Bayesian Optimization for Modular Black-Box Systems with Switching Costs

Chi-Heng Lin[1] **Joseph D. Miano**[2] **Eva L. Dyer**[1,3]

[1]School of Electrical and Computer Engineering, Georgia Tech, Atlanta, Georgia, USA
[2] School of Computer Science, Georgia Tech, Atlanta, Georgia, USA
[3]Department of Biomedical Engineering, Georgia Tech & Emory University, Atlanta, Georgia, USA

Abstract

Most existing black-box optimization methods assume that all variables in the system being optimized have equal cost and can change freely at each iteration. However, in many real-world systems, inputs are passed through a sequence of different operations or *modules*, making variables in earlier stages of processing more costly to update. Such structure induces a dynamic cost from *switching* variables in the early parts of a data processing pipeline. In this work, we propose a new algorithm for switch-cost-aware optimization called Lazy Modular Bayesian Optimization (LaMBO). This method efficiently identifies the global optimum while minimizing cost through a passive change of variables in early modules. The method is theoretically grounded which achieves a vanishing regret regularized with switching cost. We apply LaMBO to multiple synthetic functions and a three-stage image segmentation pipeline used in a neuroimaging task, where we obtain promising improvements over existing cost-aware Bayesian optimization algorithms. Our results demonstrate that LaMBO is an effective strategy for black-box optimization capable of minimizing switching costs.

1 INTRODUCTION

Bayesian optimization (BO) [Snoek et al., 2012, Srinivas et al., 2010, Mockus et al., 1978] is a popular technique that is used to optimize unknown black-box systems. Such systems arise in a wide range of applications ranging from robotics [Berkenkamp et al., 2016] and sensor networks [Garnett et al., 2010], to hyperparameter tuning in machine learning [Bergstra et al., 2011, Frazier, 2018]. In the black-box setting, the underlying function that maps variables to a reward (loss) is unknown and is instead queried. BO methods find ways to tackle this challenging setting by approximating the unknown function with a Gaussian process (GP) [Rasmussen, 2003] and updating this belief on the fly to decide which sample to generate next.

Unfortunately, when trying to optimize a complex black-box system, the cost of generating a sample can often be prohibitive. Here, costs could represent the amount of time, energy, or resources required to generate a black-box sample (i.e., test a new hyperparameter parameter configuration of interest). To account for costs to update different variables, or overall cost constraints, a wide range of different cost-aware and multi-resolution sampling strategies ranging from batch optimization [González et al., 2016, Kathuria et al., 2016], multi-fidelity model [Kandasamy et al., 2016, 2017], multi-objective optimization [Abdolshah et al., 2019], to dynamic programming [Lam and Willcox, 2017, Lam et al., 2016] have been developed over the past decade.

While the underlying black-box function that we want to optimize may be unknown, many real-world systems have costs with specific structure that are known ahead of time. An important yet simple abstraction of many systems encountered in practice is that they process their inputs through a sequence of *modules*, where the outputs from one module to the next are chained together. For instance, in many scientific applications like genomics [Davis-Turak et al., 2017] and neuroimaging [Abraham et al., 2014, Johnson et al., 2019], generating an output (sample) often involves running high-dimensional inputs through multiple stages (modules) of processing, and each module has unique hyperparameters that must be optimized. When making updates in these types of sequential systems, it becomes much more costly to update a variable at an earlier stage of processing because we must take into account the fact that all operations in subsequent stages must be rerun. Not only does this sequential structure affect the cost, but it also gives rise to *switching costs*, where the cost depends on which variables are modified between consecutive iterations. However, most of these methods are agnostic to additional information about the structure of the underlying costs in the system, and thus are

Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI 2021), PMLR 161:1024–1034.

too aggressive in changing variables across modules.

In light of these motivations, we introduce a new algorithm for black-box optimization called Lazy Modular Bayesian Optimization (LaMBO). This method leverages modular and sequential structure in a system to reduce overall cumulative costs during optimization. To quantify the cost of switching in these cases, we model the cost of each query as the aggregation of cost needed to rerun modules from the first step where a variable must be updated. By encouraging the optimization method to be lazy, analytically we show that LaMBO achieves a sublinear rate in a notion of switching-cost regularized regret. We also empirically evaluate the performance of the proposed method by applying LaMBO to a number of synthetic datasets and neuroimaging problem where the aim is to tune a modular pipeline for 3D reconstruction of neuroanatomical structures from slices of 2D images [Lee et al., 2019, Johnson et al., 2019]. Our empirical results show that hyperparameters in this three-stage system can be optimized to 95% optimality jointly over multiple modules within 1.4 hours compared with 5.6 hours obtained from the best of the alternatives. These results point to the fact that leveraging system structure, and dynamic switching costs, can be advantageous for optimizing multi-stage black-box systems.

Summary of Contributions. The contributions of this work are as follows: (*i*) In Section 3, we formulate a novel Bayesian optimization problem with *switching-cost* constraints, and propose the algorithm LaMBO to solve the problem in systems with modular structure. To the best of our knowledge, this is the first attempt to leverage modular system structure in the design of a cost-efficient algorithm for black-box optimization. (*ii*) In Section 4, we establish theoretical guarantees of LaMBO by proving a regularized regret bound taking switching-cost consumption into consideration using techniques from both the multi-armed bandit and Bayesian optimization literature. (*iii*) In Section 5, we apply our method to synthetic functions and to a 3D brain-image-segmentation task. We empirically demonstrate that the method can efficiently solve switch-cost-aware optimization across modular compositions of functions.

2 BACKGROUND AND RELATED WORK

2.1 BAYESIAN OPTIMIZATION

Black-box optimization methods aim to find the global minimum of an unknown function $f(x)$ with only a few queries. Let f^* and $\mathbf{x}^*$ be the optimal function value and optimizer, respectively. Standard algorithms seek to produce a sequence of inputs $\mathbf{x}^1, \dots, \mathbf{x}^T$ that result in (potentially) noisy observations $\mathbf{y}^1, \dots, \mathbf{y}^T$ such that $f(\mathbf{x}^t)$ will approach the optimal value f^* quickly. A common choice to measure performance of a candidate algorithm is the *cumulative regret*:

$$R(T) = \sum_{t=1}^{T} f(\mathbf{x}^t) - f^*. \tag{1}$$

Among the many different approaches for black-box optimization, BO is a celebrated probabilistic method whose statistical inferences are tractable and theoretically grounded. It uses a Gaussian process (GP) prior on the distribution of the unknown function f, which is characterized by a mean function $\mu(\mathbf{x})$ and a kernel function $k_0(\mathbf{x}, \mathbf{x}')$. Let $\mathbf{k}_t(\mathbf{x}) := [k_0(\mathbf{x}, \mathbf{x}^1), \dots, k_0(\mathbf{x}, \mathbf{x}^t)]^T$, $\mathbf{K}_t := [k_0(\mathbf{x}^i, \mathbf{x}^j)]_{1 \le i,j \le t}$, and σ^2 represent the noise variance. In this case, we can update the posterior with simple closed-form formulas:

$$\begin{aligned} \mu_{t+1}(\mathbf{x}) &= \mathbf{k}_t^T(\mathbf{x})(\mathbf{K}_t + \sigma\mathbf{I})^{-1}\mathbf{y}_t, \\ \sigma^2_{t+1}(\mathbf{x}) &= k_0(\mathbf{x}, \mathbf{x}) - \mathbf{k}_t^T(\mathbf{x})(\mathbf{K}_t + \sigma\mathbf{I})^{-1}\mathbf{k}_t(\mathbf{x}). \end{aligned} \tag{2}$$

Common classes of selection algorithms that use a BO framework include the: Upper Confidence Bound (UCB) [Srinivas et al., 2010], Expected-Improvement (EI) [Mockus, 1982], and entropy search [Wang and Jegelka, 2017] algorithms. At the heart of all of these methods is the design of an acquisition function that is used to select the next evaluation point, i.e., $\mathbf{x}^t \in \arg\min_{\mathbf{x}} \alpha^t(\mathbf{x})$. The acquisition function allows flexibility in trading-off between exploration and exploitation and are constructed using the posterior statistics. In this paper, we will adopt the UCB acquisition function due to its simplicity and success in both theory and practice. The GP-UCB acquisition function is given by

$$\alpha^t_{UCB}(\mathbf{x}) = \mu_{t-1}(\mathbf{x}) - \beta_t \sigma_{t-1}(\mathbf{x}), \tag{3}$$

where β_t is a design parameter that controls the amount of exploration in the algorithm.

2.2 SLOWLY MOVING BANDIT ALGORITHM

To incorporate switching costs into a BO sampling strategy, we adopt [Koren et al., 2017b] on solving a multi-armed bandit problem with switching costs. In this setting, optimization is formulated into a arm-selection problem where optimal variables i (arms) are selected from a set $\mathcal{K}$ to minimize an unknown loss function $\ell : \mathcal{K} \mapsto \mathbb{R}$. At each iteration t, we can query an oracle to measure the loss (inverse reward) $\ell(i^t)$ by pulling arm i^t. In the switch-cost-aware case, there is a cost metric c which incurs cost $c(i^t, i^{t-1})$ when switching between arms from $t-1$ to t. The objective is to minimize a linear combination of the loss and switching cost. In [Koren et al., 2017b], the authors propose the slowly moving bandit algorithm (SMB) to tackle the problem with a general cost metric. Here, we extend the idea to the setting of black-box optimization.

SMB is based on a multiplicative update strategy [Auer et al., 2002] that encodes the cost of switching between arms in a

tree; each arm is a leaf and the cost to switch from one arm to another is encoded in the distance from their corresponding leafs in the tree. At each iteration t, SMB chooses an arm according to a probability distribution p_t conditioned on the level of the tree (the root is level 0) selected at the last iteration. We will make the sampling distribution precise momentarily. The distribution is then updated with a standard multiplicative update rule $p_t \leftarrow p_t \exp(-\eta \tilde{\ell}_t)$, where η is the learning rate and $\tilde{\ell}_t$ is the estimated loss. Compared with basic bandit algorithms, there are two key modifications in SMB. First, it uses conditional sampling to encourage slow switching. This constrains the arm selection to be the close to the previous choice, where distance is embedded in the tree's structure. Formally, an arm is drawn according to the following conditional distribution $p(\cdot|A_{h_{t-1}}(i^{t-1}))$, where h_{t-1} is a random level chosen at previous iteration, and $A_h(i)$ denotes the leaves (arms) that belong to the subtree rooted at level h which has i as one of its leaves. This ensures that i^t remains in some small subtree as in the previous iteration. Second, to utilize the classic multiplicative method, SMB makes sure that in average the conditional sampling is equivalent to direct sampling by modifying the loss estimators $\tilde{\ell}_t$ as,

$$\tilde{\ell}_t = \bar{\ell}_{t,0} + \sum_{h=0}^{H-1} \sigma_{t,h} \bar{\ell}_{t,h}, \tag{4}$$

$$\bar{\ell}_{t,h}(i) = \log \left(\sum_{j \in A_h(i)} \frac{p_t(j) e^{-\eta(1+\sigma_{t,h-1})\bar{\ell}_{t,h-1}(j)}}{p_t(A_h(i))} \right)^{-\frac{1}{\eta}}, \tag{5}$$

where $\bar{\ell}_{t,0}$ is an unmodified loss estimator for algorithms without switching cost, and $\{\sigma_{t,k}\}_k$ are i.i.d. uniform random variables in $\{-1, 1\}$. For the purpose of self-contained, we include the pseudo-code of SMB in Supp. D.

2.3 RELATED WORK

The closest framework to ours in Bayesian optimization is the cost-aware Bayesian optimization, where instead of trying to minimize a function using the fewest samples, the methods strive to find the optimizer with least cumulative cost. The most standard method [Snoek et al., 2012, Lee et al., 2020] measures the acquisition function in the unit of the cost $\alpha^t(\mathbf{x})/c(\mathbf{x})^\gamma$, where c denotes the cost function and γ is some trade-off parameter. Another approach is to impose explicit cumulative budget constraints [Lam et al., 2016, Lam and Willcox, 2017], where the authors have used dynamic programming-based approaches. While many of these algorithms have proposed cost-efficient optimization strategies under static costs, the scenario with the *switching* cost where deviating from a previous action induces larger costs, has not been well-understood in the literature.

Multi-fidelity strategies [Kandasamy et al., 2017, Poloczek et al., 2017, Wu et al., 2020, McLeod et al., 2017] are also popular choices in which the decision maker is allowed to choose an additional fidelity parameter that controls the accuracy and the cost for function evaluations. In the sense that across subsequent resolutions there are correlations or structure in the costs of different parameters. On the surface, it seems our problem can be easily cast under the framework. However, as the dependencies on the accuracy of function approximation to variables in different modules are non-separable, one can not map a module to a fidelity. Another cost efficient BO approach similar to our work is process constrained BO [Vellanki et al., 2017], where some variables are not allowed to change due to constraints from physical system. CA-MOBO [Abdolshah et al., 2019] is also a cost-aware strategy which uses the framework of multi-objective optimization. It generalizes the UCB method to multi-dimensional outputs seeking a sweet spot in the trade-off between cost consumption and optimization accuracy. Our work differs from theirs as we are allowed to probe variables anytime but may incur different cost when changing sets of variables in different modules, and we consider costs arising from switching variables.

The switching cost optimization has been studied in multi-armed bandit literature [Kalai and Vempala, 2005, Koren et al., 2017a,b, Dekel et al., 2014, Feldman et al., 2016]. However, the arms are assumed to be uncorrelated, while in our work we assume strong dependency and leverage it by using Gaussian surrogate to explore multi-arms simultaneously.

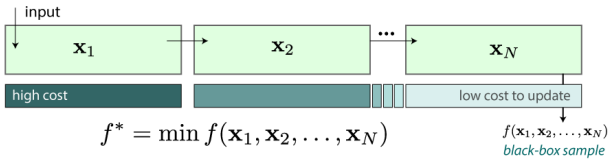

Figure 1: *Example of a modular system that consists of a sequence of operations that are applied, each with their own distinct set of variables*. When variables in early stages are changed, all the remaining modules need to run and this incurs high costs.

3 LAZY MODULAR BAYESIAN OPTIMIZATION (LAMBO)

A key assumption underlying this work is that the black-box system of interest has a modular structure, where the overarching system can be decomposed into a sequence of different sets of operations, each with a distinct set of variables that need to be optimized (Figure 1).

3.1 PROBLEM SETUP

Let $\mathbf{x}_m \in \mathcal{X}_m$ denote the variables in the m^{th} module, and let $\mathbf{x} \in \mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_N$ denote the set of variables across all modules. Our main goal is to propose a cost-efficient algorithm that finds the optimizer for a black-box function,

$$\mathbf{x}^* \in \arg\min_{\mathbf{x}\in\mathcal{X}} f(x).$$

The function f is unknown to us, but when a set of variables $\mathbf{x}$ are input into the system, this generates a noisy output $\mathbf{y} = f(\mathbf{x}) + \epsilon$, where ϵ is σ-sub-Gaussian. To ensure that our model of the cost reflects the modular structure of the system, we make the following assumptions: (i) running the m^{th} module incurs c_m cost , $\forall m = 1, \ldots, N$, (ii) a module needs to be run only if variables in some modules earlier than it in the pipeline has been changed from previous iteration. We will also assume that c_N, as any update requires updating variables in the last module, is negligible and equal to 0. Under the above modeling assumptions, the total cost incurred at iteration t is equal to,

$$\Gamma^t := \sum_{m=1}^{N-1} c_m \mathbb{1}_{\{\mathbf{x}^t_{1:m} \neq \mathbf{x}^{t-1}_{1:m}\}}, \tag{6}$$

where $\mathbb{1}_{\{\mathbf{x}^t_{1:m} \neq \mathbf{x}^{t-1}_{1:m}\}}$ is an indicator that equals to 1 when any variable in modules before the m^{th} module have been changed from the previous iteration. We refer to the quantity Γ^t as the *movement cost*. In our image analysis pipeline experiment (Section 5.2), costs can be thought of as the amount of time or the amount of compute required to re-run a specific module and all of the subsequent modules that follow. In this case, our goal is to perform an end-to-end optimization on the system to maximize the accuracy on a validation set, which can be measured with an f1-score or some other measure of the accuracy of the segmented output.

To trade-off between cost efficiency and functional optimality, we define the *movement regret* as,

$$R^+(T, \lambda) = \sum_{t=1}^{T} f(\mathbf{x}^t) - f^* + \lambda\Gamma^t. \tag{7}$$

Γ^t serves as a regularizer which is added to the standard definition of the cumulative regret. In general, the function value and the cost are measured in different units, so λ should depend on the scales of data.

3.2 ALGORITHM

This section we provide the descriptions of the 3 steps in the proposed algorithm 1 named Lazy Modular Bayesian Optimization (LaMBO).

Step 1) Modular Structure Embedding Phase. To visualize our algorithmic approach, we point the reader to Figure 2. In the first stage of our optimization procedure, we need to encode the switching costs associated with the system of interest. To do this, we take inspiration from the SMB algorithm described in Section 2.2 to encode the cost to switch variables using a tree-based approach (Figure 2B). We start by linking each arm with a region (subset) of variable space. The regions are flexible and can be partitioned in different ways, but should reflect the modular structure in the system. Thus, we choose to partition the variable space of each module separately. Specifically, $\mathcal{P}_m = \{\mathcal{C}_{m_1}, \mathcal{C}_{m_2}, \ldots \mathcal{C}_{m_l}\}$ defines a partition for the m^{th} module, where $\mathcal{X}_m = \cup_n \mathcal{C}_{m_n}$. We require these sets to be disjoint $\mathcal{C}_{mn_1} \cap \mathcal{C}_{mn_2} = \emptyset$ for $n_1 \neq n_2$. Thus, when selecting an arm, we select a joint region of the first $N-1$ modules[1], i.e., $i \equiv (\mathcal{Z}_1, \ldots, \mathcal{Z}_{N-1}) \in \mathcal{K} := \mathcal{P}_1 \times \cdots \times \mathcal{P}_{N-1}$.

Next, we represent the arms in a tree $\mathcal{T}$ to encode the cost of switching between any two variable subsets. Intuitively, we want to build a tree that encodes the cost of switching between any two sets of hyperparameters (arms) in terms of the shortest path between these two leaves in the tree. Specifically, in Line 2 of Algorithm 1, we call a subroutine ConstructMSET which returns a tree $\mathcal{T}$ (modular structure embedding tree, MSET), given a partitioning of the variables across all modules and depth parameters d_m, where d_m is the depth of the m$^{\text{th}}$ module. The partition and modular specification define the leaves of the tree and the depth parameters control the probability of switching, with higher depth in a module corresponding to lower switching probability (more laziness). In our example (Figure 2B) , the tree consists of two parts (colored with blue and red) divided by the first forks, the upper portion corresponds to the partition of the first module, while the lower portion corresponds to the partition of the second module. In this case, the depth in the second module is set to 3 to reflect higher relative costs between the two modules and encourage lazy switching behavior.

Step 2) Optimization Phase. Now the remaining task is to devise a strategy for arm selection and estimate the local optimum within its corresponding variable subset. We propose to use SMB for region (arm) selection, and then use a BO strategy to search within the selected region (Line $5-6$). The parameters of SMB and BO are updated at each iteration (Line $7-11$). Unfortunately, direct application of BO changes all variables across each iteration, which typically incurs maximum cost. Hence, we propose an alternative *lazy* strategy: when the same variable subset is selected in an early module, we will use the results from the previous iteration rather than updating the outputs from this lazy module. This means that we do not need to rerun the

[1]We exclude the last module from partitioning procedure since the cost of changing parameters in the last module is the minimum cost per iteration, and can be changed freely at each iteration.

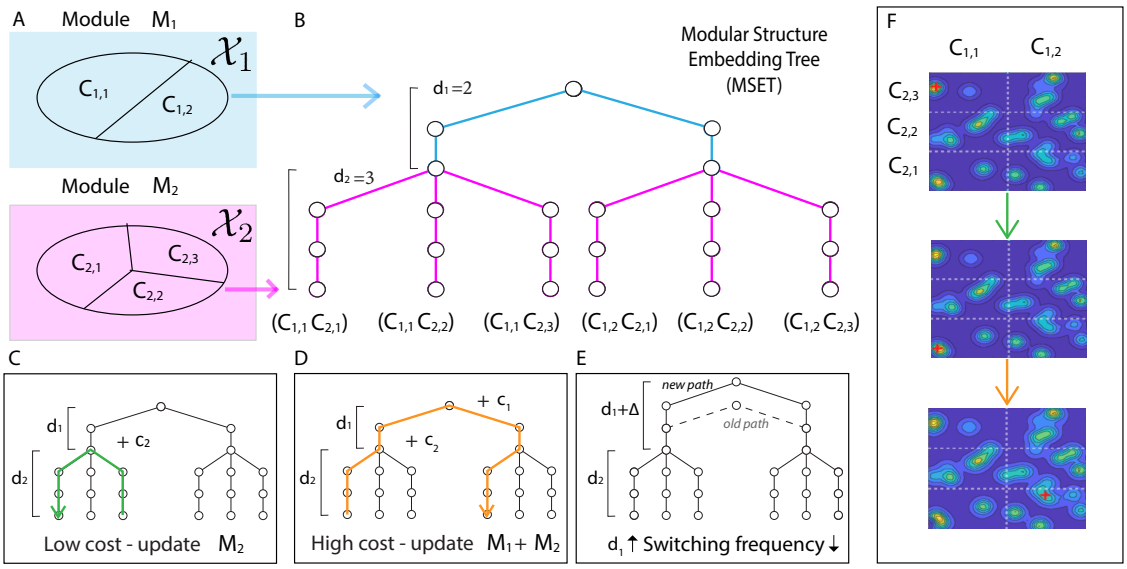

Figure 2: *Overview of our approach*. Illustration of the optimization in a modular system. In (A), we show a partition of variable spaces into regions and its corresponding MSET (B), constructed based on the partition and modular costs. An illustration of how changing regions incurs different costs (C-D), where in each case we trace the path between different arms. Changing the depth parameter $d_2 \leftarrow d_2 + \Delta$ produces a longer distance between any two arms and gives less incentive for arm changes (E). In (F), the landscapes of the BO update within regions at three consecutive iterations, corresponding to the arm changes in (C) and (D), respectively

module and thus can minimize the overall cost. Specifically, let i_t be the arm we've selected and $(\mathcal{Z}_1, \ldots, \mathcal{Z}_{N-1})$ be its associated variable region. We propose to search for a block-wise update $\mathbf{x}^t = [\mathbf{x}^{t-1}_{1:m-1}, \mathbf{u}]$ that minimizes the loss as follows:

$$\bar{\ell}_{t,0}(i_t) := \min_{\mathbf{u}\in\mathcal{U}} \alpha_t([\mathbf{x}^{t-1}_{1:m-1}, \mathbf{u}]), \;\; \mathcal{U} = (\prod_{l=m}^{N-1} \mathcal{Z}_l) \times \mathcal{X}_N, \tag{8}$$

where m is the first module that has a variable region that differs from the previous iteration $m := \min\{n : \mathcal{Z}_n \neq \mathcal{Z}^{t-1}_n\}$, and $\alpha_t(\cdot)$ is a BO acquisition function.

3.3 EMPIRICAL CONSTRUCTION OF MSET

A crucial part of algorithm is the design of the subroutine `ConstructMSET`, which involves partitioning the variables in each module, and setting the depth parameters (d_i's). From our experiments, we observe that simple bisection aligned with coordinates yields good partition on many synthetic data and on our neural data. For a MSET with $|\mathcal{K}|$ leaves with the partition, LaMBO requires solving $|\mathcal{K}|$ local BO optimization problems per iteration. Hence initially, we partition each variable space of module to two subsets only, and abandon subsets when their arm selection probability p_t is below some threshold after 10 consecutive iterations. In our experiments, we always set the threshold to be $0.1/|\mathcal{K}|$, where $|\mathcal{K}|$ denotes the number of leaves of MSET. After that, we further divide the remaining subsets again to increase the resolution. This procedure could be iterated upon further although we typically do not go beyond two stages of refinement. To avoid trapping in the local optimum, we also refresh the arm-selection probability and update the kernel hyperparameters simultaneously every 25 iterations.

In our implementation, we set the depth parameter to be $d_i = 1$ or $d_i \propto \log \lambda c_i$ when c_i could be estimated in prior. Empirically, we found that the performance is quite robust when $d_i \leq 5$ for the different cost ratios in both synthetic and real experiments we tested. To avoid accumulating cost too fast in early stages of LaMBO, we record the number of times that variable changes in the first module and dynamically increase the first depth parameters d_1 by 1 every 20 iterations when the the number has increased beyond 5 ($1/4$ of the cycle) during the period. In all experiments, we have found this simple add-on perform on par or better than fixing depth parameters through an entire run.

4 ALGORITHMIC ANALYSIS

In this section, we analyze the performance of Algorithm 1 from two perspectives: *1. Optimization accuracy* and *2.*

Algorithm 1 Lazy Modular Bayesian Optimization

Input: η, GP(μ_0, k_0), Partitions $\{\mathcal{P}_m\}_{m=1}^{N-1}$, depth parameters $\{d_m\}_{m=1}^{N-1}$.
$\mathcal{T}$ = `ConstructMSET`($\{\mathcal{P}_m\}_{m=1}^{N-1}$,$\{d_m\}_{m=1}^{N-1}$).
H = depth($\mathcal{T}$), $\mathcal{K}$ = set of leaves, p_1 = Unif($\mathcal{K}$), $h_0 = H$ and $i_0 \sim p_1$.
for $t = 1$ to T **do**
  Select arm $i_t \sim p_t(\cdot|A_{h_{t-1}}(i_{t-1}))$.
  Choose $\mathbf{x}^t$ by solving Eq. (8).
  Let $\sigma_{t,h}$, $h = 1, \ldots, H-1$, be i.i.d. Unif($\{-1, 1\}$).
  let $h_t = \min\{0 \le h \le H : \sigma_{t,h} = -1\}$ where $\sigma_{t,H} = -1$.
  Obtain loss estimators via $\tilde{\ell}_t$ = Eq. (4), Eq. (8) and

$$p_{t+1} = \frac{p_t(i)e^{-\eta\tilde{\ell}_t(i)}}{\sum_{j=1}^{|\mathcal{K}|} p_t(j)e^{-\eta\tilde{\ell}_t(j)}}, \ \forall i \in \mathcal{K}.$$

  Posterior Updates by Eq. (2).
end for

Cost efficiency. Our main result, which is stated in Theorem 1, shows that LaMBO achieves sublinear movement regret when the parameters of the input tree are set properly using the cost structure of the system.

Our results are presented in terms of *maximum information gain* defined below.

Definition 1. ***Maximum Information Gain.*** *Let $f \sim GP$ be defined in the domain $\mathcal{X}$. The observation of f at any $\mathbf{x}$ is given by the model $y = f(\mathbf{x}) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma)$. For any set $A \in \mathcal{X}$, let f_A and y_A denote the set of function values and observations at points in A, and I denote the Shannon Mutual Information. The Maximum Information Gain is defined by $\gamma_T := \max_{A \subset \mathcal{X}:|A|=T} I(y_A, f_A)$*

Analytical bounds on γ_T of common kernels are provided in Supp. A.2. To proceed with our analysis, we make the following assumption on the objective function.

Assumption 1. *The function f is L-Lipschitz, non-negative, and has a bounded norm $\|f\|_{\mathcal{H}_{k_0}} \le 1$ in the reproducing kernel Hilbert space $\mathcal{H}_{k_0}$.*

Note that our assumption is not too stringent since for any function in a Hilbert space defined above, L can be estimated by $|f(x) - f(y)| \le \|f\|_{\mathcal{H}}\|\Phi(x) - \Phi(y)\|_{\mathcal{H}}$; for instance, $L = 1/w$ for exponential kernel $k_0(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|^2/w^2)$ since $\|\Phi(x) - \Phi(y)\|_{\mathcal{H}} \le \|\mathbf{x} - \mathbf{y}\|/w$.

4.1 OPTIMIZATION ACCURACY

Our first lemma concerns the optimization capability of LaMBO under a common definition of regret, $R(T) := \sum f(x_t) - f^*$. Note that we choose to represent it in expectation instead of a probability bound for notational compactness. Conversion to one another is straightforward by common technique like Markov inequality.

Lemma 1. ***Ordinary Regret Bound.*** *Suppose the learning rate of the LaMBO is set to be $\eta = \sqrt{2^{-H}T^{-1}\log|\mathcal{K}|}$, where H is the depth of the MSET, then the expected cumulative regret of LaMBO is:*

$$\mathbb{E}[R(T)] = \mathcal{O}\left(\sqrt{2^H T \log|\mathcal{K}|}\right).$$

Remark 1. *By treating modules as arms, a natural comparison is the result of SMB in [Koren et al., 2017b] where $\mathbb{E}[R_T] = \mathcal{O}\left(\sqrt{kT\log|\mathcal{K}|}\right)$. As the arms $|\mathcal{K}|$ are represented as leaves of a binary tree with depth H, we must have $2^H \le k$, which shows that the regret bound we have is upper bounded by the result of SMB. The equality holds when the tree is complete. On the other hand, the gap between us could be potentially large. The key to this improvement is by leveraging arm correlation; by using Gaussian surrogate, each sample gives the information of not only the pulled arm itself, but also that of infinitely many others.*

4.2 ANALYSIS OF COST EFFICIENCY

Next, we analyze the cost incurred by adopting LaMBO. The following lemma shows that LaMBO is capable of accumulating sublinear cost. The result also gives an explicit recipe of choosing parameters $\{d_i\}$ of MSET from theory. Below we provide a sketch of the achievable rate and defer the detailed forms of parameters to Supp. A.1.

Lemma 2. ***Cumulative Switching Cost.*** *For sufficiently large T, there exists depth parameters $\{d_i\}$ of the MSET such that LaMBO accumulates movement cost*

$$\mathbb{E}[\sum_{t=1}^{T}\Gamma^t] = \mathcal{O}\left(\sum_{m=1}^{N-1} c_m T^{2/3}\log|\mathcal{K}|\log\frac{T^{1/3}}{\log|\mathcal{K}|}\right).$$

Remark 2. *A striking implication of Lemma 2 is that even with nonconstant cost $c_m = \Omega(1)$, the cumulative cost could still be sublinear as long as $c_m \ll o(T^{1/3})$.*

Finally combining the above two lemmas leads to our concluding theorem, which shows that a simple partition strategy, along with proper selection of the depth parameters d_i, gives sublinear movement regret defined in (7). Without additional information about how to partition each module, the simplest way to partition the space is uniformly. Hence in the analysis we adopt an uniform partition strategy characterized by r_i, where r_i denotes the Euclidean diameter of the partitioned subset $\mathcal{X}_i$.

Now we present a sketch of our main theoretical result where a proof and detailed constants could be found in Supp. A.1.

Theorem 1. ***Movement Regret Bound.*** *For $1 \leq m \leq N-1$, let D_m denote the dimension of $\mathcal{X}_m$. Suppose for all $t > 0$, $1 \leq m \leq N-1$, we set $\beta_t = \Theta(\sqrt{\gamma_{t-1} + \ln T})$, $\eta = \Theta(T^{-2/3} \sum_{m=1}^{N-1} D_m \log(LT^{1/3}/D_m \log T))$. The MSET has uniform partition of each $\mathcal{X}_m$ with diameters $r_m = \frac{D_m}{L} T^{-\frac{1}{3}} \log T$, where the depth parameters d_m are chosen according to Lemma 2, and UCB acquisition function is used. Then LaMBO achieves the expected movement regret*

$$\mathbb{E}[R^+] = \mathcal{O}((\lambda \sum_{j=1}^{N-1} c_j \sum_{m=1}^{N-1} D_m T^{\frac{2}{3}} (\log T)^2) + \gamma_T \sqrt{T}).$$

Remark 3. Comparisons to moving bandit algorithm: *A black-box optimization strategy blind to switch cost usually has $\mathbb{E}[R(T)] = o(1)$, $\mathbb{E}[\sum_{t=1}^{T} \Gamma^t] = \Omega(T)$ and thus obtain a linear movement regret $\mathbb{E}[R^+] = \mathbb{E}[R(T)] + \mathbb{E}[\sum_{t=1}^{T} \Gamma^t] = \Omega(T)$. For switch-cost-aware alternatives, the closest result on moving regret is Theorem 2 in [Koren et al., 2017b]. However, their result relies on the Lispchitz property of the movement metric, which does not hold in our setting as the cost from changing variables in modules is not even continuous. By leveraging arms correlation with BO and adapting a lazy arm selection strategy, we extend their result by achieving a sublinear rate.*

5 EXPERIMENTS

In this section, we start by testing LaMBO on benchmark synthetic functions used in other studies [Vellanki et al., 2017, Kirschner et al., 2019]. Following this, we apply LaMBO to tune a multi-stage neuroimaging pipeline that reconstructs 3D images from segmented 2D images.

Experimental setup. For simplicity, we used the squared exponential kernel and initialized it using 15 random samples before starting the inference procedure. In our experiments, the functions are normalized by their maximized absolute value for clear comparisons, the regularization parameter is fixed to $\lambda = 0.1$, the UCB parameter is set each iteration as $\beta_t = 0.2D \log 2t$, and the learning rate is set to $\eta = 1$. The sampling noise ϵ is assumed to be independent Gaussian with standard deviation 0.01. For construction of MSET, we test on the simplest case where $d_i = 1$ and partition the variable space in each module into 2 sets aligned with a random coordinate. Some practical and detailed discussions on the hyperparameter choices and partition strategies are deferred to Supp. B. The curves on synthetic data and real data were computed by averaging across 100 and 20 simulations, respectively. We compare LaMBO with common baselines GP-UCB [Srinivas et al., 2010], GP-EI [Močkus, 1975], Max-value entropy search [Wang and Jegelka, 2017], random sampling, and three cost-aware strategies: EIpu [Snoek et al., 2012], CA-MOBO [Abdolshah et al., 2019], and CArBO [Lee et al., 2020]. To adapt the cost-aware strategies to our setting, we update the cost function at each iteration to be to the switching cost Γ^t defined in Eqn. (6) in Section 3.

5.1 EXPERIMENTS ON SYNTHETIC FUNCTIONS

For synthetic benchmarks, we selected a number of common functions used to test algorithms in the literature. However, unlike our real data examples that have clear modular structure due to the different sets of operations performed at different stages, the variables in synthetic test functions do not readily admit a modular structure. Thus to simulate a 2-module or 3-module scenario, we divide the variables in each function into different groups to create effective modules.

In Figure 3 (A-D), we compare the methods on synthetic functions in a two-module setting with a cost ratio of 10 to 1. In (A-B), we show the results for two different synthetic functions Hartmann and Rastrigin, respectively (more experiments on synthetic functions could be found in Supp. E). In (C), we study the impact of splitting variables into sets of different dimensions with function Ackley 8D, a synthetic function with a sharp global optimum surrounded with multiple local ones. The result suggests LaMBO is stable among different variable configurations in modules. Under the same setting as in (A), we verify our regret analysis by plotting the cumulative movement regret curve in (D), and study the performance of the different approaches when the cost is $[1, 1]$. The former shows that LaMBO minimizes the averaged movement regret of (7) better than cost-aware and unaware baselines. The later shows that LaMBO performs better when the cost ratio between modules is large while on par with alternative when the ratio is $\simeq 1$. (F) explores the 3-module setting of Ackley 8D ($[2, 2, 4]$), with the cost $[40, 10, 1]$. In this case, we found that it performs even better than its 2-module counterpart in Figure reffig:synsupp, suggesting LaMBO's applicability in pipeline with many modules.

Overall, we find that LaMBO outperforms other approaches and really shines when the cost of earlier modules is much larger (as seen in (A) vs. (E)). When we track the optimization trajectory, we observe that LaMBO performs similarly to other methods early on, but with further iterations, LaMBO starts to outperform the alternatives. This could be explained by inaccurate estimation of the function at early stages, and the fact that aggressive input changes could outperform the more conservative or lazy strategy used in LaMBO. However, as more samples are gathered, LaMBO demonstrates more power in terms of its cost efficiency by being lazy in variable switching.

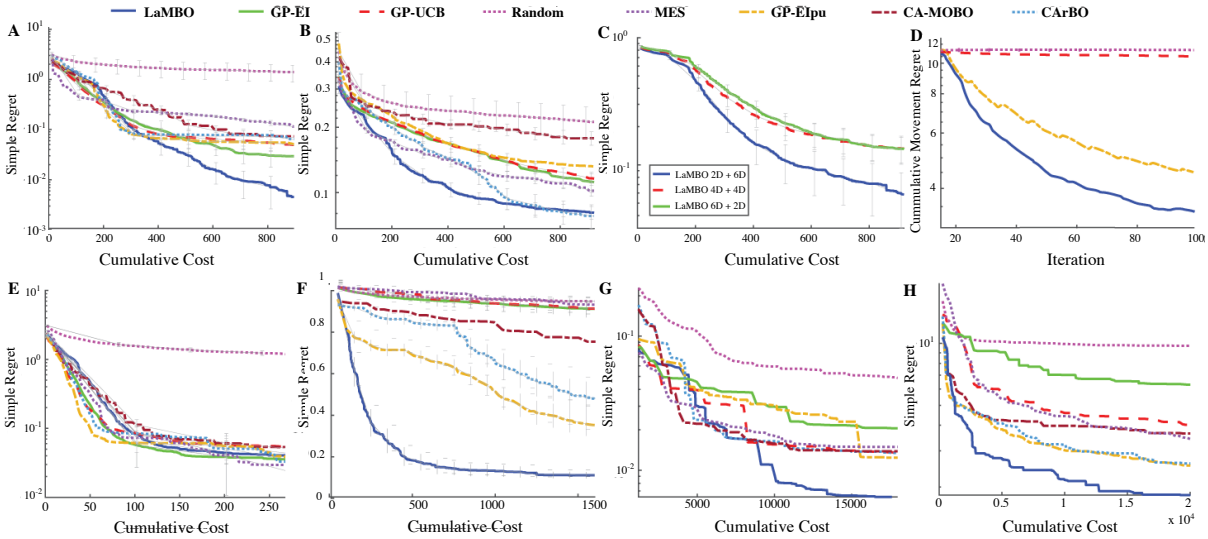

Figure 3: *Results on synthetic datasets and a brain mapping example.* In (A-D), we compare LaMBO with other BO algorithms on synthetic functions in a two module setting with a cost ratio of 10 to 1. In (A-B), we show the results for two different synthetic functions (Hartmann 6D, Rastrigin 6D). We split the variables of them into two modules with the first 3 dimensions in one module and the remaining three in the second. In (C), we study the impact of splitting variables into sets of different dimensions, by splitting the Ackley 8D into three different configurations $[2, 6]$, $[4, 4]$, and $[6, 2]$. In (D), we study the cumulative movement regret with Hartman 6D. This verifies our theory that LaMBO can effectively reduce the movement regret. In (E), we show the performance of the different approaches when the cost ratio between modules equals one, using Hartman 6D. (F) explores the three module setting of Ackley 8D splitting the variables into dimensions $[2, 2, 4]$, and define the costs by $= [40, 10, 1]$. In this case, when the costs accumulate early on, LaMBO really shines. Finally, we depict a brain mapping pipeline consisting of two (G) and (H) three modules, where the costs are modeled with estimated amount of time to execute each module ([326,325,55] sec).

5.2 APPLICATION TO A MULTI-STAGE NEUROIMAGING PIPELINE

Segmentation and identification of neural structures of interest (e.g., cell bodies, axons, or vasculature) is an important problem in connectomics and other applications of brain mapping [Helmstaedter et al., 2013, Oh et al., 2014, Dyer et al., 2017]. However, when dealing with large datasets, transfer can be challenging, and thus workflows must be re-optimized for each new dataset [Johnson et al., 2020]. Here, we consider the optimization of a relatively common three-stage pipeline, consisting of a pre-processing (image enhancement via denoising and contrast normalization), semantic segmentation for pixel-level prediction (via an U-net architecture), and a post-processing operation (to reconstruct a 3D volume). For comparison, we also consider a simplified pipeline without the pre-processing. To optimize this pipeline, we use a publicly available X-ray microCT dataset [Prasad et al., 2020] to set up the experiments in both a two-module (no pre-processing) and full three-module version of the pipeline.

In the first module, a pre-processing operation is performed where we tune a contrast parameter and denoising parameter. In the second module we train an U-Net, where in this case we tune the learning rate and batch size. The third module is in charge of post-processing and generates 3D reconstructions from the U-Net output; the hyperparameters in this module include a label purity score, cell opening size, and a shape parameter to determine whether uncertain components are either cells or blood vessels. Details of search space for each module are described in Supp. 5.2). The cost of the experiment is the aggregate recorded clock time for generating an output after changing a variable in a specific module. To test LaMBO on the problem, we gathered an offline data set consisting of $606,000$ different hyperparameters obtained by exhaustive search.

In the two-module case (Figure 3G), we observe a transition effect; when enough cost has been spent, LaMBO starts to increase its gap in performance over other methods. In the three-module case (Figure 3H) the advantage is even more pronounced, where the transition happens earlier. Quantitatively it shows that to get close to the optimum (within 5%), LaMBO can achieve this result in only 25% of the time required by the best alternative approach (1.4 vs. 5.6 hours).

6 DISCUSSION

This paper addresses a real-world problem of system optimization that is encountered in a variety of scientific disciplines. Increasingly, as we expand the size of datasets in different domains, we need automated solutions to quickly apply advanced machine learning systems to new datasets and re-optimize systems in an end-to-end manner. To tackle this problem, we introduced a new algorithm for Bayesian optimization that leverages known modular structure in an otherwise black-box system to minimize the overall cost required for global optimization. We showed how to leverage structure in such systems by incorporating a lazy switching strategy with Bayesian optimization. In the future, we would like to generalize our method to the case where both the function and switching costs are unknown, and extend to more complex cost hierarchies.

Acknowledgements

This work was supported by the NIH award 1R24MH114799-01 and awards IIS-1755871 and CCF-1740776 from the NSF.

References

Majid Abdolshah, Alistair Shilton, Santu Rana, Sunil Gupta, and Svetha Venkatesh. Cost-aware multi-objective bayesian optimisation. *arXiv preprint arXiv:1909.03600*, 2019.

Alexandre Abraham, Fabian Pedregosa, Michael Eickenberg, Philippe Gervais, Andreas Mueller, Jean Kossaifi, Alexandre Gramfort, Bertrand Thirion, and Gaël Varoquaux. Machine learning for neuroimaging with scikit-learn. *Frontiers in neuroinformatics*, 8:14, 2014.

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

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, page 2546–2554, Red Hook, NY, USA, 2011. Curran Associates Inc. ISBN 9781618395993.

Felix Berkenkamp, Angela P Schoellig, and Andreas Krause. Safe controller optimization for quadrotors with gaussian processes. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 491–496. IEEE, 2016.

Jeremy Davis-Turak, Sean M Courtney, E Starr Hazard, W Bailey Glen Jr, Willian A da Silveira, Timothy Wesselman, Larry P Harbin, Bethany J Wolf, Dongjun Chung, and Gary Hardiman. Genomics pipelines and data integration: challenges and opportunities in the research setting. *Expert review of molecular diagnostics*, 17(3): 225–237, 2017.

Ofer Dekel, Jian Ding, Tomer Koren, and Yuval Peres. Bandits with switching costs: T 2/3 regret. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 459–467, 2014.

Eva L Dyer, William Gray Roncal, Judy A Prasad, Hugo L Fernandes, Doga Gürsoy, Vincent De Andrade, Kamel Fezzaa, Xianghui Xiao, Joshua T Vogelstein, Chris Jacobsen, et al. Quantifying mesoscale neuroanatomy using x-ray microtomography. *Eneuro*, 4(5), 2017.

Michal Feldman, Tomer Koren, Roi Livni, Yishay Mansour, and Aviv Zohar. Online pricing with strategic and patient buyers. In *Advances in Neural Information Processing Systems*, pages 3864–3872, 2016.

Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

R. Garnett, M. A. Osborne, and S. J. Roberts. Bayesian optimization for sensor set selection. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, IPSN '10, page 209–219, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781605589886. doi: 10.1145/1791212.1791238. URL `https://doi.org/10.1145/1791212.1791238`.

Javier González, Zhenwen Dai, Philipp Hennig, and Neil Lawrence. Batch bayesian optimization via local penalization. In *Artificial intelligence and statistics*, pages 648–657, 2016.

Moritz Helmstaedter, Kevin L Briggman, Srinivas C Turaga, Viren Jain, H Sebastian Seung, and Winfried Denk. Connectomic reconstruction of the inner plexiform layer in the mouse retina. *Nature*, 500(7461):168–174, 2013.

Erik C Johnson, Miller Wilt, Luis M Rodriguez, Raphael Norman-Tenazas, Corban Rivera, Nathan Drenkow, Dean Kleissas, Theodore J LaGrow, Hannah Cowley, Joseph Downs, et al. Toward a reproducible, scalable framework for processing large neuroimaging datasets. *BioRxiv*, page 615161, 2019.

Erik C Johnson, Miller Wilt, Luis M Rodriguez, Raphael Norman-Tenazas, Corban Rivera, Nathan Drenkow, Dean Kleissas, Theodore J LaGrow, Hannah P Cowley, Joseph Downs, et al. Toward a scalable framework for reproducible processing of volumetric, nanoscale neuroimaging datasets. *GigaScience*, 9(12):giaa147, 2020.

Adam Kalai and Santosh Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.

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.

Kirthevasan Kandasamy, Gautam Dasarathy, Jeff Schneider, and Barnabás Póczos. Multi-fidelity bayesian optimisation with continuous approximations. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1799–1808. JMLR. org, 2017.

Tarun Kathuria, Amit Deshpande, and Pushmeet Kohli. Batched gaussian process bandit optimization via determinantal point processes. In *Advances in Neural Information Processing Systems*, pages 4206–4214, 2016.

Johannes Kirschner, Mojmír Mutnỳ, Nicole Hiller, Rasmus Ischebeck, and Andreas Krause. Adaptive and safe bayesian optimization in high dimensions via one-dimensional subspaces. *arXiv preprint arXiv:1902.03229*, 2019.

Tomer Koren, Roi Livni, and Yishay Mansour. Multi-armed bandits with metric movement costs. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4119–4128. Curran Associates, Inc., 2017a.

Tomer Koren, Roi Livni, and Yishay Mansour. Bandits with movement costs and adaptive pricing. In Satyen Kale and Ohad Shamir, editors, *Proceedings of the 2017 Conference on Learning Theory*, volume 65 of *Proceedings of Machine Learning Research*, pages 1242–1268, Amsterdam, Netherlands, 07–10 Jul 2017b. PMLR. URL `http://proceedings.mlr.press/v65/koren17a.html`.

Remi Lam and Karen Willcox. Lookahead bayesian optimization with inequality constraints. In *Advances in Neural Information Processing Systems*, pages 1890–1900, 2017.

Remi Lam, Karen Willcox, and David H Wolpert. Bayesian optimization with a finite budget: An approximate dynamic programming approach. In *Advances in Neural Information Processing Systems*, pages 883–891, 2016.

Eric Hans Lee, Valerio Perrone, Cedric Archambeau, and Matthias Seeger. Cost-aware bayesian optimization, 2020.

Kisuk Lee, Nicholas Turner, Thomas Macrina, Jingpeng Wu, Ran Lu, and H Sebastian Seung. Convolutional nets for reconstructing neural circuits from brain images acquired by serial section electron microscopy. *Current opinion in neurobiology*, 55:188–198, 2019.

Mark McLeod, Michael A Osborne, and Stephen J Roberts. Practical bayesian optimization for variable cost objectives. *arXiv preprint arXiv:1703.04335*, 2017.

J. Mockus. The bayesian approach to global optimization. In R. F. Drenick and F. Kozin, editors, *System Modeling and Optimization*, pages 473–481, Berlin, Heidelberg, 1982. Springer Berlin Heidelberg. ISBN 978-3-540-39459-4.

Jonas Močkus. On bayesian methods for seeking the extremum. In *Optimization techniques IFIP technical conference*, pages 400–404. Springer, 1975.

Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. The application of bayesian methods for seeking the extremum. *Towards global optimization*, 2(117-129): 2, 1978.

Seung Wook Oh, Julie A Harris, Lydia Ng, Brent Winslow, Nicholas Cain, Stefan Mihalas, Quanxin Wang, Chris Lau, Leonard Kuan, Alex M Henry, et al. A mesoscale connectome of the mouse brain. *Nature*, 508(7495):207–214, 2014.

Matthias Poloczek, Jialei Wang, and Peter Frazier. Multi-information source optimization. In *Advances in Neural Information Processing Systems*, pages 4288–4298, 2017.

JA Prasad, AH Balwani, E Johnson, JD Miano, V Sampathkumar, V de Andrade, M Du, R Vescovi, C Jacobsen, D Gursoy, N Kasthuri, and EL Dyer. A three-dimensional thalamocortical dataset for characterizing brain heterogeneity. *under review in Nature Scientific Data*, 2020.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 1015–1022, USA, 2010. Omnipress. ISBN 978-1-60558-907-7. URL `http://dl.acm.org/citation.cfm?id=3104322.3104451`.

Pratibha Vellanki, Santu Rana, Sunil Gupta, David Rubin, Alessandra Sutti, Thomas Dorin, Murray Height, Paul Sanders, and Svetha Venkatesh. Process-constrained batch bayesian optimisation. In *Advances in Neural Information Processing Systems*, pages 3414–3423, 2017.

Zi Wang and Stefanie Jegelka. Max-value entropy search for efficient bayesian optimization. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3627–3635. JMLR. org, 2017.

Jian Wu, Saul Toscano-Palmerin, Peter I Frazier, and Andrew Gordon Wilson. Practical multi-fidelity bayesian optimization for hyperparameter tuning. In *Uncertainty in Artificial Intelligence*, pages 788–798. PMLR, 2020.