Bayesian Optimization of Combinatorial Structures

Ricardo Baptista [1] **Matthias Poloczek** [2]

Abstract

The optimization of expensive-to-evaluate black-box functions over combinatorial structures is an ubiquitous task in machine learning, engineering and the natural sciences. The combinatorial explosion of the search space and costly evaluations pose challenges for current techniques in discrete optimization and machine learning, and critically require new algorithmic ideas. This article proposes, to the best of our knowledge, the first algorithm to overcome these challenges, based on an adaptive, scalable model that identifies useful combinatorial structure even when data is scarce. Our acquisition function pioneers the use of semidefinite programming to achieve efficiency and scalability. Experimental evaluations demonstrate that this algorithm consistently outperforms other methods from combinatorial and Bayesian optimization.

1. Introduction

We consider the problem of optimizing an expensive-to-evaluate black-box function over a set of combinatorial structures. This problem is pervasive in machine learning, engineering, and the natural sciences. Applications include object location in images (Zhang et al., 2015), drug discovery (Negoescu et al., 2011), cross-validation of hyperparameters in mixed-integer solvers (Hutter et al., 2010), food safety control (Hu et al., 2010), and model sparsification in multi-component systems (Baptista et al., 2018). We also face this problem in the shared economy: a bike sharing company has to decide where to place bike stations among locations offered by the communal administration in order to optimize its utility, as measured in a field test.

[*]Equal contribution [1]Center for Computational Engineering, Massachusetts Institute of Technology, Cambridge, MA [2]Dept. of Systems and Industrial Engineering, The University of Arizona, Tucson, AZ. Correspondence to: Ricardo Baptista <rsb@mit.edu>, Matthias Poloczek <poloczek@email.arizona.edu>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

We present a novel algorithm for this problem, *Bayesian Optimization of Combinatorial Structures* (`BOCS`), that is capable of taming the combinatorial explosion of the search domain while achieving sample-efficiency, thereby improving substantially over the state of the art. Specifically, our contributions are:

1. A novel method to obtain an approximate optimizer of the acquisition function that employs algorithmic ideas from convex optimization to achieve scalability and efficiency. This approach overcomes the inherent limited scalability of many acquisition functions to large combinatorial domains.

2. We propose a model that captures the interaction of structural elements, and show how to infer these interactions in practice when data is expensive and thus scarce. We also demonstrate the usefulness of this interpretable model on experimental data.

3. We evaluate the performance of the `BOCS` algorithm together with methods from machine learning and discrete optimization on a variety of benchmark problems, including tasks from machine learning, aerospace engineering, and food safety control.

Related Work: Bayesian optimization has emerged as a powerful technique for the optimization of expensive functions if the domain is a box-constrained subset of the real coordinate space, i.e., a tensor-product of bounded connected univariate domains (e.g., see Brochu et al. (2010); Shahriari et al. (2016b)). Recently, these techniques have been extended to certain high-dimensional problems whose box-constrained sets have a low 'effective dimensionality' (Wang et al., 2016; Binois et al., 2017) or an additive decomposition (Kandasamy et al., 2015; Li et al., 2016; Wang et al., 2017). Hutter & Osborne (2013) and Swersky et al. (2014) considered applications where parameters have conditional dependencies, becoming irrelevant if other parameters take certain values. Jenatton et al. (2017) presented a scalable algorithm when these dependencies form a tree. Shahriari et al. (2016a) proposed techniques for growing the box adaptively to optimize over unbounded continuous domains.

Structured domains have received little attention. Negoescu et al. (2011) proposed a linear parametric belief model that

can handle categorical variables. Their streamlined knowledge gradient acquisition function has cost $\Omega(d \cdot 2^d)$ for each iteration and thus is designed for applications with small dimensionality d. Hutter et al. (2011) suggested a novel surrogate model based on random forests to handle categorical data. Their SMAC algorithm uses random walks to obtain a local optimum under the expected improvement acquisition criterion (Mockus et al., 1978; Jones et al., 1998), and therefore can handle even high dimensional problems. In practice, structured domains are often embedded into a box in $\mathbb{R}^d$ to run an off-the-shelf Bayesian optimization software, e.g., see (Dewancker et al., 2016; Golovin et al., 2017). However, this is typically infeasible in practice due to the curse of dimensionality, also referred to as *combinatorial explosion*, as the number of alternatives grows exponentially with the parameters. Thus, it is not surprising that optimization over structured domains was raised as an important open problem at the NIPS 2017 Workshop on Bayesian optimization (Hernández-Lobato et al., 2017).

Methods in discrete optimization that are able to handle black-box functions include local search (Khanna et al., 1998; Selman et al., 1993; Spears, 1993) and evolutionary algorithms, e.g., particle search (Schäfer, 2013). However, these procedures are not designed to be sample efficient and hence often prohibitively expensive for the problems we consider. Moreover, the popular local search algorithms have the conceptual disadvantage that they do not necessarily converge to a global optimum. Popular techniques such as branch and bound and mathematical programming, e.g., linear, convex, and mixed-integer programming, typically cannot be applied to black-box functions. We will compare the BOCS algorithm to the methods of (Snoek et al., 2012; Hutter et al., 2011; Khanna et al., 1998; Spears, 1993; Bergstra & Bengio, 2012; Schäfer, 2013) in Sect. 4.

We formalize the problem under consideration in Sect. 2, describe the statistical model in Sect. 3.1, specify our acquisition function and the relaxation to semidefinite programming in Sects. 3.2 and 3.3, present numerical experiments in Sect. 4, and conclude in Sect. 5. Sections labeled by letters are in the supplement.

The code for this paper is available at https://github.com/baptistar/BOCS.

2. Problem Formulation

Given an expensive-to-evaluate black-box function f over a *discrete structured domain* $\mathcal{D}$ of feasible points, our goal is to find a *global optimizer* $\operatorname{argmax}_{x \in \mathcal{D}} f(x)$. We suppose that observing x provides independent, conditional on $f(x)$, and normally distributed observations with mean $f(x)$ and finite variance σ^2. For the sake of simplicity, we focus on $\mathcal{D} = \{0,1\}^d$, where x_i equals one if a certain element i is present in the design and zero otherwise. For example, we can associate a binary variable with each possible location for a bike station, with a side-chain in a chemical compound, with a possible coupling between two components in a multi-component system, or more generally with an edge in a graph-like structure. We note that BOCS generalizes to integer-valued and categorical variables and to models of higher order (see Sect. A).

3. The BOCS Algorithm

We now present the BOCS algorithm for combinatorial structures and describe its two components: a model tailored to combinatorial domains in Sect. 3.1 and its acquisition function in Sect. 3.2. Sect. 3.3 summarizes the algorithm and Sect. 3.4 presents the variant BOCS-SA. The time complexity is analyzed in Sect. 3.5.

3.1. Statistical Model

When developing a generative model for an expensive function $f(x) : \mathcal{D} \to \mathbb{R}$ defined on a combinatorial domain, it seems essential to model the interplay of elements. For example, in the above bike sharing application, the utility of placing a station at some location depends critically on the presence of other stations. Similarly, the absorption of a medical drug depends on the combination of functional groups in the molecule. A general model for f is thus given by $\sum_{S \in 2^{\mathcal{D}}} \alpha_S \prod_{i \in S} x_i$, where $2^{\mathcal{D}}$ is the power set of the domain and α_S is a real-valued coefficient. Clearly, this model is impractical due to the exponential number of monomials. Thus, we consider restricted models that contain monomials up to order k. A higher order increases the expressiveness of the model but also decreases the accuracy of the predictions when data is limited (e.g., see Ch. 14.6 in Gelman et al. (2013)). We found that second-order models provide an excellent trade-off in practice (cp. Sect. 4 and B). Thus, under our model, x has objective value

$$f_{\boldsymbol{\alpha}}(x) = \alpha_0 + \sum_j \alpha_j x_j + \sum_{i,j>i} \alpha_{ij} x_i x_j. \qquad (1)$$

While the so-called *interaction terms* are quadratic in $x \in \mathcal{D}$, the regression model is linear in $\boldsymbol{\alpha} = (\alpha_i, \alpha_{ij}) \in \mathbb{R}^p$ with $p = 1 + d + \binom{d}{2}$.

Sparse Bayesian Linear Regression: To quantify the uncertainty in the model, we propose a Bayesian treatment for $\boldsymbol{\alpha}$. For observations $(x^{(i)}, y^{(i)}(x^{(i)}))$ with $i = 1, \ldots, N$, let $\mathbf{X} \in \{0,1\}^{N \times p}$ be the matrix of predictors and $\mathbf{y} \in \mathbb{R}^N$ the vector of corresponding observations of f. Using the data model, $y^{(i)}(x^{(i)}) = f(x^{(i)}) + \varepsilon^{(i)}$ where $\varepsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$, we have $\mathbf{y} \mid \mathbf{X}, \boldsymbol{\alpha}, \sigma^2 \sim \mathcal{N}(\mathbf{X}\boldsymbol{\alpha}, \sigma^2 I_N)$.

One drawback of using a second-order model is that it has $\Theta(d^2)$ regression coefficients which may result in

high-variance estimators for the coefficients if data is scarce. To assert a good performance even for high-dimensional problems with expensive evaluations, we employ a *sparsity-inducing prior*. We use the heavy-tailed horseshoe prior (Carvalho et al., 2010):

$$\begin{aligned}\alpha_k \mid \beta_k^2, \tau^2, \sigma^2 &\sim \mathcal{N}(0, \beta_k^2\tau^2\sigma^2) \quad k = 1, \ldots, p \\ \tau, \beta_k &\sim \mathcal{C}^+(0,1) \quad k = 1, \ldots, p \\ P(\sigma^2) &= \sigma^{-2},\end{aligned}$$

where $\mathcal{C}^+(0,1)$ is the standard half-Cauchy distribution. In this model, the global, τ, and the local, β_k, hyper-parameters individually shrink the magnitude of each regression coefficient. Following Makalic & Schmidt (2016), we introduce the auxiliary variables ν and ξ to re-parameterize the half-Cauchy densities using inverse-gamma distributions. Then the conditional posterior distributions for the parameters are given by

$$\begin{aligned}
&\boldsymbol{\alpha}|\cdot \sim \mathcal{N}(\mathbf{A}^{-1}\mathbf{X}^T\mathbf{y}, \sigma^2\mathbf{A}^{-1}), \qquad (2)\\
&\mathbf{A} = (\mathbf{X}^T\mathbf{X} + \Sigma_*^{-1}), \Sigma_* = \tau^2\text{diag}(\beta_1^2, \ldots, \beta_p^2)\\
&\sigma^2|\cdot \sim IG\left(\frac{N+p}{2}, \frac{(\mathbf{y}-\mathbf{X}\boldsymbol{\alpha})^T(\mathbf{y}-\mathbf{X}\boldsymbol{\alpha})}{2} + \frac{\boldsymbol{\alpha}^T\Sigma_*^{-1}\boldsymbol{\alpha}}{2}\right)\\
&\beta_k^2|\cdot \sim IG\left(1, \frac{1}{\nu_k} + \frac{\alpha_k^2}{2\tau^2\sigma^2}\right) \quad k = 1, \ldots, p\\
&\tau^2|\cdot \sim IG\left(\frac{p+1}{2}, \frac{1}{\xi} + \frac{1}{2\sigma^2}\sum_{k=1}^{p}\frac{\alpha_k^2}{\beta_k^2}\right)\\
&\nu_k|\cdot \sim IG\left(1, 1 + \frac{1}{\beta_k^2}\right) \quad k = 1, \ldots, p\\
&\xi|\cdot \sim IG\left(1, 1 + \frac{1}{\tau^2}\right).
\end{aligned}$$

Given these closed-form conditionals, we employ a Gibbs sampler to efficiently sample from the posterior over $\boldsymbol{\alpha}$. The complexity of sampling $\boldsymbol{\alpha}$ is dominated by the cost of sampling from the multivariate Gaussian. This step has cost $\mathcal{O}(p^3)$ for a naïve implementation and hence can be prohibitive for a large number of predictors. Instead we use the exact sampling algorithm of Bhattacharya et al. (2016) whose complexity is $\mathcal{O}(N^2p)$ and therefore nearly linear in p whenever $N \ll p$. We have evaluated the different approaches and found that the proposed sparse regression performs well for several problems (see Sect. F in the supplement for details).

We note that if the statistical model for f is based on a maximum likelihood estimate (MLE) for $\boldsymbol{\alpha}$ (see Sect. E), the algorithm would exhibit a purely exploitative behavior and produce sub-optimal solutions. Thus, it seems essential to account for the uncertainty in the model for the objective, which is accomplished by sampling the model parameters from the posterior over $\boldsymbol{\alpha}$ and σ^2.

3.2. Acquisition Function

The role of the acquisition function is to select the next sample point in every iteration. Ours is inspired by Thompson sampling (Thompson, 1933; 1935) (also see the excellent survey of Russo et al. (2017)) that samples a point x with probability proportional to x being an optimizer of the unknown function. We proceed as follows. Keeping in mind that our belief on the objective f at any iteration is given by the posterior on $\boldsymbol{\alpha}$, we sample $\boldsymbol{\alpha}_t \sim P(\boldsymbol{\alpha} \mid \mathbf{X}, \mathbf{y})$ and want to find an $\operatorname{argmax}_{x\in\mathcal{D}} f_{\boldsymbol{\alpha}_t}(x)$. Since applications often impose some form of regularization on x, we restate the problem as $\operatorname{argmax}_{x\in\mathcal{D}} f_{\boldsymbol{\alpha}}(x) - \lambda\mathcal{P}(x)$, where $\mathcal{P}(x) = \|x\|_1$ or $\mathcal{P}(x) = \|x\|_2^2$ and thus cheap to evaluate. Then, for a given $\boldsymbol{\alpha}$ and $\mathcal{P}(x) = \|x\|_1$, the problem is to obtain an

$$\begin{aligned}
&\operatorname*{argmax}_{x\in\mathcal{D}} f_{\boldsymbol{\alpha}}(x) - \lambda\mathcal{P}(x) \qquad (3)\\
&= \operatorname*{argmax}_{x\in\mathcal{D}} \sum_j (\alpha_j - \lambda)x_j + \sum_{i,j>i}\alpha_{ij}x_ix_j, \qquad (4)
\end{aligned}$$

where $x \in \{0,1\}^d$. Similarly, if $\mathcal{P}(x) = \|x\|_2^2$, the problem becomes $\operatorname{argmax}_{x\in\mathcal{D}} \sum_j \alpha_j x_j + \sum_{i,j>i}(\alpha_{ij} - \lambda\delta_{ij})x_ix_j$. Thus, in both cases we are to solve a binary quadratic program of the form

$$\operatorname*{argmax}_{x\in\mathcal{D}} x^TAx + b^Tx, \qquad (5)$$

where $\mathcal{D} = \{0,1\}^d$. That is, we are to optimize a quadratic form over the vertices of the d-dimensional hypercube $\mathcal{H}_d$. This problem is known to be notoriously hard, not admitting exact solutions in polynomial time unless $\mathsf{P} = \mathsf{NP}$ (Garey & Johnson, 1979; Charikar & Wirth, 2004).

Outline: We will show how to obtain an approximation efficiently. First, we relax the quadratic program into a vector program, replacing the binary variables by high-dimensional vector-valued variables on the $(d+1)$-dimensional unit sphere $\mathcal{S}_d$. Note that the optimum value attainable for this convex relaxation is at least as large as the optimum of Eq. (5). We then rewrite this vector program as a semidefinite program (SDP) that can be approximated in polynomial time to a desired precision (Steurer, 2010; Arora & Kale, 2016; Boyd & Vandenberghe, 2004). The solution to the SDP is converted back into a collection of vectors. Finally, we apply the randomized rounding method of Charikar & Wirth (2004) to obtain a solution in $\mathcal{D}$. We found that this procedure often produces an $x^{(t)}$ that is (near)-optimal. Moreover, it has a robustness guarantee in the sense that the approximation error never deviates more than $\mathcal{O}(\log d)$ from the optimum. (This bound requires that α_0 does not carry a negative weight that is large in absolute value compared to the optimal value of the SDP, see (Charikar & Wirth, 2004) for details.) Note that the worst case guarantee is essentially the best possible under standard complexity assumptions (Arora et al., 2005).

To convert the input domain to $\{-1,1\}^d$, we replace each variable x_i by $y_i = 2x_i - 1$ and accordingly adapt the coefficients by defining $\tilde{A} = A/4$, $c = b/4 + A^T\mathbf{1}/4$, and

$$B = \begin{bmatrix} \tilde{A} & c \\ c^T & 0 \end{bmatrix}.$$

Augmenting the state with an additional variable y_0, we can rewrite (5) as the quadratic program $\operatorname{argmax}_z \; z^T B z$ with $z{=}[y,\, y_0] \in \{-1,1\}^{d+1}$. Thus, replacing y_i by the real-valued vector-variable $\nu_i \in \mathcal{S}_d$ for $0{\le}i{\le}d$ we obtain the relaxation $\operatorname{argmax} \sum_{i,j} B_{i,j}\langle \nu_i, \nu_j\rangle$. This vector program is equivalent to the SDP

$$\underset{Z \succeq 0}{\operatorname{argmax}} \; \operatorname{Tr}(B^T Z) \quad \text{s.t. } \operatorname{diag}(Z) = \mathrm{I}_{d+1}, \tag{6}$$

where Z is a symmetric $(d{+}1){\times}(d{+}1)$ real matrix.

First we obtain a solution Z^* to (6) that is then (approximately) factorized as $Z^* = (V^*)^T V^*$, where $V^* \in \mathbb{R}^{(d+1)\times(d+1)}$ contains column vectors that satisfy the constraints $\|V_i^*\|_2 = 1$ (i.e., $V_i^* \in \mathcal{S}_d$). Then, drawing a random vector $r \in \mathbb{R}^{d+1}$ with independent standard Gaussian entries, we apply the randomized geometric rounding procedure of Charikar & Wirth (2004) to obtain an approximate solution $y^* \in \{-1,1\}^d$ to the original quadratic program. Lastly, we apply the inverse transformation $x_i^* = (y_i^* + 1)/2$ to recover a solution on the d-dimensional hypercube.

3.3. Summary of the BOCS Algorithm

We now summarize the BOCS algorithm. Using an initial dataset of N_0 samples, it first computes the posterior on f based on the sparsity-inducing prior.

In the optimization phase, BOCS proceeds in iterations until the sample budget N_{max} is exhausted. In iteration $t = 1, 2, \ldots$, it samples the vector $\boldsymbol{\alpha}_t$ from the posterior over the regression coefficients that is defined by the parameters in Eq. (2). Now BOCS computes an approximate solution $x^{(t)}$ for $\max_{x\in\{0,1\}^d} f_{\boldsymbol{\alpha}_t} - \lambda\mathcal{P}(x)$ as follows: first it transforms the quadratic model into an SDP, thereby relaxing the variables into vector-valued variables on the $(d+1)$-dimensional unit-sphere. This SDP is solved (with a predescribed precision) and the next point $x^{(t)}$ is obtained by rounding the vector-valued SDP solution. The iteration ends after the posterior is updated with the new observation $y^{(t)}$ at $x^{(t)}$. BOCS is summarized as Algorithm 1.

3.4. BOCS-SA: A Low-Complexity Variant of BOCS

We propose a variant of BOCS that replaces semidefinite programming by stochastic local search. In our experimental evaluation, the solver takes only a few seconds to obtain a solution to the semidefinite program and will scale easily to a few hundred dimensions. While semidefinite programs can be approximated to a given precision in polynomial time, their complexity might become a bottleneck in future applications when the dimensionality grows large. Therefore, we also investigated alternative techniques to solve the problem in (5) and have found good performance with stochastic local search, specifically with simulated annealing.

Algorithm 1 Bayesian Optimization of Combinatorial Structures

Input: Objective function $f(x) - \lambda\mathcal{P}(x)$; Sample budget N_{max}; Size of initial dataset N_0.
Sample initial dataset D_0.
Compute the posterior on $\boldsymbol{\alpha}$ given the prior and D_0.
for $t = 1$ **to** $N_{max} - N_0$ **do**
  Sample coefficients $\boldsymbol{\alpha}_t \sim P(\boldsymbol{\alpha} \mid \mathbf{X}, \mathbf{y})$.
  Find approximate solution $x^{(t)}$ for $\max_{x\in\mathcal{D}} f_{\boldsymbol{\alpha}_t}(x) - \lambda\mathcal{P}(x)$.
  Evaluate $f(x^{(t)})$ and append the observation $y^{(t)}$ to $\mathbf{y}$.
  Update the posterior $P(\boldsymbol{\alpha} \mid \mathbf{X}, \mathbf{y})$.
end for
return $\operatorname{argmax}_{x\in\mathcal{D}} f_{\boldsymbol{\alpha}_t}(x) - \lambda\mathcal{P}(x)$.

Simulated annealing (SA) performs a random walk on $\mathcal{D}$, starting from a point chosen uniformly at random. Let $x^{(t)}$ be the point selected in iteration t. Then the next point $x^{(t+1)}$ is selected in the *neighborhood* $N(x^{(t)})$ that contains all points with Hamming distance at most one from $x^{(t)}$. SA picks $x \in N(x^{(t)})$ uniformly at random and evaluates $\mathrm{obj}(x)$: If the observed objective value is better than the observation for $x^{(t)}$, SA sets $x^{(t+1)} = x$. Otherwise, the point is adopted with probability $\exp((\mathrm{obj}(x) - \mathrm{obj}(x^{(t)}))/T_{t+1})$, where T_{t+1} is the current temperature. SA starts with a high T that encourages exploration and cools down over time to zoom in on a good solution.

In what follows, BOCS-SDP denotes the implementation of BOCS that leverages semidefinite programming. The implementation that uses simulated annealing (SA) is denoted by BOCS-SA.

3.5. Time Complexity

Recall from Sect. 3.1 that the computational cost of sampling from the posterior over $\boldsymbol{\alpha}$ and σ^2 is bounded by $\mathcal{O}(N^2 p)$, where $p = \Theta(d^2)$ for the second-order model and N is the number of samples seen so far. The acquisition function is asymptotically dominated by the cost of the SDP solver, which is polynomial in d for a given precision ε. Therefore, the total running time of a single iteration of BOCS-SDP is bounded by $\mathcal{O}(N^2 d^2 + poly(d, \frac{1}{\varepsilon}))$.

A single iteration of BOCS-SA on the other hand has time complexity $\mathcal{O}(N^2 d^2)$, since simulated annealing runs in $\mathcal{O}(d^2)$ steps for the temperature schedule of Spears

(1993). We point out that the number of alternatives is exponential in d, thus the running times of BOCS-SDP and BOCS-SA are only *logarithmic* in the size of the domain that we optimize over.

4. Numerical Results

We conduct experiments on the following benchmarks: (1) binary quadratic programming with $d = 10$ variables (Sect. 4.1), (2) sparsification of Ising models with $d = 24$ edges (Sect. 4.2), (3) contamination control of a food supply chain with $d = 25$ stages (Sect. 4.3), (4) complexity reduction of an aero-structural multi-component system with $d = 21$ coupling variables (Sect. 4.4). We evaluate the variants of the BOCS algorithm described in Sect. 3 and compare them to the following methods from machine learning and combinatorial optimization.

Expected improvement (EI) with a Gaussian process based model (Jones et al., 1998; Snoek et al., 2012) typically performs well for noise-free functions. EI uses the popular one-hot encoding that coincides with the tailored kernel of Hutter (2009) for binary variables. Although the computational cost for selecting the next candidate point is relatively low compared to other acquisition functions, EI is considerably more expensive than the other methods (see also Sect. C). SMAC (Hutter et al., 2011) addresses this problem by performing a local search for a candidate with high expected improvement. It uses a random forest-based model that is able to handle categorical and integer-valued variables.

Sequential Monte Carlo particle search (PS) (Schäfer, 2013) is an evolutionary algorithm that maintains a population of candidate solutions. PS is robust to multi-modality and often outperforms local search and simulated annealing for combinatorial domains (Del Moral et al., 2006; Schäfer, 2013).

Simulated annealing (SA) is known for its excellent performance on hard combinatorial problems (Spears, 1993; Pankratov & Borodin, 2010; Poloczek & Williamson, 2017).

Starting at a randomly chosen point, *oblivious local search* (OLS) (Khanna et al., 1998) evaluates in every iteration all points with Hamming distance one from its current point and adopts the best. We are interested in the search performance of OLS relative to its sample complexity. At each iteration, OLS requires d function evaluations to search within the neighborhood of the current solution. We also compare to *random search* (RS) of Bergstra & Bengio (2012).

We report the function value returned after t evaluations, averaged over at least 100 runs of each algorithm for the first three problems. Intervals stated in tables and error bars in plots give the mean $\pm$ 2 standard errors. Bold entries in tables highlight the best mean performance for each choice of λ. BOCS and EI are given identical initial datasets in every replication. These datasets were drawn via Monte Carlo sampling. Algorithms that do not take an initial dataset are allowed an equal number of 'free' steps before counting their function evaluations. The implementations of the above algorithms and the variants of BOCS are available at https://github.com/baptistar/BOCS.

4.1. Binary Quadratic Programming

The objective in the binary quadratic programming problem (BQP) is to maximize a quadratic function with ℓ_1 regularization, $f(x) - \lambda\mathcal{P}(x) = x^T Q x - \lambda\|x\|_1$, over $\{0,1\}^d$. $Q \in \mathbb{R}^{d\times d}$ is a random matrix with independent standard Gaussian entries that is multiplied element-wise by a matrix $K \in \mathbb{R}^{d\times d}$ with entries $K_{ij} = \exp(-(i-j)^2/L_c^2)$. The entries of K decay smoothly away from the diagonal with a rate determined by the correlation length L_c^2. We note that, as the correlation length increases, Q changes from a nearly diagonal to a denser matrix, making the optimization more challenging. We set $d = 10$, sampled 50 independent realizations for Q, and ran every algorithm 10-times on each instance with different initial datasets. Bayesian optimization algorithms received identical initial datasets of size $N_0 = 20$. Recall the performance at step t of the other algorithms (i.e., SA, OLS, and RS) corresponds to the $(t+N_0)$-th function evaluation. For $\lambda = 0$ and $L_c = 10$, Fig. 1 reports the simple regret after step t , i.e., the absolute difference between the global optimum and the solution returned by the respective algorithm.

We see that both variants of BOCS perform significantly better than the competitors. BOCS-SDP and the variant BOCS-SA based on stochastic local search are close with the best performance. EI and SA make progress slowly, whereas the other methods are clearly distanced. When considering the performance of OLS, we note that a deterministic search over a 1-flip neighborhood seems to make progress, but is eventually stuck in local optima. Similarly, MLE plateaus quickly. We discuss this phenomenon below.

We also studied the performance for L_c=100 and λ=1, see Fig. 2. Again, BOCS-SDP performs substantially better than the other algorithms, followed by BOCS-SA. Table 1 compares the performances of EI and BOCS across other settings of L_c and λ. MLE is derived from BOCS-SA by setting the regression weights to a maximum likelihood estimate (see Sect. E). We witnessed a purely exploitative behavior of this algorithm and an inferior performance that seems to plateau. This underlines the importance of sampling from the posterior of the regression weights, which enables the algorithm to explore the model space, resulting in significantly better performance.

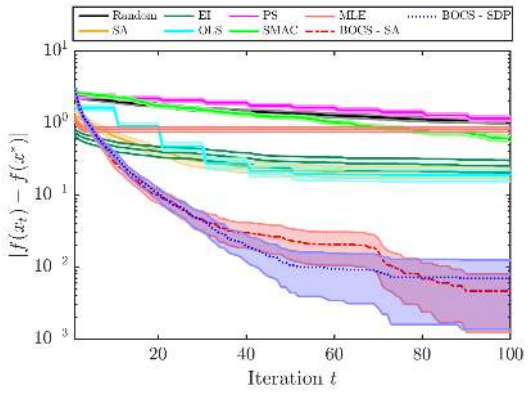

Figure 1. Random BQP instances with $L_c = 10$ and $\lambda = 0$: Both variants of BOCS outperform the competitors substantially.

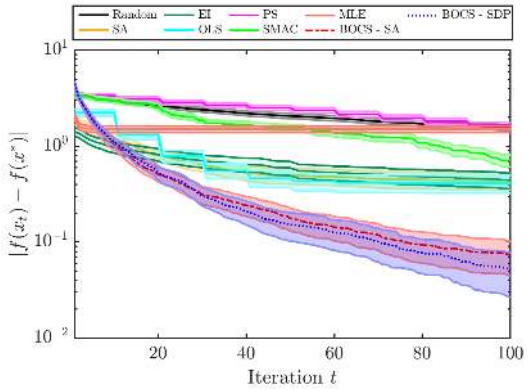

Figure 2. Random BQP instances with L_c=100 and λ=1: Both versions of BOCS outperform the other methods.

Table 1. The simple regret after 100 iterations for 10-dimensional BQP instances. The entries have been multiplied by 10. The best performance for each setting is set in bold.

(L_c, λ)	EI	BOCS-SA	BOCS-SDP
$(1, 0)$	0.49 ± 0.13	**0.02** ± 0.02	0.03 ± 0.02
$(1, 10^{-4})$	0.50 ± 0.12	**0.02** ± 0.01	0.03 ± 0.03
$(1, 10^{-2})$	0.54 ± 0.12	**0.02** ± 0.02	0.05 ± 0.05
$(10, 0)$	2.54 ± 0.51	**0.07** ± 0.05	0.07 ± 0.05
$(10, 10^{-4})$	2.49 ± 0.44	**0.06** ± 0.04	0.08 ± 0.05
$(10, 10^{-2})$	2.27 ± 0.40	**0.04** ± 0.04	0.10 ± 0.06
$(100, 0)$	3.38 ± 0.70	0.15 ± 0.07	**0.11** ± 0.06
$(100, 10^{-4})$	4.07 ± 0.77	0.16 ± 0.08	**0.15** ± 0.08
$(100, 10^{-2})$	4.25 ± 0.78	0.17 ± 0.09	**0.13** ± 0.07

4.2. Sparsification of Ising Models

We consider zero-field Ising models that admit a distribution $p(z) = \frac{1}{Z^p} \exp(z^T J^p z)$ for $z \in \{-1, 1\}^n$, where $J^p \in \mathbb{R}^{n \times n}$ is a symmetric interaction matrix and Z^p is the partition function. The support of the matrix J^p is represented by a graph $\mathcal{G}^p = ([n], \mathcal{E}^p)$ that satisfies $(i, j) \in \mathcal{E}^p$ if and only if $J^p_{ij} \neq 0$ holds.

Table 2. Sparsification of Ising models: BOCS-SDP obtains the best function values for all three settings of λ, here measured after 150 iterations. We also note that the BOCS-SDP algorithm has the lowest variability over 10 random Ising models.

λ	SA	EI	OLS
0	0.21 ± 0.05	0.20 ± 0.04	0.54 ± 0.09
10^{-4}	0.23 ± 0.05	0.20 ± 0.04	0.49 ± 0.08
10^{-2}	0.39 ± 0.05	0.39 ± 0.04	0.67 ± 0.09
λ	MLE-SA	BOCS-SA	BOCS-SDP
0	1.19 ± 0.11	0.19 ± 0.04	**0.11** ± 0.04
10^{-4}	1.21 ± 0.11	0.19 ± 0.04	**0.10** ± 0.03
10^{-2}	1.37 ± 0.11	0.37 ± 0.04	**0.33** ± 0.04

Given $p(z)$, the objective is to find a close approximating distribution $q_x(z) = \frac{1}{Z^q} \exp(z^T J^q z)$ while minimizing the number of edges in $\mathcal{E}^q$. We introduce variables $x \in \{0, 1\}^{|\mathcal{E}^p|}$ that indicate if each edge is present in $\mathcal{E}^q$ and set the edge weights as $J^q_{ij} = x_{ij} J^p_{ij}$. The distance between $p(z)$ and $q_x(z)$ is measured by the Kullback-Leibler (KL) divergence

$$D_{KL}(p||q_x) = \sum_{(i,j) \in \mathcal{E}^p} (J^p_{ij} - J^q_{ij}) \mathbb{E}_p[z_i z_j] + \log\left(\frac{Z_q}{Z_p}\right).$$

Note that the cost of computing the ratio of the partition functions grows exponentially in n, which makes the KL divergence an expensive-to-evaluate function. Summing up, the goal is to obtain an $\operatorname{argmin}_{x \in \{0,1\}^d} D_{KL}(p||q_x) + \lambda \|x\|_1$. The experimental setup consists of 4×4 zero-field Ising models with grid graphs, i.e., $n = 16$ nodes and $d = 24$ edges. The edge parameters are chosen independently and uniformly at random in the interval $[.05, 5]$. The sign of each parameter is positive with probability $1/2$ and negative otherwise. The initial dataset contains $N_0 = 20$ points. We note that with a cost of about $1.8s$ for a single evaluation of the KL divergence, enumerating all $|\mathcal{D}| = 2^{24}$ inputs to obtain an optimal solution is infeasible. Thus, we report the best value obtained after iteration t rather than the simple regret. Fig. 3 depicts the mean performance with 95% confidence intervals for $\lambda = 10^{-4}$, when averaged over 10 randomly generated Ising models and 10 initial data sets D_0 for each model. The statistics for other values of λ are stated in Table 2. Initially, BOCS-SDP, BOCS-SA, EI and SA show a similar performance. As the number of samples increases, BOCS-SDP obtains better solutions and in addition achieves a lower variability across different instances of Ising models.

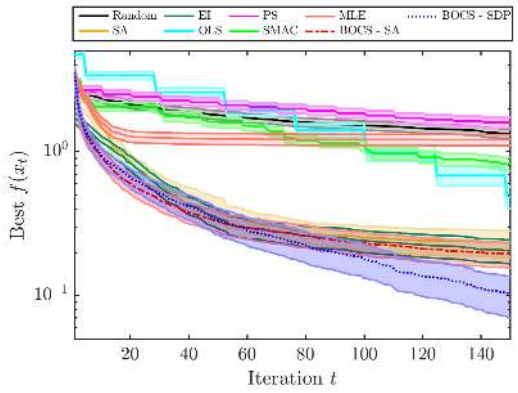

Figure 3. Sparsification of Ising models ($\lambda = 10^{-4}$): BOCS−SDP performs best followed by BOCS−SA. EI and SA also show good performance. Due to the size of the search space and the evaluation cost of the objective, we report the average best function values after t iterations rather than the simple regret.

4.3. Contamination Control

The contamination control problem (Hu et al., 2010) considers a food supply with d stages that may be contaminated with pathogenic microorganisms. Specifically, we let random variable Z_i denote the fraction of contaminated food at stage i for $1 \leq i \leq d$. (Z) evolves according to a random process that we describe next. At each stage i, a prevention effort can be made to decrease the contamination by a random rate Γ_i, incurring a cost c_i. If no prevention effort is taken, the contamination spreads with rate given by random variable Λ_i. This results in the recursive equation $Z_i = \Lambda_i(1 - x_i)(1 - Z_{i-1}) + (1 - \Gamma_i x_i)Z_{i-1}$, where $x_i \in \{0, 1\}$ is the decision variable associated with the prevention effort at stage i. Thus, the goal is to decide for each stage whether to implement a prevention effort in order to minimize the cost while ensuring the fraction of contaminated food does not exceed an upper limit U_i with probability at least $1 - \epsilon$. The random variables Λ_i, Γ_i and the initial contamination fraction Z_1 follow beta-distributions, whereas U_i=0.1 and $\epsilon = 0.05$.

We consider the Lagrangian relaxation of the problem that is given by

$$\underset{x}{\operatorname{argmin}} \sum_{i=1}^{d} \left[c_i x_i + \frac{\rho}{T} \sum_{k=1}^{T} 1_{\{Z_k > U_i\}} \right] + \lambda \|x\|_1, \qquad (7)$$

where each violation is penalized by $\rho = 1$. Recall that we have $d = 25$ stages. We set $T = 10^2$, hence the objective requires T simulations of the random process and is expensive to evaluate. The ℓ_1 regularization term encourages the prevention efforts to occur at a small number of stages.

The mean objective value (with 95% confidence intervals) of the best solution found after t iterations is shown in Fig. 4 for $\lambda = 10^{-2}$. Table 3 compares the performances for other values of λ after 250 iterations. BOCS−SDP achieves the best performance in all scenarios.

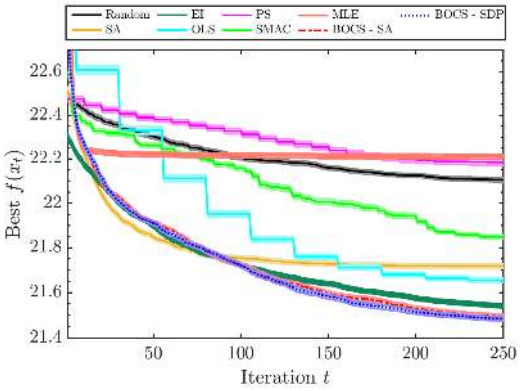

Figure 4. Contamination control ($\lambda = 10^{-2}$): Initially SA performs best but then is trapped in a local optimum. As the number of samples increases, BOCS−SDP obtains the best contamination prevention schedules, followed by BOCS−SA and EI.

Table 3. Contamination control: BOCS−SA and BOCS−SDP obtain the best function values for all three settings of λ, here measured after 250 iterations.

λ	SA	EI	OLS
0	21.58 ± 0.01	21.39 ± 0.01	21.54 ± 0.01
10^{-4}	21.60 ± 0.01	21.40 ± 0.01	21.51 ± 0.01
10^{-2}	21.72 ± 0.01	21.54 ± 0.01	21.65 ± 0.01
1	**23.33** ± 0.01	24.71 ± 0.02	25.12 ± 0.08
λ	MLE-SA	BOCS−SA	BOCS−SDP
0	22.02 ± 0.01	21.35 ± 0.01	**21.34** ± 0.01
10^{-4}	22.03 ± 0.01	21.36 ± 0.01	**21.35** ± 0.01
10^{-2}	22.21 ± 0.02	21.49 ± 0.01	**21.48** ± 0.01
1	**23.33** ± 0.01	**23.33** ± 0.01	**23.33** ± 0.01

While SA and EI initially perform well (see Fig. 4), SA becomes stuck in a local optimum. After about 80 iterations, BOCS typically finds the best prevention control schedules, and improves upon the solution found by EI.

4.4. Aero-structural Multi-Component Problem

We study the aero-structural problem of Jasa et al. (2018) that is composed of two main components, aerodynamics and structures (see Fig. 5). These blocks are coupled by 21 coupling variables that describe how aerodynamic properties affect the structure and vice versa, how loads and deflections affect the aerodynamics. For a set of inputs, e.g., airspeed, angle of the airfoil etc., with prescribed (Gaussian) probability distributions, the model computes the uncertainty in the output variables $\mathbf{y}$, which include the lift coefficient C_L,

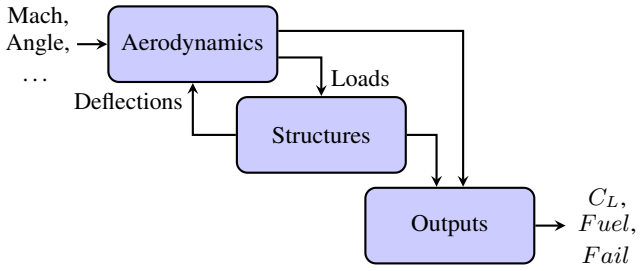

Figure 5. The aero-structural model of Jasa et al. (2018): The arrows indicate the flow of information between components. Note that the loop requires a fixed point solve whose computational cost increases quickly with the number of involved coupling variables.

the aircraft's fuelburn $Fuel$, and a structural stress failure criteria $Fail$. However, the amount of coupling in the model contributes significantly to the computational cost of performing this uncertainty quantification (UQ).

To accelerate the UQ process, we wish to identify a model with a reduced number of coupling variables that accurately captures the probability distribution of the output variables, $\pi_{\mathbf{y}}$. Let $x \in \{0,1\}^d$ represent the set of 'active' coupling variables: $x_i = 0$ denotes that coupling i from the output of one discipline is ignored and its input to another discipline is fixed to a prescribed value. The effect of this perturbation on the model outputs is measured by the KL divergence between $\pi_{\mathbf{y}}$, i.e., the output distribution of the reference model, and the output distribution $\pi^x_{\mathbf{y}}$ for the model with coupling variables x. Thus, the problem is to find an $\operatorname{argmin}_{x\in\{0,1\}^d} D_{KL}(\pi_{\mathbf{y}}||\pi^x_{\mathbf{y}}) + \lambda\|x\|_1$, where $D_{KL}(\pi_{\mathbf{y}}||\pi^x_{\mathbf{y}})$ is expensive to evaluate and $\lambda \geq 0$ trades off accuracy and sparsity of the model. Fig. 6 shows the average performances for $\lambda = 10^{-2}$. SA has the best overall performance, followed closely by EI, BOCS-SDP, and BOCS-SA that have a similar convergence profile.

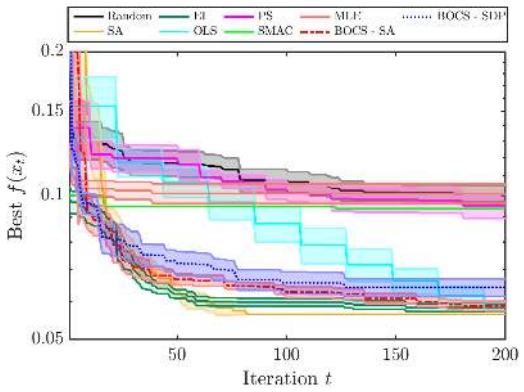

Figure 6. The aero-structural multi-component problem (λ=10^{-2}): SA performs best followed by EI, BOCS-SDP, and BOCS-SA.

Fig. 7 shows the output distribution of the reference model $\pi_{\mathbf{y}}$ (left), and the output distribution of a sparsified model $\pi^x_{\mathbf{y}}$ found by BOCS for $\lambda = 10^{-2}$ (right). We note that the distribution is closely preserved, while the sparsified solution only retains 5 out of the 21 active coupling variables present in the reference model.

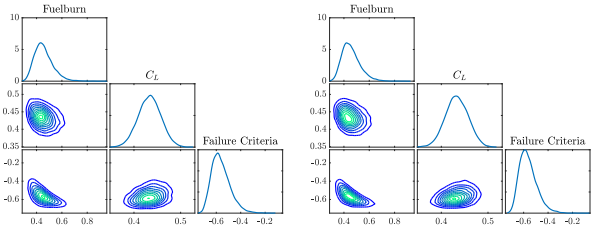

Figure 7. The aero-structural problem: the univariate and bivariate marginals of the output distribution for the reference (left) and a sparsified model (right) found by BOCS for $\lambda = 10^{-2}$.

5. Conclusion

We have proposed the first algorithm to optimize expensive-to-evaluate black-box functions over combinatorial structures. This algorithm successfully resolves the challenge posed by the combinatorial explosion of the search domain. It relies on two components: The first is a novel acquisition algorithm that leverages techniques from convex optimization for scalability and efficiency. Aside from effective handling of sparse data, the value of our model lies in its applicability to a wide range of combinatorial structures. We have demonstrated that our algorithm outperforms state-of-the-art methods from combinatorial optimization and machine learning.

Future work includes efficient optimization of other acquisition criteria, for example based on expected improvement or uncertainty reduction (e.g., see Chevalier et al. (2014); Hernández-Lobato et al. (2014)), and the development of tailored models for specific applications. For the latter, we anticipate a significant potential in the explicit modeling of combinatorial substructures that seem of relevance for a given task. For example, when optimizing over combinatorial structures such as graphical models, power grids and road networks, it seems promising to enrich the model by monomials that correspond to paths in the induced graph. Another interesting direction is to employ a deep neural networks to learn useful representations for the regression. This technique would harmonize well with our acquisition function and complement the sparse parametric model proposed here for functions with moderate evaluation costs, as it would require more training data but also scale better to large numbers of samples.

ACKNOWLEDGEMENT

This work has been supported in part by the Air Force Office of Scientific Research (AFOSR) MURI on "Managing multiple information sources of multi-physics systems," Program Officer Dr. Fariba Fahroo, Award Number FA9550-15-1-0038.

References

Arora, S. and Kale, S. A combinatorial, primal-dual approach to semidefinite programs. *Journal of the ACM (JACM)*, 63(2):12, 2016.

Arora, S., Berger, E., Elad, H., Kindler, G., and Safra, M. On non-approximability for quadratic programs. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 206–215, 2005.

Baptista, R., Marzouk, Y., Willcox, K., and Peherstorfer, B. Optimal approximations of coupling in multidisciplinary models. *AIAA Journal*, 56(6):2412–2428, 2018.

Bergstra, J. and Bengio, Y. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

Bhattacharya, A., Chakraborty, A., and Mallick, B. K. Fast sampling with gaussian scale mixture priors in high-dimensional regression. *Biometrika*, pp. asw042, 2016.

Binois, M., Ginsbourger, D., and Roustant, O. On the choice of the low-dimensional domain for global optimization via random embeddings. *arXiv preprint arXiv:1704.05318*, 2017.

Boyd, S. and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

Brochu, E., Cora, V. M., and De Freitas, N. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

Carvalho, C. M., Polson, N. G., and Scott, J. G. The horseshoe estimator for sparse signals. *Biometrika*, 97(2): 465–480, 2010.

Charikar, M. and Wirth, A. Maximizing quadratic programs: Extending grothendieck's inequality. In *Proc. of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pp. 54–60, 2004.

Chevalier, C., Bect, J., Ginsbourger, D., Vazquez, E., Picheny, V., and Richet, Y. Fast parallel kriging-based stepwise uncertainty reduction with application to the identification of an excursion set. *Technometrics*, 56(4): 455–465, 2014.

Del Moral, P., Doucet, A., and Jasra, A. Sequential monte carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, 2006.

Dewancker, I., McCourt, M., Clark, S., Hayes, P., Johnson, A., and Ke, G. A stratified analysis of Bayesian optimization methods. *arXiv preprint arXiv:1603.09441*, 2016.

Garey, M. R. and Johnson, D. S. *Computers and intractability*. W. H. Freeman and Company, 1979.

Gelman, A., Carlin, J. B., Stern, H. S., and Dunson, D. B. *Bayesian data analysis*. Chapman and Hall/CRC, 2013.

Golovin, D., Solnik, B., Moitra, S., Kochanski, G., Karro, J., and Sculley, D. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1487–1495. ACM, 2017.

Hernández-Lobato, J. M., Hoffman, M. W., and Ghahramani, Z. Predictive entropy search for efficient global optimization of black-box functions. In *Advances in Neural Information Processing Systems*, pp. 918–926, 2014.

Hernández-Lobato, J. M., Gonzalez, J., and Martinez-Cantin, R. NIPS workshop on Bayesian optimization, 2017. The problem is listed at http://bayesopt.com/. Last Accessed on 02/07/18.

Hu, Y., Hu, J., Xu, Y., Wang, F., and Cao, R. Z. Contamination control in food supply chain. In *Proceedings of the 2010 Winter Simulation Conference*, pp. 2678–2681, 2010. The source code is available at http://simopt.org/wiki/index.php?title=Contamination_Control_Problem. Last Accessed on 02/05/18.

Hutter, F. *Automated configuration of algorithms for solving hard computational problems*. PhD thesis, University of British Columbia, 2009.

Hutter, F. and Osborne, M. A. A kernel for hierarchical parameter spaces. *arXiv preprint arXiv:1310.5738*, 2013.

Hutter, F., Hoos, H., and Leyton-Brown, K. Automated configuration of mixed integer programming solvers. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pp. 186–202, 2010.

Hutter, F., Hoos, H. H., and Leyton-Brown, K. Sequential model-based optimization for general algorithm configuration. In *International Conference on Learning and Intelligent Optimization*, pp. 507–523. Springer, 2011.

Jasa, J. P., Hwang, J. T., and Martins, J. R. R. A. Open-source coupled aerostructural optimization using python. *Structural and Multidisciplinary Optimization*, 57(4): 1815–1827, 2018. The source code is available at http://github.com/mdolab/OpenAeroStruct. Last Accessed on 02/07/18.

Jenatton, R., Archambeau, C., González, J., and Seeger, M. Bayesian optimization with tree-structured dependencies. In *International Conference on Machine Learning*, pp. 1655–1664, 2017.

Jones, D. R., Schonlau, M., and Welch, W. J. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

Kandasamy, K., Schneider, J., and Póczos, B. High dimensional Bayesian optimisation and bandits via additive models. In *International Conference on Machine Learning*, pp. 295–304, 2015.

Khanna, S., Motwani, R., Sudan, M., and Vazirani, U. V. On syntactic versus computational views of approximability. *SIAM Journal on Computing*, 28(1):164–191, 1998.

Li, C.-L., Kandasamy, K., Póczos, B., and Schneider, J. High dimensional bayesian optimization via restricted projection pursuit models. In *Artificial Intelligence and Statistics*, pp. 884–892, 2016.

Makalic, E. and Schmidt, D. F. A simple sampler for the horseshoe estimator. *IEEE Signal Processing Letters*, 23 (1):179–182, 2016.

Mockus, J., Tiesis, V., and Zilinskas, A. The application of Bayesian methods for seeking the extremum. In Dixon, L. C. W. and Szego, G. P. (eds.), *Towards Global Optimisation*, volume 2, pp. 117–129. Elsevier Science Ltd., North Holland, Amsterdam, 1978.

Negoescu, D. M., Frazier, P. I., and Powell, W. B. The knowledge-gradient algorithm for sequencing experiments in drug discovery. *INFORMS Journal on Computing*, 23(3):346–363, 2011.

Pankratov, D. and Borodin, A. On the relative merits of simple local search methods for the MAX-SAT problem. In *SAT 2010*, pp. 223–236, 2010.

Poloczek, M. and Williamson, D. P. An experimental evaluation of fast approximation algorithms for the maximum satisfiability problem. *ACM Journal of Experimental Algorithmics (JEA)*, 22:1–18, 2017.

Russo, D., Van Roy, B., Kazerouni, A., and Osband, I. A tutorial on Thompson sampling. *arXiv:1707.02038*, 2017.

Schäfer, C. Particle algorithms for optimization on binary spaces. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 23(1):8, 2013.

Selman, B., Kautz, H. A., and Cohen, B. Local search strategies for satisfiability testing. In *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, pp. 521–532, 1993.

Shahriari, B., Bouchard-Cote, A., and Freitas, N. Unbounded bayesian optimization via regularization. In *Artificial Intelligence and Statistics*, pp. 1168–1176, 2016a.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and de Freitas, N. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104 (1):148–175, 2016b.

Snoek, J., Larochelle, H., and Adams, R. P. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pp. 2951–2959, 2012.

Spears, W. M. Simulated annealing for hard satisfiability problems. In *Cliques, Coloring and Satisfiability: Second DIMACS Implementation Challenge*, pp. 533–558, 1993.

Steurer, D. Fast SDP algorithms for constraint satisfaction problems. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pp. 684–697. SIAM, 2010.

Swersky, K., Duvenaud, D., Snoek, J., Hutter, F., and Osborne, M. A. Raiders of the lost architecture: Kernels for bayesian optimization in conditional parameter spaces. *arXiv preprint arXiv:1409.4011*, 2014.

Thompson, W. R. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25:285–294, 1933.

Thompson, W. R. On the theory of apportionment. *American Journal of Mathematics*, 57(2):450–456, 1935.

Wang, Z., Hutter, F., Zoghi, M., Matheson, D., and de Feitas, N. Bayesian optimization in a billion dimensions via random embeddings. *Journal of Artificial Intelligence Research*, 55:361–387, 2016.

Wang, Z., Li, C., Jegelka, S., and Kohli, P. Batched high-dimensional Bayesian optimization via structural kernel learning. *arXiv preprint arXiv:1703.01973*, 2017.

Zhang, Y., Sohn, K., Villegas, R., Pan, G., and Lee, H. Improving object detection with deep convolutional networks via Bayesian optimization and structured prediction. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.