+1, 2i)$ if $B_{h,2i-1} < B_{h,2i}$.
8: $\quad\quad$ Ties are broken at random. $P \leftarrow P \cup \{(h, i)\}$.
9: $\quad$ **end while**
10: $\quad (H, I) \leftarrow (h, i)$. Query $x_{H,I}$ at fidelity z_H and receive value Y. $\mathcal{T} \leftarrow \mathcal{T} \cup \{(H, I)\}$.
11: $\quad$ Let $n = n + 1$ and let $x_n \triangleq x_{H,I}$. Update $C = C + \lambda(z_h)$.
12: $\quad$ **for** all $(h, i) \in P$ **do**
13: $\quad\quad T_{h,i} \leftarrow T_{h,i} + 1$
14: $\quad\quad \hat{\mu}_{h,i} \leftarrow (1 - 1/T_{h,i})\hat{\mu}_{h,i} + Y/T_{h,i}$.
15: $\quad$ **end for**
16: $\quad$ **for** all $(h, i) \in P$ **do**
17: $\quad\quad U_{h,i} \leftarrow \hat{\mu}_{h,i} + \sqrt{2\sigma^2 \log n / T_{h,i}} + \nu\rho^h + \zeta(z_h)$.
18: $\quad$ **end for**
19: $\quad B_{H+1,2I-1} = B_{H+1,2I} = \infty$
20: $\quad$ Starting from the leaves down to the root maintain: $B_{h,i} \leftarrow \min\{U_{h,i}, \max\{B_{h+1,2i-1}, B_{h+1,2i}\}\}$.
21: **end while**
22: Return a point among $x_1, x_2..., x_n$ chosen uniformly at random.

---

ated. In our implementation we return a point $\hat{x}_\Lambda \in \{x_1, ..., x_n\}$ such that $\hat{x}_\Lambda = \arg\max_{x_i} f_{z_i}(x_i) - \zeta(z_i) + \epsilon_i$. Note that $f_{z_i}(x_i) - \zeta(z_i)$ is a lower bound on the value $f(x_i)$. We return a point that approximately maximizes this lower-bound.

When (ν^*, ρ^*) are not known: Grill. et al. [11] have recently developed a technique for searching for the optimal smoothness parameters for HOO [3]. The technique can be extended to our algorithm MFHOO in the multi-fidelity setup. This leads us to our second algorithm MFPOO (Algorithm 2).

Algorithm 2 MFPOO: Multi-Fidelity Parallel Optimistic Optimization

1: **Arguments:** (ν_{max}, ρ_{max}), $\zeta(z)$, $\lambda(z)$, Λ, σ
2: Let $N = (1/2)D_{max} \log(\Lambda/\log(\Lambda))$ where $D_{max} = \log 2 / \log(1/\rho_{max})$
3: **for** $i = 1$ to N **do**
4: $\quad$ Spawn MFHOO with parameters $(\nu_{max}, \rho_i = \rho_{max}^{N/(N-i-1)})$ with budget $(\Lambda - N\lambda(1))/N$
5: **end for**
6: Let $\hat{x}_{\Lambda,i}$ be the point returned by the ith MFHOO instance for $i \in \{0, .., N-1\}$. Evaluate all $\{\hat{x}_{\Lambda,i}\}_i$ at $z = 1$. Return the point $\hat{x}_\Lambda = \hat{x}_{\Lambda,i^*}$ where $i^* = \arg\max_i f(x_{\Lambda,i}) + \epsilon_i$.

The key idea of the algorithm is to spawn several MFHOO instances with different smoothness parameters $\rho_1, ..., \rho_N$. The sequence $\rho_1, ..., \rho_N$ is chosen carefully according to the strategy introduced in [11]. The budget is uniformly allocated in between all the MFHOO instances spawned. The i-th MFHOO instance is spawned with the parameters $(\nu_{max}, \rho_i = \rho_{max}^{N/(N-i-1)})$. It is only required that $\rho_{max} \geq \rho^*$ and $\nu_{max} \geq \nu^*$. In Theorem 2 we show that at least one of the MFHOO instances spawned by MFPOO has a simple regret guarantee of MFHOO run at the optimal parameters (ν^*, ρ^*) but with a budget $(\Lambda - N\lambda(1))/N$. We provide more details and intution about this algorithm in Appendix A.

5 Theoretical Results

In this section we provide our main theoretical results: Simple regret bounds for MFHOO (Algorithm 1) and MFPOO (Algorithm 2). First we present Theorem 1, which provides a simple regret bound for Algorithm 1.

Theorem 1. *If Algorithm 1 is run with parameters (ν, ρ) that satisfy Assumption 1 and given a total cost budget Λ, then the simple regret is bounded as follows,*

$$S(\Lambda) = \mathcal{O}\left(C(\nu, \rho)^{\frac{1}{d(\nu,\rho)+2}} n(\Lambda)^{-\frac{1}{d(\nu,\rho)+2}} \times (\log n(\Lambda))^{1/(d(\nu,\rho)+2)}\right),$$

where $n(\Lambda) = \max\{n : \sum_{h=1}^{n} \lambda(z_h) \leq \Lambda\}$. Here, $z_h = \zeta^{-1}(\nu\rho^h)$.

Comparison with HOO [3]: The simple regret bound that is attained by HOO [3] (operating at the highest fidelity) given the same cost budget Λ is $S'(\Lambda) = \mathcal{O}((\Lambda/\lambda(1))^{-1/(d(\nu,\rho)+2)}(\log(\Lambda/\lambda(1)))^{1/(d(\nu,\rho)+2)})$. It is easy to verify that $S(\Lambda) < S'(\Lambda)$, as $\lambda(z_h) \leq \lambda(1)$ for all z_h. In many real-world situations like hyperparameter tuning the regret of MFHOO can be much less as compared to HOO operating at the highest fidelity. In fact the real gain in MFHOO is observed in situations where evaluating at the highest fidelity is extremely expensive and Λ is of the order of $\lambda(1)$. We will now provide a corollary that highlights this, which is motivated by the following *illustrative example*. The setting below and analogous corollaries for the *noiseless case* is available in [34].

Illustrative Example: Let us consider our hyperparameter tuning example again, however let us use the fidelity range to model the number of iterations of an iterative learning algorithm. For concreteness, we will assume that the learning iterations are gradient descent steps on a smooth strongly convex objective. Let $z = 1$ represent training to completion which

might take N iterations or descent steps. Cheaper fidelities correspond to training for fewer iterations and validating, for instance $z_n < 1$ corresponds to training till $n < N$ iterations and is $O(n/N)$. The cost is linearly proportional to the fidelity, while the error of gradient descent at fidelity z_n is $O(r^n)$ for some $r \in (0, 1)$. Thus, if $\zeta(z_n)$ scales as $\nu_* \rho_*^h$, then n scales as $O(h0)$. It then follows that $\lambda(z_n) = O(h)$. It should also be noted that in the context of optimizing deep networks, where training till completion can take many hours, the total cost budget Λ is usually a small multiple of $\lambda(1)$ (evaluation cost at the highest fidelity). This motivates the following condition and the corollary that follows.

Condition 1. *$\zeta(.)$ and $\lambda(.)$ are such that $\lambda(z_h^*) \leq \min\{\beta h, \lambda(1)\}$ for some constant $\beta > 0$. Here, $z_h^* = \zeta^{-1}(\nu_* \rho_*^h)$. Further, we assume that $\Lambda \leq \lambda(1)^{1+\epsilon}$ for some $\epsilon \in (0, 1)$. Here, β is a universal constant which is much less than $\lambda(1)$.*

Under Condition 1, we get the following corollary of Theorem 1.

Corollary 1. *If Algorithm 1 is run with parameters (ν, ρ) that satisfy Assumption 1 such that $\nu > \nu^*$ and $\rho > \rho^*$ with a total cost budget Λ, then the simple regret is bounded under Condition 1 as,*

$$S(\Lambda) = \mathcal{O}\left(C(\nu, \rho)^{\frac{1}{d(\nu,\rho)+2}} n_g(\Lambda)^{-\frac{1}{d(\nu,\rho)+2}} \times \right.$$
$$\left. (\log n_g(\Lambda))^{1/(d(\nu,\rho)+2)} \right),$$

where $n_g(\Lambda) \geq \sqrt{2(\Lambda - \lambda(1))/\beta}$.

Thus, Corollary 1 implies that under Condition 1 the simple regret of MFHOO scales as $S(\Lambda) = \mathcal{O}((\log \Lambda / \sqrt{\Lambda})^{1/(d(\nu,\rho)+2)})$. On the other hand, HOO [3] would only be able to evaluate $\Lambda/\lambda(1)$ points. Thus, the simple regret of HOO would scale as $S'(\Lambda) = \mathcal{O}\left((\log \Lambda / \Lambda^{\epsilon/(1+\epsilon)})^{1/(d(\nu,\rho)+2)}\right)$, as $\Lambda \leq \lambda(1)^{1+\epsilon}$. Thus, in this setting $S(\Lambda)$ can be order-wise less than $S(\Lambda')$, as $\epsilon < 1$.

Our next result in Theorem 2 shows that at least one of the MFHOO instances spawned by Algorithm 2 has a simple regret close to that of an MFHOO run with the parameters (ν^*, ρ^*). Thus, MFPOO (Algorithm 2) can recover the performance of MFHOO run with the optimal parameters when supplied with just an upper bound on ν^* and ρ^* respectively.

Theorem 2. *If Algorithm 2 is run with parameters $(\nu_{max}(\geq \nu^*), \rho_{max}(\geq \rho^*))$ and given a total cost budget Λ, then the simple regret of at least one of the MFHOO instances spawned by Algorithm 2 is bounded*

as follows,

$$S(\Lambda) = \mathcal{O}\left((\nu_{max}/\nu^*)^{D_{max}} C(\nu^*, \rho^*)^{\frac{1}{d(\nu^*,\rho^*)+2}} \times \right.$$
$$\left. \left(\frac{\log n(\Lambda/\log\Lambda)}{n(\Lambda/\log\Lambda)}\right)^{\frac{1}{2+d(\nu^*,\rho^*)}} \right). \tag{2}$$

The simple regret bound in Theorem 2 should be compared to that of Theorem 1 when Algorithm 1 is run with the best parameters (ν^*, ρ^*). The expression is Theorem 2 is order-wise same as the simple regret achieved by MFHOO run at the optimal parameters (ν^*, ρ^*) but with a budget of $\Lambda/\log\Lambda$. This is a minor loss in terms of simple regret and is achieved without exact knowledge of the optimal parameters. For instance, under Condition 1, the simple regret of MFPOO is only a factor of $O((\log \Lambda)^{1/(2+d(\nu^*,\rho^*))})$ away from that of MFHOO run with parameters (ν^*, ρ^*). Note that there are differences between the style of results in [11] and Theorem 2 (more details in Appendix A).

6 Empirical Results

In this section we empirically validate the performance of our algorithms as compared to other benchmark algorithms for the multi-fidelity black-box optimization setting on real and synthetic data-sets. We first compare the algorithms on popular synthetic benchmark functions commonly used in the black-box optimization literature. We also empirically validate the performance of MFPOO against other algorithms for real-world use cases of hyper-parameter tuning. The algorithms under contention are: (i) BOCA [19] which is a multi-fidelity Gaussian Process (GP) based algorithm that can handle continuous fidelity spaces, (ii) MF-GP-UCB [18] which is a GP based multi-fidelity method that can handle finite fidelities, (iii) GP-EI criterion in bayesian optimization [15], (iv) MF-SKO, the multi-fidelity sequential kriging optimisation method [13], (v) GP-UCB [38] and (vi) MFPOO (Algorithm 2) and (vii) POO [11].

In our implementation of MFPOO, we *do not* assume access to a known bias function. In all our experiments it is assumed that the bias function has a parametric form $\zeta(z) = c(1 - z)$. The parameter c can be initialized and then updated online owing to the fact that different MFHOO instances spawned by MFPOO query the same node at different fidelities. In all our experiments we set $\rho_{max} = 0.95$. We provide more implementation details in Appendix D.1, in the interest of space. All experiments were performed on a 32-core Intel(R) Xeon(R) @ 2.60GHz machine, with a Nvidia 1080 Ti GPU.
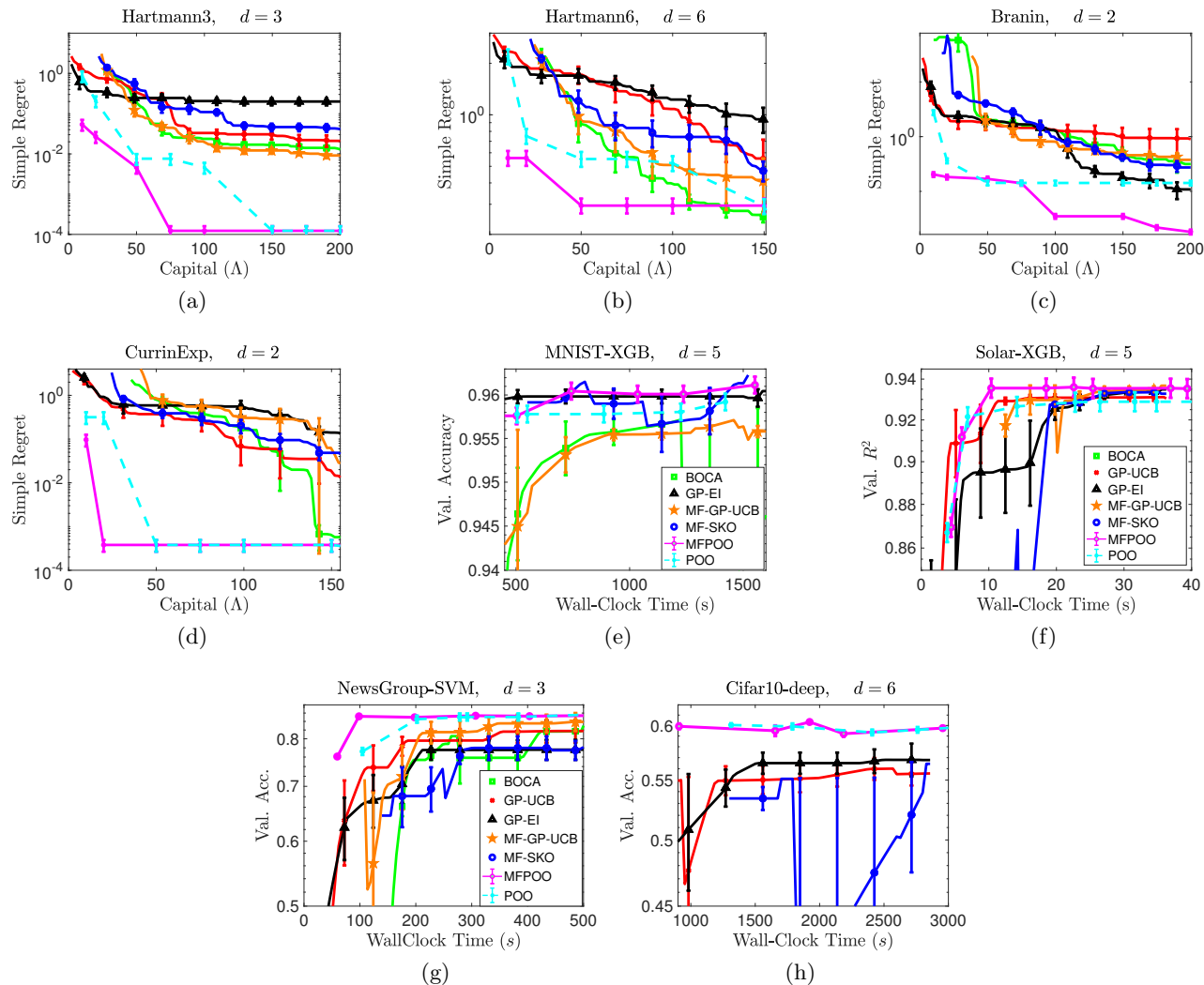
Figure 1: The *common legend* for all the plots is presented in Fig. 1f, in the interest of space and clarity. Figures (*a*) to (*d*) consists of experiments on multi-fidelity versions of synthetic functions. The experiments are averaged over 10 runs and the corresponding confidence bars are plotted. Figure (*e*) shows the 5-fold cross-validation accuracy achieved vs. wall-clock time, for tuning XGB on the MNIST data-set. GP-UCB is omitted in this figure due to poor performance. Figure (*f*) shows the 5-fold cross-validation R-square achieved vs. wall-clock time, for tuning XGB on the Solar-Radiation regression data-set. BOCA is omitted in this figure due to poor performance. Figure (*g*) shows the performance of the algorithms for tuning SVM on the 20-News Group dataset. Figure (*h*) shows the comparison of various algorithms for tuning the hyper-parameters of a ConvNet on the Cifar-10 data-set. The code base provided for BOCA and MF-GP-UCB failed to converge for this data-set. All the experiments are averaged over 5 runs.

An implementation of our algorithm can be found at https://github.com/rajatsen91/MFTreeSearchCV.git.

Synthetic Experiments: We now provide empirical results on commonly used synthetic benchmark functions. The multi-fidelity setup is introduced into the benchmark functions following the methodology in [19]. The exact details of the functions at different fidelities are provided in Appendix D.2. Note that the bias function is *not* assumed to be known however the cost function is known. We add Gaussian noise in the function evaluations at different variances σ^2 as specified in Appendix D.2. The performance of the algorithms on 4 different benchmark functions are

shown in Fig. 1 (a) - (d). The functions used are Hartmann3, Hartmann6, Branin [8] and CurinExp [6]. At the top of each sub-figure, we mention the function name and the dimension of the domain (*d*). We can observe that the tree search based methods (MFPOO and POO) outperform the other benchmarks. Among the two, MFPOO performs better that POO, because it can effectively use multiple fidelities.

XGB on MNIST: As our second experiment, we consider the task of tuning XGBOOST [5] on the MNIST data-set [27]. We consider a a subset consisting of 20000 images. The black-box function being evaluated is the 5-Fold cross-validation

accuracy at the the highest fidelity $z = 1$, which refers to using the whole data-set. The fidelity range $\mathcal{Z} = [0, 1]$ is mapped to $[500, 20000]$, that is using a fidelity $z \in [0, 1]$ implies using a randomly sub-sampled data-set consisting of $\lfloor z*(15000) + 500 \rfloor$ samples in order to measure the cross-validation error. The hyper-parameters being tuned and the respective ranges are: max_depth: [2,13], $colsample\_bytree$: [0.2,0.9], $n\_estimators$: [10,400], $gamma$: [0,0.7], $learning\_rate$: [0.05,0.3]. We plot the cross-validation accuracy achieved by different methods as a function of time in Fig. 1e. MFPOO outperforms the other algorithms in terms of validation accuracy achieved. GP-EI is also promising on this data-set. The final cross-validation accuracy achieved by MFPOO and GP-EI are 0.9611 and 0.9597 respectively. Note that in this experiment a single experiment at the highest fidelity takes approximately 200 seconds. The results are averaged over 5 experiments. σ in our algorithm is set to 0.05.

XGB on Solar Data: We test the algorithms on a regression problem that involved predicting the level of solar radiation given several weather indicators [37]. The data-set has 32684 samples. The fidelities are mapped to the range $[700, 32684]$ similar to the MNIST example above. The hyper-parameters and their ranges are also identical to the experiment above. The function value at the highest fidelity is the 5-Fold cross-validation R-square on the whole data-set. The performances of the algorithms are plotted in Fig. 1f. It can be observed that MFPOO outperforms the other algorithms especially in the lower-time horizons. Note that a single experiment at the highest fidelity for this data-set takes 2 seconds. All experiments were performed on the same machine. The results are averaged over 5 experiments. More details are in Appendix D.1.

SVM on 20 News Group: In Fig. 1g, we test the algorithms for tuning hyper-parameters of scikit-learn's SVM classifier module on the News Group dataset [26]. The hyper-parameters to being tuned are: (i) the regularization penalty in the range [1e-5,1e5] (accessed in the log. scale), (ii) the kernel temperature (γ) also in the range [1e-5,1e5] and (iii) kernel type between {'rbf','poly'}. We use a subset of 7000 samples for training i.e $z = 1$ corresponds to using all 7000 samples and $z = 0$ corresponds to a randomly chosen subset of size 100. The black-box function corresponds to the 5-fold cross-validation accuracy at the chosen fidelity. We can observe that MFPOO outperforms the other algorithms especially in lower budget settings. One evaluation at the highest fidelity takes 40.

ConvNet on Cifar-10: In Fig. 1h, we employ the algorithms for tuning the hyper-parameters of a deep convolutional network for classifying the cifar-10 [23] dataset. As the training set we use a subset of $50k$ samples from the original training data. The black-box function is the accuracy on a fixed validation set (randomly chosen half of the official test set) after 30-epochs. Note that we want to test the relative accuracy obtained by each of the tuning algorithms and therefore in the interest of time we set maximum number of epochs to be 30, even though higher accuracy can be obtained by training for more epochs. We use the AlexNet [24] architecture. The hyper-parameters being tuned are: (i) number of output channels in first conv. layer in the range [32,128], (ii) kernel size in first layer in [5,14], (iii) number of output channels in second layer in [128,256], (iv) kernel size in second layer in [3,13], (v) learning rate of Adam optimizer in [1e-5,1e-2] accessed in log-scale and (vi) dropout probability in the last layer in the range [0,0.4]. The fidelity is the number of samples used for training where $z = 0$ corresponding to 1000 randomly chosen training samples while $z = 1$ means using $50k$ samples for training. It takes about 600 seconds for one evaluation at $z = 1$. We can see that both the tree based methods clearly outperform the other algorithms in this experiment. Note that POO does not work for a budget of 1000 seconds but MFPOO does.

7 Conclusion

We study noisy black-box optimization using *tree-like* hierarchical partitions of the parameter space, when low-cost approximations are available. We propose two algorithms, MFHOO (Algorithm 1) and MFPOO (Algorithm 2) for this problem and provide simple regret guarantees for both our algorithms. Our algorithms are empirically validated against various benchmarks showing superior performance in both simulations and in real world hyper-parameter tuning examples over a wide range of datasets and learning algorithms. We believe that this paper opens up several interesting research problems, for instance developing more adaptive algorithms that query different areas of the domain at different fidelities even at the same height of the tree. We also believe that a more nuanced analysis of the algorithm is possible leading to better simple regret guarantees.

References

[1] Alekh Agarwal, John C Duchi, Peter L Bartlett, and Clement Levrard. Oracle inequalities for computationally budgeted model selection. In *COLT*, 2011.

[2] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[3] Sébastien Bubeck, Rémi Munos, Gilles Stoltz, and Csaba Szepesvári. X-armed bandits. *Journal of Machine Learning Research*, 12(May):1655–1695, 2011.

[4] Valerii V Buldygin and Yu V Kozachenko. Sub-gaussian random variables. *Ukrainian Mathematical Journal*, 32(6):483–489, 1980.

[5] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

[6] Carla Currin. A bayesian approach to the design and analysis of computer experiments. Technical report, ORNL Oak Ridge National Laboratory (US), 1988.

[7] Mark Cutler, Thomas J. Walsh, and Jonathan P. How. Reinforcement Learning with Multi-Fidelity Simulators. In *ICRA*, 2014.

[8] L. C. W. Dixon and George Philip Szegö (eds.). *Towards global optimisation 2*, volume 2. North Holland, 1978.

[9] Daniel E Finkel. Direct optimization algorithm user guide. *Center for Research in Scientific Computation, North Carolina State University*, 2, 2003.

[10] Alexander I. J. Forrester, András Sóbester, and Andy J. Keane. Multi-fidelity optimization via surrogate modelling. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, 2007.

[11] Jean-Bastien Grill, Michal Valko, and Rémi Munos. Black-box optimization of noisy functions with unknown smoothness. In *Advances in Neural Information Processing Systems*, pages 667–675, 2015.

[12] D. Huang, T.T. Allen, W.I. Notz, and R.A. Miller. Sequential kriging optimization using multiple-fidelity evaluations. *Structural and Multidisciplinary Optimization*, 2006.

[13] Deng Huang, Theodore T Allen, William I Notz, and R Allen Miller. Sequential kriging optimization using multiple-fidelity evaluations. *Structural and Multidisciplinary Optimization*, 32(5):369–382, 2006.

[14] Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *Artificial Intelligence and Statistics*, pages 240–248, 2016.

[15] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

[16] Kirthevasan Kandasamy, Gautam Dasarathy, Junier B Oliva, Jeff Schneider, and Barnabás Póczos. Gaussian process bandit optimisation with multi-fidelity evaluations. In *Advances in Neural Information Processing Systems*, pages 992–1000, 2016.

[17] Kirthevasan Kandasamy, Gautam Dasarathy, Junier B Oliva, Jeff Schneider, and Barnabas Poczos. Multi-fidelity gaussian process bandit optimisation. *arXiv preprint arXiv:1603.06288*, 2016.

[18] Kirthevasan Kandasamy, Gautam Dasarathy, Barnabas Poczos, and Jeff Schneider. The multi-fidelity multi-armed bandit. In *Advances in Neural Information Processing Systems*, pages 1777–1785, 2016.

[19] Kirthevasan Kandasamy, Gautam Dasarathy, Jeff Schneider, and Barnabas Poczos. Multi-fidelity bayesian optimisation with continuous approximations. *arXiv preprint arXiv:1703.06240*, 2017.

[20] Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast bayesian optimization of machine learning hyperparameters on large datasets. *arXiv preprint arXiv:1605.07079*, 2016.

[21] Robert Kleinberg, Aleksandrs Slivkins, and Eli Upfal. Multi-armed bandits in metric spaces. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 681–690. ACM, 2008.

[22] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

[23] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[25] Rémi Lam, Douglas L Allaire, and Karen E Willcox. Multifidelity optimization using statistical surrogate modeling for non-hierarchical information sources. In *56th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, page 0143, 2015.

[26] Ken Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339, 1995.

[27] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[28] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *arXiv preprint arXiv:1603.06560*, 2016.

[29] R. Martinez-Cantin, N. de Freitas, A. Doucet, and J. Castellanos. Active Policy Learning for Robot Planning and Exploration under Uncertainty. In *Proceedings of Robotics: Science and Systems*, 2007.

[30] Rémi Munos. Optimistic optimization of a deterministic function without the knowledge of its smoothness. In *Advances in neural information processing systems*, pages 783–791, 2011.

[31] David Parkinson, Pia Mukherjee, and Andrew R Liddle. A Bayesian model selection analysis of WMAP3. *Physical Review*, 2006.

[32] Matthias Poloczek, Jialei Wang, and Peter I Frazier. Multi-information source optimization. *arXiv preprint arXiv:1603.00389*, 2016.

[33] A Sabharwal, H Samulowitz, and G Tesauro. Selecting near-optimal learners via incremental data allocation. In *AAAI*, 2015.

[34] Rajat Sen, Kirthevasan Kandasamy, and Sanjay Shakkottai. Multi-fidelity black-box optimization with hierarchical partitions. In *International Conference on Machine Learning*, pages 4545–4554, 2018.

[35] Xuedong Shang, Emilie Kaufmann, and Michal Valko. Adaptive black-box optimization got easier: Hct only needs local smoothness. In *European Workshop on Reinforcement Learning*, 2017.

[36] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian Optimization of Machine Learning Algorithms. In *Advances in Neural Information Processing Systems*, 2012.

[37] Solar radiation kaggle competition. https://www.kaggle.com/dronio/SolarEnergy/data. Accessed: 2018-05-13.

[38] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

[39] Michal Valko, Alexandra Carpentier, and Rémi Munos. Stochastic simultaneous optimistic optimization. In *International Conference on Machine Learning*, pages 19–27, 2013.

[40] C. Zhang and K. Chaudhuri. Active Learning from Weak and Strong Labelers. In *NIPS*, 2015.