Mixed-Variable Bayesian Optimization

Erik Daxberger[*,†,1,2], **Anastasia Makarova**[*,3], **Matteo Turchetta**[2,3] and **Andreas Krause**[3]
[1]Department of Engineering, University of Cambridge
[2]Max Planck Institute for Intelligent Systems, Tübingen
[3]Department of Computer Science, ETH Zurich
ead54@cam.ac.uk,{anmakaro,matteo.turchetta,krausea}@inf.ethz.ch

Abstract

The optimization of expensive to evaluate, black-box, mixed-variable functions, i.e. functions that have continuous and discrete inputs, is a difficult and yet pervasive problem in science and engineering. In Bayesian optimization (BO), special cases of this problem that consider fully continuous or fully discrete domains have been widely studied. However, few methods exist for mixed-variable domains and none of them can handle discrete constraints that arise in many real-world applications. In this paper, we introduce MiVaBo, a novel BO algorithm for the efficient optimization of mixed-variable functions combining a linear surrogate model based on expressive feature representations with Thompson sampling. We propose an effective method to optimize its acquisition function, a challenging problem for mixed-variable domains, making MiVaBo the first BO method that can handle complex constraints over the discrete variables. Moreover, we provide the first convergence analysis of a mixed-variable BO algorithm. Finally, we show that MiVaBo is significantly more sample efficient than state-of-the-art mixed-variable BO algorithms on several hyperparameter tuning tasks, including the tuning of deep generative models.

1 Introduction

Bayesian optimization (BO) [Močkus, 1975] is a well-established paradigm to optimize costly-to-evaluate, complex, black-box objectives that has been successfully applied to many scientific domains. Most of the existing BO literature focuses on objectives that have purely *continuous* domains, such as those arising in tuning of continuous hyperparameters of machine learning algorithms, recommender systems, and preference learning [Shahriari *et al.*, 2016]. More recently, problems with purely *discrete* domains, such as food safety control and model-sparsification in multi-component systems [Baptista and Poloczek, 2018] have been considered.

However, many real-world optimization problems in science and engineering are of *mixed-variable* nature, involving *both* continuous and discrete input variables, and exhibit complex constraints. For example, tuning the hyperparameters of a convolutional neural network involves both continuous variables, e.g., learning rate and momentum, and discrete ones, e.g., kernel size, stride and padding. Also, these hyperparameters impose validity constraints, as some combinations of kernel size, stride and padding define invalid networks. Further examples of mixed-variable, potentially constrained, optimization problems include sensor placement [Krause *et al.*, 2008], drug discovery [Negoescu *et al.*, 2011], optimizer configuration [Hutter *et al.*, 2011] and many others. Nonetheless, only few BO methods can address the unconstrained version of such problem and no existing method can handle the constrained one. This work introduces the first algorithm that can efficiently optimize mixed-variable functions subject to known constraints with provable convergence guarantees.

Related Work. Extending *continuous* BO methods [Shahriari *et al.*, 2016] to mixed inputs requires ad-hoc relaxation methods to map the problem to a fully continuous one and rounding methods to map the solution back. This ignores the original domain structure, makes the solution quality dependent on the relaxation and rounding methods, and makes it hard to handle discrete constraints. Extending *discrete* BO methods [Baptista and Poloczek, 2018; Oh *et al.*, 2019] to mixed inputs requires a discretization of the continuous domain part, the granularity of which is crucial: If it is too small, the domain becomes prohibitively large; if it is too large, the domain may only contain poorly performing values of the continuous inputs. Few BO methods address the mixed-variable setting. SMAC [Hutter *et al.*, 2011] uses a random forest surrogate model. However, its frequentist uncertainty estimates may be too inaccurate to steer the sampling. TPE [Bergstra *et al.*, 2011] uses kernel density estimation to find inputs that will likely improve upon and unlikely perform worse than the incumbent solution. While SMAC and TPE can handle hierarchical constraints, they cannot handle more general constraints over the discrete variables, e.g., cardinality constraints. They also lack convergence guarantees. Hyperband (HB) [Li *et al.*, 2018] uses cheap but less accurate approximations of the objective to dynamically allocate resources for function evaluations. BOHB [Falkner *et al.*, 2018] is the model-based counterpart of HB, based on TPE. They thus extend existing mixed-variable methods to the multi-fidelity setting rather than propos-

[*]Equal contribution. [†]Work done while at ETH Zurich.

ing new ones, which is complementary to our approach, rather than in competition with it. [Garrido-Merchán and Hernández-Lobato, 2018] propose a Gaussian process kernel to model discrete inputs without rounding bias. Their method lacks guarantees and cannot handle discrete constraints. We instead use discrete optimizers for the acquisition function, which avoid bias by only making integer evaluations. Finally, while [Hernández-Lobato *et al.*, 2015; Gardner *et al.*, 2014; Sui *et al.*, 2015] extend continuous BO methods to handle unknown constraints, no method can handle known discrete constraints in a mixed-variable domain.

Contributions. We introduce MIVABO, the first BO algorithm for efficiently optimizing mixed-variable functions subject to known linear and quadratic integer constraints, encompassing many of the constraints present in real-world domains (e.g. cardinality, budget and hierarchical constraints). It relies on a linear surrogate model that decouples the continuous, discrete and mixed components of the function using an expressive feature expansion (Sec. 3.1). We exploit the ability of this model to efficiently draw samples from the posterior over the objective (Sec. 3.2) by combining it with Thompson sampling, and show how to optimize the resulting constrained acquisition function (Sec. 3.3). While in continuous BO, optimizing the acquisition function is difficult but has well-established solutions, this is not true for mixed-variable spaces and doing this efficiently and accurately is a key challenge that hugely impacts the algorithm's performance. We also provide the first convergence analysis of a mixed-variable BO algorithm (Sec. 3.5). Finally, we demonstrate the effectiveness of MIVABO on a set of complex hyperparameter tuning tasks, where it outperforms state-of-the-art methods and is competitive with human experts (Sec. 4).

2 Problem Statement

We consider the problem of optimizing an unknown, costly-to-evaluate function defined over a mixed-variable domain, accessible through noisy evaluations and subject to known linear and quadratic constraints. Formally, we aim to solve

$$\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) \quad \text{s.t. } g^c(\mathbf{x}) \geq 0,\ g^d(\mathbf{x}) \geq 0, \qquad (1)$$

where $\mathcal{X} \subseteq \mathcal{X}^c \times \mathcal{X}^d$ with continuous subspace $\mathcal{X}^c$ and discrete subspace $\mathcal{X}^d$. Both constraints $g^c(\mathbf{x}) \geq 0$ over $\mathcal{X}^c$ and $g^d(\mathbf{x}) \geq 0$ over $\mathcal{X}^d$ are known, and specifically $g^d(\mathbf{x})$ are linear or quadratic. We assume, that the domain of the continuous inputs is box-constrained and can thus, w.l.o.g., be scaled to the unit hypercube, $\mathcal{X}^c = [0,1]^{D_c}$. We further assume, w.l.o.g., that the discrete inputs are binary, i.e., vectors $\mathbf{x}^d \in \mathcal{X}^d = \{0,1\}^{D_d}$ are vertices of the unit hypercube. This representation can effectively capture the domain of any discrete function. For example, a vector $\mathbf{x}^d = [x_i^d]_{i=1}^{D_d} \in \mathcal{X}^d$ can encode a subset A of a ground set of D_d elements, such that $x_i^d = 1 \Leftrightarrow a_i \in A$ and $x_i^d = 0 \Leftrightarrow a_i \notin A$, yielding a set function. Alternatively, $\mathbf{x}^d \in \mathcal{X}^d$ can be a binary encoding of integer variables, yielding a function defined over integers.

Background. BO algorithms are iterative black-box optimization methods which, at every step t, select an input $\mathbf{x}_t \in \mathcal{X}$ and observe a noise-perturbed output $y_t \triangleq f(\mathbf{x}_t) + \epsilon$ with $\epsilon \overset{\text{iid}}{\sim} \mathcal{N}(0, \beta^{-1})$, $\beta > 0$. As evaluating f is costly, the goal is to query inputs based on past observations to find a global minimizer $\mathbf{x}_* \in \arg\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$ as efficiently and accurately as possible. To this end, BO algorithms leverage two components: (i) a *probabilistic function model* (or *surrogate*), that encodes the belief about f based on the observations available, and (ii) an *acquisition function* $\alpha : \mathcal{X} \to \mathbb{R}$ that expresses the informativeness of input $\mathbf{x}$ about the location of $\mathbf{x}_*$, given the surrogate of f. Based on the model of f, we query the best input measured by the acquisition function, then update the model with the observation and repeat this procedure. The goal of the acquisition function is to simultaneously learn about inputs that are likely to be optimal and about poorly explored regions of the input space, i.e., to trade-off exploitation against exploration. Thus, BO reduces the original hard black-box optimization problem to a series of cheaper problems $\mathbf{x}_t \in \arg\max_{\mathbf{x}\in\mathcal{X}} \alpha_t(\mathbf{x})$. However, in our case, these optimization problems involve mixed variables and exhibit linear and quadratic constraints and are thus still challenging. We now present MIVABO, an algorithm to efficiently solve the optimization problem in Eq. (1).

3 MIVABO Algorithm

We first introduce the linear model used to represent the objective (Sec. 3.1) and describe how to do inference with it (Sec. 3.2). We then show how to use Thompson sampling to query informative inputs (Sec. 3.3) and, finally, provide a bound on the regret incurred by MIVABO. (Sec. 3.5).

3.1 Model

We propose a surrogate model that accounts for both discrete and continuous variables in a principled way, while balancing two conflicting goals: Model expressiveness versus feasibility of Bayesian inference and of the constrained optimization of the mixed-variable acquisition function. Linear models defined over non-linear feature mappings, $f(\mathbf{x}) = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x})$, are a class of flexible parametric models that strike a good trade-off between model capacity, interpretability and ease of use through the definition of features $\boldsymbol{\phi} : \mathcal{X} \to \mathbb{R}^M$. While the complexity of the model is controlled by the number of features, M, its capacity depends on their definition. Therefore, to make the design of a set of expressive features more intuitive, we treat separately the contribution to the objective f from the discrete part of the domain, from the continuous part of the domain, and from the interaction of the two,

$$f(\mathbf{x}) = \textstyle\sum_{j\in\{d,c,m\}} {\mathbf{w}^j}^\top \boldsymbol{\phi}^j(\mathbf{x}^j) \qquad (2)$$

where, for $j \in \{d, c, m\}$, $\boldsymbol{\phi}^j(\mathbf{x}^j) = [\phi_i^j(\mathbf{x}^j)]_{i=1}^{M_j} \in \mathbb{R}^{M_j}$ and $\mathbf{w}^j \in \mathbb{R}^{M_j}$ are the feature and weight vector for the *d*iscrete, *c*ontinuous and *m*ixed function component, respectively.

In many real-world domains, a large set of features can be discarded *a priori* to simplify the design space. It is common practice in high-dimensional BO to assume that only low-order interactions between the variables contribute significantly to the objective, which was shown for many practical problems [Rolland *et al.*, 2018; Mutný and Krause, 2018], including deep neural network hyperparameter tuning [Hazan *et al.*, 2017]. Similarly, we focus on features defined over small subsets of the inputs. Formally, we consider

$\boldsymbol{\phi}(\mathbf{x}) = [\phi_k(\mathbf{x}_k)]_{k=1}^M$, where $\mathbf{x}_k$ is a subvector of $\mathbf{x}$ containing exclusively continuous or discrete variables or a mix of both. Thus, the objective $f(\mathbf{x})$ can be decomposed into a sum of low-dimensional functions $f_k(\mathbf{x}_k) \triangleq w_k\phi_k(\mathbf{x}_k)$ defined over subspaces $\mathcal{X}_k \subseteq \mathcal{X}$ with $\dim(\mathcal{X}_k) \ll \dim(\mathcal{X})$. This defines a *generalized additive model* [Rolland *et al.*, 2018; Hastie, 2017], where the same variable can be included in multiple subvectors/features. The complexity of this model is controlled by the *effective dimensionality* (ED) of the subspaces, which is crucial under limited computational resources. In particular, let $\bar{D}_d \triangleq \max_{k\in[M]} \dim(\mathcal{X}_k^d)$ denote the ED of the discrete component in Eq. (2), i.e. the dimensionality of the largest subspace that exclusively contains discrete variables. Analogously, $\bar{D}_c$ and $\bar{D}_m$ denote the EDs of the continuous and mixed component, respectively. Intuitively, the ED corresponds to the maximum order of the variable interactions present in f. Then, the number of features $M \in \mathcal{O}\big(D_d^{\bar{D}_d} + D_c^{\bar{D}_c} + (D_d + D_c)^{\bar{D}_m}\big)$ scales exponentially in the EDs only (as modeling up to L-th order interactions of N inputs requires $\sum_{l=0}^{L}\binom{N}{l} \in \mathcal{O}(N^L)$ terms), which are usually small, even if the true dimensionality is large.

Discrete Features $\boldsymbol{\phi}^d$. We aim to define features ϕ^d that can effectively represent the discrete component of Eq. (2) as a linear function, which should generally be able to capture arbitrary interactions between the discrete variables. To this end, we consider all subsets S of the discrete variables in $\mathcal{X}^d$ (or, equivalently, all elements S of the powerset $2^{\mathcal{X}_d}$ of $\mathcal{X}_d$) and define a monomial $\prod_{j\in S} x_j^d$ for each subset S (where for $S = \emptyset$, $\prod_{j\in\emptyset} x_j^d = 1$). We then form a weighted sum of all monomials to yield the multi-linear polynomial ${\mathbf{w}^d}^\top \boldsymbol{\phi}^d(\mathbf{x}^d) = \sum_{S\in 2^{\mathcal{X}_d}} w_S \prod_{j\in S} x_j^d$. This functional representation corresponds to the Fourier expansion of a *pseudo-Boolean function* (PBF) [Boros and Hammer, 2002]. In practice, an exponential number of features can be prohibitively expensive and may lead to high-variance estimators as in BO one typically does not have access to enough data to robustly fit a large model. Alternatively, [Baptista and Poloczek, 2018; Hazan *et al.*, 2017] empirically found that a second-order polynomial in the Fourier basis provides a practical balance between expressiveness and efficiency, even when the true function is of higher order. In our model, we also consider quadratic PBFs, ${\mathbf{w}^d}^\top \boldsymbol{\phi}^d(\mathbf{x}^d) = w_\emptyset + \sum_{i=1}^n w_{\{i\}} x_i^d + \sum_{1\le i<j\le n} w_{\{i,j\}} x_i^d x_j^d$, which induces the discrete feature representation $\boldsymbol{\phi}^d(\mathbf{x}^d) \triangleq [1, \{x_i^d\}_{i=1}^{D_d}, \{x_i^d x_j^d\}_{1\le i<j\le D_d}]^\top$ and reduces the number of model weights to $M_d \in \mathcal{O}(D_d^2)$.

Continuous Features $\boldsymbol{\phi}^c$. In BO over continuous spaces, most approaches are based on Gaussian process (GP) models [Williams and Rasmussen, 2006] due to their flexibility and ability to capture large classes of continuous functions. To fit our linear model formulation, we leverage GPs' expressiveness by modeling the continuous part of our model in Eq. (2) using feature expansions $\boldsymbol{\phi}^c(\mathbf{x}^c)$ that result in a finite linear approximation of a GP. One simple, yet theoretically sound, choice is the class of Random Fourier Features (RFFs) [Rahimi and Recht, 2008], which use Monte Carlo integration for a randomized approximation of a GP. Alternatively, one can use Quadrature Fourier Features [Mutný and Krause, 2018], which instead use numerical integration for a deterministic approximation, which is particularly effective for problems with low effective dimensionality. Both feature classes were successfully used in BO [Jenatton *et al.*, 2017; Mutný and Krause, 2018]. In our experiments, we use RFFs approximating a GP with a squared exponential kernel, which we found to best trade off complexity vs. accuracy in practice.

Mixed Features $\boldsymbol{\phi}^m$. The mixed term should capture as rich and realistic interactions between the discrete and continuous variables as possible, while keeping model inference and acquisition function optimization efficient. To this end, we stack products of all pairwise combinations of features of the two variable types, i.e. $\boldsymbol{\phi}^m(\mathbf{x}^d, \mathbf{x}^c) \triangleq [\phi_i^d(\mathbf{x}^d)\cdot\phi_j^c(\mathbf{x}^c)]_{1\le i\le M_d, 1\le j\le M_c}^\top$. This formulation provides a good trade-off between modeling accuracy and computational complexity. In particular, it allows us to reduce $\boldsymbol{\phi}^m$ to the discrete feature representation $\boldsymbol{\phi}^d$ when conditioned on a fixed assignment of continuous variables $\boldsymbol{\phi}^c$ (and vice versa). This property is crucial for optimizing the acquisition function, as it allows us to optimize the mixed term of our model by leveraging the tools for optimizing the discrete and continuous parts individually. The proposed representation contains $M_d M_c$ features, resulting in a total of $M = M_d + M_c + M_d M_c$. To reduce model complexity, prior knowledge about the problem can be incorporated into the construction of the mixed features. In particular, one may consider the following approaches. Firstly, one can exploit a known interaction structure between variables, e.g., in form of a dependency graph, and ignore the features that are known to be irrelevant. Secondly, one can start by including all of the proposed pairwise feature combinations and progressively discard not-promising ones. Finally, for high-dimensional problems, one can do the opposite and progressively add pairwise feature combinations, starting from the empty set.

3.2 Model Inference

Let $\mathbf{X}_{1:t} \in \mathbb{R}^{t\times D}$ be the matrix whose i^{th} row contains the input $\mathbf{x}_i \in \mathcal{X}$ queried at iteration i, $\dim\mathcal{X} = D$, and let $\mathbf{y}_{1:t} = [y_1, \ldots, y_t]^\top \in \mathbb{R}^t$ be the array of the corresponding noisy function observations. Also, let $\mathbf{\Phi}_{1:t} \in \mathbb{R}^{t\times M}$ be the matrix whose i^{th} row contains the featurized input $\boldsymbol{\phi}(\mathbf{x}_i) \in \mathbb{R}^M$. The formulation of f in Eq. (2) and the noisy observation model induce the Gaussian likelihood $p(\mathbf{y}_{1:t}|\mathbf{X}_{1:t}, \mathbf{w}) = \mathcal{N}(\mathbf{\Phi}_{1:t}\mathbf{w}, \beta^{-1}\mathbf{I})$. To reflect our *a priori* belief about the weight vector $\mathbf{w}$ and thus f, we specify a prior distribution over $\mathbf{w}$. A natural choice for this is a zero-mean isotropic Gaussian prior $p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{0}, \alpha^{-1}\mathbf{I})$, with precision $\alpha > 0$, which encourages $\mathbf{w}$ to be uniformly small, so that the final predictor is a sum of all features, each giving a small, nonzero contribution. Given the likelihood and prior, we infer the posterior $p(\mathbf{w}|\mathbf{X}_{1:t}, \mathbf{y}_{1:t}, \alpha, \beta) \propto p(\mathbf{y}_{1:t}|\mathbf{X}_{1:t}, \mathbf{w}, \beta)p(\mathbf{w}|\alpha)$, which due to conjugacy is Gaussian, $p(\mathbf{w}|\mathbf{X}_{1:t}, \mathbf{y}_{1:t}) = \mathcal{N}(\mathbf{m}, \mathbf{S}^{-1})$, with mean $\mathbf{m} = \beta\mathbf{S}^{-1}\mathbf{\Phi}_{1:t}^\top\mathbf{y}_{1:t} \in \mathbb{R}^M$ and precision $\mathbf{S} = \alpha\mathbf{I} + \beta\mathbf{\Phi}_{1:t}^\top\mathbf{\Phi}_{1:t} \in \mathbb{R}^{M\times M}$ [Williams and Rasmussen, 2006]. This simple analytical treatment of the posterior distribution over $\mathbf{w}$ is a main benefit of this model, which can be viewed as a GP with a linear kernel in feature space.

3.3 Acquisition Function

We propose to use Thompson sampling (TS) [Thompson, 1933], which samples weights $\widetilde{\mathbf{w}} \sim p(\mathbf{w}|\mathbf{X}_{1:t}, \mathbf{y}_{1:t}, \alpha, \beta)$ from the posterior and chooses the next input by solving $\widehat{\mathbf{x}} \in \arg\min_{\mathbf{x}\in\mathcal{X}} \widetilde{\mathbf{w}}^\top \boldsymbol{\phi}(\mathbf{x})$. TS intuitively focuses on inputs that are plausibly optimal and has previously been successfully applied in both discrete and continuous domains [Baptista and Poloczek, 2018; Mutný and Krause, 2018].

TS requires solving $\widehat{\mathbf{x}} \in \arg\min_{\mathbf{x}\in\mathcal{X}} \widetilde{\mathbf{w}}_t^\top \boldsymbol{\phi}(\mathbf{x})$, which is a challenging mixed-variable optimization problem. However, as $\widetilde{\mathbf{w}}_t^\top \boldsymbol{\phi}(\mathbf{x})$ decomposes as in Eq. (2), we can naturally use an alternating optimization scheme which iterates between optimizing the discrete variables $\mathbf{x}^d$ conditioned on a particular setting of the continuous variables $\mathbf{x}^c$ and vice versa, until convergence to some local optimum. While this scheme provides no theoretical guarantees, it is simple and thus widely and effectively applied in many contexts where the objective is hard to optimize. In particular, we iteratively solve $\widehat{\mathbf{x}}^d \in \arg\min_{\mathbf{x}^d\in\mathcal{X}^d} \big(\widetilde{\mathbf{w}}^{d\top} \boldsymbol{\phi}^d(\mathbf{x}^d) + \widetilde{\mathbf{w}}^{m\top} \boldsymbol{\phi}^m(\mathbf{x}^d, \mathbf{x}^c = \widehat{\mathbf{x}}^c)\big)$, $\widehat{\mathbf{x}}^c \in \arg\min_{\mathbf{x}^c\in\mathcal{X}^c} \big(\widetilde{\mathbf{w}}^{c\top} \boldsymbol{\phi}^c(\mathbf{x}^c) + \widetilde{\mathbf{w}}^{m\top} \boldsymbol{\phi}^m(\mathbf{x}^d = \widehat{\mathbf{x}}^d, \mathbf{x}^c)\big)$. Importantly, using the mixed features proposed in Sec. 3.1, these problems can be optimized by purely discrete and continuous optimizers, respectively. This also holds in the presence of mixed constraints $g^m(\mathbf{x}) \geq 0$ if those decompose accordingly into discrete and continuous constraints.

This scheme leverages independent subroutines for discrete and continuous optimization: For the discrete part, we exploit the fact that optimizing a second-order pseudo-Boolean function is equivalent to a binary integer quadratic program (IQP) [Boros and Hammer, 2002], allowing us to exploit commonly-used efficient and robust solvers such as Gurobi or CPLEX. While solving general binary IQPs is NP-hard [Boros and Hammer, 2002], these optimizers are in practice very efficient for the dimensionalities we consider (i.e., $D_d < 100$). This approach allows us to use any functionality offered by these tools, such as the ability to optimize objectives subject to linear constraints $\mathbf{A}\mathbf{x}^d \leq \mathbf{b}$, $\mathbf{A} \in \mathbb{R}^{K\times D_d}, \mathbf{b} \in \mathbb{R}^K$ or quadratic constraints $\mathbf{x}^{d\top}\mathbf{Q}\mathbf{x}^d + \mathbf{q}^\top\mathbf{x}^d \leq b$, $\mathbf{Q} \in \mathbb{R}^{D_d\times D_d}, \mathbf{q} \in \mathbb{R}^{D_d}, b \in \mathbb{R}$. For the continuous part, one can use optimizers commonly used in continuous BO, such as L-BFGS or DIRECT. In our experiments, we use Gurobi as the discrete and L-BFGS as the continuous solver within the alternating optimization scheme, which we always run until convergence.

3.4 Model Discussion

BO algorithms are comprised of three major design choices: the surrogate model to estimate the objective, the acquisition function to measure informativeness of the inputs and the acquisition function optimizer to select queries. Due to the widespread availability of general-purpose optimizers for continuous functions, continuous BO is mostly concerned with the first two design dimensions. However, this is different for mixed-variable constrained problems. We show in Sec. 4 that using a heuristic optimizer for the acquisition function optimization leads to poor queries and, therefore, poor performance of the BO algorithm. Therefore, the tractability of the acquisition function optimization influences and couples the other design dimensions. In particular, the following considerations make the choice of a linear model and TS the ideal combination of surrogate and acquisition function for our problem. Firstly, the linear model is preferable to a GP with a mixed-variable kernel as the latter would complicate the acquisition function optimization for two reasons: (i) the posterior samples would be arbitrary nonlinear functions of the discrete variables and (ii) it would be non-trivial to evaluate them at arbitrary points in the domain. In contrast, our explicit feature expansion solves both problems, while second order interactions provide a valid discrete function representation [Baptista and Poloczek, 2018; Hazan *et al.*, 2017] and lead to tractable quadratic MIPs with capacity for complex discrete constraints. Moreover, Random Fourier Features approximate common GP kernels arbitrarily well, and inference in MiVaBo scales linearly with the number of data points, making it applicable in cases where GP inference, which scales cubically with the number of data points, would be prohibitive. Secondly, TS induces a simple relation between the surrogate and the resulting optimization problem for the acquisition function, allowing to trade off model expressiveness and optimization tractability, which is a key challenge in mixed-variable domains. Finally, the combination of TS and the linear surrogate facilitates the convergence analysis described in Sec. 3.5, making MiVaBo the first mixed-variable BO method with theoretical guarantees.

3.5 Convergence Analysis

Using a linear model and Thompson sampling, we can leverage convergence analysis from linearly parameterized multi-armed bandits, a well-studied class of methods for solving structured decision making problems [Abeille *et al.*, 2017]. These also assume the objective to be linear in features $\boldsymbol{\phi}(\mathbf{x}) \in \mathbb{R}^M$ with a fixed but unknown weight vector $\mathbf{w} \in \mathbb{R}^M$, i.e. $\mathbb{E}[f(\mathbf{x})|\boldsymbol{\phi}(\mathbf{x})] = \mathbf{w}^\top\boldsymbol{\phi}(\mathbf{x})$, and aim to minimize the *total regret* up to time T: $\mathcal{R}(T) = \sum_{t=1}^T (f(\mathbf{x}_*) - f(\mathbf{x}_t))$. We obtain the following regret bound for MiVaBo:

Proposition 1. *Assume that the following assumptions hold in every iteration $t = 1, \dots, T$ of the* MiVaBo *algorithm:*

1. $\widetilde{\mathbf{w}}_t \sim \mathcal{N}(\mathbf{m}, 24M \ln T \ln \frac{1}{\delta}\mathbf{S}^{-1})$*, i.e. with scaled variance.*
2. $\mathbf{x}_t = \arg\min_{\mathbf{x}} \widetilde{\mathbf{w}}^\top \boldsymbol{\phi}(\mathbf{x})$ *is selected exactly.*[1]
3. $\|\widetilde{\mathbf{w}}_t\|_2 \leq c, \|\boldsymbol{\phi}(\mathbf{x}_t)\|_2 \leq c, \|f(\mathbf{x}_*) - f(\mathbf{x}_t)\|_2 \leq c, c \in \mathbb{R}^+$.

Then, $\mathcal{R}(T) \leq \tilde{\mathcal{O}}\left(M^{3/2}\sqrt{T}\ln\frac{1}{\delta}\right)$ *with probability* $1-\delta$.

Prop. 1 follows from Theorem 1 in [Abeille *et al.*, 2017] and works for infinite arms $\mathbf{x} \in \mathcal{X}, |\mathcal{X}| = \infty$. In our setting, both the discrete and continuous Fourier features (and, thus, the mixed features) satisfy the standard boundedness assumption, such that the proof indeed holds. Prop. 1 implies *no-regret*, $\lim_{T\to\infty} \mathcal{R}(T)/T = 0$, i.e., convergence to the global minimum, since the minimum found after T iterations is no further away from $f(\mathbf{x}_*)$ than the mean regret $\mathcal{R}(T)/T$. To our knowledge, MiVaBo is the first mixed-variable BO algorithm for which such a guarantee is known to hold.

[1]To this end, one can use more expensive but theoretically backed optimization methods instead of the alternating one, such as the powerful and popular *dual decomposition* [Sontag *et al.*, 2011].

4 Experiments

We present experimental results on tuning the hyperparameters of two machine learning algorithms, namely gradient boosting and a deep generative model, on multiple datasets.

Experimental Setup. For MIVABO[2], we set the prior variance α, observation noise variance β, and kernel bandwidth σ to 1.0, and scale the variance as stated in Prop. 1. We compare against SMAC, TPE, random search, and the popular GPyOpt BO package. GPyOpt uses a GP model with the upper confidence bound acquisition function [Srinivas *et al.*, 2010], and accounts for mixed variables by relaxing discrete variables to be continuous and later rounding them to the nearest discrete neighbor. To separate the influence of model choice and acquisition function optimization, we also consider the MIVABO model optimized by simulated annealing (SA) (MIVABO-SA) and the GP approach optimized by SA (GP-SA). We compare against the SA-based variants only in constrained settings, using more principled methods in unconstrained ones. To handle constraints, SA assigns high energy values to invalid inputs, making the probability of moving there negligible. We use SMAC, TPE and GPyOpt and SA with their respective default settings.

4.1 Gradient Boosting Tuning

The OpenML database [Vanschoren *et al.*, 2014] contains evaluations for various machine learning methods trained on several datasets with many hyperparameter settings. We consider extreme gradient boosting (XGBoost) [Chen and Guestrin, 2016], one of the most popular OpenML benchmarks, and tune its ten hyperparameters – three are discrete and seven continuous – to minimize the classification error on a held-out test set (without any constraints). We use two datasets, each containing more than 45000 hyperparameter settings. To evaluate hyperparameter settings for which no data is available, we use a surrogate modeling approach based on nearest neighbor [Eggensperger *et al.*, 2015], meaning that the objective returns the error of the closest (w.r.t. Euclidean distance) setting available in the dataset. Fig. 1 shows that MIVABO achieves performance which is either significantly stronger than (left dataset) or competitive with (right dataset) the state-of-the-art mixed-variable BO algorithms on this challenging task. GPyOpt performs poorly, likely because it cannot account for discrete variables in a principled way. As compared to TPE and SMAC, MIVABO seems to benefit from more sophisticated uncertainty estimation.

4.2 Deep Generative Model (DGM) Tuning

DGMs recently received considerable attention in the machine learning community. Despite their popularity and importance, effectively tuning their hyperparameters is a major challenge. We consider tuning the hyperparameters of a variational autoencoder (VAE) [Kingma and Welling, 2014] composed of a convolutional encoder and a deconvolutional decoder [Salimans *et al.*, 2015]. The VAEs are evaluated on stochastically binarized `MNIST`, as in [Burda *et al.*, 2016], and `FashionMNIST`. They are trained on 60000 images for 32 epochs, using Adam with a mini-batch size of 128. We report the negative log-likelihood (NLL; in nats) achieved by the VAEs on a held-out test set of 10000 images, as estimated via importance sampling using 32 samples per test point. To our knowledge, no other BO paper considered DGM tuning.

VAE tuning is difficult due to the high-dimensional and structured nature of its hyperparameter space, and, in particular, due to constraints arising from dependencies between some of its parameters. We tune 25 discrete parameters defining the model architecture, e.g. the number of convolutional layers, their stride, padding and filter size, the number and width of fully-connected layers, and the latent space dimensionality. We further tune three continuous parameters for the optimizer and regularization. Crucially, mutual dependencies between the discrete parameters result in complex constraints, as certain combinations of stride, padding and filter size lead to invalid architectures. Particularly, for the encoder, the shapes of all layers must be integral, and for the decoder, the output shape must match the input data shape, i.e., one channel of size 28×28 for {`Fashion`}`MNIST`. The latter constraint is especially challenging, as only a small number of decoder configurations yield the required output shape. Thus, even for rather simple datasets such as {`Fashion`}`MNIST`, tuning such a VAE is significantly more challenging than, say, tuning a convolutional neural network for classification.

While MIVABO can conveniently capture these restrictions via linear and quadratic constraints, the competing methods cannot. To enable a comparison that is as fair as possible, we thus use the following sensible heuristic to incorporate the knowledge about the constraints into the baselines: If a method tries to evaluate an invalid parameter configuration, we return a penalty error value, which will discourage a model-based method to sample this (or a similar) setting again. However, for fairness, we only report valid observations and ignore all configurations that violated a constraint. We set the penalty value to 500 nats, which is the error incurred by a uniformly random generator. We investigated the impact of the penalty value (e.g., we also tried 250 and 125 nats) and found that it does *not* qualitatively affect the results.

Fig. 3 shows that MIVABO significantly outperforms the competing methods on this task, both on `MNIST` (left) and `FashionMNIST` (right). This is because MIVABO can naturally encode the constraints and thus directly optimize over the feasible region in parameter space, while TPE, SMAC and GPyOpt need to learn the constraints from data. They fail to do so and get stuck in bad local optima early on. The model-based approaches likely struggle due to sharp discontinuities in hyperparameter space induced by the constraint violation penalties (i.e., as invalid configurations may lie close to well-performing configurations). In contrast, random search is agnostic to these discontinuities, and thus notably outperforms the model-based methods. Lastly, GP-SA and MIVABO-SA struggle as well, suggesting that while SA can avoid invalid inputs, the effective optimization of complex constrained objectives crucially requires more principled approaches for acquisition function optimization, such as the one we propose. This shows that all model choices for MIVABO (as discussed in Sec. 3.4) are necessary to achieve such strong results.

[2]We provide a `Python` implementation of MIVABO at `https://github.com/edaxberger/mixed_variable_bo`.

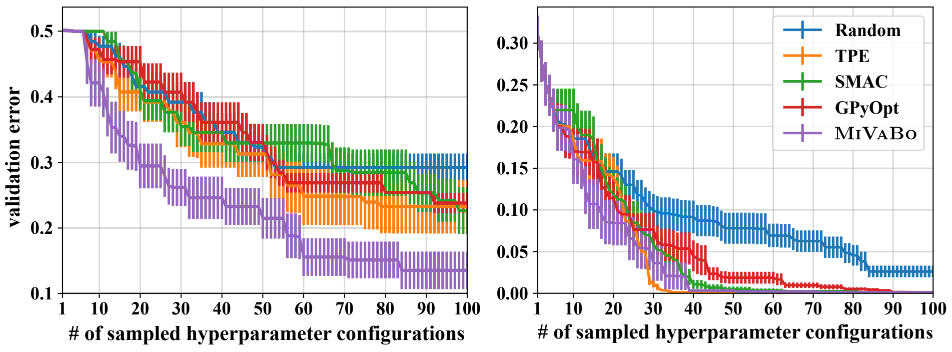

Figure 1: **XGBoost hyperparameter tuning** on `monks-problem-1` (left) and `steel-plates-fault` (right). Mean ± one std. of the validation error over 16 random seeds. MiVaBo significantly outperforms the baselines on the first dataset, and is competitive on the second.

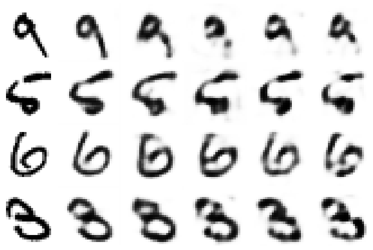

Figure 2: Randomly chosen `MNIST` test images (left column) and their reconstructions by the best VAE models found by MiVaBo, random search, GPyOpt, TPE and SMAC (left to right), thus ordered by NLL values, which seem to capture visual quality.

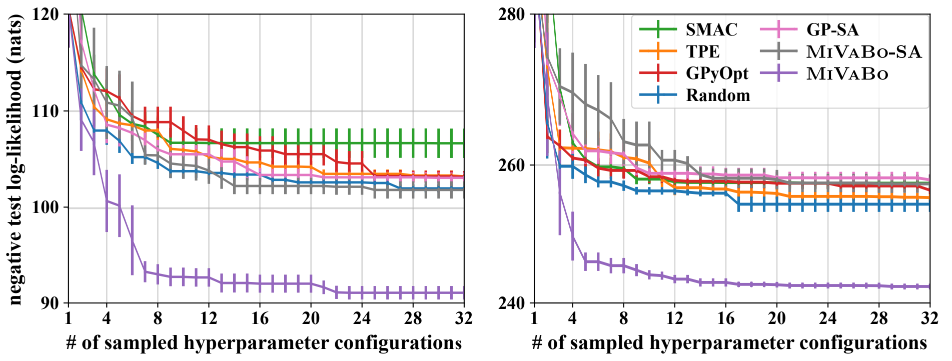

Figure 3: **VAE hyperparameter tuning** on `MNIST` (left) and `FashionMNIST` (right). Mean ± one std. of the NLL in nats, estimated using 32 importance samples, over 8 random seeds. Every model was trained for 32 epochs. MiVaBo significantly outperforms the state-of-the-art baselines, demonstrating its ability to handle the complex constrained nature of the VAE's parameter space.

Method	Time	NLL
SMAC	0.32s	99.09
TPE	0.12s	97.05
GPyOpt	0.65s	97.33
Random	0.01s	93.74
MiVaBo	7.39s	**84.25**

Figure 4: Mean wall-clock time of one iteration (excluding function evaluation time) and mean negative log-likelihood (NLL) in nats, estimated with 5000 importance samples, of the best VAEs found after 32 BO iterations (as in Fig. 3), when trained for 3280 epochs. Human expert baseline for even deeper models is 82-83 nats.

Although log-likelihood scores allow for a quantitative comparison, they are hard to interpret for humans. Thus, for a qualitative comparison, Fig. 2 visualizes the reconstruction quality achieved on `MNIST` by the best VAE configuration found by all methods after 32 BO iterations. The VAEs were trained for 32 epochs each, as in Fig. 3. The likelihoods seem to correlate with the quality of appearance, and the model found by MiVaBo arguably produces the visually most appealing reconstructions among all models. Note that while MiVaBo requires more time than the baselines (see Fig. 4), this is still negligible compared to the cost of a function evaluation, which involves training a deep generative model. Finally, the best VAE found by MiVaBo achieves 84.25 nats on `MNIST` when trained for 3280 epochs and using 5000 importance samples for log-likelihood estimation, i.e. the setting used in [Burda *et al.*, 2016] (see Fig. 4). This is comparable to the performance of 82-83 nats achieved by human expert tuned models, e.g. as reported in [Salimans *et al.*, 2015] (which use even more convolutional layers and a more sophisticated inference method), highlighting MiVaBo's effectiveness in tuning complex deep neural network architectures.

5 Conclusion

We propose MiVaBo, the first method for efficiently optimizing expensive mixed-variable black-box functions subject to linear and quadratic discrete constraints. MiVaBo combines a linear model of expressive features with Thompson sampling, making it simple yet effective. Moreover, it is highly flexible due to the modularity of its components, i.e., the mixed-variable features, and the optimization oracles for the acquisition procedure. This allows practitioners to tailor MiVaBo to specific objectives, e.g. by incorporating prior knowledge in the feature design or by leveraging optimizers handling specific types of constraints. We show that MiVaBo enjoys theoretical convergence guarantees that competing methods lack. Finally, we empirically demonstrate that MiVaBo significantly improves optimization performance as compared to state-of-the-art methods for mixed-variable optimization on complex hyperparameter tuning tasks.

Acknowledgements

This research has been partially supported by SNSF NFP75 grant 407540_167189. Matteo Turchetta was supported through the ETH-MPI Center for Learning Systems. Erik Daxberger was supported through the EPSRC and Qualcomm. The authors thank Josip Djolonga, Mojmír Mutný, Johannes Kirschner, Alonso Marco Valle, David R. Burt, Ross Clarke, Wolfgang Roth as well as the anonymous reviewers of an earlier version of this paper for their helpful feedback.

References

[Abeille *et al.*, 2017] Marc Abeille, Alessandro Lazaric, et al. Linear Thompson sampling revisited. *EJS*, 2017.

[Baptista and Poloczek, 2018] Ricardo Baptista and Matthias Poloczek. Bayesian optimization of combinatorial structures. In *ICML*, 2018.

[Bergstra *et al.*, 2011] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *NIPS*, 2011.

[Boros and Hammer, 2002] Endre Boros and Peter L Hammer. Pseudo-boolean optimization. *Discrete applied mathematics*, 123(1-3):155–225, 2002.

[Burda *et al.*, 2016] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. In *ICLR*, 2016.

[Chen and Guestrin, 2016] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *KDD*, 2016.

[Eggensperger *et al.*, 2015] Katharina Eggensperger, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Efficient benchmarking of hyperparameter optimizers via surrogates. In *AAAI*, 2015.

[Falkner *et al.*, 2018] Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In *ICML*, 2018.

[Gardner *et al.*, 2014] Jacob R. Gardner, Matt J. Kusner, Zhixiang Xu, Kilian Q. Weinberger, and John P. Cunningham. Bayesian optimization with inequality constraints. In *ICML*, 2014.

[Garrido-Merchán and Hernández-Lobato, 2018] Eduardo C Garrido-Merchán and Daniel Hernández-Lobato. Dealing with categorical and integer-valued variables in bayesian optimization with gaussian processes. *CoRR*, 2018.

[Hastie, 2017] Trevor J Hastie. Generalized additive models. In *Statistical models in S*. Routledge, 2017.

[Hazan *et al.*, 2017] Elad Hazan, Adam Klivans, and Yang Yuan. Hyperparameter optimization: A spectral approach. In *ICRL*, 2017.

[Hernández-Lobato *et al.*, 2015] José Miguel Hernández-Lobato, Michael A Gelbart, Matthew W Hoffman, Ryan P Adams, and Zoubin Ghahramani. Predictive entropy search for bayesian optimization with unknown constraints. *JMLR*, 2015.

[Hutter *et al.*, 2011] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *LION*, 2011.

[Jenatton *et al.*, 2017] Rodolphe Jenatton, Cedric Archambeau, Javier González, and Matthias Seeger. Bayesian optimization with tree-structured dependencies. In *ICML*, 2017.

[Kingma and Welling, 2014] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[Krause *et al.*, 2008] Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *JMLR*, 2008.

[Li *et al.*, 2018] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *JMLR*, 2018.

[Močkus, 1975] Jonas Močkus. On Bayesian methods for seeking the extremum. In *Optimization Techniques*, 1975.

[Mutný and Krause, 2018] Mojmir Mutný and Andreas Krause. Efficient high dimensional bayesian optimization with additivity and quadrature fourier features. In *NIPS*, 2018.

[Negoescu *et al.*, 2011] Diana M Negoescu, Peter I Frazier, and Warren B Powell. The knowledge-gradient algorithm for sequencing experiments in drug discovery. *INFORMS*, 2011.

[Oh *et al.*, 2019] Changyong Oh, Jakub M. Tomczak, Efstratios Gavves, and Max Welling. Combinatorial bayesian optimization using graph representations. *NeurIPS*, 2019.

[Rahimi and Recht, 2008] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *NIPS*, 2008.

[Rolland *et al.*, 2018] Paul Rolland, Jonathan Scarlett, Ilija Bogunovic, and Volkan Cevher. High-dimensional Bayesian optimization via additive models with overlapping groups. In *AISTATS*, 2018.

[Salimans *et al.*, 2015] Tim Salimans, Diederik P Kingma, and Max Welling. Markov chain monte carlo and variational inference: Bridging the gap. In *ICML*, 2015.

[Shahriari *et al.*, 2016] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *IEEE*, 2016.

[Sontag *et al.*, 2011] David Sontag, Amir Globerson, and Tommi Jaakkola. Introduction to dual composition for inference. In *Optimization for Machine Learning*. 2011.

[Srinivas *et al.*, 2010] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *ICML*, 2010.

[Sui *et al.*, 2015] Yanan Sui, Alkis Gotovos, Joel Burdick, and Andreas Krause. Safe exploration for optimization with gaussian processes. In *ICML*, 2015.

[Thompson, 1933] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 1933.

[Vanschoren *et al.*, 2014] Joaquin Vanschoren, Jan N Van Rijn, Bernd Bischl, and Luis Torgo. OpenML: networked science in machine learning. *ACM SIGKDD*, 15(2):49–60, 2014.

[Williams and Rasmussen, 2006] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*. MIT Press Cambridge, MA, 2006.