A More on MFPOO (Algorithm 2)

In this section we will provide more insights about our second algorithm MFPOO (Algorithm 2). We would like to note that in practice if the evaluations of performed by the series of MFHOO instances are stored, then the subsequent MFHOO instances can reuse this information and save on the cost budget. Our implementation can account for this aspect and this provides a significant performance improvement in practice. In Algorithm 1 and Algorithm 2 it has been assumed that the bias function is known, which may not be true in practice. However, we can assume a simple parameteric form of the bias function and update it online. We will provide more details about this in Appendix D.1.

We would also like to highlight the following about Theorem 2, which provides simple regret guarantees for MFPOO. Note that Theorem 2 only states that one of the MFHOO instances spawned by Algorithm 2 has a simple regret bound stated in the theorem. However, we do not provide any theoretical guarantees regarding the fact that with high probability i^* selected in step 6 of Algorithm 2 is the MFHOO instance with the best performance. The analysis of POO [11] can provide such a guarantee in a non-multi-fidelity setup by keeping track of the average value of the points evaluated by each of the HOO instances spawned. This analysis cannot be extended as the points evaluated by different MFHOO instances are all at different gradations of the fidelity and thus have varying biases. However, it should be noted that in practice the point $x_{\Lambda,i}$ returned by MFHOO instance i is chosen according to the scheme in Remark 1. Therefore, the function value of the point returned is a good indicator of the overall performance of the MFHOO instance and thus it is expected that the best performing MFHOO instance is selected in step 6. This is also corroborated by the strong empirical performance of MFPOO in our real and synthetic experiments in Section 6.

B Regret Guarantees when (ν^*, ρ^*) are known

The analysis of Algorithm 1 proceeds in these steps:

(i) We first prove that $N(\Lambda)$ (the number of steps performed by the algorithm) has to be at least some quantity $n(\Lambda)$ with probability one, when MFHOO is run with a cost budget of Λ.

(ii) Given that the algorithm performs $n(\Lambda)$ evaluations we prove that the cumulative regret (see the definition in [3]) incurred by the algorithm till then is bounded by $R_{n(\Lambda)}$. Therefore, owing to step 22 of the algorithm, the simple regret $s(\Lambda)$ is bounded by $R_{n(\Lambda)}/n(\Lambda)$. The main challenge in the analysis is to show that the guarantees similar to that of HOO [3] can be achieved in the presence of fidelity biases and under the new set of assumptions.

Lemma 1 is the main result for Step-**(i)** of the analysis.

Lemma 1. *When Algorithm 1 is run with a budget of Λ, $N(\Lambda) \geq n(\Lambda) + 1$ where,*

$$n(\Lambda) = \max\{n : \sum_{h=1}^{n} \lambda(z_h) < \Lambda\}.$$

Proof. Note that the structure of MFHOO is such that at each step a child of a current leaf node is expanded and the child is queried at a fidelity z_{h+1} where h was the height of the leaf node. Also note that for all h, $z_{h+1} > z_h$, and therefore $\lambda(z_{h+1}) > \lambda(z_h)$. Suppose, n steps of MFHOO has been performed. The worst case cost in those n steps can therefore be in the case where the algorithm explores without branching, that is at time-step $t \in [n]$ a node at depth $h = t$ is queried.

In this case the cost incurred is $\sum_{h=1}^{n} \lambda(z_h)$. Therefore, in this dominating corner case the algorithm will query at least $n(\Lambda)$ times. Thus, in all other cases the algorithm is bound to perform at least $n(\Lambda)$ queries. □

The main result for step **(ii)** of the analysis if provided as Theorem 3.

Theorem 3. *If Algorithm 1 is allowed to run for n queries, then the cumulative regret R_n accumulated is bounded as,*

$$R_n = \mathcal{O}\left(n^{\frac{d(\nu,\rho)+1}{d(\nu,\rho)+2}} (\log n)^{1/(d(\nu,\rho)+2)}\right).$$

The first step in the proof is equivalent to Lemma 14 in [3], which we provide below for completeness.

Lemma 2. *Let (h, i) be a sub-optimal node. Let $0 \leq k \leq h-1$ be the largest height such that (k, i_k^*) is on the path from the root to (h, i). Then for all integers $u \geq 0$, we have,*

$$\mathbb{E}[T_{h,i}(n)] \leq u + \sum_{t=u+1}^{n} \mathbb{P}\left\{[U_{s,i_s^*}(t) \leq f^* \text{ for some } s \in \{k+1, ..., t-1\}] \text{ or } [T_{h,i}(t) > u, U_{h,i}(t) > f^*]\right\}$$

Proof. It follows directly from Lemma 14 in [3]. □

Lemma 3. *For all optimal nodes (h, i) and for all integers $n \geq 1$,*

$$\mathbb{P}\left\{U_{h,i}(n) \leq f^*\right\} \leq n^{-3}.$$

Proof. We only consider the case where $T_{h,i}(n) \geq 1$, because otherwise the lemma is true trivially. Note that by Assumption 1, we have that $f^* - f(x) \leq \nu\rho^h$ for all $x \in (h,i)$. Hence, we have the following,

$$\sum_{t=1}^{n} \left(f(X_t) + \nu\rho^h - f^*\right) \mathbb{1}_{(H_t,I_t)\in\mathcal{C}(h,i)} \geq 0. \quad (3)$$

We now have the following chain,

$$\begin{aligned}
&\mathbb{P}\left\{U_{h,i}(n) \leq f^*\right\} \\
&= \mathbb{P}\left\{\hat{\mu}_{h,i}(n) + \sqrt{\frac{2\sigma^2 \log n}{T_{h,i}(n)}} + 2\nu\rho^h \leq f^*\right\} \\
&= \mathbb{P}\left\{T_{h,i}(n)\hat{\mu}_{h,i}(n) + T_{h,i}(n)(2\nu\rho^h - f^*)\right. \\
&\left.\leq -\sqrt{2\sigma^2 T_{h,i}(n)\log n}\right\} \\
&\overset{(a)}{\leq} \mathbb{P}\left\{\sum_{t=1}^{n}\left(Y_t - f_{Z_t}(X_t)\right)\mathbb{1}_{(H_t,I_t)\in\mathcal{C}(h,i)}\right. \\
&+ \sum_{t=1}^{n}\left(f(X_t) + \nu\rho^h - f^*\right)\mathbb{1}_{(H_t,I_t)\in\mathcal{C}(h,i)} \\
&\left.\leq -\sqrt{2\sigma^2 T_{h,i}(n)\log n}\right\} \\
&\leq \mathbb{P}\left\{\sum_{t=1}^{n}\left(Y_t - f_{Z_t}(X_t)\right)\mathbb{1}_{(H_t,I_t)\in\mathcal{C}(h,i)}\right. \\
&\left.\leq -\sqrt{2\sigma^2 T_{h,i}(n)\log n}\right\}.
\end{aligned}$$

Here, step (a) follows from the fact that $\zeta(Z_t) \leq \zeta(z_h) = \nu\rho^h$, and therefore $f(X_t) - f_{Z_t}(X_t) \leq \nu\rho^h$. The last term can be bounded by n^{-3} using an union bound and Azuma-Hoeffding for martingale differences, similar to the last part of Lemma 15 in [3].

□

Lemma 4. *For all integers $t \leq n$, and for all sub-optimal nodes (h,i) such that $\Delta_{h,i} > \nu\rho^h$, and $u \geq 8\sigma^2 \log n/(\Delta_{h,i} - \nu\rho^h)^2$, we have,*

$$\mathbb{P}\left\{U_{h,i}(t) > f^*, T_{h,i}(t) \geq u\right\} \leq tn^{-4}. \quad (4)$$

Proof. Note that for these values of u we have,

$$\sqrt{\frac{2\sigma^2 \log t}{u}} + \nu\rho^h \leq \frac{\Delta_{h,i} + \nu\rho^h}{2}.$$

Therefore, we have the following chain,

$$\begin{aligned}
&\mathbb{P}\left\{U_{h,i}(t) > f^*, T_{h,i}(t) > u\right\} \\
&= \mathbb{P}\left\{\hat{\mu}_{h,i}(t) + \sqrt{\frac{2\sigma^2 \log t}{T_{h,i}(t)}} + 2\nu\rho^h > f^*_{h,i} + \Delta_{h,i}, T_{h,i}(t) > u\right\} \\
&\leq \mathbb{P}\left\{\hat{\mu}_{h,i}(t) + \nu\rho^h > f^*_{h,i} + \frac{\Delta_{h,i} - \nu\rho^h}{2}, T_{h,i}(t) > u\right\} \\
&\leq \mathbb{P}\left\{T_{h,i}(t)(\hat{\mu}_{h,i}(t) - (f^*_{h,i} - \nu\rho^h)) >\right. \\
&\left.\frac{\Delta_{h,i} - \nu\rho^h}{2} T_{h,i}(t), T_{h,i}(t) > u\right\} \\
&\leq \mathbb{P}\left\{\sum_{s=1}^{t}\left(Y_s - f_{Z_s}(X_s)\right)\mathbb{1}_{(H_s,I_s)\in\mathcal{C}(h,i)}\right. \\
&\left.> \frac{\Delta_{h,i} - \nu\rho^h}{2} T_{h,i}(t), T_{h,i}(t) > u\right\}.
\end{aligned}$$

Now, following the same techniques as in Lemma 3 (and also in Lemma-16 in [3]) it can be shown that the last term is less than tn^{-4}. □

Combining Lemma 3 and 4 we get the following result.

Lemma 5. *For all sub-optimal nodes (h,i) with $\Delta_{h,i} > \nu\rho^h$ we have,*

$$\mathbb{E}\left[T_{h,i}(n)\right] \leq \frac{8\sigma^2 \log n}{(\Delta_{h,i} - \nu\rho^h)^2} + 4.$$

Proof of Theorem 3. Let H be a fixed integer greater than 1, which is to be chosen later. Let I_h be the nodes at height h that are $2\nu\rho^h$ optimal. Let τ_h be the set of nodes at height h which are not in I_h but whose parents are in I_{h-1}. We will partition the nodes of the infinite tree into three subsets, $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2 \cup \mathcal{T}_3$. Let $\mathcal{T}_1$ be all the descendants of I_H and the nodes in I_H. Let $\mathcal{T}_2 \triangleq \cup_{0 \leq h < H} I_h$. Let $\mathcal{T}_3$ be the all descendants of $\cup_{0 \leq h < H} \tau_h$, including the nodes themselves. We define the following partitioned cumulative regret quantities,

$$R_{n,i} = \sum_{t=1}^{n}(f^* - f(X_t))\mathbb{1}_{\{(H_t,I_t)\in\mathcal{T}_i\}}, \text{ for } i = 1,2,3. \quad (5)$$

Note that we have, $R_n = \sum_{i=1}^{3} \mathbb{E}[R_{n,i}]$.

(i) Let us first bound $\mathbb{E}[R_{n,1}]$. Since, all nodes in $\mathcal{T}_1$ are $2\nu\rho^H$ optimal, therefore by Assumption 1 all points that lie in these cells are $3\nu\rho^H$ optimal. Therefore, we have that $\mathbb{E}[R_{n,1}] \leq 3\nu\rho^H n$.

(ii) All nodes that belong to I_h are $2\nu\rho^h$ optimal. Therefore, all points belonging to I_h is $3\nu\rho^h$ optimal. Also, by Definition 1, we have that $|I_h| \leq$

$C(\nu,\rho)\rho^{-d(\nu,\rho)h}$. Thus, we have the following,

$$\mathbb{E}[R_{n,2}] \leq \sum_{h=0}^{H-1} 3\nu\rho^h C(\nu,\rho)\rho^{-d(\nu,\rho)h} \leq 3\nu C(\nu,\rho)\sum_{h=0}^{H-1}\rho^{h(1-d(\nu,\rho))}.$$

(iii) All nodes in τ_h have their parents in I_{h-1}. So, all the points in these nodes are at least $2\nu\rho^{h-1}$ optimal. Therefore, we have the following chain.

$$\mathbb{E}[R_{n,3}] \leq \sum_{h=1}^{H} 3\nu\rho^{h-1}\sum_{i:(h,i)\in\tau_h}\mathbb{E}[T_{h,i}(n)] \leq \sum_{h=1}^{H} 3\nu\rho^{h-1}2C(\nu,\rho)\rho^{-d(\nu,\rho)(h-1)}\left(\frac{8\sigma^2\log n}{(\nu\rho^h)^2}+4\right)$$

Combining the above three steps we arrive at,

$$R_n \leq 3\nu\rho^H + 3\nu C(\nu,\rho)\sum_{h=0}^{H-1}\rho^{h(1-d(\nu,\rho))} + \sum_{h=1}^{H} 6\nu\rho^{h-1}C(\nu,\rho)\rho^{-d(\nu,\rho)(h-1)}\left(\frac{8\sigma^2\log n}{(\nu\rho^h)^2}+4\right) \leq \alpha_1 n\rho^H + \alpha_2 C(\nu,\rho)\rho^{-H(1+d(\nu,\rho))}\sigma^2\log n,$$

where α_1 and α_2 are universal constants.

Now, we choose H such that the two terms in the above equations are order-wise equal. This gives us the following regret bound,

$$R_n \leq \tau C(\nu,\rho)^{1/(d(\nu,\rho)+2)} n^{\frac{d(\nu,\rho)+1}{d(\nu,\rho)+2}}(\sigma^2\log n)^{1/(d(\nu,\rho)+2)}, \quad (6)$$

where τ is an universal constant. □

Now, we can combine the above results to arrive at one of our main results.

Proof of Theorem 1. Note that in Step-22 of Algorithm 1 one of the points seen so far is randomly chosen. Therefore, if the algorithm has evaluated n points so far and incurred a cumulative regret of R_n, then the simple regret so far is given by R_n/n. Now, we have the following chain,

$$S(\Lambda) \leq \mathbb{E}\left[\mathbb{E}\left[\frac{R_n}{n}\middle| N(\Lambda)=n\right]\right] \leq \mathbb{E}\left[\mathbb{E}\left[\tau C(\nu,\rho)^{1/(d(\nu,\rho)+2)} n^{-\frac{1}{d(\nu,\rho)+2}}(\sigma^2\log n)^{1/(d(\nu,\rho)+2)}\middle| N(\Lambda)=n\right]\right].$$

Note, that the expression inside the conditional expectation is decreasing with the value of n. Also, according to Lemma 1 $N(\Lambda)\geq n(\Lambda)$ almost surely. Therefore, we have

$$S(\Lambda) \leq \tau C(\nu,\rho)^{\frac{1}{d(\nu,\rho)+2}} n(\Lambda)^{-\frac{1}{d(\nu,\rho)+2}}(\sigma^2\log n(\Lambda))^{1/(d(\nu,\rho)+2)}.$$

□

C Recovering optimal scaling with unknown smoothness

In this section we will prove Theorem 2. The proof of this theorem is very similar to the analysis in [11]. We will first use a function lemma from [11] that will be key in proving Theorem 2.

Lemma 6. *Consider the parameters $\nu > \nu^*$ and $\rho > \rho^*$. Let $h_{min} \triangleq \log(\nu/\nu^*)\log(1/\rho)$. Then we have the following,*

$$\mathcal{N}_h(2\nu\rho^h) \leq \max\left(C(\nu^*,\rho^*)2^{(\log\rho^*+\log\nu^*-\log\nu)/\log\rho}, 2^{h_{min}}\right) \times \rho^{-h[d(\nu^*,\rho^*)+\log 2(1/\log(1/\rho)-1/\log(1/\rho^*))]}$$

Proof. It follows directly from the analysis of Theorem 1 in appendix B.1 of [11]. □

Lemma 6 implies the following,

$$C(\nu,\rho) \leq \max\left(C(\nu^*,\rho^*)2^{(\log\rho^*+\log(\nu^*/\nu))/\log\rho}, 2^{h_{min}}\right)$$
$$d(\nu,\rho) \leq d(\nu^*,\rho^*) + \log 2(1/\log(1/\rho) - 1/\log(1/\rho^*)) \quad (7)$$

Proof of Theorem 2. Let $S(\Lambda)_{(\nu,\rho)}$ the simple regret of an MFHOO instance run with parameters (ν,ρ) satisfying the condition in Theorem 2, given a budget of Λ. Then from the proof of Theorem 2 we have the following chain,

$$\log S(\Lambda)_{(\nu,\rho)} \leq \log\tau + \frac{\log C(\nu,\rho)}{2+d(\nu,\rho)} - \frac{\log(n(\Lambda)/(\sigma^2\log n(\Lambda)))}{2+d(\nu,\rho)} \leq \log\tau + \frac{\log C(\nu,\rho)}{2+d(\nu,\rho)} - \frac{\log(n(\Lambda)/(\sigma^2\log n(\Lambda)))}{2+d(\nu^*,\rho^*)}\left(1-\frac{d(\nu,\rho)-d(\nu^*,\rho^*)}{2+d(\nu^*,\rho^*)}\right).$$

The last inequality follows from Eq. (7). Recall that MFPOO spawns N (defined in Algorithm 2) MFHOO instances each with budget Λ/N. By Equation. (7), we have that out of the N parameters $\rho_1, ..., \rho_N$, there is at least one say $\bar{\rho}\geq\rho^*$ such that,

$$d(\nu_{max},\bar{\rho}) - d(\nu^*,\rho^*) \leq \frac{D_{max}}{N}.$$

Thus we have that,

$$\log S(\Lambda/N)_{(\nu_{max},\bar{\rho})} \leq \log \tau + \frac{\log C(\nu_{max},\bar{\rho})}{2+d(\nu,\bar{\rho})} + \log\left(\frac{\log n(\Lambda/N)}{n(\Lambda/N)}\right)\left(\frac{1}{d(\nu^*,\rho^*)} - \frac{D_{max}/N}{(2+d(\nu^*,\rho^*))^2}\right). \tag{8}$$

Using Equation 7 and following the steps in B.3 on page 12 in [11] it can be shown that,

$$\frac{\log C(\nu_{max},\bar{\rho})}{2+d(\nu,\bar{\rho})} \leq \beta + \frac{D_{max}}{2+d(\nu^*,\rho^*)}\log(\nu_{max}/\nu^*) \tag{9}$$

Finally using the fact that $N = 0.5 D_{max}\log(\Lambda/\log\Lambda)$ and that $n(\Lambda) \leq \Lambda$ we have the following:

$$-\log\left(\frac{\log n(\Lambda/N)}{n(\Lambda/N)}\right)\frac{D_{max}/N}{(2+d(\nu^*,\rho^*))^2} \leq 2. \tag{10}$$

Putting together Equation (10),(9) and (8) we have the following,

$$S(\Lambda/N)_{(\nu_{max},\bar{\rho})} \leq \tau\exp(\beta+2)\left(D_{max}(\nu_{max}/\nu^*)^{D_{max}} \log n(\Lambda/\log(\Lambda))/(n(\Lambda/\log(\Lambda)))\right)^{1/(2+d(\nu^*,\rho^*))}.$$

This proves that at least one of the MFHOO instances spawned has the regret specified in Theorem 2. □

D More on Experiments

D.1 Implementation Details

In this section, we provide the following implementation details about our algorithm:

Updating the Bias Function: As mentioned before, we assume that the bias function if of the form $\zeta(z) = c(1-z)$. The parameter c is estimated online as follows: (i) We start by choosing a random point $x \in \mathcal{X}$, which is queried at $z_1 = 0.8$ and $z_2 = 0.2$, giving observations Y_1 and Y_2. We initialize $c = 2|Y_1 - Y_2|/|z_1 - z_2|$. We also set $\nu_{max} = 2 * c$. Note that this uses up a small portion of the budget $(\lambda(0.2) + \lambda(0.8))$. The structure of MFPOO is such that while running the parallel MFPOO instances the sames cells (representative point in the cell) is queried again at different fidelities say z_1 and z_2, yielding function values Y_1 and Y_2. If at any point $|Y_1 - Y_2|/|z_1 - z_2| > c$, we update $c \leftarrow 2c$.

Saving on Parallel MFHOO's: In practice we can save significant portions of the cost budget by making use of the fact that two MFHOO instances can query the same cell at fidelities z_1 and z_2 which are very close to each other. We set of tolerance $\tau = 0.01$. If an MFHOO (spawned by MFPOO) queries a cell at a fidelity z_2, and that cell had already been queried before at z_1, then we reuse the previous evaluation if $|z_1 - z_2| < \tau$. This provides significant gains in practice.

Hierarchical Partitions: The hierarchical partitioning scheme followed is similar to that of the DIRECT algorithm [9]. Each time when a cell needs to be broken into children cells, the coordinate direction in which the cell width is maximum is selected, and the children are divided into halves in the direction of that coordinate.

Real-Data Implementations: Our tree-search implementations are python objects that can take in as input a wrapper class which converts a classification/regression problem into a black-box function object with multiple fidelities. We implement our regressors and classifiers (scikit-learn XGB) within the black-box function objects. We use a 16 Core machine, where XGBoost can be run on parallel threads. We set *nthreads* = 5 and the 5-Fold cross-validation is also performed in parallel.

D.2 Description of Synthetic Functions

We use multi-fidelity versions of commonly used benchmark functions in the black-box optimization literature. These multi-fidelity versions have been previously used in [17, 34].

Currin exponential function [6]: This is a two dimensional function with domain $[0,1]^2$. The cost function is $\lambda(z) = 0.1 + z^2$ and the noise variance is $\sigma^2 = 0.5$. The multi-fidelity object as a function of (x,z) is,

$$f_z(x) = \left(1 - 0.1(1-z)\exp\left(\frac{-1}{2x_2}\right)\right) \times \left(\frac{2300x_1^3 + 1900x_1^2 + 2092x_1 + 60}{100x_1^3 + 500x_1^2 + 4x_1 + 20}\right).$$

Hartmann functions [8]: We use two Hartmann functions in 3 and 6 dimensions. The functional form of the multi-fidelity object is $f_z(x) = \sum_{i=1}^{4}(\alpha_i - \alpha'(z))\exp\left(-\sum_{j=1}^{3} A_{ij}(x_j - P_{ij})^2\right)$ where $\alpha = [1.0, 1.2, 3.0, 3.2]$ and $\alpha'(z) = 0.1(1-z)$. In the case of 3 dimensions, the cost function is $\lambda(z) = 0.05 + (1-0.05)z^3$, $\sigma^2 = 0.01$ and,

$$A = \begin{bmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix}, \quad P = 10^{-4} \times \begin{bmatrix} 3689 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8828 \end{bmatrix}.$$

Moving to the 6 dimensional case, the cost function is

$\lambda(z) = 0.05 + (1 - 0.05)z^3$, $\sigma^2 = 0.05$ and,

$$A = \begin{bmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}, \quad P = 10^{-4} \times \begin{bmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{bmatrix}.$$

.

When $z = 1$, these functions reduce to the commonly used Hartmann benchmark functions.

Branin function [8]**:** For this function the domain is $\mathcal{X} = [[-5, 10], [0, 15]]^2$. The multi-fidelity object is given by,

$$f_z(x) = a(x_2 - b(z)x_1^2 + c(z)x_1 - r)^2 + s(1 - t(z))\cos(x_1) + s,$$

where $a = 1$, $b(z) = 5.1/(4\pi^2) - 0.01(1 - z)$ $c(z) = 5/\pi - 0.1(1 - z)$, $r = 6$, $s = 10$ and $t(z) = 1/(8\pi) + 0.05(1 - z)$. At $z = 1$, this becomes the standard Branin function. The cost function is $\lambda(z) = 0.05 + z^3$ and $\sigma^2 = 0.05$ is the noise variance.