Batch Bayesian Optimization via Local Penalization

Javier González
Dept. of Computer Science
University of Sheffield

Zhenwen Dai
Dept. of Computer Science
University of Sheffield

Philipp Hennig
Max Planck Institute
for Intelligent Systems

Neil Lawrence
Dept. of Computer Science
University of Sheffield

Abstract

The popularity of Bayesian optimization methods for efficient exploration of parameter spaces has lead to a series of papers applying Gaussian processes as surrogates in the optimization of functions. However, most proposed approaches only allow the exploration of the parameter space to occur *sequentially*. Often, it is desirable to simultaneously propose *batches* of parameter values to explore. This is particularly the case when large parallel processing facilities are available. These could either be computational or physical facets of the process being optimized. Batch methods, however, require the modeling of the interaction between the different evaluations in the batch, which can be expensive in complex scenarios. We investigate this issue and propose a highly effective heuristic based on an estimate of the function's *Lipschitz constant* that captures the most important aspect of this interaction—local repulsion—at negligible computational overhead. A *penalized acquisition function* is used to collect batches of points minimizing the non-parallelizable computational effort. The resulting algorithm compares very well, in run-time, with much more elaborate alternatives.

1 Introduction

Many problems, such as the configuration of machine learning algorithms [Snoek et al., 2012] or the experimental design of biological experiments [González et al., 2014] require the optimization of an unknown, possibly noisy, function f. Bayesian optimization (BO) has emerged in this scenario as an efficient heuristic to optimize f if function evaluations are costly and the overall number of evaluations must be kept low [Jones et al., 1998].

The task is to solve the global optimization problem of finding

$$\mathbf{x}_{max} = \arg\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}). \quad (1)$$

We assume that f is a *black-box* from which only perturbed evaluations of the type $y_i = f(\mathbf{x}_i) + \epsilon_i$, with $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, are available. We will assume that the objective of interest can be described well by a L-Lipschitz continuous function $f : \mathcal{X} \to \mathbb{R}$ defined on a compact subset $\mathcal{X} \subseteq \mathbb{R}^d$.

In sequential BO the goal is to make a series of evaluations $\mathbf{x}_1, \ldots, \mathbf{x}_N$ of f such that the maximum of f is evaluated as quickly as possible. After n points are available, BO proposes a new location $\mathbf{x}_{n+1}$ using a probabilistic model for f, conditioned on all previous observations $\mathcal{D}_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$. Typically the model is a Gaussian process (GP) $p(f) = \mathcal{GP}(\mu; k)$ with mean function μ and positive-definite covariance function (kernel) k that in this work we assume is stationary. Under Gaussian likelihoods, the posterior distribution of f (for a sample of size n) is also a GP, with posterior mean and variance given by

$$\mu_n(\mathbf{x}^*) = \mathbf{k}_n(\mathbf{x}^*)^\top [\mathbf{K}_n + \sigma_n^2 \mathbf{I}]^{-1} \mathbf{y}_n$$

and

$$\sigma_n^2(\mathbf{x}^*) = k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}_n(\mathbf{x}^*)^\top [\mathbf{K}_n + \sigma_n^2 \mathbf{I}]^{-1} \mathbf{k}_n(\mathbf{x}^*),$$

where $\mathbf{K}_n$ is the matrix such that $(\mathbf{K}_n)_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, $\mathbf{k}_n(\mathbf{x}^*) = [k(\mathbf{x}_1, \mathbf{x}^*), \ldots, k(\mathbf{x}_n, \mathbf{x}^*)]^\top$ [Rasmussen and Williams, 2005] and $\mathbf{x}^*$ is the point where the GP is evaluated.

This posterior is used to form the acquisition function $\alpha(\mathbf{x}; \mathcal{I}_n)$, where $\mathcal{I}_n$ represents the available data set $\mathcal{D}_n$ and the GP structure (kernel, likelihood and parameter values) when n data points are available. The next evaluation is placed at the (numerically estimated) global maximum $\mathbf{x}_{n+1}$ of this acquisition function. A number of possible acquisition functions are

Appearing in Proceedings of the 19^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2016, Cadiz, Spain. JMLR: W&CP volume 41.

now available, ranging from fast heuristics [Osborne, 2010, Jones et al., 1998] to non-local entropy-based approaches [Hennig and Schuler, 2012, Hernández-Lobato et al., 2014].

While the goal of Bayesian optimization is to keep the number of evaluations of f as low as possible, in high-dimensional and or otherwise complex problems, the number of required evaluations can still be considerable. Parallel approaches arise as the natural solution to circumvent the computational bottleneck around these evaluations of f. We focus on cases in which the cost of evaluating f in a batch of points of size n_b is the same as evaluating f in a single point. Such scenarios appear, for instance, in the optimization of computer models where several cores are available to run in parallel, or in wet-lab experiments when the cost of testing one experimental design is the same as testing a batch of them. In these settings, the set of available pairs $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ can be augmented with the evaluations of f on batches of data points $\mathcal{B}_t^{n_b} = \{\mathbf{x}_{t,1}, \dots, \mathbf{x}_{t,nb}\}$, for $t = 1, \dots, m$, rather than on single observations. Our goal here is to define a design rule for such batches $\mathcal{B}_1^{n_b}, \dots, \mathcal{B}_m^{n_b}$. The batch selection problem can be generalized further, *e.g.* by adapting the batch size [Azimi et al., 2012] or by collecting batches asynchronously [Ginsbourger et al., 2011, Janusevskis et al., 2012, Snoek et al., 2012]. For simplicity of exposition these ideas will not feature further here.

1.1 Optimal batch design and previous work

The goal of any batch criterion is to mimic the decisions that would be made under the equivalent (optimal) sequential policy: Consider the choice of selecting $\mathbf{x}_{t,k}$, the k-th element of the t-th batch. Under a sequential policy, in which the evaluations of f at all locations prior to $\mathbf{x}_{t,k}$ are available, the decision is to take $\mathbf{x}_{t,k}$ as the maximizer of $\alpha(\mathbf{x}; \mathcal{I}_{t,k-1})$. In the batch case, the decision about where to collect $\mathbf{x}_{t,k}$ has to incorporate the uncertainty about the locations $\mathbf{x}_{t,1}, \dots, \mathbf{x}_{t,k-1}$, and the outcomes of the evaluation of f there. Iteratively marginalizing these sources of uncertainty gives

$$\mathbf{x}_{t,k} = \arg\max_{\mathbf{x}\in\mathcal{X}} \int \alpha(\mathbf{x}; \mathcal{I}_{t,k-1}) \prod_{j=1}^{k-1} p(y_{t,j}|\mathbf{x}_{t,j}, \mathcal{I}_{t,j-1}) p(\mathbf{x}_{t,j}|\mathcal{I}_{t,j-1}) \mathrm{d}\mathbf{x}_{t,j} \mathrm{d}y_{t,j}, \quad (2)$$

where

$$p(y_{t,j}|\mathbf{x}_{t,j}, \mathcal{I}_{t,j-1}) = \mathcal{N}\left(y_{t,j}; \mu_n(\mathbf{x}_{t,j}), \sigma_n^2(\mathbf{x}_{t,j})\right)$$

is the predictive distribution of the GP at $\mathbf{x}_{t,j}$ when a total of n points are available and

$$p(\mathbf{x}_{t,j}|\mathcal{I}_{t,j-1}) = \delta(\mathbf{x}_{t,j} - \arg\max_{\mathbf{x}\in\mathcal{X}} \alpha(\mathbf{x}; \mathcal{I}_{t,j-1}))$$

reflects the optimization step required to obtain $\mathbf{x}_{t,j}$ after the evaluations of f at previous batch-elements have been marginalized.

The optimization in Eq. (2) is intractable even for small batch-sizes, due to the optimization-marginalization loop required to obtain $\mathbf{x}_{t,k}$. The literature in batch BO has tried to avoid this computational burden by means of different strategies, most of which involve the explicit use of the predictive distributions $p(y_{t,j}|\mathbf{x}_{t,j}, \mathcal{I}_{t,j-1})$, for $j = 1, \dots, n_b$. Exploratory approaches [Schonlau et al., 1998, Contal et al., 2013] search for a reduction in system uncertainty. This is using the property that the variance of $p(y_{t,j}|\mathbf{x}_{t,j}, \mathcal{I}_{t,j-1})$ does not depend on the value of the objective there. Other methods use $p(y_{t,j}|\mathbf{x}_{t,j}, \mathcal{I}_{t,j-1})$ to generate 'fake' observations of the model [Azimi et al., 2012, 2011, Bergstra et al., 2011] and avoid the marginalization step. In statistics, the suitability of the expected improvement utility has been studied for the design of batches [Chevalier and Ginsbourger, 2013, Frazier, 2012]. In contrast to the previous mentioned works, these methods use the joint distribution of $y_{t_1}, \dots y_{t,nb}$ to simultaneously optimize elements on the batch [Azimi et al., 2010]. These non-greedy strategies are very well founded from a theoretical perspective in practice but tend to scale poorly with the dimension of the problem and the sizes of the batches. Other theoretical properties of batch BO have been studied in the context of bandits [Srinivas et al., 2010], Bayesian networks [Oenek and Schwarz, 2000], multi-armed bandits [Desautels et al., 2012], robust optimization [Taddy et al., 2009] and the optimal balance between exploration and exploitation in batch designs [Jalali et al., 2013].

1.2 Goal and Contributions of this work

Using $p(y_{t,j}|\mathbf{x}_{t,j}, \mathcal{I}_{t,j-1})$ to model the interaction between batch elements has a computational overhead of $\mathcal{O}(n^3)$, since the GP needs to be updated after each batch location is selected to jointly optimize all the elements in the batch. The motivation of this work is to develop a *heuristic approximation* of Eq. (2) at lower computational cost, while incorporating information about global properties of f from the GP model into the batch design.

Our approach rests on the hypothesis that f is a Lipschitz continuous function, which is a common assumption in global optimization [Floudas and Pardalos, 2009]. For easy reference: a real-valued function $f : \mathcal{X} \to \mathbb{R}$ on a compact subset $\mathcal{X} \subseteq \mathbb{R}^d$ of the d-dimensional real vector space is said to be L-Lipschitz if it satisfies

$$|f(\mathbf{x}_1) - f(\mathbf{x}_2)| \leq L\|\mathbf{x}_1 - \mathbf{x}_2\|, \qquad \forall\, \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X} \quad (3)$$

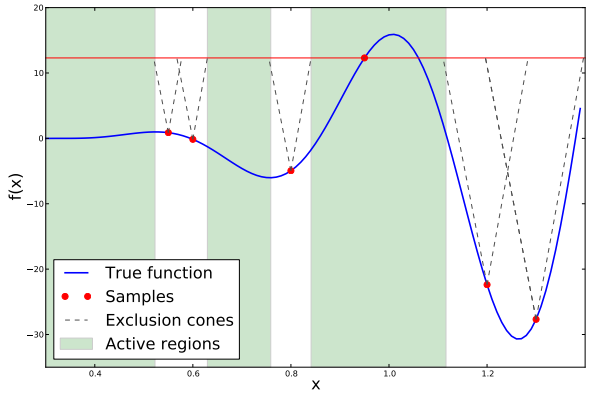


Figure 1: Forrester function $f(x) = (6x-2)^2 \sin(12x-4)$ in the interval $[0.3, 01, 4]$. We take 6 evaluations $\mathbf{x}_1, \ldots, \mathbf{x}_6$ of the function, $M = \max_i f(\mathbf{x}_i)$ and $L = 400$. The exclusion zones for the maximum of f determined by the balls $B_r(\mathbf{x}_i)$ are shown.

where L is a global positive constant, and $\|\cdot\|$ is the ℓ^2-norm on $\mathbb{R}^d$, a property that has been previously exploited in global optimization [Horst and Pardalos, 1995, Strongin and Sergeyev, 2000].

In the context of parallelizing Bayesian optimization, a beneficial aspect of the Lipschitzian assumption is that it naturally allows us to place bounds on how far the optimum of f is from a certain location. See Figure 1 for details. As explained below, this information can be used to define policies to collect a batch of points multiple steps ahead without evaluating f, by mimicking the hypothesized behavior of a sequential policy. The main challenge is that, in practice, the constant L is unknown. In the literature, this problem has been addressed from different angles [Floudas and Pardalos, 2009]. We explore a new alternative: inferring the Lipschitz constant directly from the Gaussian process model for f.

Our contributions are: (i) A new batch BO heuristic, BBO-LP, that selects batches of points by an iterative *maximization-penalization* loop around the the acquisition function. This leads to efficient parallelization of BO and can be used with any acquisition function. (ii) A probabilistic framework to approximately infer the Lipschitz constant of f, termed GP-LCA, that uses the properties of the gradients of the GP. The inferred value of L is used to improve batch selection. (iii) A python implementation of several batch BO methods is published in conjunction with this work.[1] (iv) Confirmation of the effectiveness of the approach is demonstrated through several simulated experiments, an algorithm configuration problem, and a real wet-lab experimental design. In particular, the local penalization approach performs equal or better than current batch BO methods in terms of the convergence to the maximum, but shows better performance in terms of the *wall-clock time*.

[1] http://sheffieldml.github.io/GPyOpt/.

2 Maximization-Penalization Strategy for Batch Design

The intuition behind our approach is that for most GP priors in practical use for BO, the dominant effect of a function evaluation on the acquisition function is a *local exclusion* around the new evaluation. This shape of the acquisition function will be modeled through the Lipschitz properties of f, to distribute the elements in each batch. This should be understood as a heuristic to the shape of $\alpha(\mathbf{x}; \mathcal{I}_{t,k-1})$ if all previous observations were available, mimicking the effect a sequential policy. This is especially useful in cases in which the acquisition function shows *multi-modal shape*, a common situation in the first iterations of BO algorithms. The following definition is helpful for the formalization of the algorithm:

Definition 1 *A function $\varphi(\mathbf{x}; \mathbf{x}_j)$, $\mathbf{x} \in \mathcal{X}$, is a* local penalizer *of a generic acquisition function $\alpha(\mathbf{x})$ at $\mathbf{x}_j$ if $\varphi(\mathbf{x}; \mathbf{x}_j)$ is differentiable, $0 \leq \varphi(\mathbf{x}; \mathbf{x}_j) \leq 1$ and $\varphi(\mathbf{x}; \mathbf{x}_j)$ is an non-decreasing function in $\|\mathbf{x} - \mathbf{x}_j\|$.*

We propose to replace the *maximization-marginalization* loop in Eq. 2 by a *maximization-penalization* strategy: while the optimization is carried out in a similar fashion, the marginalization step is replaced by the direct penalization of $\alpha(\mathbf{x}; \mathcal{I}_{t,k-1})$ around its most recent maximum, *i.e*, the previous batch element. Figure 2 gives a graphical illustration. The maximization-penalization strategy selects $\mathbf{x}_{t,k}$ as

$$\mathbf{x}_{t,k} = \arg\max_{x \in \mathcal{X}} \left\{ g(\alpha(\mathbf{x}; \mathcal{I}_{t,0})) \prod_{j=1}^{k-1} \varphi(\mathbf{x}; \mathbf{x}_{t,j}) \right\}, \quad (4)$$

where $\varphi(\mathbf{x}; \mathbf{x}_{t,j})$ are local local penalizers centered at $\mathbf{x}_{t,j}$ and $g : \mathbb{R} \to \mathbb{R}^+$ is a differentiable transformation of $\alpha(\mathbf{x})$ that keeps it strictly positive without changing the location of its extrema. We will use $g(z) = z$ if $\alpha(\mathbf{x})$ is already positive and the *soft-plus* transformation $g(z) = \ln(1 + e^z)$ elsewhere. This does not require re-estimation of the GP model after each location is selected, just a new optimization of the penalized utility.

The effect of a local penalizer is to smoothly reduce the value of the acquisition function in a neighborhood of $\mathbf{x}_j$. A 'good' local penalizer centered at $\mathbf{x}_j$ should reflect the belief about the distance from $\mathbf{x}_j$ to $\mathbf{x}_M$: If we suspect that $\mathbf{x}_M$ is far from $\mathbf{x}_j$, a broad $\varphi(\mathbf{x}; \mathbf{x}_j)$

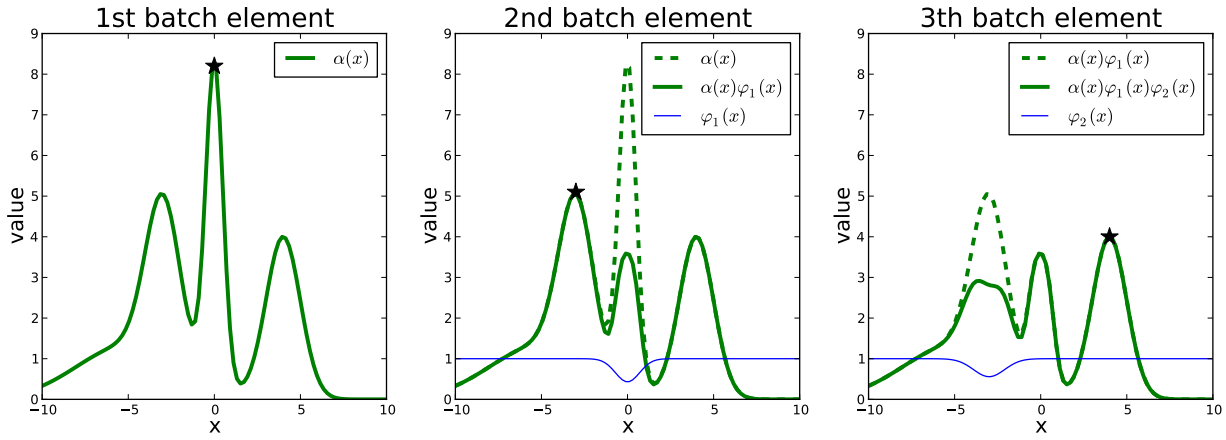


Figure 2: Illustration of three iterations of the *maximization-penalization* loop. The main task of good batch design is to explore the modes of the acquisition function, achieved by iterative maximization (black stars) and penalization (using $\varphi_1(\mathbf{x}), \varphi_2(\mathbf{x})$) of the acquisition function $\alpha(\mathbf{x})$.

will discard a large portion of $\mathcal{X}$ in which we don't need to collect any sample. On the other hand, if we believe that $\mathbf{x}_M$ and $\mathbf{x}_j$ are close, ideally we want to minimize the penalization of $\alpha(\mathbf{x})$ and keep collecting samples is a close neighborhood. This local penalization mimics the acquisition function's dynamics under a sequential policy in the following sense: the modes of the acquisition functions correspond to regions in which either $\mu_n(\mathbf{x})$ or $\sigma_n^2(\mathbf{x})$ (or both) are large. Evaluating, for instance, where $\sigma_n(\mathbf{x})$ is large will reduce uncertainty in that region, decreasing $\alpha(\mathbf{x})$ in a neighborhood. The functions $\varphi(\mathbf{x}; \mathbf{x}_j)$ are surrogates for this neighborhood.

2.1 Choosing Local Penalizers $\varphi(\mathbf{x}; \mathbf{x}_j)$

We now construct penalizing functions $\varphi(\mathbf{x}; \mathbf{x}_j)$ that incorporate into $\alpha(\mathbf{x})$ the current belief about the distance from the batch locations to $\mathbf{x}_M$. Take $M = \max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$, and a valid Lipschitz constant L. Consider the ball

$$B_{r_j}(\mathbf{x}_j) = \{\mathbf{x} \in \mathcal{X} : \|\mathbf{x}_j - \mathbf{x}\| \leq r_j\} \tag{5}$$

where

$$r_j = \frac{M - f(\mathbf{x}_j)}{L}.$$

To simplify the notation we write $r_j = r(\mathbf{x}_j)$ for the radius of the ball around $\mathbf{x}_j$. If f in (5) is the true optimization objective, then $\mathbf{x}_M \notin B_{r_j}(\mathbf{x})$—otherwise the Lipschitz condition would be violated. The size of $B_{r_j}(\mathbf{x}_j)$ depends on L, M and the value of f at $\mathbf{x}_j$. Both large variability in f (large L) and proximity of $f(\mathbf{x}_j)$ to the optimum M shrink $B_{r_j}(\mathbf{x}_j)$.

In the BO context, under the assumption $f(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$, we choose $\varphi(\mathbf{x}; \mathbf{x}_j)$ as the probability that $\mathbf{x}$, any point in $\mathcal{X}$ that is a potential candidate to be a maximum, does not belong to $B_{r_j}(\mathbf{x}_j)$:

$$\varphi(\mathbf{x}; \mathbf{x}_j) = 1 - p(\mathbf{x} \in B_{r_j}(\mathbf{x}_j)). \tag{6}$$

The following proposition (proof in Supp. Materials shows that this local penalizer can be computed in closed form.

Proposition 1 *Let $f(\mathbf{x})$ be a $\mathcal{GP}$ with posterior mean $\mu_n(\mathbf{x})$ and posterior variance $\sigma_n^2(\mathbf{x})$. The function $\varphi(\mathbf{x}; \mathbf{x}_j)$ in Eq. (6) is a valid local penalizer of $\alpha(\mathbf{x})$ at $\mathbf{x}_j$ such that:*

$$\varphi(\mathbf{x}; \mathbf{x}_j) = \frac{1}{2} erfc\,(-z)$$

where $z = \frac{1}{\sqrt{2\sigma_n^2(\mathbf{x}_j)}}\left(L\|\mathbf{x}_j - \mathbf{x}\| - M + \mu_n(\mathbf{x}_j)\right)$, for erfc the complementary error function, $M = \max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$ and L a valid Lipschitz constant.

The functions $\varphi(\mathbf{x}; \mathbf{x}_j)$ thus create exclusion zones whose size is governed by L. If $\mu_n(\mathbf{x}_j)$ is close to M, then $\varphi(\mathbf{x}; \mathbf{x}_j)$ will have a smaller and more localized effect on $\alpha(\mathbf{x})$ (a smaller exclusion area). On the other hand, if $\mu_n(\mathbf{x}_j)$ is far from M, $\varphi(\mathbf{x}; \mathbf{x}_j)$ will produce a wider yet less intense correction on $\alpha(\mathbf{x})$. The value of L also affects the size of the effect of $\varphi(\mathbf{x}; \mathbf{x}_j)$ on $\alpha(\mathbf{x})$, decreasing it as L increases.

2.2 Selecting the parameters L and M

The values of M and L are unknown in general. To approximate M, one can take $\hat{M} = \max_{\mathcal{X}} \mu_n(\mathbf{x})$ or, to avoid solving this maximization problem, use the even rougher approximation $\hat{M} = \max_i\{y_i\}$.

Regarding the parameter L note that the definition of Lipschitz continuity in Eq. (3) does not uniquely

Algorithm 1 Batch Bayesian Optimization with Local Penalization (BBO-LP).

Input: dataset $\mathcal{D}_1 = \{\mathbf{x}_i, y_i\}_{i=1}^n$, batch size n_b, iteration budget m, acquisition transformation g.
for $t = 1$ **to** m **do**
  Fit a GP to $\mathcal{D}_t$ and the acquisition function $\alpha(\mathbf{x}, \mathcal{I}_{t,0})$.
  $\tilde{\alpha}_{t,0}(\mathbf{x}) \leftarrow g(\alpha(\mathbf{x}, \mathcal{I}_{t,0}))$.
  $\hat{L} \leftarrow \max_{\mathcal{X}} \|\mu_\nabla(\mathbf{x})\|$.
  for $j = 1$ **to** n_b **do**
    1. *M-step:* $\mathbf{x}_{t,j} \leftarrow \arg\max_{x\in\mathcal{X}} \{\tilde{\alpha}_{t,j-1}(\mathbf{x})\}$.
    2. *P-step:* $\tilde{\alpha}_{t,j}(\mathbf{x}) \leftarrow \tilde{\alpha}_{t,0}(\mathbf{x}) \prod_{j=1}^k \varphi(\mathbf{x}; \mathbf{x}_{t,j}, \hat{L})$.
  end for
  $\mathcal{B}_t^{n_b} \leftarrow \{\mathbf{x}_{t,1}, \ldots, \mathbf{x}_{t,n_b}\}$.
  $y_{t,1}, \ldots, y_{t,n_b} \leftarrow$ Parallel evaluations of f at $\mathcal{B}_t^{n_b}$.
  $\mathcal{D}_{t+1} \leftarrow \mathcal{D}_t \cup \{(\mathbf{x}_{t,j}, y_{t,j})\}_{j=1}^{n_b}$.
end for
Fit GP to $\mathcal{D}_n$.
Returns: $\hat{\mathbf{x}}_M = \arg\max_{x\in\mathcal{X}} \{\mu(\mathbf{x})\}$.

identify L. In the BO penalization context, small but feasible values of L are preferred, because they produce large exclusion zones and thus more efficient search. Given access to the true objective f, one can show that $L_\nabla = \max_{\mathbf{x}\in\mathcal{X}} \|\nabla f(\mathbf{x})\|$ is a valid Lipschitz constant (see Supp. Materials for further details). Note that L_∇ is the smallest value of L that satisfies the Lipschitz condition Eq. (3) in the limit $\mathbf{x}_1 \to \mathbf{x}_2$ in (3).

We now construct an approximation for L_∇. Assuming that f is a draw from a GP with a (at least) twice differentiable kernel k, the gradient of f at $\mathbf{x}^*$ is distributed as a multivariate Gaussian $\nabla f(\mathbf{x}^*)|\mathbf{X}, \mathbf{y}, \mathbf{x}^* \sim \mathcal{N}(\mu_\nabla(\mathbf{x}^*), \Sigma_\nabla^2(\mathbf{x}^*))$ with mean vector

$$\mu_\nabla(\mathbf{x}^*) = \partial\mathbf{K}_{n,*}(\mathbf{x}^*)\tilde{\mathbf{K}}_n^{-1}\mathbf{y},$$

and covariance matrix

$$\Sigma_\nabla^2(\mathbf{x}^*) = \partial^2\mathbf{K}_{*,*} - \partial\mathbf{K}_{n,*}(\mathbf{x}^*)\tilde{\mathbf{K}}_n^{-1}\partial\mathbf{K}_{n,*}(\mathbf{x}^*)^\top$$

for $\tilde{\mathbf{K}}_n = \mathbf{K}_n + \sigma^2\mathbf{I}$ and where, for $i, j = 1, \ldots, d$ and $l = 1, \ldots, n$,

$$(\partial\mathbf{K}_{n,*})_{i,l} = \frac{\partial\mathbf{k}_N(\mathbf{x}^*)}{\partial x^{(i)}}, \quad (\partial^2\mathbf{K}_{*,*})_{ij} = \frac{\partial^2 k(\mathbf{x}^*, \mathbf{x}^*)}{\partial x^{(i)}\partial x^{(j)}}.$$

We choose

$$\hat{L}_{GP-LCA} = \max_{\mathcal{X}} \|\mu_\nabla(\mathbf{x}^*)\|$$

and call this the Gaussian Process Lipschitz Constant Approximation criterion (GP-LCA). Note that this definition of $\hat{L}_{GP-LCA}$ ignores the variance of the gradient, which could be used to identify candidate points to improve the approximation of L_∇ in a Bayesian optimization fashion. The supplement contains further experiments supporting the quality of this approximation. See Algorithm 1 for a description of all the steps described in this section.

2.3 Heteroscedastic scenarios

The use of an unique value of L assumes that the function to optimize is Lipschitz homocedastic. Although this is a typical hypothesis for most BO methods, recent works have pointed out that some real problems do not satisfy this condition [Assael et al., 2014]. It is not the goal of this work to analyze this case further but, interestingly, the method proposed here can be extended to non-Lipschitz cases by replacing L in the penalizers $\varphi(\mathbf{x}; \mathbf{x}_j, \hat{L})$ by a local values of L. For instance, a possible approach would be to replace the local penalizers by $\varphi(\mathbf{x}; \mathbf{x}_j, \hat{L}_j)$ where $\hat{L}_j = \|\mu_\nabla(\mathbf{x}_j)\|$.

2.4 Optimizing the penalized acquisition function

The optimization of (4) can be performed most easily by any gradient-based method in the log space because there, the gradients have an additive form. More formally, when the transformation used to make the acquisition positive is the *soft-plus* function, $g(z) = \ln(1 + e^z)$, the gradient of $\log\tilde{\alpha}_{t,k}(\mathbf{x}; \mathcal{I}_{t,0})$, being $\tilde{\alpha}_{t,k}(\mathbf{x}; \mathcal{I}_{t,0})$ the penalized acquisition in Eq. (4), is:

$$\begin{aligned}\nabla \ln\tilde{\alpha}_{t,k}(\mathbf{x}, \mathcal{I}_{t,0}) \quad = \quad & \frac{1}{\ln(1 + e^{\alpha(\mathbf{x};\mathcal{I}_{t,0})})}\frac{e^{\alpha(\mathbf{x};\mathcal{I}_{t,0})}}{1 + e^{\alpha(\mathbf{x};\mathcal{I}_{t,0})}} \cdot \\ & \nabla\alpha(\mathbf{x}; \mathcal{I}_{t,0}) + \sum_{j=1}^{k-1} \varphi(\mathbf{x}; \mathbf{x}_{t,j})^{-1} \cdot \\ & \nabla\varphi(\mathbf{x}; \mathbf{x}_{t,j}),\end{aligned}$$

where $\nabla\alpha(\mathbf{x}; \mathcal{I}_{t,0})$ is the (assumed known) gradient of the original acquisition function and $\nabla\varphi(\mathbf{x}; \mathbf{x}_{t,j})$ are the gradients of the local penalizers

$$\nabla\varphi(\mathbf{x}; \mathbf{x}_{t,j}) = \frac{e^{-z^2}}{\sqrt{2\pi\sigma_n^2(\mathbf{x}_j)}}\frac{2L}{\|\mathbf{x}_j - \mathbf{x}\|}(\mathbf{x}_j - \mathbf{x}),$$

See Supp. Materials for details.

3 Experimental Section

This section compares the performance of Algorithm 1 with the state-of-the-art methods for batch BO. We label the different methods by means of the batch design type followed by the acquisition used: Rand is used when the first element in the batch is collected maximizing the acquisition and the remaining ones randomly, B and PE denote the exploratory approaches in [Schonlau et al., 1998] and [Contal et al., 2013], Pred is used in cases when the model is used to generate 'fake'

d	n_b	EI	UCB	Rand-EI	Rand-UCB	SM-UCB	B-UCB
	5			0.32±0.05	**0.31±0.05**	1.86±1.06	0.56±0.03
2	10	0.31±0.03	0.32±0.06	0.65±0.32	0.79±0.42	4.40±2.97	**0.59±0.00**
	20			0.67±0.31	0.75±0.32	-	0.57±0.01
	5			9.19±5.32	10.59±5.04	137.2±113.0	**6.01±0.00**
5	10	8.84±3.69	11.89±9.44	1.74±1.47	2.20±1.85	108.7±74.38	3.77±0.00
	20			2.18±2.30	2.76±3.06	-	2.53±0.00
	5			**690.5±947.5**	1825±2149	9e+04±7e+04	2098±0.00
10	10	559.1±1014	1463±1803	200.9±455.9	1149±1830	9e+04±1e+05	857.8±0.00
	20			639.4±1204	385.9±642.9	-	1656±0.00

d	n_b	PE-UCB	Pred-EI	Pred-UCB	qEI	LP-EI	LP-UCB
	5	0.99±0.74	0.41±0.15	0.45±0.16	1.53±0.86	0.35±0.11	**0.31±0.06**
2	10	0.66±0.29	1.16±0.70	1.26±0.81	3.82±2.09	0.66±0.48	0.69±0.51
	20	0.75±0.44	1.28±0.93	1.34±0.77	-	**0.50±0.21**	0.58±0.21
	5	123.5±81.43	10.43±4.88	11.77±9.44	15.70±8.90	11.85±5.68	10.85±8.08
5	10	120.8±78.56	9.58±7.85	11.66±11.48	17.69±9.04	3.88±4.15	**1.88±2.46**
	20	98.60±82.60	8.58±8.13	10.86±10.89	-	6.53±4.12	**1.44±1.93**
	5	2e+05±2e+05	793.0±1226	1412±3032	-	1881±1176	1194±1428
10	10	6e+04±8e+04	442.6±717.9	1725±3205	-	1042±1562	**100.4±338.7**
	20	5e+04±4e+04	1091±1724	2231±3110	-	1249±1570	**20.75±50.12**

Table 1: Results for the gSobol function across different dimensions, batch sizes and methods. For each algorithm, the mean and standard deviation are shown. Best results among the batch methods are highlighted in bold. '-' represents that the method could not complete the first iteration within the time budget. The value of f at the minimum is always zero. EI and UCB represent the Expected improvement and the upper confidence bound acquisitions. Rand stands for the random batch collection. SM is the simulating and matching approach. Pred is the predictive approach and LP the local penalization method presented in this work. qEI is the multi-point expected improvement.

batch observations as in [Azimi et al., 2012], SM identifies the simulating and matching method [Azimi et al., 2010] and LP stands for our local penalization method. The multi-point expected improvement [Chevalier and Ginsbourger, 2013] is denoted by qEI. Two acquisition functions are used: the expected improvement (EI) defined as $\alpha_{EI}(\mathbf{x};\mathcal{I}_n) = (u\Phi(u) + \phi(u))\,\sigma_n(\mathbf{x})$, where $u = (\mu_n(\mathbf{x}) - y_{\min})/\sigma_n(\mathbf{x})$ and $\Phi(\cdot)$, $\phi(\cdot)$ are the standard Gaussian distribution and density functions respectively and $y_{\min}$ is the best current location and the Upper Confidence Bound (UCB) defined as $\alpha_{UCB}(\mathbf{x};\mathcal{I}_n) = \mu_n(\mathbf{x}) + \kappa\sigma_n(\mathbf{x})$, with $\kappa \geq 0$. The batch methods that can be used with an arbitrary acquisition function are tested using both, with the exception of the SM whose implementation is only available with the UCB. When used in a sequential setting (for baseline reference) the EI and UCB are referred by their acronyms. In total, we use 2 sequential and 10 batch methods. To run the B, PE, SM, methods, we use the available MATLAB code.[2] The implementation of these methods optimize f by searching its optimum in a fine grid, which is an advantage computationally but a drawback in terms of precision. The qEI was taken from the R-package DiceOptim.[3] Unless specified otherwise, the default implemented settings of all the previous methods are used.

We perform: (i) a simulation in which the performance of the algorithms is compared for a fixed time budget across different problem dimensions, batch sizes and acquisition functions and (ii) a comparison of the *gained information per second rate* in three objective functions with different evaluation costs. We always minimize the objective, minimizing $-f$ in examples in which the goal is to find maximum of f. In all the experiments the exponentiated quadratic (EQ) covariance $k(\mathbf{x},\mathbf{x}') = \theta\exp(-\gamma\|\mathbf{x}-\mathbf{x}'\|^2)$, $\theta,\gamma > 0$ is used in the GP, whose parameters are optimized by maximizing the marginal likelihood from the best of 10 random initializations. The results are taken over 20 replicates with different initial values. All the simulations were done on Amazon EC2 servers with Intel Xeon E5-2666 processors and 2 virtual CPUs except the SVR tuning

[2] http://econtal.perso.math.cnrs.fr/software/. Alternative implementations of GP-B-UCB is available at http://www.its.caltech.edu/ tadesaut/GPBUCBCode/ but we used the former one for comparative consistency.

[3] http://cran.r-project.org/web/packages/DiceOptim

with 16 virtual CPUs.

3.1 Comparisons in terms of the dimension, batch size and acquisition function

We consider the gSobol function (see Supp. Materials **??**) to compare the above mentioned methods across dimensions $d = 2, 5, 10$ and batch sizes, $n_b = 5, 10, 20$. For methods using the UCB, κ was fixed to 2, which allows us to compare the different batch designs using the same acquisition function. For dimension 2, 5 and 10, we use a time budget of 1, 5 and 10 mins. respectively. Table 3 shows the averaged best value found by each algorithm for all the iterations completed within the time limit. In general, the batch methods using the UCB show a better performance that methods using the EI, especially in dimensions 5 and 10. The overall best technique is the LP-UCB, that achieves the best results in 5 of the 9 cases. It is also notable that it exhibits fairly small standard deviations compared with the rest of the methods and it is coherent accumulating information about the optimum of f in terms of the batch size: as n_b increases the results are consistently better. In dimension 2 and batch size 5, the LP-EI is the best method. There are three cases in which the LP batch designs are not the most competitive (although still providing good results). Exploratory approaches works well in low dimensional cases, being the B-UCB the best method in two scenarios.

3.2 Comparisons in terms of the cost to evaluate the objective

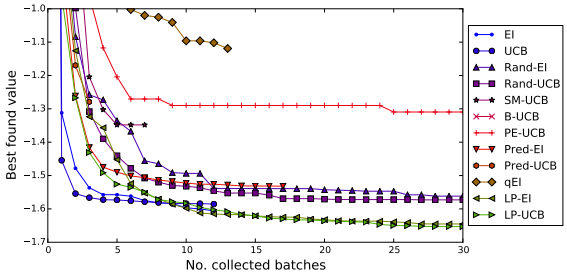
We choose three scenarios to compare the algorithms in terms of the running time. The examples correspond to three functions that are cheap, moderate and expensive to evaluate. More specifically, the first experiment uses a function (Cosines) that is inexpensive to evaluate but quite multi-modal. The second experiment is motivated by a wet-lab experimental design. We work with a surface that emulates the performance of mammalian cells in protein production given different gene designs. The function has dimension 71 and is is moderately expensive to evaluate since it corresponds to the predictive mean of a GP trained over 1,500 data instances. The qEI was not used in this experiment due to the huge computational effort required to jointly optimize the batches in dimension 71. The third experiment involves the tuning of the three parameters of a support vector regression (SVR) [Drucker et al., 1997] in a example with 45730 instances and 9 continuous attributes [Bache and Lichman, 2013]. The objective function is the cross-validation error of the model, which is expensive to evaluate due to the amount of data used. See Supp. Materials for further details. We take a batch size of $n_b = 5$ for the Cosines function and $n_b = 10$ for the wet-lab and SVR experiments. We compare the averaged best found results in terms of the number of collected batches and the wall-clock time. In the last experiment we use the SVR implementation available in scikit–learn[4] and only the methods implemented in python are used (EI, UCB, Rand-EI, Rand-UCB, Pred-EI, Pred-UCB, LP-EI and LP-UCB).

In the Cosines experiment both the sequential EI and UCB policies achieve the best results during the first 10 iterations of the algorithms (2 full batches). As the algorithms progress, however, a significant improvement is observed by the LP-EI and LP-UCB methods in terms of the number iterations and in terms of the wall-clock-time. When many points are collected, the update of the GP is more expensive and a good batch design is able to explore regions that the sequential method cannot. The rest of the batch methods, however, are not able to do this exploration efficiently, which leads to poorer results. Similar results are obtained for the wet-lab experiment. The LP-EI and LP-UCB are again the most competitive techniques improving the rest of the batch methods and the sequential policies. The differences are even more significant in this scenario. Since f is now more expensive to evaluate, the parallelization of the evaluations makes the search much more efficient, specially for the LP-UCB method. Regarding the last experiment, the cost of evaluating the function dominates the cost of designing the batch. In this case the performance of the different batch methods is comparable but significantly better than the sequential policies due to the parallel evaluations of f. The results for the three functions are coherent with those observed in Section 3.1 showing that the BBP-LP methods is overall the most efficient method for batches collection in BO.

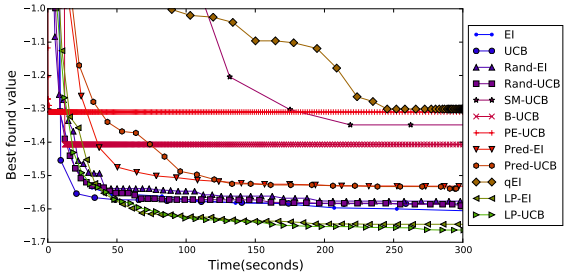
4 Discussion

We have investigated a new heuristic for batch BO, BBO-LP, that significantly reduces the computational burden of non-parallelizable tasks. The resulting method can be used with any acquisition function and it is able to make fast and appropriate decisions about the locations where f should be evaluated. When the batch evaluations of f are parallelizable this is an important advantage, meaning that they don't lead to considerable additional computational overhead. We have found other interesting results. In simple scenarios, batch policies based on random exploration work reasonably well in terms of the information gained per second. When the complexity of the problem increases, however, methods that make use of some information about f improve the random policy. In par-

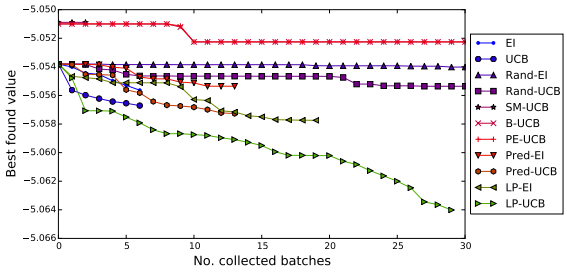
[4] http://scikit-learn.org/stable/index.html.



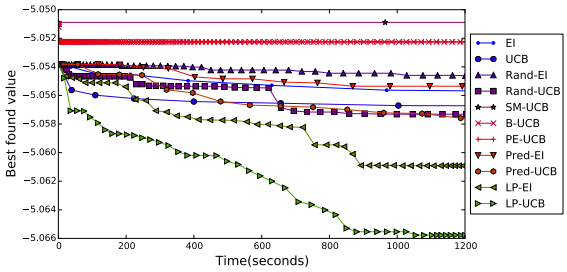
(a) Results for the Cosines function - batch iterations.



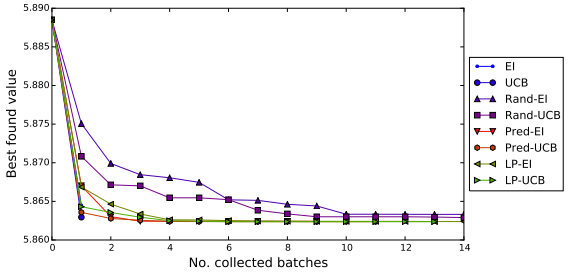
(b) Results for the Cosines function - running time.



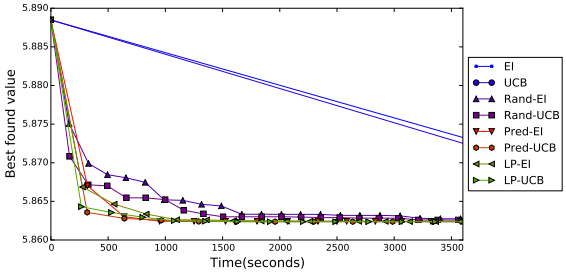
(c) Results for the wet-lab function - batch iterations.



(d) Results for the wet-lab function - running time.



(e) Results for the SVR function - batch iterations.



(f) Results for the SVR function - running time.

Figure 3: Results for the three test functions described in Section 3.2. Geometric figures on top of the lines represent the moments in which the batches are evaluated. See caption of Table 3 for details on the acronyms.

ticular, the approach here proposed makes use of the Lipschitz continuity of f to model the interaction between the elements in the batch. In spirit, this is similar to use the GP to predict the evaluations of f but, in practice, is much more efficient because it avoids the re-computation of the GP after every point is selected. The limitations of this approach are, however, determined by the ability to learn correctly a small enough, and valid, Lipschitz constant for f.

One could also wonder whether it is necessary to require that sample paths from the GP measure on f should be Lipschitz-continuous themselves. This would severely restrict the applicability of this notion, because the relationship between regularity of the kernel and the sample paths is complicated. Even if the kernel is Lipschitz-continuous, sample paths may not be Lipschitz [Adler, 1981]. However, our approach only tries to model the effect of evaluations on the BO objective, not the GP probability measure itself. Many BO objectives, in particular the EI and UCB, are smooth functions of only the sufficient statistics (mean and covariance function) of the GP posterior. Both the posterior mean and covariance function are members of the Reproducing kernel Hilbert space induced by the kernel (i.e. they are weighted sums of kernel functions). Thus, if the kernel is Lipschitz, so is the acquisition function, even if the GP measure itself has non-Lipschitz sample paths. Finally, note that our local repulsion criterion naturally suggests a Latin square design for the case when no functional values have been been acquired. The latin square design is widely suggested for this domain [Jones et al., 1998].

Acknowledgements: The authors found great inspiration for this work at the *1st Braitenberg Round table on Probabilistic numerics and Random Geometries*. The authors thank the financial support of RADIANT EU FP7-HEALTH Ref 305626 and BBSRC No BB/K011197/1 projects.

References

Robert J. Adler. *The geometry of random fields*. Wiley, 1981.

John-Alexander M. Assael, Ziyu Wang, and Nando de Freitas. Heteroscedastic treed bayesian optimisation. *CoRR*, abs/1410.7172, 2014.

Javad Azimi, Alan Fern, and Xiaoli Fern. Batch Bayesian optimization via simulation matching. In *Advances in Neural Information Processing Systems*, pages 109–117, 2010.

Javad Azimi, Ali Jalali, and Xiaoli Fern. Dynamic batch Bayesian optimization. *CoRR*, abs/1110.3347, 2011.

Javad Azimi, Ali Jalali, and Xiaoli Zhang Fern. Hybrid batch Bayesian optimization. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

Kevin Bache and Moshe Lichman. UCI machine learning repository, 2013.

James Bergstra, Rémy Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *NIPS'2011*, 2011.

Clment Chevalier and David Ginsbourger. Fast computation of the multi-points expected improvement with applications in batch selection. In Giuseppe Nicosia and Panos M. Pardalos, editors, *LION*, volume 7997 of *LNCS*, pages 59–69. Springer, 2013. ISBN 978-3-642-44972-7.

Emile Contal, David Buffoni, Alexandre Robicquet, and Nicolas Vayatis. Parallel Gaussian process optimization with upper confidence bound and pure exploration. *CoRR*, abs/1304.5350, 2013.

Thomas Desautels, Andreas Krause, and Joel W. Burdick. Parallelizing exploration-exploitation tradeoffs with Gaussian process bandit optimization. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

Harris Drucker, Chris, Burges L. Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression machines. In *Advances in Neural Information Processing Systems 9*, pages 155–161, 1997.

Christodoulos A. Floudas and Panos M. Pardalos, editors. *Encyclopedia of Optimization, Second Edition*. Springer, 2009.

P. I. Frazier. Parallel global optimization using an improved multi-points expected improvement criterion. In *INFORMS Optimization Society Conference, Miami FL*, 2012.

David Ginsbourger, Janis Janusevskis, and Rodolphe Le Riche. Dealing with asynchronicity in parallel Gaussian Process based global optimization. Technical report, 2011.

Javier González, Joseph Longworth, David James, and Neil Lawrence. Bayesian optimisation for synthetic gene design. *NIPS Workshop on Bayesian Optimization in Academia and Industry*, 2014.

Philipp Hennig and Christian J. Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13, 2012.

José M. Hernández-Lobato, Matthew W. Hoffman, and Zoubin Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. In *Advances in Neural Information Processing Systems 27*, pages 918–926. Curran Associates, Inc., 2014.

Reiner Horst and Panos M. Pardalos, editors. *Handbook of global optimization*. Nonconvex optimization and its applications. Kluwer Academic Publishers, Dordrecht, Boston, 1995.

Ali Jalali, Javad Azimi, Xiaoli Fern, and Ruofei Zhang. A lipschitz exploration-exploitation scheme for Bayesian optimization. In *Machine Learning and Knowledge Discovery in Databases*, pages 210–224, 2013.

Janis Janusevskis, Rodolphe Le Riche, David Ginsbourger, and Ramunas Girdziusas. Expected improvements for the asynchronous parallel global optimization of expensive functions: Potentials and challenges. In Y. Hamadi and M. Schoenauer, editors, *LION*, volume 7219 of *LNCS*, pages 413–418. Springer, 2012.

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

Michael Osborne. *Bayesian Gaussian Processes for Sequential Prediction, Optimisation and Quadrature*. PhD thesis, PhD thesis, University of Oxford, 2010.

Ji Oenek and Josef Schwarz. The parallel Bayesian optimization algorithm. In Peter Sink, Jn Vak, Vladimr Kvasnika, and Radko Mesiar, editors, *The State of the Art in Computational Intelligence*, volume 5 of *Advances in Soft Computing*, pages 61–67. Physica-Verlag HD, 2000.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

Matthias Schonlau, William J. Welch, and Donald R. Jones. *Global versus local search in constrained optimization of computer models*, volume Volume 34 of

Lecture Notes–Monograph Series, pages 11–25. Institute of Mathematical Statistics, Hayward, CA, 1998.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems 25*, pages 2960–2968, 2012.

Niranjan Srinivas, Andreas Krause, Matthias Seeger, and Sham M. Kakade. Gaussian process optimization in the bandit setting: No regret and experimental design. In Johannes Frnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 1015–1022. Omnipress, 2010.

Roman G. Strongin and Yaroslav D. Sergeyev. *Global optimization with non-convex constraints : sequential and parallel algorithms*. Nonconvex Optimization and Its Applications. Kluwer academic publishers, Dordrecht, Boston, Londres, 2000.

Matthew A. Taddy, Herbert K. H. Lee, Genetha A. Gray, and Joshua D. Griffin. Bayesian guided pattern search for robust local optimization. *Technometrics*, 51(4):389–401, 2009. doi: 10.1198/TECH.2009.08007.