Computers & Industrial Engineering 157 (2021) 107310

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Bayesian optimization for a multiple-component system with target values

Jihwan Jeong [1], Hayong Shin [*]

Department of Industrial & Systems Engineering, KAIST, 291 Daehak-ro, Yuseong-gu, Daejeon 34141, Republic of Korea

A R T I C L E I N F O

Keywords:
Multiple-component system
Bayesian optimization
Sequential experimental design
Metamodeling
Gaussian processes

A B S T R A C T

Bayesian optimization (BO) that employs the Gaussian process (GP) as a surrogate model has recently gained much attention in optimization of expensive black-box functions. In BO, the number of experiments necessary to optimize a function can be considerably reduced by sequentially selecting next design points that are optimal with respect to some sampling criterion. However, little research has been done to address the optimization of a multiple-component system where each component has a certain target value to meet. In this paper, we aim to find an optimal design parameter in the sense that the response function is close to the target value for every component. To this end, the squared errors from the targets are aggregated to produce an objective function. Instead of modeling this objective using GP as in the standard BO formulation, we place the GP prior over the response function. As a result, the distribution over the objective function follows that of the weighted sum of non-central chi-squared random variables (WSNC) due to the inter-dependency between responses. When components of the system are changed, the standard BO suffers inefficiency; however, our formulation enables us to retain a learned model, resulting in better efficiency. We compare the rates of convergence of different BO methods and other black-box optimization baselines using several test functions. The performance of our model is comparable to the standard BO when there is no change in the system, but the superiority of our method becomes clear when changes in the components occur.

1. Introduction

In modern mass production systems, batch processing is frequently used. It is quite a challenge to determine optimal process parameters for batch processing of heterogeneous parts (e.g. components of different shape and size). An interesting example of batch processing of multiple heterogeneous component is PCB (Printed Circuit Board) assembly process using SMT (Surface Mount Technology), which is a main workhorse in electronic appliance manufacturing. Briefly speaking, SMT process typically consists of the following three steps: (1) solder paste is printed on PCB, (2) chips are placed on the board, then (3) the board is fed into an oven for soldering. Step (1) and (3) are examples of batch processing of heterogeneous multiple components. If we focus on step (1) solder paste printing, each PCB has multiple locations, called pads, on which solder paste is transferred through apertures of a stencil mask. It is crucial to have the right amount of paste for each pad. In this case, each pad is a component, and the amount of solder paste printed on it is *the process response*, for which target value is given respectively. The process engineer wants to find the optimal process parameters for the solder paste printer. There are many other examples sharing the same characteristics (heterogeneous multiple components with the same kind of process response) in manufacturing or service systems. In many cases, we have to rely on experimental trials in order to find 'good' process parameters.

When such a system is expensive-to-evaluate, how to design experiments to find optimal parameters within fewest possible trials is of crucial importance. This has been successfully dealt with by Bayesian optimization (BO) for single-component systems (Hennig & Schuler, 2012; Jones, Schonlau, & Welch, 1998; Snoek, Larochelle, & Adams, 2012). The idea of BO is to build a surrogate model (e.g. Gaussian Process, GP) and use it to sequentially select next design parameters in an intelligent manner via maximization of an *acquisition function*. The BO framework has also been applied to optimize systems with target values. For example, Picheny, Ginsbourger, Roustant, Haftka, and Kim (2010) adaptively selected next designs in order to reduce the model uncertainty around a target value. Similarly, Ranjan, Bingham, and Michailidis (2008) tackled the contour estimation problem using the expected improvement as the acquisition function.

* Corresponding author.
E-mail addresses: jhjeong@mie.utoronto.ca (J. Jeong), hyshin@kaist.ac.kr (H. Shin).
[1] The first author is currently affiliated with University of Toronto.

https://doi.org/10.1016/j.cie.2021.107310
Received 29 May 2019; Received in revised form 17 January 2021; Accepted 8 April 2021
Available online 20 April 2021

However, there has been little work on optimizing multiple-component systems with target values using BO methods. In principle, one could employ multi-objective BO methods (MOBO) by setting one objective per each component. The MOBO framework, however, is most useful when the objectives compete with one another, and the trade-off between them is critical (Knowles, 2006). Furthermore, state-of-the-art MOBO methods often assume that the objectives are independent of each other (Belakaria, Deshwal, & Doppa, 2019; Hernandez-Lobato, Hernandez-Lobato, Shah, & Adams, 2016). However, as we assume that process responses are of the same kind, making use of the dependency between the responses from components turns out to be effective. Furthermore, unlike the multi-objective setting, the tradeoffs between objectives—related to responses from different components—are not clear in the problem setting we consider.

Hence, to avoid being overly-complicated, we rather opt for minimizing *the weighted sum of squared errors*, with each error being incurred by one component due to a deviation from the assigned target value. Here, we assume that each component c of the system can be fully represented and uniquely determined by a feature vector $\mathbf{y}_c$. We further assume that the response from the component c is a scalar function of a controllable parameter vector $\mathbf{x}$ and the feature vector $\mathbf{y}_c$, i.e. $f(\mathbf{x}, \mathbf{y}_c)$ where $\mathbf{x} \in \mathbb{R}^{d_X}, \mathbf{y}_c \in \mathbb{R}^{d_Y}$, and $f : \mathbb{R}^{d_X+d_Y} \rightarrow \mathbb{R}$. Then, the objective function is of the form below,

$$\mathcal{L}_{\mathbf{T}}(\mathbf{x}) = \sum_{c=1}^{C} w_c (f(\mathbf{x}, \mathbf{y}_c) - \mathrm{T}_c)^2 \tag{1}$$

where w_c denotes a weight given to the c-th component, $\mathbf{T} = (\mathrm{T}_1, \ldots, \mathrm{T}_C)^\top$ is the associated target vector, and C is the total number of components.

Standard BO methods would place a GP prior over the objective $\mathcal{L}_{\mathbf{T}}(\mathbf{x})$. When a changeover—which we define as addition, removal or replacement of any components in the system—occurs, those methods suffer from loss of data and information obtained so far. This is because changes in components or target values induce the change in the objective function, and then the GP surrogate model approximating the previous objective function becomes no longer valid.

In this paper, we develop a BO framework for multiple-component systems with target values, which can be reused when the system design is changed. The main technical contribution follows from the fact that we place a GP prior over the response f, namely the result of an experiment from each component. Then, we show that the objective function $\mathcal{L}_{\mathbf{T}}(\mathbf{x})$ follows the distribution of *a weighted sum of non-central chi-squared random variables (WSNC)* from our modeling assumption. This understanding enables us to define and compute the expected improvement (EI) acquisition function, which in turn helps us sequentially find the global optimum of $\mathcal{L}_{\mathbf{T}}(\mathbf{x})$.

We review Gaussian Process Regression and Bayesian Optimization in Section 2. Then, we propose our novel BOMCT method (Bayesian Optimization for a Multiple-Component system with Target values) **in Section** 3. With a few commonly used simulated test functions, we demonstrate the validity and effectiveness of our method. The optimization process of the Branin test function is illustrated in Section 4.1, followed by the results comparing the rates of convergence of different methods on several test functions (Section 4.2). As substantiated by the results, our method not only quickly reduces objective function values but also outperforms other methods—in terms of the number of iterations needed—when some components of the system are changed.

2. Background

2.1. Gaussian process regression (GPR)

A Gaussian process (GP), which defines a distribution over functions, can be fully described by its mean and covariance function, that is,

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{2}$$

Here, $k(\mathbf{x}, \mathbf{x}')$ is a positive definite covariance function also known as a kernel. The mean function, $m(\mathbf{x})$, is usually set to the zero vector because we are interested in the posterior distribution rather than the prior in most cases.

After observing data $D = \{\mathrm{X}, \mathbf{t}\} = \{(\mathbf{x}_i, t_i) | i = 1, \ldots, n\}$, we obtain the predictive distribution as the conditional distribution of $\mathbf{f}_*$ at test points X_* given $\mathbf{t}$. Since this is also Gaussian, it can be put as follows (Rasmussen & Williams, 2006):

$$\mathbf{f}_* | \mathrm{X}_*, \mathrm{X}, \mathbf{t} \sim \mathcal{N}(\bar{\mathbf{f}}_*, cov(\mathbf{f}_*)) \tag{3}$$

$$\begin{aligned} cov(\mathbf{f}_*) &= K(\mathrm{X}_*, \mathrm{X}_*) - K(\mathrm{X}_*, \mathrm{X})[K(\mathrm{X}, \mathrm{X}) + \sigma_n^2 \mathrm{I}]^{-1} K(\mathrm{X}, \mathrm{X}_*) \\ \bar{\mathbf{f}}_* &= \mathbb{E}[\mathbf{f}_* | \mathrm{X}, \mathbf{t}, \mathrm{X}_*] = K(\mathrm{X}_*, \mathrm{X})[K(\mathrm{X}, \mathrm{X}) + \sigma_n^2 \mathrm{I}]^{-1} \mathbf{t} \end{aligned} \tag{4}$$

An independent noise $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$ is assumed here, resulting in $\mathbf{t} = f(\mathrm{X}) + \epsilon \sim \mathcal{GP}(\mathbf{0}, k'(\cdot, \cdot))$ where $k'(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_i, \mathbf{x}_j) + \sigma_n^2 \delta_{ij}$, with $\sigma_{ij} = 1$ if $i = j$ or 0 otherwise.

At a test point $\mathbf{x}_*$, the predictive distribution gives an estimator of the function value as $\bar{\mathbf{f}}_*$, and the uncertainty is quantified by $cov(\mathbf{f}_*)$. Although the accuracy of prediction largely depends on the selection of kernel function and its hyperparameters, GPR is by far the most preferred surrogate in BO due to its flexibility as well as tractability (Krause, Singh, & Guestrin, 2008; Snoek et al., 2012). More comprehensive treatment of GPR can be found in Rasmussen and Williams (2006).

2.2. Surrogate modeling and Bayesian optimization

Consider the optimization problem of a scalar objective function $J : \mathbb{R}^d \rightarrow \mathbb{R}$ defined on a d-dimensional continuous domain, $\mathcal{X}$. When J indicates the result of some expensive experiment, then direct optimization methods requiring a number of function evaluations are prohibitive. As such, we may resort to methods which employ a surrogate model that approximates the original function while requiring far less computation in optimizing it (Crombecq, Laermans, & Dhaene, 2011).

That being said, we can roughly describe the BO methodology as a surrogate-based optimization scheme that harnesses convenient features of the Gaussian distribution in optimizing a complex function. Frazier (2018) has summarized the common properties of objective functions that are relevant to BO, and some of them are presented below:

1. J is an expensive function requiring a considerable amount of costs to evaluate; hence, the number of possible evaluations of the function is limited.
2. J is a black-box function in the sense that it lacks a known structure (e.g. concavity) and analytic form that would otherwise simplify the optimization.
3. Often, no information except the values of observed J is available. Thus, the first and second-order optimization methods cannot be used.

Due to (ii) and (iii), we are left with derivative-free methods. In addition, the number of function evaluations should be minimized in order to account for (i). BO addresses these issues by (1) using a GP model as a surrogate for J, (2) sampling based on some criterion computed from the posterior distribution of the GP, (3) finally, sequentially updating the posterior distribution. This sampling criterion is called an acquisition function. The general schematic description of BO is summarized below in Algorithm 1.

Algorithm 1. General Scheme of BO (Frazier, 2018)

The most popular choice of acquisition function is the expected

```
input: Objective function ($J : \mathbb{R}^d \to \mathbb{R}$) and its domain ($\mathcal{X} \subset \mathbb{R}^d$)
       Experimental budget ($N$)
       Number of initial design points ($n_0$) and initial design (DoE)
Place a GP prior on $J$
Observe $J$ at $n_0$ points according to some initial experimental design, and
  store them to the dataset $D_{n_0}$
Set $n = n_0$
while $n \le N$ do
    Update the posterior distribution on $J$ using the dataset collected
    Compute the maximizer ($\mathbf{x}_{new}$) of the acquisition function using the
      posterior distribution
    Observe $J_{new} = J(\mathbf{x}_{new})$ and add it to the dataset $D_n$
    Increment $n$
end
return Either $argmin_{\mathbf{x} \in D_n} J(\mathbf{x})$, or the point having the minimum mean
  posterior response
```

improvement (EI), which was proposed by Močkus (1974) and later popularized by Jones et al. (1998). The 'improvement' at a point $\mathbf{x}$ at the nth iteration of BO is defined as $I_n(\mathbf{x}) = \left[J_n^{min} - \widehat{J}(\mathbf{x})\right]^+$. Note that $\widehat{J}$ denotes a GP surrogate model, J_n^{min} is the minimum objective value observed so far, and $[\cdot]^+$ is the positive part of a value inside the brackets. This improvement is a random variable because of $\widehat{J}$; therefore, we take the expectation under the posterior distribution on J and obtain a real-valued score.

It is easy to see that the expected improvement at the nth step, $\mathrm{EI}_n(\mathbf{x})$, is:

$$\mathbb{E}[I_n(\mathbf{x})] = \mathbb{E}_n[\max(0, J_n^{\min} - \widehat{J}(\mathbf{x}))|\mathcal{D}_n, \mathbf{x}] = \begin{cases} (J_n^{\min} - \mu(\mathbf{x}))\Phi(\mathrm{Z}) + \sigma(\mathbf{x})\phi(\mathrm{Z}) & , \text{if } \sigma(\mathbf{x}) > 0 \\ 0 & , \text{otherwise} \end{cases} \tag{5}$$

where $\mathrm{Z} = \frac{f(\mathbf{x}) - \mu(\mathbf{x})}{\sigma(\mathbf{x})} \sim \mathcal{N}(0,1)$, and ϕ, Φ refer to the density and distribution functions of the standard normal distribution, respectively (Brochu, Cora, & de Freitas, 2010). This acquisition automatically balances the exploitation and exploration. To see this, note that the EI is large when $\mu(\mathbf{x})$ is small or $\sigma(\mathbf{x})$ is large. In other words, the EI exploits current estimation by giving a high score to $\mathbf{x}$ when $\mu(\mathbf{x})$ is small, while it also encourages points that are located in a less-explored region with high $\sigma(\mathbf{x})$ to be sampled.

2.3. Optimizing the sum of squared errors using Bayesian optimization

As described in Section 1, we often want a multiple-component system to output a result in which $f(\mathbf{x}, \mathbf{y}_c)$ is close to a certain target T_c. In the common BO setting, we simply place a GP prior on $\mathcal{L}_{\mathbf{T}}(\mathbf{x})$ of Eq. 1 and follow Algorithm 1 accordingly.

This approach is efficient only when it is guaranteed that the components of the system are not changing at all times. In reality, however, they can be frequently replaced with different ones. When $\mathbf{y}_c$ changes, so does $f(\mathbf{x}, \mathbf{y}_c)$, which in turn alters the objective function $\mathcal{L}_{\mathbf{T}}(\mathbf{x})$. This implies that we have to learn $\mathcal{L}_{\mathbf{T}}(\mathbf{x})$ all over again, leading to critical inefficiency. That is, when $\mathbf{y}_c$ changes, we have no choice but to discard the information collected so far if we model $\mathcal{L}_{\mathbf{T}}(\mathbf{x})$ with a GP surrogate.

In Section 3, we instead directly model the response function $f(\mathbf{x}, \mathbf{y})$ using a GP surrogate and subsequently optimize $\mathcal{L}_{\mathbf{T}}(\mathbf{x})$ under the guidance of an appropriately developed acquisition function. This proposed method retains efficiency even if the features $\mathbf{y}_c$ change. That way, we can utilize the information—obtained from the experiments performed with one set of components—in optimizing the parameters of the multiple-component system with another set of components.

3. Model description

3.1. Overview

As noted in previous sections, we are interested in optimizing a black-box multiple-component system that is expensive-to-evaluate. Let the function $f : \mathbb{R}^{d_X + d_Y} \to \mathbb{R}$ represent scalar outputs of evaluations conducted on this system. We assume that f depends on two kinds of variables: $\mathbf{x}$ and $\mathbf{y}$. The design parameter $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^{d}_{\mathbb{X}}$ is the one that we would like to optimize, whereas $\mathbf{y} \in \mathcal{Y} \subset \mathbb{R}^{d}_{\mathbb{Y}}$ is the variable that uniquely and fully represents each component. In other words, each component c has one $\mathbf{y}_c$ vector that is fixed for the component unless otherwise mentioned.

In this paper, we adjust the design parameter $\mathbf{x}$ such that the response from a component c gets close to a certain target T_c for all c ($c = 1, \ldots, C$). That is to say, we would like $|f(\mathbf{x}, \mathbf{y}_c) - \mathrm{T}_c|$ to be small. Naturally, we can set up a loss function by summing up the squared errors from the corresponding targets at all components. This loss function will significantly penalize large deviations from the targets. A diagonal weight matrix $diag(\mathbf{w})$—that is user-defined and assumed to adequately reflect the importance of each component—is introduced because we are aggregating all errors. Additionally, with the response vector $\mathrm{f}(\mathbf{x}) =$

$[f(\mathbf{x},\mathbf{y}_1),\ldots,f(\mathbf{x},\mathbf{y}_C)]^\top$, the target vector $\mathbf{T} = [\mathrm{T}_1,\ldots,\mathrm{T}_C]$, and the $C\times C$ identity matrix I, we can reiterate Eq. 1 in vectorized form,

$$\mathcal{L}_\mathbf{T}(\mathbf{x}) = \sum_{c=1}^{C} w_c(f(\mathbf{x},\mathbf{y}_c) - \mathrm{T}_c)^2 = (\mathrm{f}(\mathbf{x}) - \mathbf{T})^\top \mathit{diag}(\mathbf{w})\,(\mathrm{f}(\mathbf{x}) - \mathbf{T}) \tag{6}$$

Then, we put a GP prior on $f(\mathbf{x},\mathbf{y})$:

$$\widehat{f}(\mathbf{x},\mathbf{y}) \sim \mathcal{GP}(m(\mathbf{x},\mathbf{y}), k((\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y}'))) \tag{7}$$

Note that hereafter the GP model is denoted as $\widehat{f}$, while the actual response function is f. Similarly, $\widehat{\mathcal{L}}_\mathbf{T}$ denotes the loss function induced by $\widehat{f}$. We can see that an experiment at one design point $\mathbf{x}$ yields C responses, i.e. $\mathrm{f}(\mathbf{x}) = [f(\mathbf{x},\mathbf{y}_1),\ldots,f(\mathbf{x},\mathbf{y}_C)]^\top$, which are then combined to give one $\mathcal{L}_\mathbf{T}(\mathbf{x})$ value. After having experimented at n different design points $(\mathbf{x}_1,\ldots,\mathbf{x}_n)$, we have the dataset $D_n = \{\mathbf{X},\mathbf{f}\}$, where $\mathbf{X}$ is a $nC\times(d_X+d_Y)$ input matrix and $\mathbf{f} = [\mathrm{f}^\top(\mathbf{x}_1),\ldots,\mathrm{f}^\top(\mathbf{x}_n)]^\top \in \mathbb{R}^{nC\times 1}$ is the response vector.

Given the GP prior (Eq. 7) and the dataset D_n, the posterior distribution at some query point $\mathbf{x}_*$ is similarly computed as in Eq. 3 and Eq. 4. One critical difference is that we should always consider the predictive distribution for all C responses. Thus, we augment $\mathbf{x}_*$ with $\{\mathbf{y}_c\}_{c=1}^C$ to obtain a $C\times(d_X+d_Y)$ query matrix $\mathbf{X}_*$:

$$\mathbf{X}_* = \begin{bmatrix} \mathbf{x}_* & \mathbf{y}_1 \\ \vdots & \vdots \\ \mathbf{x}_* & \mathbf{y}_C \end{bmatrix} \tag{8}$$

Then, the predictive distribution at $\mathbf{x}_*$ forms a multivariate normal distribution defined as follows (assuming identical Gaussian noise, σ):

$$\widehat{\mathrm{f}}(\mathbf{x}_*)|\mathbf{X}_*,\mathbf{X},\mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}_*,\Sigma_*)\boldsymbol{\mu}_* = K(\mathbf{X}_*,\mathbf{X})\left[K(\mathbf{X},\mathbf{X}) + \sigma^2\mathrm{I}\right]^{-1}\mathbf{f} \in \mathbb{R}^{C\times 1}\Sigma_* \\ = K(\mathbf{X}_*,\mathbf{X}_*) - K(\mathbf{X}_*,\mathbf{X})\left[K(\mathbf{X},\mathbf{X}) + \sigma^2\mathrm{I}\right]^{-1}K(\mathbf{X},\mathbf{X}_*) \in \mathbb{R}^{C\times C} \tag{9}$$

Since we are minimizing $\mathcal{L}_\mathbf{T}(\mathbf{x})$ in Eq. 6, the improvement at $\mathbf{x}_*$ is defined as $I_n(\mathbf{x}_*) = \left[\mathcal{L}_n^{\min} - \widehat{\mathcal{L}}_\mathbf{T}(\mathbf{x}_*)\right]^+$, where $\mathcal{L}_n^{\min}$ denotes the minimum loss we have obtained until the nth iteration. In the standard BO, $\widehat{\mathcal{L}}_\mathbf{T}(\mathbf{x}_*)$ is modeled by a GP prior. In this case, the expected improvement is straightforward to compute (Section 2.2). When we place a GP prior on the response function, however, $\widehat{\mathcal{L}}_\mathbf{T}(\mathbf{x}_*)$ becomes a quadratic form in the Gaussian random variables of Eq. 9 associated with the diagonal weight matrix $\mathit{diag}(\mathbf{w})$. Because $f(\mathbf{x}_*,\mathbf{y}_c)$ are all correlated as per Σ_*, $\widehat{\mathcal{L}}_\mathbf{T}(\mathbf{x}_*)$ *does not* follow the non-central chi-squared distribution.

In the following subsection, we show that $\widehat{\mathcal{L}}_\mathbf{T}(\mathbf{x}_*)$ follows the distribution of a weighted sum of non-central chi-squared random variables (WSNC) and present a method to compute the expected improvement of the random variable.

3.2. *The expected improvement acquisition function*

Suppose that Q is a continuous non-negative random variable that has a differentiable cumulative function (cdf) F_Q and the density function (pdf) q. If we define an improvement of Q with respect to $Q_{\min}$ as $I = \max\{0, Q_{\min} - Q\}$, then the expected improvement can be computed as follows:

$$\begin{aligned} EI_Q = \;& \mathbb{E}_Q[I] = \mathbb{E}_Q\left[\left(Q_{\min} - Q\right)\cdot 1_{Q_{\min}\geqslant Q}\right] \\ =\;& Q_{\min}\cdot\mathbb{P}(Q\leqslant Q_{\min}) - \int_0^{Q_{\min}} t\cdot q(t)dt \\ =\;& Q_{\min}\cdot F_Q(Q_{\min}) - [t\cdot F_Q(t)]_0^{Q_{\min}} + \int_0^{Q_{\min}} F_Q(t)dt = \int_0^{Q_{\min}} F_Q(t)dt \end{aligned} \tag{10}$$

This result indicates that the expected improvement of the random variable Q boils down to the one-dimensional definite integral of the cdf of Q. Hence, the cdf of $\widehat{\mathcal{L}}_\mathbf{T}(\mathbf{x}_*)$ would suffice to compute the expected improvement at any query point $\mathbf{x}_*$.

3.2.1. *The squared sum of dependent Gaussian random variables*

Let Y denote an m-dimensional multivariate Gaussian random vector whose mean vector and positive definite covariance matrix are $\boldsymbol{\mu}$ and Σ, respectively. Then, $Q = Y^\top AY$ is called *the quadratic form in Y associated with a symmetric matrix A*. We can show that Q becomes the weighted sum of non-central chi-squared random variables (WSNC) (Ha & Provost, 2013; Mathai & Provost, 1992).

Proof. Define a random vector $Z = L^{-1}(Y - \boldsymbol{\mu})$, where L is the lower triangular matrix from the Cholesky decomposition of $\Sigma = LL^\top$. Then, we see that $Z\sim\mathcal{N}(\mathbf{0},\mathrm{I})$, with an $m\times m$ identity matrix I. By putting $Y = L(Z + L^{-1}\boldsymbol{\mu})$ into $Q = Y^\top AY$, we get the following:

$$\begin{aligned} Q = Y^\top AY =\;& (Z + L^{-1}\boldsymbol{\mu})^\top L^\top AL(Z + L^{-1}\boldsymbol{\mu}) \\ =\;& (Z + L^{-1}\boldsymbol{\mu})^\top P\Lambda P^\top(Z + L^{-1}\boldsymbol{\mu}) \end{aligned} \tag{11}$$

$$\begin{aligned} &= (P^\top Z + P^\top L^{-1}\boldsymbol{\mu})^\top\Lambda(P^\top Z + P^\top L^{-1}\boldsymbol{\mu}) \\ &= (U + \boldsymbol{\delta})^\top\Lambda(U + \boldsymbol{\delta}) \end{aligned} \tag{12}$$

$$= \sum_{i=1}^{m}\lambda_i(U_i + \delta_i)^2 = \sum_{i=1}^{m}\lambda_i\chi^2(\delta_i^2) \tag{13}$$

In Eq. 11, the orthogonal diagonalization of the symmetric matrix $L^\top AL$ yields the orthogonal matrix P and the diagonal matrix Λ whose elements (λ_i) are the eigenvalues. Subsequently, $U = P^\top Z\sim\mathcal{N}(\mathbf{0},\mathrm{I})$ and $\boldsymbol{\delta} = P^\top L^{-1}\boldsymbol{\mu}$ are introduced in Eq. 12. It can be easily checked that the random vector U indeed follows the multivariate standard normal distribution as below,

$$\begin{aligned} &\mathbb{E}[U] = P^\top\mathbb{E}[Z] = \mathbf{0} \\ &\mathrm{Cov}(U) = \mathbb{E}[UU^\top] = \mathbb{E}[P^\top ZZ^\top P] = P^\top\mathbb{E}[ZZ^\top]P = P^\top P = \mathrm{I} \end{aligned}$$

This means that all the elements of U are independent. Simply expanding the matrix multiplication in Eq. 12 gives Eq. 13. Then, we can see that $(U_i + \delta_i)^2$ is the non-central chi-squared random variable with the non-centrality parameter δ_i, where U_i and δ_i denote the ith element of U and $\boldsymbol{\delta}$, respectively. □

3.2.2. *The loss function as a quadratic form in Gaussian random variables*

Suppose that we have the dataset $D_n = \{\mathbf{X},\mathbf{f}\}$ and the GP model $\widehat{f}(\cdot,\cdot)$, as described in Section 3.1. When a candidate design $\mathbf{x}_*$ is given, the predictive distribution over the response from each of C components is specified by Eq. 9. Also, the distribution over the loss function Eq. 6 is specified by the random variable induced by $\widehat{\mathrm{f}}(\mathbf{x}_*)$:

$$\widehat{\mathcal{L}}_\mathbf{T}(\mathbf{x}_*) = \left(\widehat{\mathrm{f}}(\mathbf{x}_*) - \mathbf{T}\right)^\top \mathit{diag}(\mathbf{w})\left(\widehat{\mathrm{f}}(\mathbf{x}_*) - \mathbf{T}\right) \tag{14}$$

Then, $\widehat{\mathcal{L}}_\mathbf{T}(\mathbf{x}_*)$ follows the distribution of the WSNC random variable, whose parameters are λ_c and δ_c:

$$\widehat{\mathcal{L}}_\mathbf{T}(\mathbf{x}_*) = \sum_{c=1}^{C}\lambda_c\chi^2(\delta_c^2) \tag{15}$$

This result is a straightforward application of the analysis given in Section 3.2.1, recognizing that $\left[\widehat{\mathrm{f}}(\mathbf{x}_*) - \mathbf{T}\right]\sim\mathcal{N}(\boldsymbol{\mu}',\Sigma')$ where $\boldsymbol{\mu}' = \boldsymbol{\mu}_* - \mathbf{T}$ and $\Sigma' = \Sigma_*$ are from Eq. 9. Consequently, we obtain λ_c and δ_c from $\Sigma' = LL^\top, L^\top \mathit{diag}(\mathbf{w})L = P\Lambda P^\top$, and $\boldsymbol{\delta} = P^\top L^{-1}\boldsymbol{\mu}'$.

Although there is no analytic form of the cdf for the WSNC random variable, several computing libraries provide numerical tools of approximation (Duchesne & de Micheaux, 2010). Because Eq. (10) is a definite integral of the cdf, it can be easily obtained via quadrature

(Picheny, Gramacy, Wild, & Le Digabel, 2016).

Algorithm 2. BOMCT, optimizing a multiple-component system via BO

```
input: Boundaries of the design space and the feature vector
       Number of initial designs ($n_0$) and budget ($N$)
Place a GP prior on $f : \mathbb{R}^{d_X+d_Y} \to \mathbb{R}$
Conduct $n_0$ experiments to obtain the initial dataset,
    $D_{n_0} \leftarrow \{\mathbf{X}_{n_0}, \mathbf{f}_{n_0}\}$ where $\mathbf{X}_{n_0} \in \mathbb{R}^{n_0 C\times(d_X+d_Y)}$ and $\mathbf{f}_{n_0} \in \mathbb{R}^{n_0 C\times 1}$
$n \leftarrow n_0$
while $n \leq N$ do
    Update the posterior distribution on $f$ using $D_n$
    Compute $\mathbf{x}_{new} = \operatorname{argmin}_{\mathbf{x}_*} \mathrm{EI}_n(\mathbf{x}_*)$ using Algorithm 3 and
        e.g. L-BFGS-B
    Augment $\mathbf{x}_{new}$ with $\{\mathbf{y}_c\}_{c=1}^{C}$ to obtain
        $\mathbf{x}_{new}^{aug} \leftarrow [(\mathbf{x}_{new}, \mathbf{y}_1)^\top, \ldots, (\mathbf{x}_{new}, \mathbf{y}_C)^\top]^\top$
    Observe $\mathrm{f}_{new} \leftarrow \mathrm{f}(\mathbf{x}_{new}^{aug}) \ : \ \mathbb{R}^{C\times 1}$
    $D_{n+1} \leftarrow D_n + \{\mathbf{x}_{new}^{aug}, \mathbf{f}_{new}\}$
    $n \leftarrow n + 1$
end
return Either $\operatorname{argmin}_{\mathbf{x}\in D_n} \mathcal{L}_{\mathbf{T}}(\mathbf{x})$ or the point having the minimum mean
  posterior response for $\hat{\mathcal{L}}_{\mathbf{T}}(\mathbf{x})$
```

3.3. *Bayesian optimization scheme for a multiple-component system with a target vector (BOMCT)*

Algorithm 2 summarizes BOMCT. We have described in Section 3.2 how to compute the expected improvement acquisition function at every candidate design point $\mathbf{x}_*$ at the nth iteration step ($\mathrm{EI}_n(\mathbf{x}_*)$), which is detailed in Algorithm 3.

Note that changes in components or target values can be easily accommodated while we also retain the learned GP model. When components have changed, then we just need to augment the new feature vector $\{\mathbf{y}_c^{\text{new}}\}_{c=1}^{C_{\text{new}}}$. If we want to optimize the system with different target values, a new target vector $\mathbf{T}$ is simply subtracted from $\boldsymbol{\mu}_*$ in Algorithm 3. Different weight vectors can also be applied with little effort.

Algorithm 3. Computing $\mathrm{EI}_n(\mathbf{x}_*)$ at the nth iteration

```
Input Minimum loss observed so far ($\mathcal{L}_n^{\min}$) Feature vectors of $C$ components ($\{\mathbf{y}_c\}_{c=1}^{C}$)
  Target vector ($\mathbf{T}$) and weight vector ($\mathbf{w}$) Dataset ($D_n = \{\mathbf{X}, \mathbf{f}\}$) and a query location
  ($\mathbf{x}_*$)
Augment $\mathbf{x}_*$ with $\{\mathbf{y}_c\}_{c=1}^{C}$,
  $\mathbf{x}_*^{aug} \leftarrow [(\mathbf{x}_*, \mathbf{y}_1)^\top, \ldots, (\mathbf{x}_*, \mathbf{y}_C)^\top]^\top$
Compute the GP predictive mean ($\boldsymbol{\mu}_*$) and covariance ($\Sigma_*$) using Eq. 9
Perform Cholesky decomposition on $\Sigma_*$,
  $\Sigma_* = LL^\top$
Compute parameters of the WSNC random variable,
  $L^\top diag(\mathbf{w})L = P\Lambda P^\top; \delta = P^\top L^{-1}\boldsymbol{\mu}'$
Return $\mathrm{EI}_n(\mathbf{x}_*) = \int_0^{\mathcal{L}_n^{\min}} F_{\widehat{\mathcal{L}_{\mathbf{T}}}}(t)dt$ using $\delta$ and $\Lambda$ via quadrature
```

4. Numerical results

In this section, we present the results of numerical simulations on some popular test functions to validate BOMCT. Firstly, we visualize the optimization process of the Branin function using the proposed method in Section 4.1. Then, the rates of convergence of several baselines are compared on four simulated test functions (Section 4.2). In Section 4.2.1, the systems consist of a small number of components, where each component has a one-dimensional feature value y. In Section 4.2.2, we examine the effect of the number of components using the 6-D Ackley function. In this experiment, each component has either a 1-dimensional or 2-dimensional feature.

The test functions are treated as a black box throughout the simulations, which means that we do not have access to the functional forms of the underlying response function. Moreover, the functions are assumed to be expensive-to-evaluate; therefore, our primary goal is to minimize the objective as fast as possible.

Simulations are conducted in Python using the libraries ‘GPy’ and ‘GPyOpt’ (GPy, 2012; GPyOpt authors, 2016). After sampling from the initial design points or whenever a new observation is made, the hyperparameters of any GP models are re-estimated via maximum likelihood estimation (MLE). Since the distribution function of WSNC is not available in Python, we use the ‘rpy2’ library to utilize the ‘CompQuadForm’ package from R (Duchesne & de Micheaux, 2010) as well as the implementation done in Picheny et al. (2016).

4.1. *Illustration of the proposed BO*

We first illustrate the patterns of the proposed acquisition function with respect to not only the objective function but also the true response function and its GP surrogate model. We use the Branin function for this purpose defined as below (Surjanovic & Bingham, 2017):

$$f(x,y) = a(y - bx^2 + cx - r)^2 + s(1-t)\cos(x) + s \tag{16}$$

where $a = 1, b = 5.1/(4\pi^2), c = 5/\pi, r = 6, s = 10$, and $t = 1/8\pi$. Here, the design variable subject to optimization is x.

As mentioned in Frazier, Powell, and Dayanik (2009) and indicated in Algorithm 1, some initial designs are necessary for the estimation of the hyperparameters of the GP model in the beginning. In the case of the

Table 1
Experimental settings: d_X and d_Y are the dimensionality of the design variable and the feature vector, respectively. n_{comp} is the number of components. The Matérn 52 kernel is used for GP models in BOMCT throughout all simulations. Details are described in the main text regarding how the feature vectors are chosen.

Test function	Target	d_X	d_Y	n_{comp}	Domain	Covariance Kernel
Branin	100	1	1	3	(-5,10) (1, 15)	Matérn 52
6-D Ackley	5	4 or 5	2 or 1	3 or 20	$(-5,5)^6$	
2-D Ackley	5	1	1	5	$(-5,5)^2$	
2-D Griewank	2	1	1	5	$(-5,5)^2$	

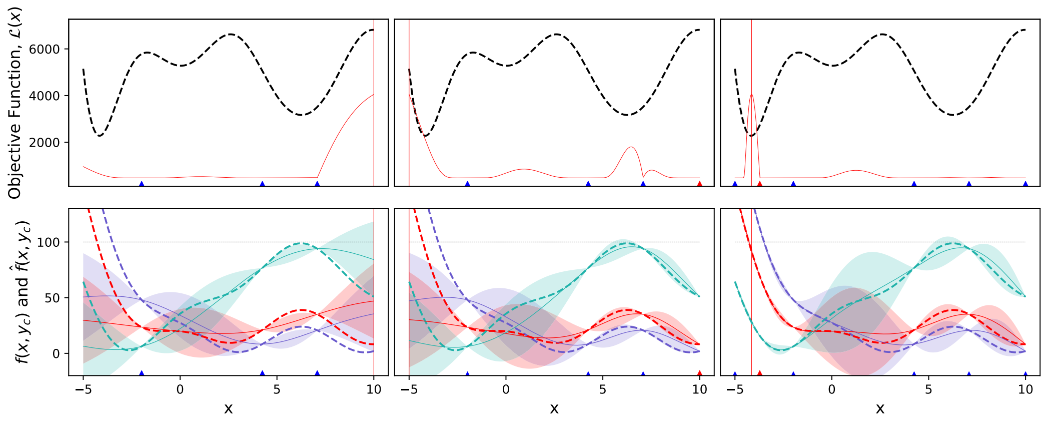
Branin function, initial evaluations have been made at three randomly chosen points. Other experimental settings used in this example are identical to those used in Section 4.2.1, which is reported in Table 1. Two out of three feature values are changed after 4 iterations to examine how the acquisition function behaves subject to such changes and whether the new optimum can be found quickly.

Fig. 1a: In the first two iterations, the acquisition is large where the uncertainty of the GP model is high. Furthermore, when the GP predictive means of components are closer to the target value, the acquisition tends to be higher. Only 6 function evaluations, including those at three initial designs, are enough for the model to accurately pick out the global optimum of the objective function.

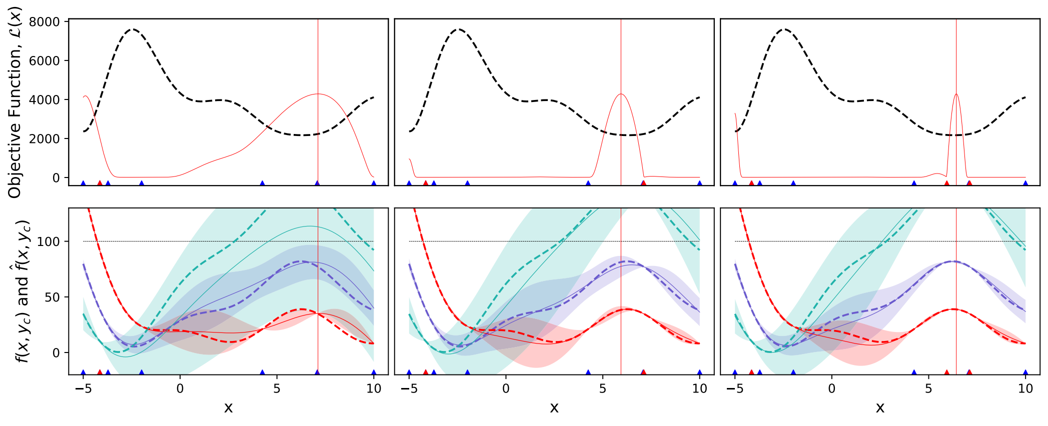
As mentioned repeatedly, it is not unusual that one or more components in a multiple-component system are changed (termed ‘changeover’). When there is a changeover in the system, we now need to optimize a different objective function as the underlying response function has changed (Fig. 1a). Note that after the changeover, one function evaluation is conducted at the previously sampled point.

In the leftmost figure of Fig. 1b, the GP predictive variances corresponding to the changed y are large (purple and green lines). The GP model is almost certain about the true response functions near the lower bound since the function has been evaluated around there, whereas large predictive variances are observed where the optimum lies. The acquisition peaks at two design points: one at which the GP mean indicates the objective to be small, and the other at which the GP variances are high as well as GP means are relatively close to the target value.

Even though the objective function has changed, it only takes few more iterations to figure out the new optimum, thanks to the previously trained GP model (Fig. 1b). Observe also that we do not have to sample from a certain interval in the middle, which saves resources that would otherwise be wasted.

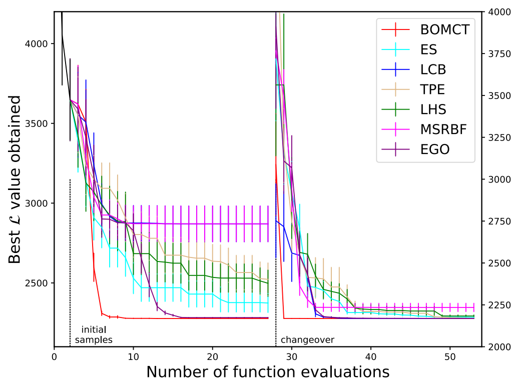


(a) Iterations 1, 2 and 4 before changeover, respectively

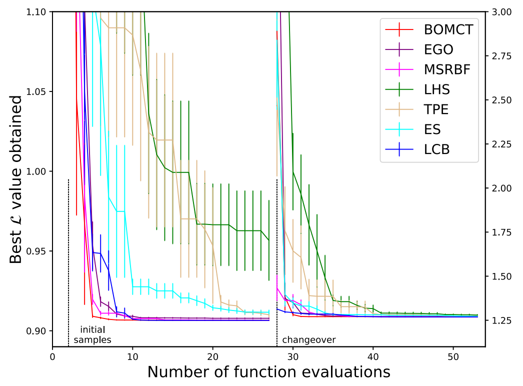


(b) Iterations 1, 2 and 3 after changeover, respectively

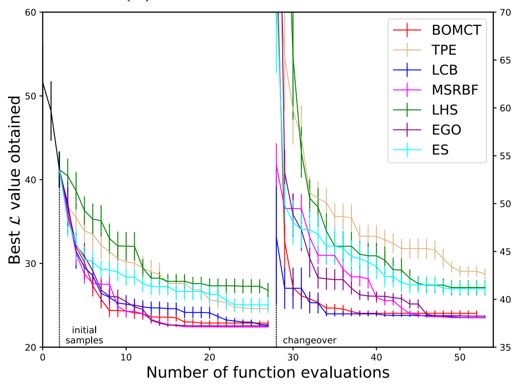
Fig. 1. Illustration of BOMCT. The objective function (black dotted line) and the acquisition (red line, in arbitrary units) are plotted together. Shown below are the target value (horizontal dotted line, set at 100); the true response functions (dotted lines of different colors) and their corresponding GP predictive means (solid lines of matching colors); the 95% credible regions as shaded parts. Sampled points are drawn as triangles on x axis. The red triangles indicate where the latest function evaluation has been made.



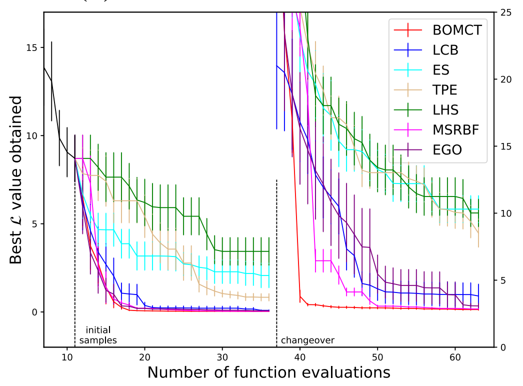
(a) The Branin function



(b) The 2-D Griewank function



(c) The 2-D Ackley function



(d) The 6-D Ackley function

Fig. 2. Comparison of the rates of convergence. The average of the best objective values from 15 simulation runs is plotted along with corresponding standard errors. Read the right axis in each plot for the values after the changeover.

4.2. *Comparison of rates of convergence*

We now investigate how BOMCT performs compared to some baselines. The compared methods are as follows:

- Latin Hypercube Sampling (LHS) (McKay, Beckman, & Conover, 1979) is a fixed space-filling experimental design where design points are randomly sampled. This method is inherently non-sequential and fixes all evaluation points prior to starting experiments.
- EGO (Jones et al., 1998) is a standard BO method which uses the expected improvement as the acquisition function.
- LCB (Srinivas, Krause, Kakade, & Seeger, 2010) is another standard BO method which uses the lower confidence bound as the acquisition function.
- ES (Hennig & Schuler, 2012) is a relatively recent standard BO method which uses the entropy search as the acquisition function.
- MSRBF (Regis & Shoemaker, 2007) is a sequential black-box optimization method that relies on using the RBF function as a surrogate model.
- Tree-structured Parzen Estimator (TPE) (Bergstra, Bardenet, Bengio, & Kégl, 2011) is a method which is often used in hyperparameter optimization of expensive machine learning algorithms.

By the standard BO, we mean the BO methods that model the objective function in Eq. 6 directly as a GP surrogate and then follow Algorithm 1 while using EI, LCB or ES as the acquisition function.

We compare the rates of convergence simulated on four different test functions. The settings of the simulations are specified in Table 1. For simplicity, we assume that every component has an equal weight value. As clearly stated, our goal is to minimize the objective function with fewest possible function evaluations. Thus, we record the best (i.e., the smallest) $\mathcal{L}_T$ values obtained until each iteration during optimization.

After initial designs, a model chooses 25 more samples before a changeover at which some or all components of the system are changed. At the changeover, a small number of evaluations are made before restarting the optimization process (three in the 6-D Ackley function and one in the other functions). This is especially done for the standard BO methods and MSRBF since they require initial designs to start with. Additionally, we have reset the best $\mathcal{L}_T$ value at the changeover since the objective function is changed.

For each compared method and test function, we have conducted 15 simulation runs with different initial designs for each run; however within each run, all methods start with the same initial samples for fair comparison. We plot means and standard errors of all methods over the optimization iterations. Also, we present box plots comparing BOMCT, EGO and MSRBF in more detail.

4.2.1. *Systems with a small number of components*

The test functions examined in this study are the Branin, 6-D Ackley, 2-D Ackley and 2-D Griewank functions (see Table 1). We refer readers to Surjanovic and Bingham (2017) for the exact definitions of these

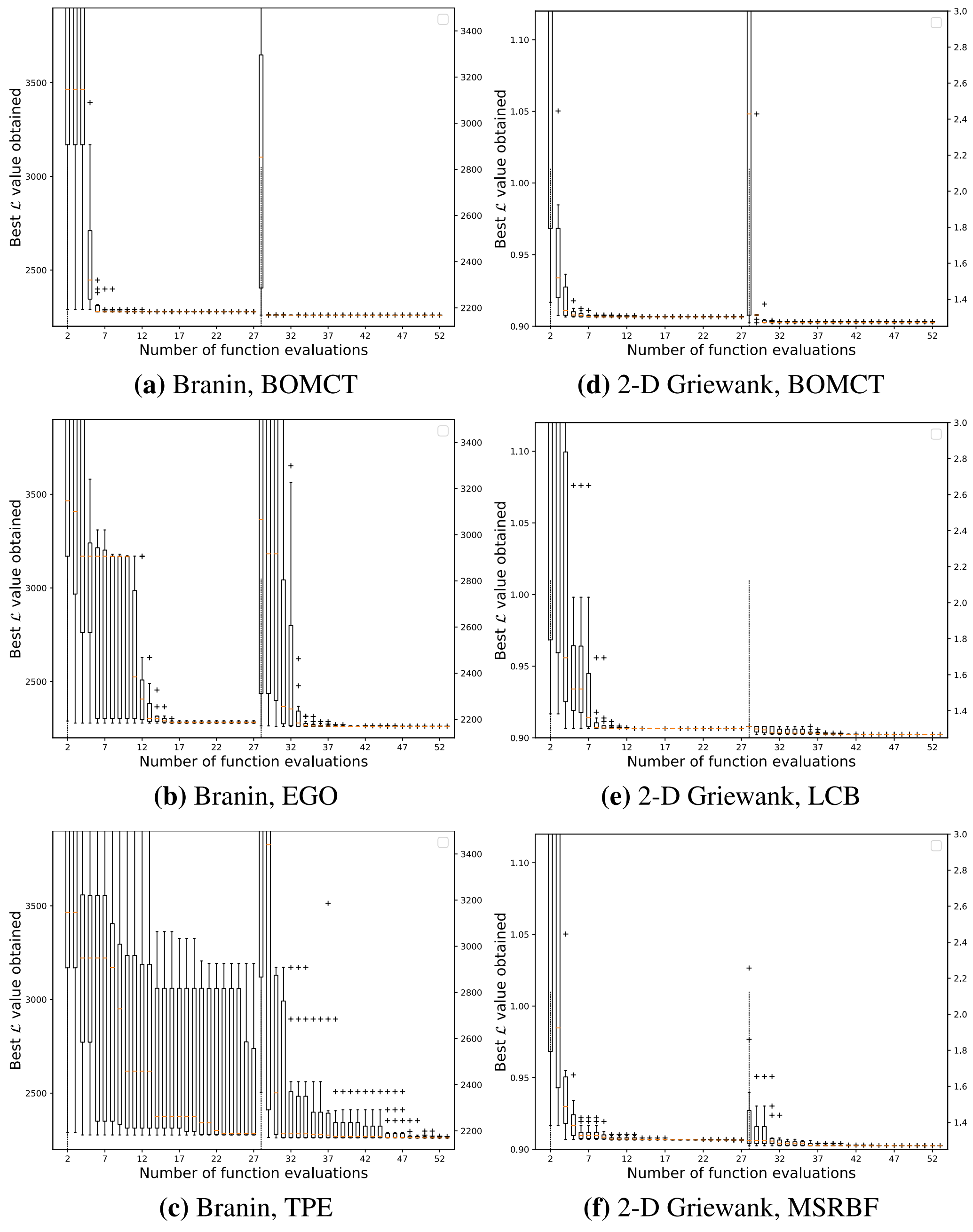


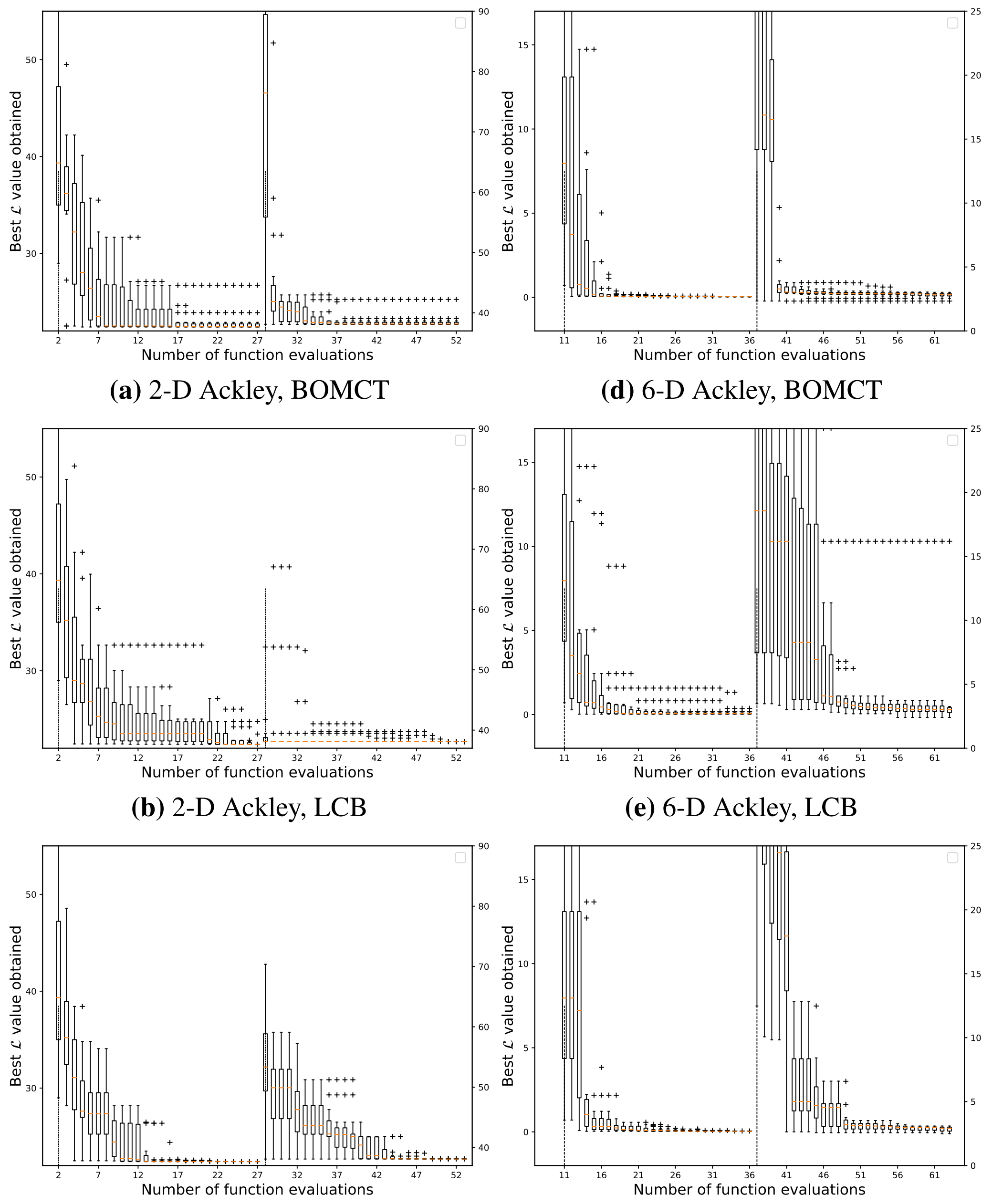
Fig. 3. Detailed comparison of the rates of convergence on the Branin and 2-D Griewank functions. The shown values are the results from 15 simulation runs. Read the right axis in each plot for the values after the changeover.

functions. For BOMCT, the Matérn 52 kernel function is used, while the squared exponential (SE) or the Matérn 52 kernels are used in the standard BO baselines. Note that the last one or two dimensions of the test functions are used for feature vectors $\mathbf{y}_c$.

We manually select the feature vectors for the test functions when there are 3 components; the vectors are chosen randomly when there are more than 5 components. The feature values in the Branin function change from (3.2, 5.5, 10.0) to (5.5, 9.0, 12.5). In the 6-D Ackley function with 1-dimensional feature, (-1.0, 0, 1.0) and (-1.0, 2.0, 3.5) are used before and after the changeover, respectively. For the test functions with 5 components, 3 out of 5 features are changed at the changeover.

Fig. 2 shows the results of the experiments. Note that the best $\mathcal{L}_\mathbf{T}$ values soar to different levels when a new set of features is applied to the multiple-component system. This is ascribed to the fact that the objective function has changed and that the first evaluation after the changeover is made at the latest design point, which would be different for different methods.

In general, we can observe superiority of the sequential methods over the fixed design, i.e., LHS. Especially, BOMCT performs nicely across all the test functions. In the 2-D Griewank function, the optima before and after the changeover are located in the vicinity, leading to fast optimization among the sequential methods. Hence, the improvement of BOMCT is only marginal. The performance gain is particularly conspicuous when it comes to the rates of convergence after the changeovers. For example, in Fig. 2d, shortly after the three samples given to all methods for adaptation to the changeover, BOMCT approaches the new optimum at once. In contrast, ES, LCB and EGO have effectively failed to minimize the objective within the available experimental budget.



(a) 2-D Ackley, BOMCT **(d)** 6-D Ackley, BOMCT

(b) 2-D Ackley, LCB **(e)** 6-D Ackley, LCB

(c) 2-D Ackley, MSRBF **(f)** 6-D Ackley, MSRBF

Fig. 4. Detailed comparison of the rates of convergence on the 2-D and 6-D Ackley functions. The shown values are the results from 15 simulation runs. Read the right axis in each plot for the values after the changeover.

MSRBF, on the other hand, proves to be quite useful in optimizing the 6-D Ackley function.

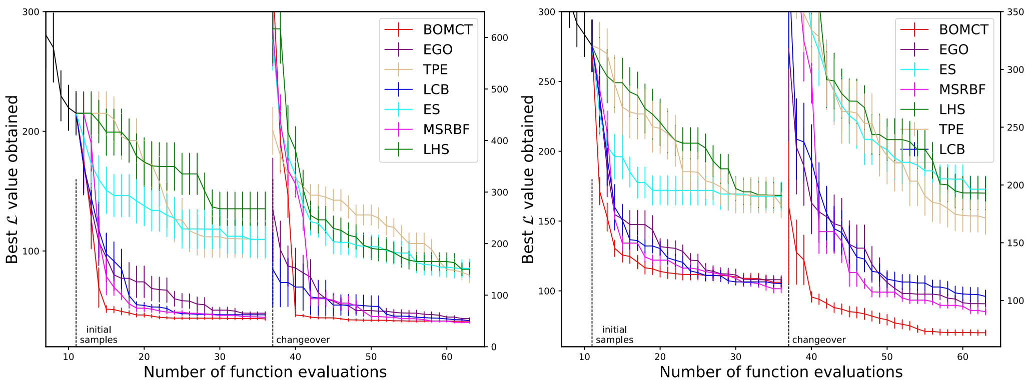
Notably, our method appears to be at least slightly better than other methods even before the changeovers. In general, when we try to put the GP prior on $f(\mathbf{x}, \mathbf{y})$ instead of $\mathcal{L}_\mathbf{T}(\mathbf{x})$, we may suffer from increased model complexity; on the other hand, we are able to use C times more data compared to when modeling $\mathcal{L}_\mathbf{T}(\mathbf{x})$. Additionally, squaring the error between a response function and a target would result in a function with more complex shapes. Hence, it seems that the benefits of modeling $f(x, y)$ using a GP prior outweigh the increased model complexity in these specific examples.

Fig. 3 and Fig. 4 show detailed comparison of (1) BOMCT (2) EGO or LCB, and (3) TPE or MSRBF using box plots. LCB seems to work particularly well in optimizing the 2-D Ackley and Griewank functions (Fig. 3e; Fig. 4b), while fails to minimize the Branin and 6-D Ackley functions (Fig. 2a; Fig. 4e). More importantly, we can reaffirm that BOMCT can reliably minimize the objectives before and after a changeover across all test functions.

4.2.2. *Systems with a large number of components*

The simulated systems studied so far are comprised of three or five components. However, it is likely that there are more of them in real world applications. Therefore, we have increased the number of components in a system to 20 in order to see if the performance of our method is affected by the size of the system. Furthermore, we use a 2-dimensional feature vector in the second example to analyze potential impacts of the dimensionality of the feature.

Firstly, each of the 20 components is set to have a feature value that



(a) 1-dimensional feature **(b)** 2-dimensional feature

Fig. 5. Comparison of the rates of convergence on the 6-D Ackley function with 20 components and different feature dimensions. The average obtained from 15 simulation runs is plotted with the standard errors as error bars.

is equidistantly distributed within the corresponding domain bound of the test function. Then, the changeover is implemented by choosing 10 feature values and adding to them some random numbers sampled from the standard normal distribution while ensuring the resulting features remain inside the bounds. The 2-dimensional feature vectors are similarly constructed. At the time of changeover, however, the first elements of the feature vectors are randomly shuffled and then recombined with the second elements that are also shuffled. This has resulted in 19 out of 20 feature vectors being changed.

Empirically, we have found that using larger values of $\mathcal{L}^{\min}$ in the computation of the EI (Algorithm 3) encourages the model to better explore the design space. This is especially the case when a function lives in a high dimensional space and objective values are small as in the 6-D Ackley function.

The general trend indicated in Fig. 5a (1D-feature case) aligns with the observations from the previous subsection. BOMCT turns out to be more effective even before a changeover occurs, but improvement over baselines is much clearer after the changeover.

Interesting results have been obtained from the 2D-feature case (Fig. 5b). BOMCT not only reduces $\mathcal{L}_{\mathrm{T}}$ faster than other methods—before and after the changeover, but also achieves significantly better values in the end. Some may argue that this is due in part to the small number of initial samples given to the baselines at the changeover (3 in our experiments). However, even if we give 12 initial samples for these methods, it appears that a gap of the similar size still remains. As pointed out previously, squaring the errors seems to have resulted in a more complex, and possibly wiggly, function that a GP model cannot easily learn. When the surrogate model is imprecise, it is reasonable to see that the BO fails to find out the global optimum value.

This may also be the case in the first phase of the optimization in the 2D-feature case. An exhaustive grid search has found that the objective function can decrease as low as around 87. However, all the sequential methods (except for TPE) have found objective values that are slightly above 100 in average. In such a case, one may attempt to use a covariance kernel that is more suitable for the particular response function to improve the performance.

We show detailed comparison of BOMCT and some baselines in Fig. 6. As discussed, when the system has 2D features, the baselines are unsuccessful in optimizing the function after the changeover; BOMCT can effectively minimize the objective. Additionally, the vertical widths of box plots indicate that BOMCT is much more reliable than other methods in that there is less fluctuation across 15 runs. LCB (Fig. 6b) is comparable to BOMCT in the 1D feature case, while MSRBF is better than the standard BO methods in the 2D feature case.

5. Conclusion

In this paper, we have presented BOMCT—a novel Bayesian optimization method for an expensive-to-evaluate multiple-component system that has target values to satisfy. We have implied in Introduction that a number of systems can potentially be formulated in this way.

Concretely, the sum of squared errors from the target values is set as the objective function so that large deviations from the targets are greatly penalized. Instead of placing a GP prior over the objective function *per se*, we put the prior over the individual responses from the components. We showed that this formulation enables us to model the objective as the weighted sum of non-central chi-squared random variables, which in turn allows us to compute the expected improvement acquisition function via quadrature. We have demonstrated the effectiveness of BOMCT using four test functions. Overall, BOMCT can reliably and efficiently find the optimal design parameters of the system across all tested functions. In particular, our proposed method is much more efficient than the standard BO methods in optimizing a system whose components change over time. This has been validated by comparing the rates of convergence of different methods on the test functions.

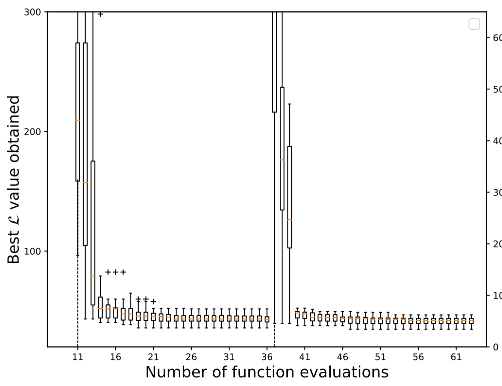
Despite the improvements, the aggregated squared errors do not allow us to incorporate specification limits (if any) which require responses to stay within some range containing the target values. Also, it may not be an easy task to engineer feature vectors of a system in some cases. Hence, future work could focus on how to force responses to occur within pre-specified ranges. Also, the computation of quadrature using 'CompQuadForm' package in R cannot be vectorized currently, which leads to longer running time for BOMCT. Once a more efficient tool becomes available, it would be interesting to see how much more performance gain we can obtain by computing the acquisition function with increased granularity.

CRediT authorship contribution statement

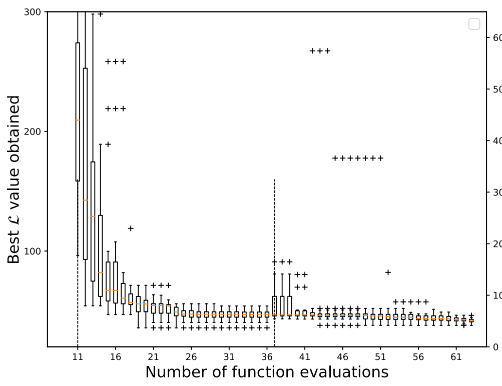
Jihwan Jeong: Methodology, Formal analysis, Software, Writing - original draft, Writing - review & editing. **Hayong Shin:** Conceptualization, Supervision, Funding acquisition, Resources.

Declaration of Competing Interest

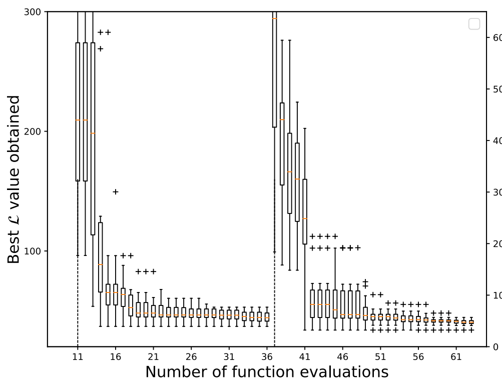
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.



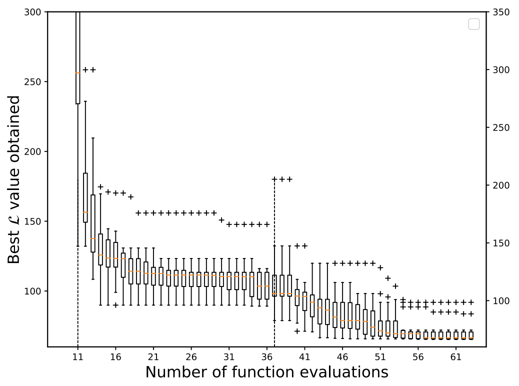
(a) Ackley (1D feature), BOMCT



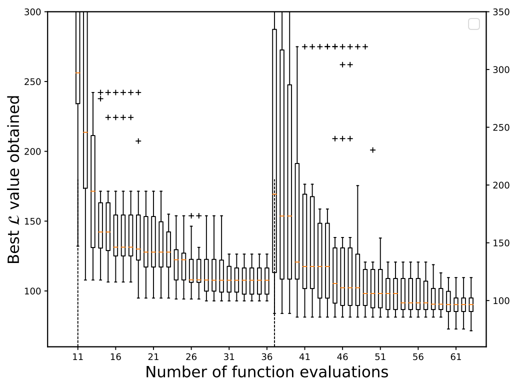
(b) Ackley (1D feature), LCB



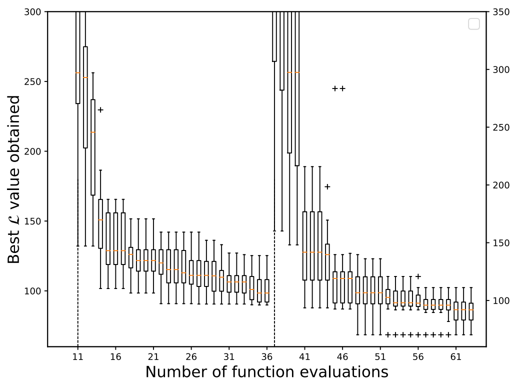
(c) Ackley (1D feature), MSRBF



(d) Ackley (2D feature), BOMCT



(e) Ackley (2D feature), EGO



(f) Ackley (2D feature), MSRBF

Fig. 6. Detailed comparison of the rates of convergence on the 6-D Ackley function when there are 20 components in the system. The shown values are the results from 15 simulation runs. Read the right axis in each plot for the values after the changeover.

Acknowledgements

This research was supported by Basic Science Research Program through the National Research Foundation of Korea funded by the Ministry of Science and ICT (2017R1A2B4006290).

References

Belakaria, S., Deshwal, A., & Doppa, J. R. (2019). Max-value entropy search for multi-objective bayesian optimization. In *Advances in Neural Information Processing Systems* (Vol. 32, pp. 7825–7835).

Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems* (pp. 2546–2554). Curran Associates, Inc.

Brochu, E., Cora, V. M., & de Freitas, N. (2010). A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. arXiv: 1012.2599.

Crombecq, K., Laermans, E., & Dhaene, T. (2011). Efficient space-filling and non-collapsing sequential design strategies for simulation-based modeling. *European Journal of Operational Research, 214*, 683–696. https://doi.org/10.1016/j.ejor.2011.05.032

Duchesne, P., & de Micheaux, P. L. (2010). Computing the distribution of quadratic forms: Further comparisons between the liu-tang-zhang approximation and exact methods. *Computational Statistics and Data Analysis, 54*, 858–862.

Frazier, P. I. (2018). A tutorial on Bayesian optimization. arXiv: 1807.02811.

Frazier, P., Powell, W., & Dayanik, S. (2009). The knowledge-gradient policy for correlated normal beliefs. *INFORMS Journal on Computing, 21*, 599–613. https://doi.org/10.1287/ijoc.1080.0314

GPy (2012). GPy: A gaussian process framework in python. http://github.com/SheffieldML/GPy.

GPyOpt authors (2016). Gpyopt: A bayesian optimization framework in python. http://github.com/SheffieldML/GPyOpt.

Ha, H.-T., & Provost, S. B. (2013). An accurate approximation to the distribution of a linear combination of non-central chi-square random variables. *REVSTAT Statistical Journal, 11*, 231–254.

Hennig, P., & Schuler, C. J. (2012). Entropy search for information-efficient global optimization. *Journal of Machine Learning Research, 13*, 1809–1837.

Hernandez-Lobato, D., Hernandez-Lobato, J., Shah, A., & Adams, R. (2016). Predictive entropy search for multi-objective Bayesian optimization. In *PMLR* (pp. 1492–1501).

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

Knowles, J. (2006). Parego: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation, 10*, 50–66.

Krause, A., Singh, A., & Guestrin, C. (2008). Near-optimal sensor placements in Gaussian processes: theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research, 9*, 235–284.

Mathai, A. M., & Provost, S. B. (1992). *Quadratic forms in random variables: Theory and applications*. New York: Marcel Kekker Inc.

McKay, M. D., Beckman, R. J., & Conover, W. J. (1979). Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics, 21*, 239–245.

Močkus, J. (1974). On bayesian methods for seeking the extremum. In *Proceedings of the IFIP Technical Conference* (pp. 400–404). London, UK, UK: Springer-Verlag.

Picheny, V., Ginsbourger, D., Roustant, O., Haftka, R. T., & Kim, N.-H. (2010). Adaptive designs of experiments for accurate approximation of a target region. *Journal of Mechanical Design, 132*. https://doi.org/10.1115/1.4001873

Picheny, V., Gramacy, R. B., Wild, S., & Le Digabel, S. (2016). Bayesian optimization under mixed constraints with a slack-variable augmented Lagrangian. In *Advances in Neural Information Processing Systems* (Vol. 29, pp. 1435–1443).

Ranjan, P., Bingham, D., & Michailidis, G. (2008). Sequential Experiment Design for Contour Estimation from Complex Computer Codes. *Technometrics, 50*, 527–541. https://doi.org/10.1198/004017008000000541

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian processes for machine learning*. MIT Press.

Regis, R., & Shoemaker, C. (2007). A stochastic radial basis function method for the global optimization of expensive functions. *INFORMS Journal on Computing, 19*, 497–509. https://doi.org/10.1287/ijoc.l060.0182

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems* (Vol. 25, pp. 2951–2959).

Srinivas, N., Krause, A., Kakade, M. S., & Seeger, M. (2010). Gaussian process optimization in the bandit setting: No regret and experimental design. arXiv: 0912.3995.

Surjanovic, S., & Bingham, D. (2017). Virtual library of simulation experiments: Test functions and datasets. https://www.sfu.ca/ssurjano/optimization.html.