A sequential Monte Carlo approach to Thompson sampling for Bayesian optimization

Hildo Bijl H.J.BIJL@TUDELFT.NL
Delft Center for Systems and Control
Delft University of Technology
Delft, The Netherlands

Thomas B. Schön THOMAS.SCHON@IT.UU.SE
Department of Information Technology
Uppsala University
Uppsala, Sweden

Jan-Willem van Wingerden J.W.VANWINGERDEN@TUDELFT.NL
Delft Center for Systems and Control
Delft University of Technology
Delft, The Netherlands

Michel Verhaegen M.VERHAEGEN@TUDELFT.NL
Delft Center for Systems and Control
Delft University of Technology
Delft, The Netherlands

Abstract

Bayesian optimization through Gaussian process regression is an effective method of optimizing an unknown function for which every measurement is expensive. It approximates the objective function and then recommends a new measurement point to try out. This recommendation is usually selected by optimizing a given acquisition function. After a sufficient number of measurements, a recommendation about the maximum is made. However, a key realization is that the maximum of a Gaussian process is not a deterministic point, but a random variable with a distribution of its own. This distribution cannot be calculated analytically. Our main contribution is an algorithm, inspired by sequential Monte Carlo samplers, that approximates this maximum distribution. Subsequently, by taking samples from this distribution, we enable Thompson sampling to be applied to (armed-bandit) optimization problems with a continuous input space. All this is done without requiring the optimization of a nonlinear acquisition function. Experiments have shown that the resulting optimization method has a competitive performance at keeping the cumulative regret limited.

Keywords: Gaussian processes, optimization methods, particle methods, model-free, controller tuning.

1. Introduction

Consider the problem of maximizing a continuous nonlinear reward function $f(\boldsymbol{x})$ (or equivalently minimizing a cost function) over a compact set X_f. In the case where $f(\boldsymbol{x})$ can be easily evaluated, where derivative data is available and where the function is convex (or

concave), the solution is relatively straightforward, as is for instance discussed by Boyd and Vandenberghe (2004). However, we will consider the case where convexity and derivative data are not known. In addition, every function evaluation is expensive and we can only obtain noisy measurements of the function. In this case, as was also indicated by Jones et al. (1998), we need a data-driven approach to optimize the function.

The main idea is to try out certain inputs $\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}$. After selecting a so-called *try-out input* $\boldsymbol{x_k}$, we feed it into the function and obtain a noisy measurement $y_k = f(\boldsymbol{x_k}) + \varepsilon$, with $\varepsilon = \mathcal{N}\left(0, \sigma_n^2\right)$. We then use all measurements obtained so far to make a Bayesian approximation of the function $f(\boldsymbol{x})$, based on which we choose the next try-out input $\boldsymbol{x_{k+1}}$. As such, this problem is known as *Bayesian optimization* (Lizotte, 2008; Brochu et al., 2010; Shahriari et al., 2016). In particular, we can approximate $f(\boldsymbol{x})$ through Gaussian process regression (Rasmussen and Williams, 2006). This gives us a mean $\mu(\boldsymbol{x})$ and a standard deviation $\sigma(\boldsymbol{x})$ for our estimate of $f(\boldsymbol{x})$. The resulting optimization method is also known as *Gaussian process optimization* (Osborne et al., 2009). Bayesian methods like Gaussian process regression are known to efficiently deal with data, requiring only little data to make relatively accurate approximations. This makes these techniques suitable for a data-driven approach to problems in which data is expensive.

The main question is how to choose the try-out inputs $\boldsymbol{x_k}$. There are two different problem formulations available. In the first, after performing all n measurements, we have to give a recommendation $\hat{\boldsymbol{x}}^*$ of what we believe is the true optimum $\boldsymbol{x}^*$. The difference $f^* - \hat{f}^*$ between the corresponding function values $f^* \equiv f(\boldsymbol{x}^*)$ and $\hat{f}^* \equiv f(\hat{\boldsymbol{x}}^*)$ is known as the *error* or the *instantaneous regret*. As such, this problem formulation is known as the *error minimization formulation* or also as the *probabilistic global optimization* problem. It is useful in applications like sensor placement (Osborne, 2010) and controller tuning in damage-free environments (Lizotte et al., 2007; Marco et al., 2016). These are all applications in which every try-out input (every experiment) has the same high set-up cost.

In the second problem formulation, our aim is to maximize the sum of all the rewards $f(\boldsymbol{x_1}), \ldots, f(\boldsymbol{x_n})$, which is known as the *value* V. Equivalently, we could also minimize the *(cumulative) regret*

$$\sum_{k=1}^{n} \left(f^* - f(\boldsymbol{x_k})\right) = nf^* - V. \tag{1}$$

This formulation is known as the *regret minimization formulation* or also as the *continuous armed bandit problem*. It is useful in applications like advertisement optimization (Pandey and Olston, 2007) or controller tuning for damage minimization (see Section 4.4). These are applications where the reward or cost of an experiment actually depends on the result of the experiment. Because our applications fall in the latter category, we will focus on the regret minimization formulation in this paper. However, with the two formulations being similar, we also take error minimization strategies into account.

The main contribution of this paper is a new algorithm, inspired by the sequential Monte Carlo method (Del Moral et al., 2006), that approximates the maximum distribution. This algorithm can then be used to sample from the maximum distribution. This enables us to formulate an efficient Bayesian optimization algorithm with Thompson sampling for problems with a continuous input space, which is highly suitable for regret minimization

problems. To the best of our knowledge, such an approach has not been applied before in literature and it is hence our main contribution.

We start by providing links to related work in Section 2. We will then present our main developments resulting in the Monte Carlo maximum distribution algorithm for approximating the distribution of the maximum in Section 3. We also analyze it and examine how we can use it to apply Thompson sampling. Experimental results are presented in Section 4, with conclusions and recommendations given in Section 5.

2. Related work

Both the error minimization and the regret minimization problems have been examined in literature before. In this section we examine the solutions that have already been proposed.

2.1 Existing error minimization methods

Several Bayesian optimization methods already exist. Good overviews are given by Lizotte (2008); Brochu et al. (2010); Shahriari et al. (2016), though we will provide a brief summary here. The recurring theme is that, when selecting the next input $\boldsymbol{x_k}$, we optimize some kind of *Acquisition Function*. In the literature, the discussion is mainly concerned with selecting and tuning an acquisition function.

The first to suggest the *Probability of Improvement* (PI) acquisition function was Kushner (1964). This function is defined as $\mathrm{PI}(\boldsymbol{x}) = \mathbb{P}(f(\boldsymbol{x}) \geq y_+)$, where $\mathbb{P}(A)$ denotes the probability of event A to occur and y_+ denotes the highest value of the observation obtained so far. This was expanded by Torn and Zilinskas (1989); Jones (2001) to the form $\mathrm{PI}(\boldsymbol{x}) = \mathbb{P}(f(\boldsymbol{x}) \geq y_+ + \xi)$, with ξ being a tuning parameter trading off between exploration (high ξ) and exploitation (zero ξ).

Later on, Mockus et al. (1978) suggested an acquisition function which also takes the magnitude of the potential improvement into account. It is known as the *Expected Improvement* (EI) acquisition function $\mathrm{EI}(\boldsymbol{x}) = \mathbb{E}\left[\max(0, f(\boldsymbol{x}) - y_+)\right]$. Similar methods were used by others. For instance, multi-step lookahead was added by Osborne (2010), a trust region to ensure small changes to the tried inputs $\boldsymbol{x_k}$ was used by Park and Law (2015), and an additional exploration/exploitation parameter ξ similar to the one used in the PI acquisition function was introduced by Brochu et al. (2010). An analysis was performed by Vazquez and Bect (2010).

Alternatively, Cox and John (1997) suggested the *Upper Confidence Bound* (UCB) acquisition function $\mathrm{UCB}(\boldsymbol{x}) = \mu(\boldsymbol{x}) + \kappa\sigma(\boldsymbol{x})$. Here the parameter κ determines the amount of exploration/exploitation, with high values resulting in more exploration. Often $\kappa = 2$ is used. The extreme case of $\kappa = 0$ is also known as the *Expected Value* (EV) acquisition function $\mathrm{EV}(\boldsymbol{x}) = \mu(\boldsymbol{x})$. It applies only exploitation, so it is not very useful by itself. Methods to determine the value of κ optimizing regret bounds were studied by Srinivas et al. (2012).

A significant shift in focus was made through the introduction of the so-called *entropy search* method. This method was first developed by Villemonteix et al. (2009), although Hennig and Schuler (2012) independently set up a similar method and introduced the name entropy search. The method was subsequently developed further as *predictive entropy search* by Hernández-Lobato et al. (2014). The main idea here is to look at the so-called *maximum distribution*: the probability $p_{\max}(\boldsymbol{x}) \equiv \mathbb{P}(\boldsymbol{x} = \boldsymbol{x^*})$ that a certain point

$\boldsymbol{x}$ equals the (unknown) optimum $\boldsymbol{x}^*$, or for continuous problems the corresponding probability density. We then focus on the relative entropy (the Kullback-Leibler divergence) of the maximum distribution compared to the flat probability density function over X_f. Initially this relative entropy is zero, but the more information we gain, the higher this relative entropy becomes. As such, we want to pick the try-out point $\boldsymbol{x_k}$ which is expected to increase the relative entropy the most.

At the same time, *portfolio methods* were developed with the aim to optimally use a whole assortment (a portfolio) of acquisition functions. These methods were introduced by Hoffman et al. (2011), using results from Auer et al. (1995); Chaudhuri et al. (2009) and subsequently expanded on by Shahriari et al. (2014), who suggested to use the change in entropy as criterion to select recommendations.

2.2 Existing regret minimization methods

In the error minimization formulation, the focus is on obtaining as much information as possible. The regret minimization formulation is more involved, since it requires a trade-off between obtaining information and incurring costs (regret). Here, most of the research has focused on the case where the number of possible inputs $\boldsymbol{x}$ is finite. It is then known as the armed bandit problem and has been analyzed by for instance Kleinberg (2004); Grünewälder et al. (2010); de Freitas et al. (2012).

One of the more promising acquisition methods for the armed bandit problem is *Thompson sampling*. This method was first suggested by Thompson (1933) and has more recently been analyzed by Chapelle and Li (2011); Agrawal and Goyal (2012). It is fundamentally different from other methods, because it does not use an acquisition function. Instead, we select an input point $\boldsymbol{x}$ as the next try-out point $\boldsymbol{x_k}$ with probability equal to the probability that $\boldsymbol{x}$ is the optimal input $\boldsymbol{x}^*$. This is equivalent to sampling $\boldsymbol{x_k}$ from the maximum distribution $p_{\max}(\boldsymbol{x})$. Generally this distribution is not known though. When only finitely many different input points $\boldsymbol{x}$ are possible, the solution is to consider the vector $\boldsymbol{f} \equiv f(X)$ of all possible function outputs. Using Bayesian methods, we approximate $\boldsymbol{f}$ as a random variable, take a sample $\hat{\boldsymbol{f}}$ from it, find for which input point $\boldsymbol{x}$ this sample has its maximum, and subsequently use that input $\boldsymbol{x}$ as the next try-out point $\boldsymbol{x_k}$.

This method has proven to work well when the number of input points is finite. When there are infinitely many possible input points, like in continuous problems, it is impossible to sample from $\boldsymbol{f}$. This means that a new method to sample from the maximum distribution $p_{\max}(\boldsymbol{x})$ is needed. However, in the existing literature this maximum distribution is not studied much at all. The idea of it was noted (but not evaluated) by Lizotte (2008). The maximum distribution was calculated by Villemonteix et al. (2009) through a brute force method. An expansion to this was developed by Hennig and Schuler (2012), who used a method from Minka (2001) to approximate the minimum distribution. Though the approximation method used is quite accurate, it has a runtime of $\mathcal{O}\left(n^4\right)$, making it infeasible to apply to most problems. An alternative method was described by Hernández-Lobato et al. (2014) who approximated function samples of a Gaussian process through a finite number of basis functions and then optimized these function samples to generate samples from the maximum distribution. Though effective, this method requires solving a

nonlinear optimization problem for each sample, which is computationally expensive and subject to the risk of finding only a local optimum.

3. Finding the maximum distribution

In this section we introduce an algorithm to find/approximate the distribution of the maximum of a Gaussian process. We then apply this algorithm to implement Thompson sampling.

3.1 A Gaussian process and its maximum

Consider a function $f(\boldsymbol{x})$. We assume that we have taken n measurements $y_i = f(\boldsymbol{x_i}) + \varepsilon$, where $\varepsilon \sim \mathcal{N}\left(0, \sigma_n^2\right)$ is Gaussian white noise. We merge all the measurement (training) input points $\boldsymbol{x_i}$ into a set X and all the measured output values y_i into a vector $\boldsymbol{y}$.

Now suppose that we want to predict the (noiseless) function values $\boldsymbol{f_*} = f(X_*)$ at a given set of trial (test) input points X_*. In this case we can use the standard GP regression equation from Rasmussen and Williams (2006). We use a mean function $m(\boldsymbol{x})$ and a covariance function $k(\boldsymbol{x}, \boldsymbol{x'})$, and we shorten $m(X_a)$ to $\boldsymbol{m_a}$ and $k(X_a, X_b)$ to K_{ab} for any subscripts a and b. (The subscript for the training set is omitted.) We then have

$$\begin{aligned} \boldsymbol{f_*} &\sim \mathcal{N}\left(\boldsymbol{\mu_*}, \Sigma_{**}\right), \\ \boldsymbol{\mu_*} &= \boldsymbol{m_*} + K_*^T \left(K + \Sigma_n\right)^{-1} \left(\boldsymbol{y} - \boldsymbol{m}\right), \\ \Sigma_{**} &= K_{**} - K_*^T \left(K + \Sigma_n\right)^{-1} K_*. \end{aligned} \tag{2}$$

A Gaussian process can be seen as a distribution over functions. That is, we can take samples of $\boldsymbol{f_*}$ and plot those as if they are functions, as is for instance done in Figure 1. These sample functions generally have their maximum at different locations $\boldsymbol{x^*}$. This implies that $\boldsymbol{x^*}$ is a random variable, and hence has a distribution $p_{\max}(\boldsymbol{x})$. An example of this is shown in Figure 2.

The distribution $p_{\max}(\boldsymbol{x})$ cannot be analytically calculated, but it can be approximated through various methods. The most obvious one is through brute force: for a finite number of trial input points X_*, we take a large number of samples $\boldsymbol{f_*}$, for each of these samples we find the location of the maximum, and through a histogram we determine the distribution of $\boldsymbol{x^*}$. This method is far from ideal as it is computationally very intensive, even for low-dimensional functions, but it is guaranteed to converge to the *true maximum distribution*.

For larger problems the brute force method is too computationally intensive, motivating the need for a way of approximating the maximum distribution. Methods to do so already exist, like those used by Hennig and Schuler (2012); Hernández-Lobato et al. (2014). However, these methods are all also computationally intensive for larger problems, and so a different way to approximate $p_{\max}(\boldsymbol{x})$ would be beneficial.

3.2 Approximating the maximum distribution

We propose a new algorithm, inspired by Sequential Monte Carlo (SMC) samplers, to find the maximum distribution $p_{\max}(\boldsymbol{x})$. Note that the algorithm presented here is *not* an actual SMC sampler, but merely uses techniques also found in SMC samplers. For more background, see e.g. Del Moral et al. (2006); Owen (2013).

The main idea is that we have n_p so-called particles at positions $\boldsymbol{x^1}, \ldots, \boldsymbol{x^{n_p}}$. Each of these particles has a corresponding weight $w^1, \ldots, w^{n_p}$. Eventually these particles are supposed to converge to the maximum distribution, at which time we can approximate this distribution through kernel density estimation as

$$p_{\max}(\boldsymbol{x}) \approx \frac{\sum_{i=1}^{n_p} w^i k_x(\boldsymbol{x}, \boldsymbol{x^i})}{\sum_{i=1}^{n_p} w^i}, \tag{3}$$

with $k_x(\boldsymbol{x}, \boldsymbol{x'})$ some manually chosen kernel. It is common to make use of a squared exponential kernel with a small length scale.

Initially we distribute these particles $\boldsymbol{x^i}$ at random positions across the input space. That is, we sample the particles $\boldsymbol{x^i}$ from the flat distribution $q(\boldsymbol{x}) = c$. Note that, because we have assumed that the input space X_f is compact, the constant c is nonzero.

To learn more about the position of the maximum, we will *challenge* existing particles. To challenge an existing particle $\boldsymbol{x^i}$, we first sample a number n_c of random challenger particles $\boldsymbol{x^i_{c_1}}, \ldots, \boldsymbol{x^i_{c_{n_c}}}$ from a proposal distribution $q'(\boldsymbol{x})$. We then set up the joint distribution

$$\begin{bmatrix} f(\boldsymbol{x^i}) \\ f(\boldsymbol{x^i_{c_1}}) \\ \vdots \\ f(\boldsymbol{x^i_{c_{n_c}}}) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(\boldsymbol{x^i}) \\ \mu(\boldsymbol{x^i_{c_1}}) \\ \vdots \\ \mu(\boldsymbol{x^i_{c_{n_c}}}) \end{bmatrix}, \begin{bmatrix} \Sigma(\boldsymbol{x^i}, \boldsymbol{x^i}) & \Sigma(\boldsymbol{x^i}, \boldsymbol{x^i_{c_1}}) & \cdots & \Sigma(\boldsymbol{x^i}, \boldsymbol{x^i_{c_{n_c}}}) \\ \Sigma(\boldsymbol{x^i_{c_1}}, \boldsymbol{x^i}) & \Sigma(\boldsymbol{x^i_{c_1}}, \boldsymbol{x^i_{c_1}}) & \cdots & \Sigma(\boldsymbol{x^i_{c_1}}, \boldsymbol{x^i_{c_{n_c}}}) \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma(\boldsymbol{x^i_{c_{n_c}}}, \boldsymbol{x^i}) & \Sigma(\boldsymbol{x^i_{c_{n_c}}}, \boldsymbol{x^i_{c_1}}) & \cdots & \Sigma(\boldsymbol{x^i_{c_{n_c}}}, \boldsymbol{x^i_{c_{n_c}}}) \end{bmatrix} \right), \tag{4}$$

and subsequently generate a sample $\begin{bmatrix} \hat{f}^i & \hat{f}^i_{c_1} & \cdots & \hat{f}^i_{c_{n_c}} \end{bmatrix}^T$ from it. Finally, we find the largest element from this vector. If this element equals $\hat{f}^i$, we do nothing. If, however, it equals $\hat{f}^i_{c_j}$, then we have $\hat{f}^i_{c_j} > \hat{f}^i$. In this case there is a challenger that has 'beaten' the current particle and it takes its place. In other words, we replace the particle $\boldsymbol{x^i}$ by $\boldsymbol{x^i_{c_j}}$.

The challenger particle also has a weight associated with. In SMC methods this weight is usually given by

$$w^i_c = \frac{q(\boldsymbol{x^i_c})}{q'(\boldsymbol{x^i_c})}. \tag{5}$$

However, to speed up convergence, we use a proposal distribution $q'(\boldsymbol{x})$ based on the ideas of mixture importance sampling and defensive importance sampling. Specifically, we use

$$q'(\boldsymbol{x}) = \alpha p_{\max}(\boldsymbol{x}) + (1 - \alpha) q(\boldsymbol{x}). \tag{6}$$

Here, α is manually chosen (often roughly $\frac{1}{2}$) and $p_{\max}(\boldsymbol{x})$ is approximated through the mixture proposal distribution (3), based on the current particle distribution. To generate a challenger particle $\boldsymbol{x^i_{c_j}}$, we hence randomly (according to the particle weights) select one of the particles $\boldsymbol{x^k}$. Then, in a part α of the cases, we sample $\boldsymbol{x^i_{c_j}}$ from $k_x(\boldsymbol{x}, \boldsymbol{x^k})$, while in the remaining $(1 - \alpha)$ part of the cases, we sample $\boldsymbol{x^i_{c_j}}$ from $q(\boldsymbol{x})$. If we sample our challenger particles in this way, it is computationally more efficient to use the weight

$$w^i_{c_j} = \frac{q(\boldsymbol{x^i_{c_j}})}{\alpha k_x(\boldsymbol{x^i_{c_j}}, \boldsymbol{x^k}) + (1 - \alpha) q(\boldsymbol{x^i_{c_j}})}. \tag{7}$$

Based on this formulation, we will challenge every existing particle once. This is called one *round* of challenges. Afterwards, we apply systematic resampling (Kitagawa, 1996) to make sure all particles have the same weight again. We repeat this until the distribution of particles has mostly converged.

We call the resulting algorithm the *Monte Carlo Maximum Distribution* (MCMD) algorithm. Pseudo-code for it is given in Algorithm 1.

Data: A known Gaussian process, user-defined parameters n_p, n_c, α and a kernel $k_x(\boldsymbol{x}, \boldsymbol{x'})$.
Result: An approximate distribution $p_{\max}(\boldsymbol{x})$ of the optimal input $\boldsymbol{x^*}$, given through (3).
Initialization:
  for $i \leftarrow 1$ **to** n_p **do**
    Sample $\boldsymbol{x^i}$ from $q(\boldsymbol{x})$. Assign $w^i = 1$.
  end
end
Iteration:
  repeat
    Apply systematic resampling to all particles.
    for $i \leftarrow 1$ **to** n_p **do**
      for $j \leftarrow 1$ **to** n_c **do**
        Select a random particle $\boldsymbol{x^k}$, taking into account weights w^k.
        if *we select a challenger based on* $\boldsymbol{x^k}$ *(probability* α*)* **then**
          Sample a challenger particle $\boldsymbol{x^i_{c_j}}$ from the kernel $k_x(\boldsymbol{x}, \boldsymbol{x^j})$.
        else
          Sample a challenger particle $\boldsymbol{x^i_{c_j}}$ from the flat distribution $q(\boldsymbol{x})$.
        end
      end
      Sample a vector $\begin{bmatrix} \hat{f}^i & \hat{f}^i_{c_1} & \cdots & \hat{f}^i_{c_{n_c}} \end{bmatrix}^T$ based on (4) and find its maximum.
      if *the maximum equals* $\hat{f}^i_{c_j} > \hat{f}^i$ **then**
        Replace particle $\boldsymbol{x^i}$ by its challenger $\boldsymbol{x^i_{c_j}}$.
        Set the new weight w^i according to (7).
      end
    end
  until *a sufficient number of rounds has passed*;
end

Algorithm 1: The Monte Carlo maximum distribution algorithm. Self-normalized importance sampling, mixture importance sampling and systematic resampling are used.

3.3 Analysing the limit distribution of the algorithm

The distribution of the particles converges to a *limit distribution*. But does this limit distribution equal the true maximum distribution? We can answer this question for a few special cases.

First consider the case where $n_c \to \infty$. In this case, the algorithm is equivalent to the brute force method of finding the maximum distribution. Assuming that a sufficient number of particles n_p is used, it hence is guaranteed to find the true maximum distribution directly, in only a single round of challenges.

Using $n_c \to \infty$ challenger particles is infeasible, because generating a sample from (4) takes $\mathcal{O}(n_c^3)$ time. Instead, we consider a very simplified case with $n_c = 1$ and $\alpha = 0$. Additionally, we consider the discrete case, where there are finitely many possible input points $\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}$. With finitely many points, we can use the kernel $k_x(\boldsymbol{x}, \boldsymbol{x'}) = \delta(\boldsymbol{x} - \boldsymbol{x'})$, with $\delta(\ldots)$ the delta function. In this simplified case, we can analytically calculate the distribution that the algorithm converges to.

Consider the given Gaussian process. Let us denote the probability $\mathbb{P}(f(\boldsymbol{x_i}) > f(\boldsymbol{x_j}))$, based on the data in this Gaussian process, as P_{ij}. Here we have

$$P_{ij} = \mathbb{P}(f(\boldsymbol{x_i}) > f(\boldsymbol{x_j})) = \Phi\left(\frac{\mu(\boldsymbol{x_i}) - \mu(\boldsymbol{x_j})}{\sqrt{\Sigma(\boldsymbol{x_i}, \boldsymbol{x_i}) + \Sigma(\boldsymbol{x_j}, \boldsymbol{x_j}) - 2\Sigma(\boldsymbol{x_i}, \boldsymbol{x_j})}}\right), \tag{8}$$

where $\Phi(\ldots)$ is the standard Gaussian cumulative density function. Through this expression we find the matrix P element-wise. Additionally, we write the part of the particles that will eventually be connected to the input $\boldsymbol{x_i}$ as p_i. In this case, the resulting vector $\boldsymbol{p}$ (with elements p_i) can be shown to satisfy

$$(P - \operatorname{diag}(1_n P))\,\boldsymbol{p} = \boldsymbol{0}, \tag{9}$$

where 1_n is an $n \times n$ matrix filled with ones, and $\operatorname{diag}(\ldots)$ is the function which sets all non-diagonal elements to zero. If we also use the fact that the sum of all probabilities $\boldsymbol{1}^T\boldsymbol{p}$ equals 1, we can find $\boldsymbol{p}$ for this discrete problem.

If the algorithm would converge to the true maximum distribution in this simplified case (with $n_c = 1$) then we must have $p_i = p_{\max}(\boldsymbol{x_i})$. In other words, the vector $\boldsymbol{p}$ would then describe the maximum distribution. However, since we can calculate the values p_i analytically, while it is known to be impossible to find $p_{\max}(\boldsymbol{x_i})$ like this, we already know that this is not the case. p_i must be different from $p_{\max}(\boldsymbol{x_i})$, and the algorithm hence does *not* converge to the maximum distribution when $n_c = 1$. However, the example from Figure 2 on page 11 does show that the algorithm gives a fair approximation. The limit distribution of the algorithm is generally less peaked than the true maximum distribution, which means it contains less information about where the maximum is (lower relative entropy) but overall its predictions are accurate. Furthermore, the difference will decrease when the variance present within the Gaussian process decreases, or when we raise n_c.

3.4 Applying the MCMD algorithm for Thompson sampling

We can now use the MCMD algorithm to apply Thompson sampling in a Gaussian process optimization setting. To do so, we sample an input point $\boldsymbol{x}$ from the approximated maximum distribution $p_{\max}(\boldsymbol{x})$ whenever we need to perform a new measurement.

The downside of this method is that samples are not drawn from the true maximum distribution, but only from an approximation of it. However, the upside is that this approximation can be obtained by making simple comparisons between function values. No large matrix equations need to be solved or nonlinear function optimizations need to be performed, providing a significant computational advantage over other methods that approximate the maximum distribution.

4. Experimental results

Here we show the results of the presented algorithm. First we study how the MCMD algorithm works for a fixed one-dimensional Gaussian process. Then we apply it through Thompson sampling to optimize the same function, expand to a two-dimensional problem and finally apply it to a real-life application. Code related to the experiments can be found through Bijl (2017b) (Chapter 6).

4.1 Execution of the MCMD algorithm

Consider the function

$$f(x) = \cos(3x) - \frac{1}{9}x^2 + \frac{1}{6}x. \tag{10}$$

From this function, we take 20 noisy measurements, at random locations in the interval $[-3, 3]$, with $\sigma_n = 0.3$ as standard deviation of the white noise. We then apply GP regression with a squared exponential covariance function with predetermined hyperparameters. The subsequent GP approximating these measurements is shown in Figure 1.

We can apply the MCMD algorithm to approximate the maximum distribution $p_{\max}(\boldsymbol{x})$ of this Gaussian process. This approximation, during successive challenge rounds of the algorithm with $\alpha = \frac{1}{2}$ and $n_c = 1$, is shown in Figure 2. (We always use $n_c = 1$ in these experiments, because it allows us to analytically calculate the limit distribution. For real-life experiments we would recommend larger values.) In this figure we see that the algorithm has mostly converged to the limit distribution after $n_r = 10$ rounds of challenges, but this limit distribution has a slightly higher entropy compared to the true maximum distribution.

4.2 Application to an optimization problem

We will now apply the newly developed method for Thompson sampling to Bayesian optimization. We will compare it with the UCB, the PI and the EI acquisition functions. After some tuning their parameters were set to $\kappa = 2$ and $\xi = 0.1$, which gave the best results we could obtain for these algorithms. To optimize these acquisition functions, we use a multi-start optimization method, because otherwise we occasionally end up with a local optimum of the acquisition function, resulting in a detrimental performance. We do not compare our results with entropy search or portfolio methods, because they are designed for the error minimization formulation.

The first problem we apply these methods to is the maximization of the function $f(x)$ of (10). We use $n = 50$ input points $x_1, \ldots, x_n$ and look at the obtained regret. To keep the memory and runtime requirements of the GP regression algorithm somewhat limited, given the large number of experiments that will be run, we will apply the FITC approximation

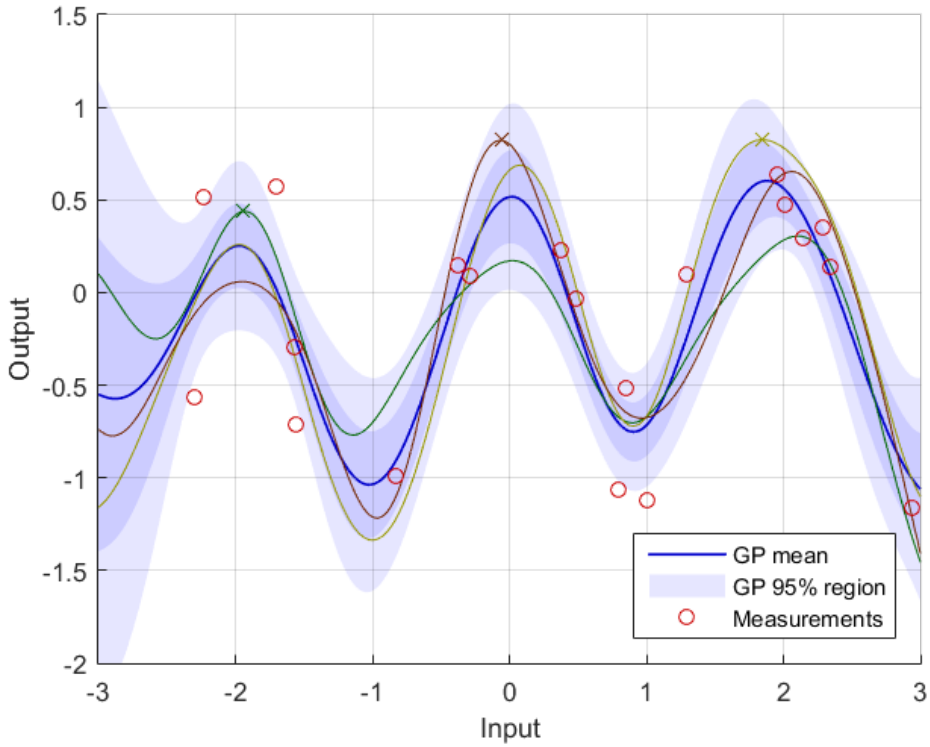

Figure 1: An example Gaussian process. The circles denote the measurements from which the GP was generated. The thick line denotes the (posterior) mean of the GP and the grey area represents the 95% certainty region. The three thinner lines are samples from the GP distribution. It is worthwhile to note that they have their maximum values (the crosses) at very different positions.

described by Candela and Rasmussen (2005), implemented in an online fashion according to Bijl et al. (2015). As inducing input points, we use the chosen input points, but only when they are not within a distance d_u (decreasing from 0.3 to 0.02 during the execution of the algorithm) of any already existing inducing input point. For simplicity the hyperparameters are assumed known and are hence fixed to reasonable values. Naturally, it is also possible to learn hyperparameters on-the-go as well, using the techniques described by Rasmussen and Williams (2006).

The result is shown in Figure 3. In this particular case, it seems that Thompson sampling and the PI acquisition function applied mostly exploitation: they have a better short term performance. On the other hand, the UCB and EI acquisition functions apply more exploration: the cost of quickly exploring is higher, but because the optimum is found sooner, it can also be exploited sooner.

It should also be noted that all algorithms occasionally end up in the wrong optimum (near $x = 2$). This can be seen from the fact that the regret graph does not level out. For this particular problem, the UCB acquisition function seems to be the best at avoiding the

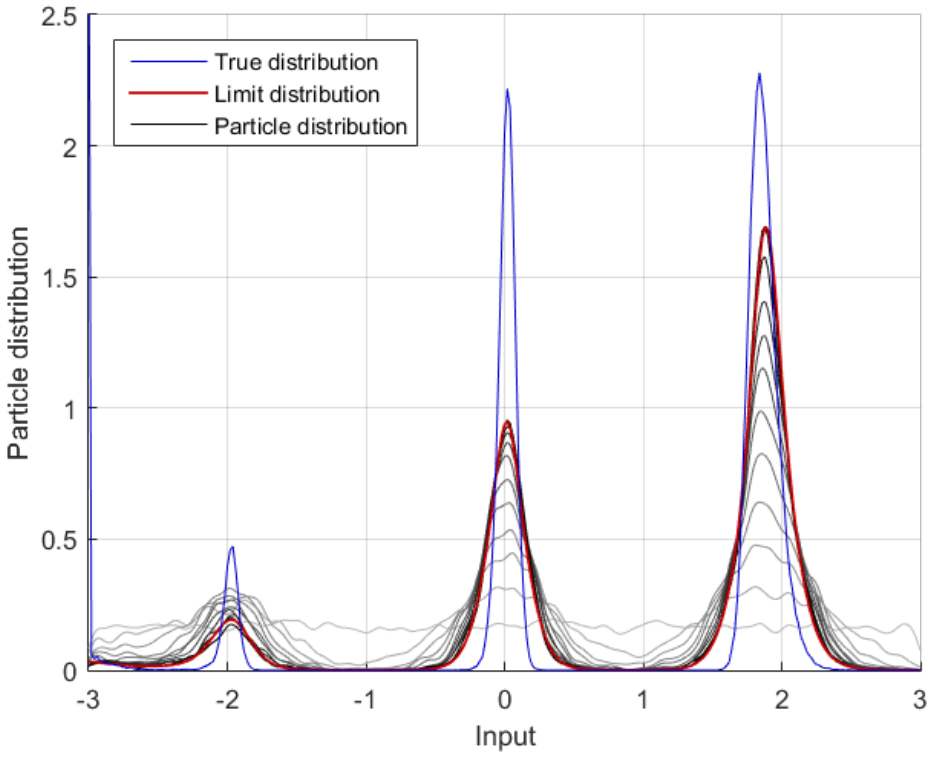

Figure 2: The maximum distribution for the Gaussian process shown in Figure 1. The black/grey lines represent the approximate maximum distribution after $1, 2, \ldots, 10$ rounds of challenges for $n_p = 10\,000$ particles. The red line is the limit distribution of the particles as derived in Section 3.3. The blue line is the true maximum distribution, found through brute force methods.

local optima, but it still falls for them every now and then. As noted earlier, only Thompson sampling has the guarantee to escape local optima given infinitely many measurements.

4.3 Extension to a two-dimensional problem

Next, we apply the optimization methods to a two-dimensional problem. We will minimize the well-known Branin function from (among others) Dixon and Szegö (1978). Or equivalently, we maximize the negative Branin function,

$$f(x_1, x_2) = -\left(x_2 - \frac{51x_1^2}{40\pi^2} + \frac{5x_1}{\pi} - 6\right)^2 - 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) - 10, \tag{11}$$

where $x_1 \in [-5, 10]$ and $x_2 \in [0, 15]$. This function is shown in Figure 4. We can find analytically that the optima occur at $\left(-\pi, \frac{491}{40}\right)$, $\left(\pi, \frac{91}{40}\right)$ and $\left(3\pi, \frac{99}{40}\right)$, all with value $-\frac{5}{4\pi}$.

The performance of the various optimization methods, averaged out over fifty full runs, is shown in Figure 5. Here we see that Thompson sampling now performs significantly better at keeping the regret small compared to the UCB ($\kappa = 2$), the PI and the EI ($\xi = 2$)

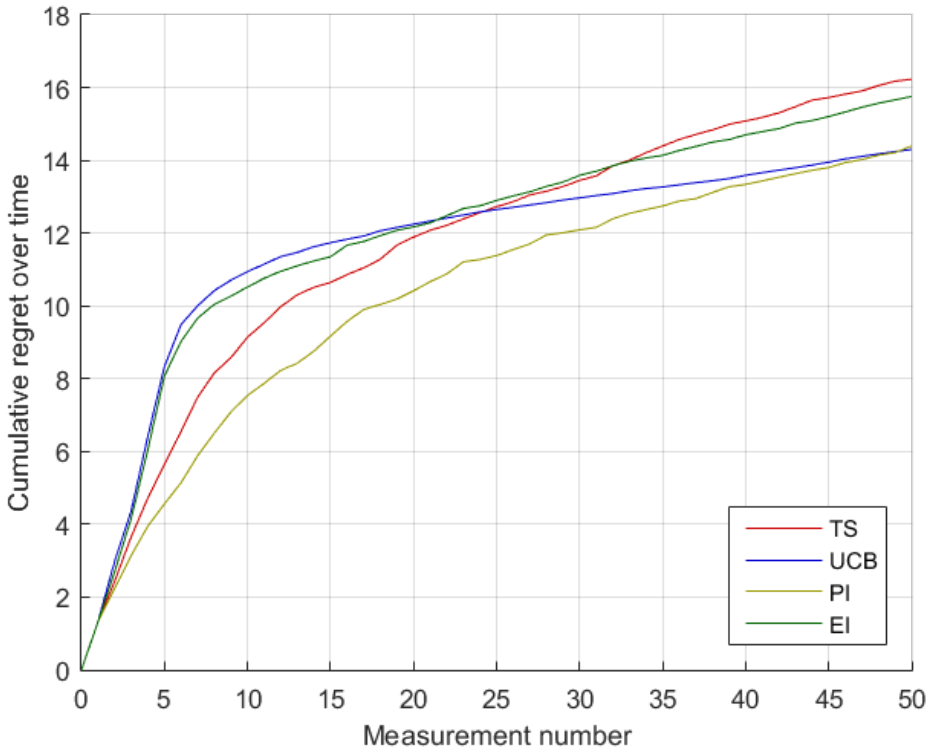

Figure 3: The cumulative regret (1) of the various Bayesian optimization algorithms for the function in (10). Results shown are the mean performance of fifty complete runs of each algorithm.

acquisition functions. We can find the reason behind this, if we look at which try-out points the various algorithms select. When we do (not shown here), we see that all acquisition functions often try out points at the border of the input space, while Thompson sampling does not. In particular, the acquisition functions (nearly) always try out all four corners of the input space, including the very detrimental point $(-5, 0)$. It is this habit which makes these acquisition functions perform worse on this specific problem.

Other than this, it is also interesting to note that in all examined runs, all optimization methods find either two or three of the optima. So while multiple optima are always found, it does regularly happen that one of the three optima is not found. All of the methods have shown to be susceptible to this. In addition, the three acquisition functions have a slightly lower average recommendation error than Thompson sampling, but since all optimization methods find various optimums, the difference is negligible. On the flip side, an advantage of using the MCMD algorithm is that it can provide us with the posterior distribution of the maximum, given all the measurement data. An example of this is shown in Figure 6.

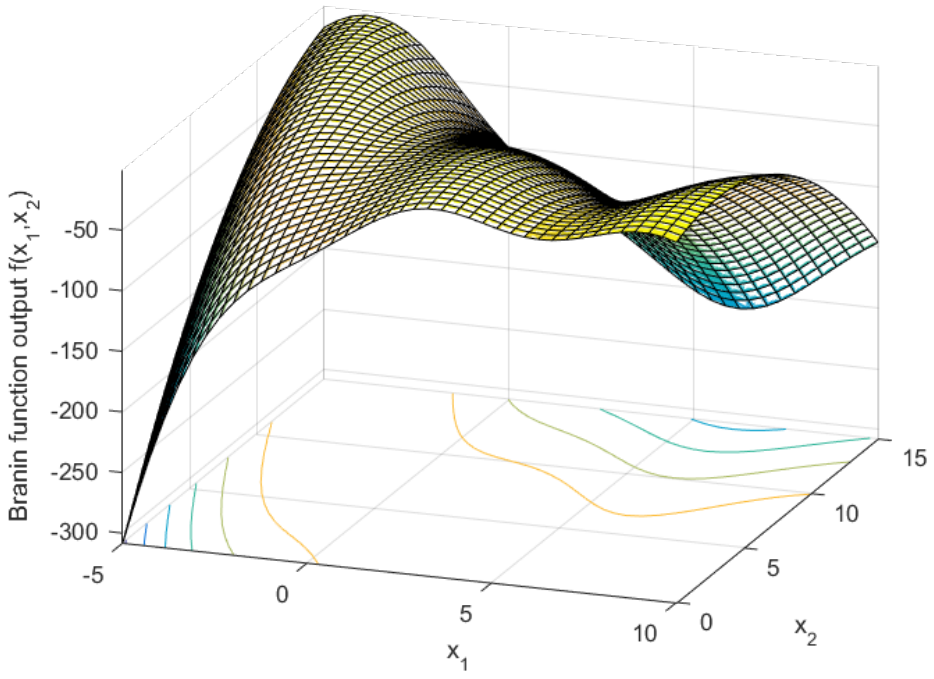

Figure 4: The (negative) Branin function, defined by (11).

4.4 Optimizing a wind turbine controller

Finally we test our algorithm on an application: data-based controller tuning for load mitigation within a wind turbine. More specifically, we use a linearized version of the so-called TURBU model, described by van Engelen and Braam (2004). TURBU is a fully integrated wind turbine design and analysis tool. It deals with aerodynamics, hydrodynamics, structural dynamics and control of modern three bladed wind turbines, and as such gives very similar results as an actual real-life wind turbine.

We will consider the case where trailing edge flaps have been added to the turbine blades. These flaps should then be used to reduce the vibration loads within the blades. To do so, the Root Bending Moment (RBM) of the blades is used as input to the control system.

To determine the effectiveness of the controller, we look at two quantities. The first is the Damage Equivalent Load (DEL; see Freebury and Musial (2000)). The idea here is that the blades are subject to lots of vibrations, some with large magnitudes and some with small magnitudes. For fatigue damage, large oscillations are much more significant. To take this into account, we look at which 1 Hz sinusoidal load would result in the same fatigue damage as all measured oscillations put together. To accomplish this, the RBM signal is separated into individual oscillations using the rainflow algorithm (Niesłony, 2009). We then use Miner's rule (Wirsching et al., 1995), applying a Wöhler exponent of $m = 11$ for the glass fiber composite blades (Savenije and Peeringa, 2009), to come up with an equivalent 1 Hz load.

The second quantity to optimize is the mean rate of change of the input signal. The reason here is that the lifetime of bearings is often expressed in the number of revolutions, or equivalently in the angular distance traveled, and dividing this distance traveled by the time

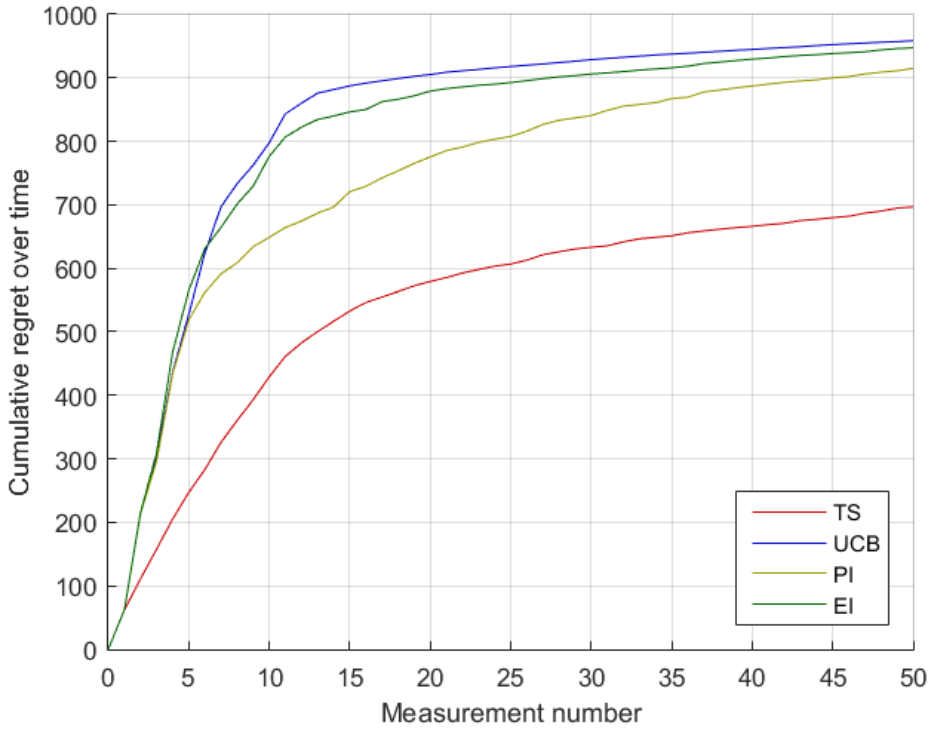

Figure 5: The cumulative regret (1) of the various Bayesian optimization algorithms for the Branin function (11). Results shown are the mean performance of fifty complete runs of each algorithm.

passed will result in the mean rate of change of the flap angle. The eventual performance score for a controller will now be a linearly weighted sum of these two parameters, where a lower score is evidently better.

As controller, we apply a proportional controller. That is, we take the RBM in the fixed reference frame (so after applying a Coleman transformation; see van Solingen and van Wingerden (2015)) and feed the resulting signal, multiplied by a constant gain, to the blade flaps. Since the wind turbine has three blades, there are three gains which we can apply. The first of these, the collective flap mode, interferes with the power control of the turbine. We will hence ignore this mode and only tune the gains of the tilt and yaw modes. Very low gains (in the order of 10^{-8}) will result in an inactive controller which does not reduce the RBM, while very high gains (in the order of 10^{-5}) will react to every small bit of turbulence, resulting in an overly aggressive controller with a highly varying input signal. Both are suboptimal, and the optimal controller will have gains somewhere between these two extreme values.

To learn more about the actual score function, we can apply a brute force method – just applying 500 random controller settings – and apply GP regression. This gives us Figure 7. Naturally, this is not possible in real life as it would cause unnecessary damage to the wind turbine. It does tell us, however, that the score function is mostly convex and that there does not seem to exist any local optimums.

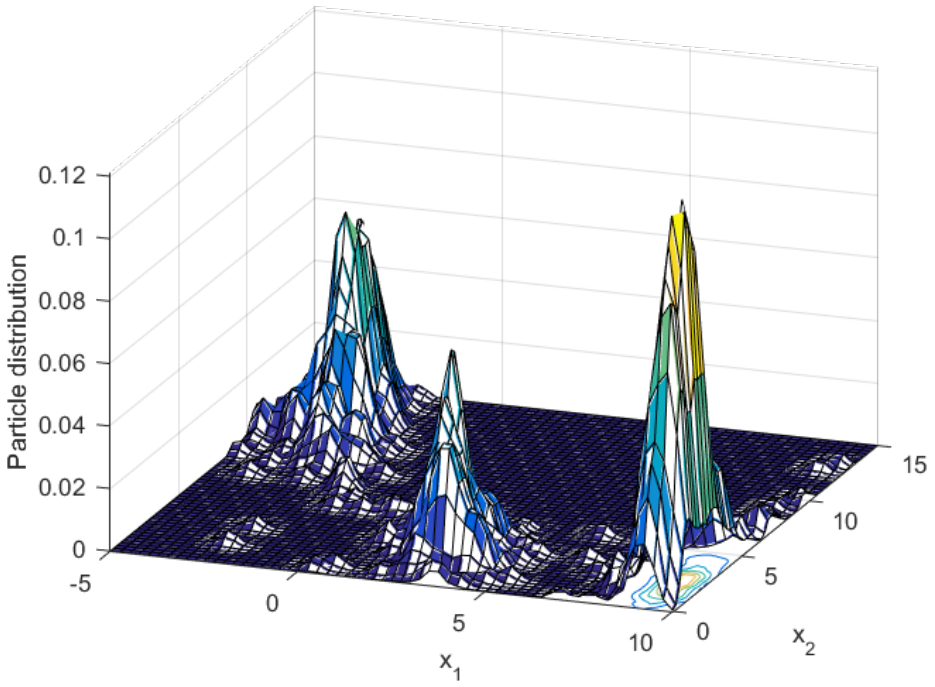

Figure 6: The probability distribution of the maximum of the GP approximating the Branin function, after generating measurements according to Thompson sampling. The three optimums have been identified, some stray particles still reside in the lesser explored regions, and no particles remain in the part of the input space that has been explored but was found suboptimal.

The results from the Bayesian optimization experiments, which are remarkably similar to earlier experiments, are shown in Figure 8. (We used $\kappa = 1$ and $\xi = 0.005$ here.) They once more show that Thompson sampling has a competitive performance at keeping the regret limited. A similar experiment, though with far fewer measurements, has been performed on a scaled wind turbine placed in a wind tunnel, and the results there were similar as well. See Bijl (2017a) for further details on this wind tunnel test.

5. Conclusions and recommendations

We have introduced the MCMD algorithm, which uses particles to approximate the distribution of the maximum of a Gaussian process. This particle approximation can then be used to set up a Bayesian optimization method using Thompson sampling. Such optimization methods are suitable for tuning the parameters of systems with large amounts of uncertainty in an online data-based way. As an example, we have tuned the controller gains of a wind turbine simulation to reduce the fatigue load using performance data that was obtained during the operation of the wind turbine.

The main advantage of Thompson sampling with the MCMD algorithm is that it does not require the optimization of a nonlinear function to select the next try-out point. In

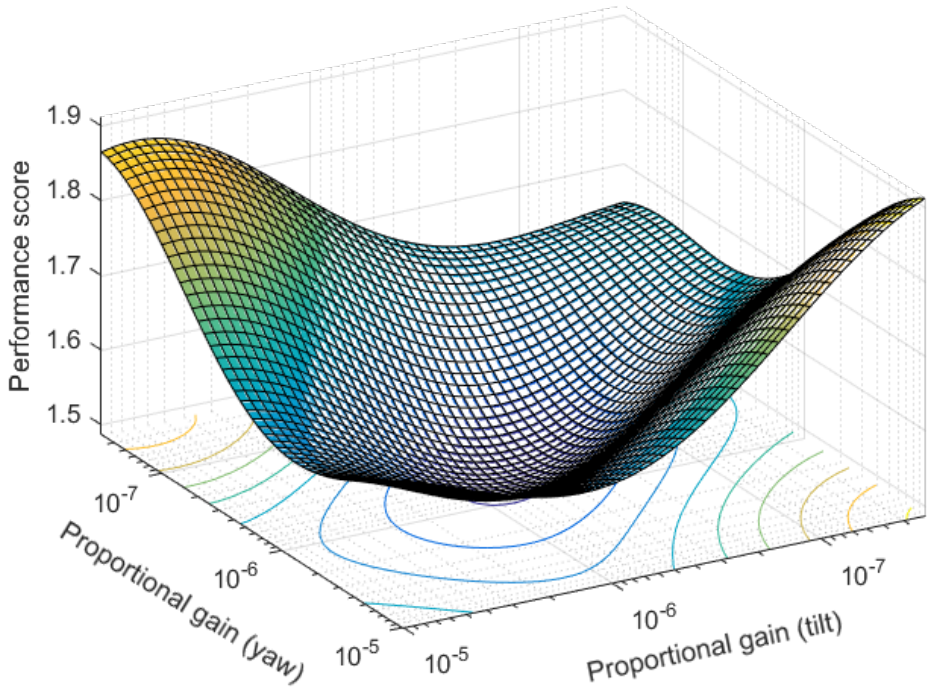

Figure 7: An approximation of the wind turbine controller score with respect to the controller gain. This approximation was made by taking 500 random points and applying a GP regression algorithm to the resulting measurements.

addition, it has shown to have a competitive performance at keeping the cumulative regret limited. However, we cannot conclude that Thompson sampling, or any other optimization method, works better than its competitors. Which method works well depends on a variety of factors, like how much the method has been fine-tuned to the specific function that is being optimized, as well as which function is being optimized in the first place. Also the number of try-out points used matters, where a lower number gives the advantage to exploitation-based methods, while a higher number benefits the more exploration-based methods. It is for this very reason that any claim that a Bayesian optimization works better than its competitors may be accepted only after very careful scrutiny.

Acknowledgments

This research is supported by the Dutch Technology Foundation STW, which is part of the Netherlands Organisation for Scientific Research (NWO), and which is partly funded by the Ministry of Economic Affairs. The work was also supported by the Swedish research Council (VR) via the projects *NewLEADS - New Directions in Learning Dynamical Systems* and *Probabilistic modeling of dynamical systems* (Contract number: 621-2016-06079, 621-

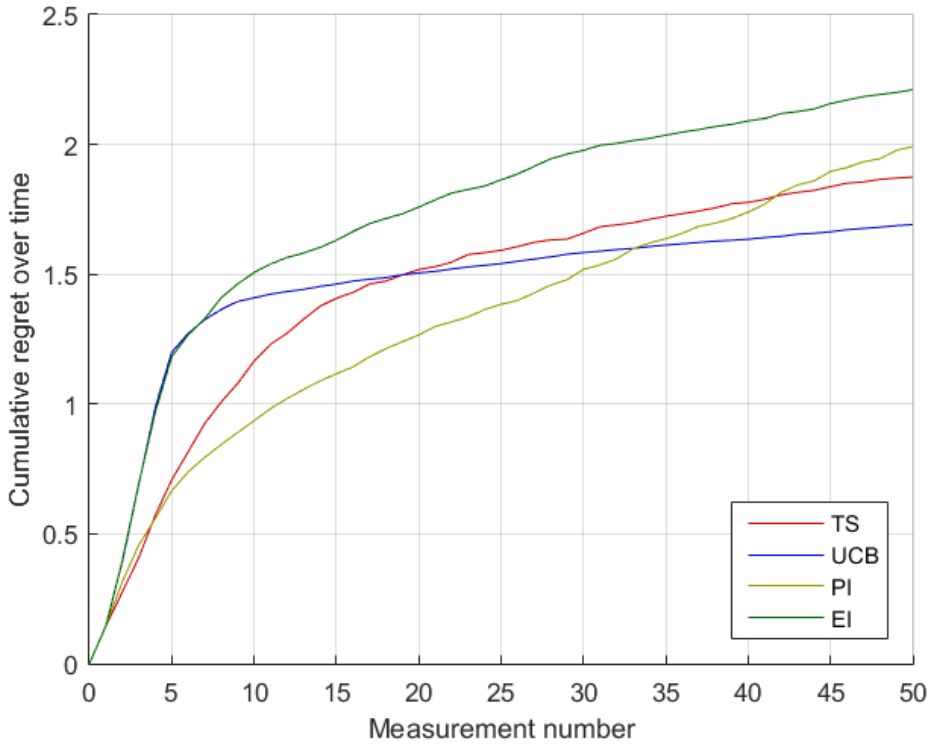

Figure 8: The cumulative regret (1) of the various Bayesian optimization algorithms for the wind turbine controller. Results shown are the mean performance of fifty complete runs of each algorithm.

2013-5524) and by the Swedish Foundation for Strategic Research (SSF) via the project *ASSEMBLE* (Contract number: RIT15-0012).

References

Shipra Agrawal and Navin Goyal. Analysis of Thompson sampling for the multi-armed bandit problem. In *JMLR Workshop and Conference Proceedings*, volume 23, pages 39.1–39.26, 2012.

Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 322–331, 1995.

Hildo Bijl. *Gaussian process regression techniques*. PhD thesis, Delft University of Technology, 2017a.

Hildo Bijl. Gaussian progress regression techniques source code, 2017b. URL `https://github.com/HildoBijl/GPRT`.

Hildo Bijl, Jan-Willem van Wingerden, Thomas B. Schön, and Michel Verhaegen. Online sparse Gaussian process regression using FITC and PITC approximations. In *Proceedings of the IFAC symposium on System Identification, SYSID, Beijing, China*, October 2015.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Eric Brochu, Vlad M Cora, and Nando de Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. Technical report, University of British Columbia, 2010.

Joaquin Q. Candela and Carl E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005.

Olivier Chapelle and Lihong Li. An empirical evaluation of Thompson sampling. In *Advances in Neural Information Processing Systems*, volume 24, pages 2249–2257. Curran Associates, Inc., 2011.

Kamalika Chaudhuri, Yoav Freund, and Daniel J. Hsu. A parameter-free hedging algorithm. In *Advances in Neural Information Processing Systems*, volume 22, pages 297–305, 2009.

Dennis D. Cox and Susan John. SDO: A statistical method for global optimization. In *Multidisciplinary Design Optimization: State-of-the-Art*, pages 315–329, 1997.

Nando de Freitas, Alex Smola, and Masrour Zoghi. Regret bounds for deterministic gaussian process bandits. Technical report, arXiv.org, 2012.

Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, 2006.

L.C.W. Dixon and G.P. Szegö. The global optimisation problem: an introduction. In L.C.W. Dixon and G.P. Szegö, editors, *Towards global optimization*, volume 2, pages 1–15. North-Holland Publishing, 1978.

Gregg Freebury and Walter Musial. Determining equivalent damage loading for full-scale wind turbine blade fatigue tests. In *Proceedings of the 19th American Society of Mechanical Engineers (ASME) Wind Energy Symposium, Reno, Nevada*, 2000.

Steffen Grünewälder, Jean-Yves Audibert, Manfred Opper, and John Shawe-Taylor. Regret bounds for Gaussian process bandit problems. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

Philipp Hennig and Christian J. Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13:1809–1837, 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*. Curran Associates, Inc., 2014.

Matthew Hoffman, Eric Brochu, and Nando de Freitas. Portfolio allocation for Bayesian optimization. In *Uncertainty in Artificial Intelligence (UAI)*, pages 327–336, 2011.

Donald R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21(4):345–383, 2001.

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.

G. Kitagawa. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996.

Robert D. Kleinberg. Nearly tight bounds for the continuum-armed bandit problem. In *Advances in Neural Information Processing Systems 17*, pages 697–704. MIT Press, 2004.

Harold J. Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86(1):97–106, 1964.

Daniel Lizotte, Tao Wang, Michael Bowling, and Dale Schuurmans. Automatic gait optimization with Gaussian process regression. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, pages 944–949, 2007.

Daniel James Lizotte. *Practical Bayesian Optimization*. PhD thesis, University of Alberta, 2008.

Alonso Marco, Philipp Hennig, Jeannette Bohg, Stefan Schaal, and Sebastian Trimpe. Automatic LQR tuning based on Gaussian process global optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) 2016*. IEEE, May 2016.

Thomas P. Minka. Expectation propagation for approximate bayesian inference. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, 2001.

Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. *The application of Bayesian methods for seeking the extremum*. Elsevier, Amsterdam, 1978.

Adam Niesłony. Determination of fragments of multiaxial service loading strongly influencing the fatigue of machine components. *Mechanical Systems and Signal Processing*, 23 (8):2712–2721, November 2009.

M. A. Osborne, R. Garnett, and S. J. Roberts. Gaussian processes for global optimization. In *Proceedings of the 3rd international conference on learning and intelligent optimization (LION3)*, pages 1–15, Trento, Italy, January 2009.

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

Art B. Owen. Monte Carlo theory, methods and examples. Unpublished manuscript, 2013.

Sandeep Pandey and Christopher Olston. Handling advertisements of unknown quality in search advertising. In *Advances in Neural Information Processing Systems*, volume 19, pages 1065–1072. MIT Press, 2007.

Jinkyoo Park and Kincho H. Law. Bayesian Ascent (BA): A data-driven optimization scheme for real-time control with application to wind farm power maximization. *IEEE Transactions on Control Systems Technology*, November 2015.

Carl E. Rasmussen and Christopher K.I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Feike J. Savenije and J.M. Peeringa. Aero-elastic simulation of offshore wind turbines in the frequency domain. Technical Report Report ECN-E-09-060, Energy research centre ECN, The Netherlands, 2009.

Bobak Shahriari, Ziyu Wang, Matthew W. Hoffman, Alexandre Bouchard-Côté, and Nando de Freitas. An entropy search portfolio for Bayesian optimization. Technical report, University of Oxford, 2014.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, January 2016.

Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias W. Seeger. Information-theoretic regret bounds for Gaussian process optimization in the bandit setting. *IEEE Transactions on Information Theory*, 58(5):3250 – 3265, May 2012.

William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

Aimo Torn and Antanas Zilinskas. *Global Optimization*. Springer-Verlag New York, Inc., 1989.

T. van Engelen and H. Braam. TURBU Offshore; Computer program for frequency domain analysis of horizontal axis offshore wind turbines - Implementation. Technical Report Report ECN-C-04-079, ECN, 2004.

E. van Solingen and J. W. van Wingerden. Linear individual pitch control design for two-bladed wind turbines. *Wind Energy*, 18:677–697, 2015.

Emmanuel Vazquez and Julien Bect. Convergence properties of the expected improvement algorithm with fixed mean and covariance functions. *Journal of Statistical Planning and Inference*, 140(11):3088–3095, 2010.

Julien Villemonteix, Emmanuel Vazquez, and Eric Walter. An informational approach to the global optimization of expensive-to-evaluate functions. *Journal of Global Optimization*, 43(2):373–389, March 2009.

Paul H. Wirsching, Thomas L. Paez, and Keith Ortiz. *Random Vibrations, Theory and Practice*. John Wiley & Sons, Inc., 1995.