# Bayesian Optimization without Acquisition Functions

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

Bayesian optimization (BO) is a powerful global optimization technique which is highly efficient when it comes to the optimization of expensive black box functions. It usually requires, however, an auxiliary global optimizer in each iteration to optimize an acquisition function. It is customary in the BO literature to use DIvided RECTangles (DIRECT) to accomplish such a task. Despite the effectiveness of DIRECT, this approach suffers from two shortcomings. Firstly, it is often hard to know whether DIRECT could indeed find the maximum of the acquisition function. As theoretical guarantees of BO algorithms often relies on finding the exact optimum of the acquisition function, failure to do so could possibly lead to failure in finding the true global optimum. Secondly, the use of DIRECT can be costly as it has to be run in each iteration of BO algorithms. In this report, we introduce a new technique for efficient global optimization which we call SOO-UCB. By combining BO with a different global optimization scheme, we are able to perform BO without the need to optimize acquisition functions. We demonstrate in our experiments that SOO-UCB not only outperforms GP-UCB but also does so in with less computation. We also discuss some theoretical properties of the proposed algorithm.

## 1 Introduction

Bayesian optimization (BO) [2, 10, 13, 9, 12] is a powerful global optimization technique. It is used to find a good approximation of the global optimum of a function $f : \mathcal{X} \to \mathbb{R}$. BO is most suitable in the case where the objective function can only be evaluated point-wise and each evaluation is expensive. The success of BO rests on two powerful techniques. First, BO assume a function prior which encapsulate our belief about the objective function. The prior afford us a posterior distribution of functions which gives very rich information about the objective function. In the BO literature, a Gaussian Process (GP) prior is commonly assumed. Second, BO uses an acquisition function to trade off exploration and exploitation when it decide which point to sample next. Many variants of acquisition functions exist. Among the most commonly used are GP Upper Confidence Bound (GP-UCB) and Expected Improvement (EI). Given certain regularity conditions, rate of convergence has been shown for both EI and GP-UCB. BO has been successfully applied to a variety of problems including robot gait planning [14], sensor placement [20], adaptive MCMC [15], and hyper-parameter optimization [19, 1]. We refer the curious readers to a tutorial treatment on the subject [2].

Despite its efficiency, BO usually requires an auxiliary global optimizer in each iteration to optimize the acquisition function. It is customary in the BO literature to use DIvided RECTangles (DIRECT) [8, 2] to accomplish such a task. Other global optimization algorithms like CMA-ES could also be applied [6]. Despite the effectiveness of DIRECT, this approach suffers from two shortcomings. Firstly, it is often hard to know whether DIRECT could indeed find the maximum of the acquisition function. As theoretical guarantees of BO algorithms often relies on finding the exact optimum, failure to do so could possibly lead to failure in finding the true global optimum. Secondly, the use of DIRECT can be unnecessarily costly as it has to be run in each iteration of BO algorithms. This is exacerbated by the fact that for any two consecutive iterations, the acquisition function may not change drastically which questions the necessity of re-optimization in each iteration. As a general global optimization scheme, DIRECT without modifications would not allow the possibility of sharing information across iterations as it is oblivious to the possible change of the underlying acquisition function. Other standard off the shelf global optimization schemes that we may employ would most likely suffer from the same problems, as these methods like DIRECT are also general global optimization schemes.

Apart from BO, there also exist a different class of methods for doing global optimization [11, 3, 17]. Instead of deriving posterior distribution, this class of methods build space partitioning trees. Like in the case of the acquisition function, these algorithms expands the leaves of the tree that is of high function value or of high variance. Out of these methods, a method named Simultaneous Optimistic Optimization (SOO) [17] is worth mentioning as it is able to optimize an objective function globally without the knowledge of its smoothness. We will describe this method in more detail in Section 2. It is interesting to note that these methods do not require the optimization of acquisition functions. However, due to the lack of a posterior that interpolates between the sampled points, it is possible this class of method may not be as competitive as BO when it comes to functions that do satisfy the prior assumption of BO. Such claims have to yet to be backed up by empirical comparisons.

To amend the aforementioned shortcomings of using a global optimizer within BO, we propose in this paper a different approach to do BO by abandoning the practice of optimizing the acquisition function in each iteration. Instead, we use SOO to optimize the underlying objective function directly while disallowing it to sample points that are deemed unfit by the GP posterior by following the approach detailed in [5]. We call this algorithm SOO-UCB. As demonstrated by our experiments, the proposed algorithm preserves the ability of traditional BO on using a minimum number of sample points to optimize and at the same time avoids the shortcomings of optimizing acquisition functions in each iteration.

The report is organized in the following fashion. In section 2, we discuss the relevant works and introduce the algorithm. We discuss the theoretical properties of the algorithm in this section. In section 3, we demonstrate the effectiveness of our approach by comparing it to GP-UCB and SOO. Finally, we conclude the report with potential future works.

## 2 SOO-UCB

In this section, we describe the relevant works and introduce the new algorithm SOO-UCB. We also briefly discuss the theoretical properties of the proposed algorithm.

### 2.1 GP-UCB

As mentioned in the introduction, Bayesian optimization has two ingredients that need to be specified: The prior and the acquisition function. In this work, we adopt GP priors. We review GPs very briefly and refer the interested reader to [18]. A GP is a distribution over functions specified by its mean function $m(\cdot)$ and covariance $k(\cdot,\cdot)$. More specifically, given a set of points $\mathbf{x}_{1:t}$, with $\mathbf{x}_i \subseteq \mathbb{R}^D$, we have

$$\mathbf{f}(\mathbf{x}_{1:t}) \sim \mathcal{N}(\mathbf{m}(\mathbf{x}_{1:t}), \mathbf{K}(\mathbf{x}_{1:t}, \mathbf{x}_{1:t})),$$

where $\mathbf{K}(\mathbf{x}_{1:t}, \mathbf{x}_{1:t})_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$ serves as the covariance matrix. A common choice of $k$ is the squared exponential function which is defined as $k_l^d(\mathbf{y}^{(1)}, \mathbf{y}^{(2)}) = \exp\left(-\frac{\|\mathbf{y}^{(1)}-\mathbf{y}^{(2)}\|^2}{2l^2}\right)$ which length scale parameter $l > 0$. Many other choices are possible depending on our degree of belief about the smoothness of the objective function. Note that $k(x,y) = 1$ when $x = y$ and as $\|x - y\|_2$ increases $k(x,y)$ decreases. This means that two points that are close by have a bigger covariance and points that are far away from each other have smaller covariance.

An advantage of using GPs lies in their analytical tractability. In particular, given observations $\mathbf{x}_{1:n}$ with corresponding values $\mathbf{f}_{1:t}$, where $f_i = f(\mathbf{x}_i)$, and a new point $\mathbf{x}^*$, the joint distribution is given by:

$$\begin{bmatrix}\mathbf{f}_{1:t} \\ f^*\end{bmatrix} \sim \mathcal{N}\left(\mathbf{m}(\mathbf{x}_{1:t}), \begin{bmatrix}\mathbf{K}(\mathbf{x}_{1:t}, \mathbf{x}_{1:t}) & \mathbf{k}(\mathbf{x}_{1:t}, \mathbf{x}^*) \\ \mathbf{k}(\mathbf{x}^*, \mathbf{x}_{1:t}) & k(\mathbf{x}^*, \mathbf{x}^*)\end{bmatrix}\right).$$

For simplicity, we assume that $\mathbf{m}(\mathbf{x}_{1:t}) = \mathbf{0}$. Using the Sherman-Morrison-Woodbury formula, one can easily arrive at the posterior predictive distribution:

$$f^*|\mathcal{D}_t, \mathbf{x}^* \sim \mathcal{N}(\mu(\mathbf{x}^*|\mathcal{D}_t), \sigma(\mathbf{x}^*|\mathcal{D}_t)),$$

with data $\mathcal{D}_t = \{\mathbf{x}_{1:t}, \mathbf{f}_{1:t}\}$, mean $\mu(\mathbf{x}^*|\mathcal{D}_t) = \mathbf{k}(\mathbf{x}^*, \mathbf{x}_{1:t})\mathbf{K}(\mathbf{x}_{1:t}, \mathbf{x}_{1:t})^{-1}\mathbf{f}_{1:t}$ and variance $\sigma(\mathbf{x}^*|\mathcal{D}_t) = k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}(\mathbf{x}^*, \mathbf{x}_{1:t})\mathbf{K}(\mathbf{x}_{1:t}, \mathbf{x}_{1:t})^{-1}\mathbf{k}(\mathbf{x}_{1:t}, \mathbf{x}^*)$. That is, we can compute the posterior predictive mean $\mu(\cdot)$ and variance $\sigma(\cdot)$ exactly for any point $\mathbf{x}^*$.

At each iteration of Bayesian optimization, one has to re-compute the predictive mean and variance. These two quantities are used to construct the second ingredient of Bayesian optimization: The acquisition function. In this work, we report results for the GP-UCB acquisition function $u(\mathbf{x}|\mathcal{D}_t) = \text{ucb}(\mathbf{x}|\mathcal{D}_t) = \mu(\mathbf{x}|\mathcal{D}_t) + \beta_t\sigma(\mathbf{x}|\mathcal{D}_t)$ [20, 5]. We define $\text{lcb}(\mathbf{x}|\mathcal{D}_t) = \mu(\mathbf{x}|\mathcal{D}_t) - \beta_t\sigma(\mathbf{x}|\mathcal{D}_t)$. In above definitions, $\beta_t = \sqrt{2\log(t^{d/2+2}\pi^2/3\delta)}$ where $d$ is the dimensionality of the objective and $\delta$ is the probability with which $f(\mathbf{x})$ is bounded above and below by $\text{ucb}(\mathbf{x}|\mathcal{D}_t)$ and $\text{lcb}(\mathbf{x}|\mathcal{D}_t)$ respectively. The next query is: $\mathbf{x}_{t+1} = \arg\max_{\mathbf{x}\in\mathcal{X}} u(\mathbf{x}|\mathcal{D}_t)$. Note that this utility favors the selection of points with high variance (points in regions not well explored) and points with high mean value (points worth exploiting). the optimization of the closed-form acquisition function is often carried out by off-the-shelf global optimization procedures like DIRECT. Other acquisition functions like Expected Improvement (EI) exist [16, 21, 4] and often yield similar results. Some researchers have also used a portfolio of acquisition functions to obtain better results [7]. We do not consider these acquisition function for brevity. The Bayesian optimization procedure is shown in Algorithm 1.

---

**Algorithm 1** GP-UCB

1: **for** $t = 1, 2, \ldots$ **do**
2: Find $\mathbf{x}_{t+1} \in \mathbb{R}^D$ by optimizing the acquisition function $u$: $\mathbf{x}_{t+1} = \arg\max_{\mathbf{x}\in\mathcal{X}} u(\mathbf{x}|\mathcal{D}_t)$.
3: Augment the data $\mathcal{D}_{t+1} = \{\mathcal{D}_t, (\mathbf{x}_{t+1}, f(\mathbf{x}_{t+1}))\}$
4: **end for**

---

Finite sample bound of the GP-UCB algorithm exist [20]. But the bounds depend on the ability in each iteration to optimize the acquisition function exactly. Since the optimization of the acquisition function in each iteration is often achieved through a global optimization scheme with a fixed budget, we may not be able to guarantee that we can find the exact optimum. The procedure is also very inefficient. As across the iterations, the landscape of the acquisition function may not change very much but we still have to restart the optimizer in each iteration which may be unnecessarily costly.

### 2.2 The Shrinkage of Feasible Regions

In [5], de Freitas et al. introduced another scheme to trade off exploration and exploitation. Instead of optimizing the acquisition function in each iteration, the authors proposed to sample, on the $T^{th}$ iteration, the objective function on a finite lattice within a feasible region $R_T$. The authors were able to show that if we double the density in each iteration, the feasible region which is defined to be

$$R_T = \{\mathbf{x} : \mu_T(x) + \beta_T\sigma_T(x) > \sup_{\mathbf{x}\in R_{T-1}} \mu_T(x) - \beta_T\sigma_T(x)\}$$

shrinks very fast. Notice, with high probability, the optimizer lies within $R_T$. Thus as the feasible region shrinks, the algorithm locates the optimizer of the objective function.

With this approach, the authors did not resort to optimization of the objective function. However, even in moderate dimensions, the algorithm becomes impractical since a lattice often becomes too large to be sampled in a reasonable amount of time. To overcome this problem, an optimistic strategy may have to be employed to sample the promising regions first in order to avoid the computational cost associated with covering the space. In the next subsection, we introduce such an optimistic strategy.

### 2.3 SOO

SOO is another way of doing global optimization. Instead of assuming that the target function is a sample of the with GP prior, it assumes a symmetric semi-metric $\ell$. such that $f(x*) - f(x) \leq$

$\ell(x, x*)$. To optimize the objective function SOO partitions the space $\mathcal{X}$ hierarchically by building a tree. Let us assume that each node of the tree has $k$ children. A node at level $h$ of the tree would be denoted as $(h, j)$ and its children $\{(h+1, kj+i)\}_{0 \le i < k}$. The children partitions their parent's cell $X_{h,j}$ with the cell of the root node being the whole space $\mathcal{X}$. A node is always evaluated at the center of the cell which we denote as $x_{h,j}$. SOO at each round expands (evaluates all its children) at most one leaf per depth, and a leaf is expanded only if it has the largest value among all leaves of same or lower depths. The SOO algorithm takes as parameter a function $t \to h_{\max}(t)$ which limits the maximum height of the tree after $t$ node expansions. The full SOO algorithm is summarized in Algorithm 2.

**Algorithm 2** SOO

```
Evaluate f(x_{0,0}).
Initialize T_1 = {0, 0} (root node).
Set t = 1.
for t = 1, 2, ... do
    Set ν_max = −∞.
    for h = 0 : min{depth(T_t), h_max(t)} do
        Set (h, j) = arg max_{j ∈ {j | (h,j) ∈ T_t}} f(x_{h,j})
        if f(x_{h,j}) > ν_max then
            Evaluate the children {(h + 1, kj + i)}_{0≤i<k} of (h, j)
            Add the children of (h, j) to T_t
            Set ν_max = f(x_{h,j}).
            Set t = t + 1;
        end if
    end for
end for
```

SOO is optimistic in the sense that it only expands cells that has the best objective values at their level and the levels below it. In this sense it exploits. Also it is easy to see that given enough iterations, every cell will be eventually expanded. Thus SOO will sample points arbitrarily close to the optimizer. It is somewhat astonishing that a finite sample performance bound also exist for SOO without assuming knowledge of the semi-metric $\ell$.

### 2.4 SOO-UCB

SOO offers a different way of trading off exploration and exploration which does not require the optimization of an acquisition function. However, because of the simplicity of the assumptions of SOO, it does not utilize the available information given by the evaluated points as effectively. To improve upon SOO, we consider an additional assumption that objective function is a sample from the GP prior as in the case of GP-UCB. Instead of optimizing UCB, we use SOO to propose points to sample and reject the proposal if the UCB of the proposed point is less than the function value of the best point already sampled. We call this algorithm SOO-UCB as it takes advantage of both SOO and the bounds provided by UCB. The algorithm in effect is a combination of the two algorithms described in the previous two sections. It improves upon SOO by using the information available more efficiently and by using an optimistic proposal it avoids the need to sample exhaustively before shrinking the feasible region. The full algorithm is summarized in Algorithm 3.

### 2.5 Theoretical Considerations

We believe that finite sample performance bounds could be derived for the newly introduced algorithm SOO-UCB. Because of the restriction of time, however, we have not been able to derive such finite sample bounds. But we note that SOO-UCB is indeed consistent. That is asymptotically SOO-UCB will sample points whose objective value is arbitrarily close to the optimum. Since we use SOO to propose points to sample, we can argue that a leaf that contains the optimizer will eventually be expanded after finite time. Also if a proposed point is such that its value is greater than or equal to the incumbent, it will be sampled with high probability. This is because with high probability, the UCB of this point would be greater than or equal to its true function value which is no less than the value of the incumbent. Thus as the leaves that contains the optimizer are expanded,

**Algorithm 3** SOO-UCB

Set $g_{0,0} = f(x_{0,0})$.
Set $f^+ = g_{0,0}$.
Initialize $\mathcal{T}_1 = \{0, 0\}$ (root node).
Set $t = 1$, $n = 1$.
Set $\mathcal{D}_1 = \{(x_{0,0}, g(x_{0,0})\}$
**while** true **do**
  Set $\nu_{\max} = -\infty$.
  **for** $h = 0$ to $\min\{\text{depth}(\mathcal{T}_t), h_{\max}(t)\}$ **do**
    Set $(h, j) = \arg\max_{j \in \{j | (h,j) \in \mathcal{T}_t\}} g(x_{h,j})$
    **if** $g(x_{h,j}) > \nu_{\max}$ **then**
      **for** $i = 0$ to $k - 1$ **do** {// loop over all Children of node $(h, j)$}
        **if** $\text{ucb}(x_{h+1,kj+i} | \mathcal{D}_n) \geq f^+$ **then** {// To sample the objective function or not.}
          Set $g(x_{h+1,kj+i}) = f(x_{h+1,kj+i})$
          Set $n = n + 1$.
          Augment the data $\mathcal{D}_n = \{\mathcal{D}_{n-1}, (x_{h+1,kj+i}, g(x_{h+1,kj+i})\}$
        **else** {// Do not sample the objective function, estimate it by $\mu(\cdot|\mathcal{D}_n)$.}
          Set $g(x_{h+1,kj+i}) = \mu(x_{h+1,kj+i} | \mathcal{D}_n)$
        **end if**
        **if** $g(x_{h+1,kj+i}) > f^+$ **then**
          Set $f^+ = g(x_{h+1,kj+i})$
        **end if**
      **end for**
      Add the children of $(h, j)$ to $\mathcal{T}_t$
      Set $\nu_{\max} = g(x_{h,j})$.
      Set $t = t + 1$;
    **end if**
  **end for**
**end while**

the children of these expanded leaves that has a greater objective value than the incumbent will be sampled. If the objective function is continuous (which is true with probability 1 a sample of the GP given many popular kernels [5]), we can have the consistency result.

## 3 Experiments

In this section, we validate the proposed algorithm with a series of experiments comparing the three algorithms (GP-UCB, SOO, SOO-UCB) on global optimization benchmarks. We omitted the feasible region shrinking algorithm (described in Section 2.2) as it is not practical for problems of even moderate dimensions. In our experiments, for each test function we used the same hyper-parameters for GP-UCB and SOO-UCB. We randomized the first sample point for SOO-UCB so that it is not deterministic. To optimize the acquisition function for GP-UCB, we used a combination of DIRECT which is followed by a local optimization method using gradients.

We use in total 5 test functions: Branin, Rosenbrock, Hartmann3, Hartmann6, and Shekel. All the test functions are common to the global optimization literature. Except for the Rosenbrock function, the test functions are multi-modal. [1] We used as out evaluation metric the log distance to the true optimum which is defined as $\log_{10}(f^* - f^+)$ where $f^+$ is the best objective value sampled so far and $f^*$ is the maximum value of the objective. For each test function, we repeat our experiments $50$ times for GP-UCB and SOO-UCB and run SOO once as SOO is a deterministic strategy. We plot the mean and a confidence bound of one standard deviation of our metric across all the runs for all the tests. The plots are presented in Figure 1 and Figure 2.

[1] We refer to the reader the following website for details and formulas which are omitted for space reasons: `http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page364.htm`.

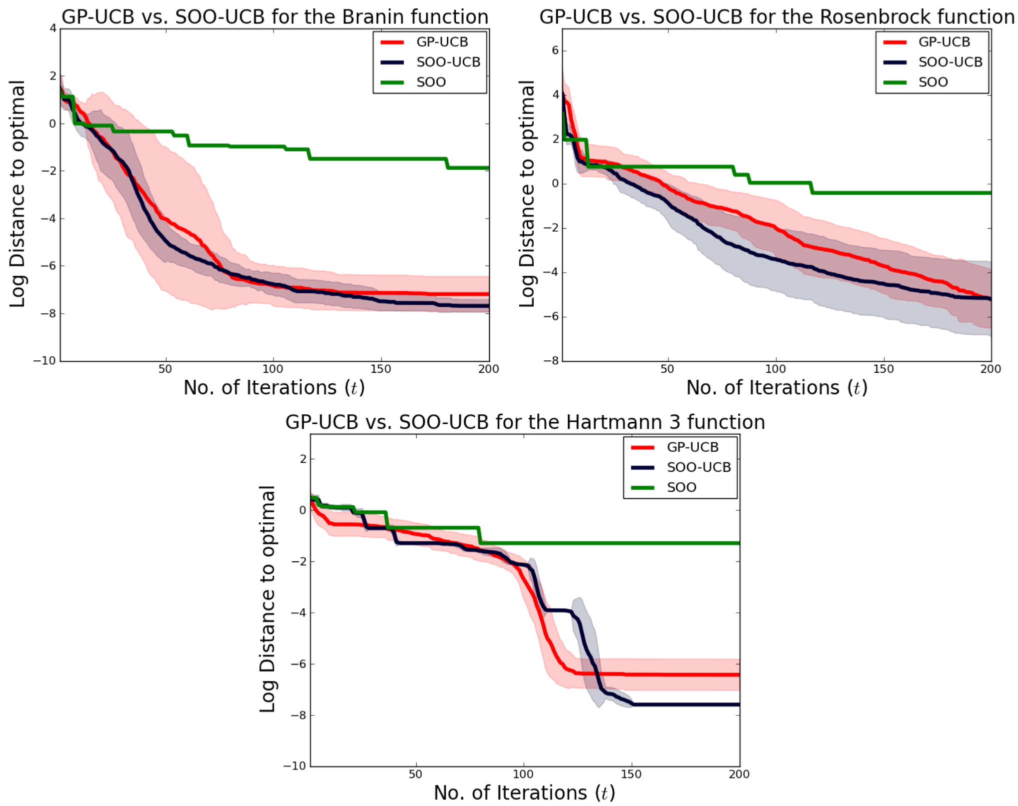


Figure 1: Comparison of GP-UCB, SOO, and SOO-UCB on multi-modal test functions of low dimensionality. In this set of experiments, GP-UCB and SOO-UCB performs similarly whereas SOO does poorly. The poor performance of SOO is caused by having weaker assumptions on the smoothness of the objective function. The good performance of GP-UCB indicates that when the dimensionality is low optimizing the acquisition function might be a reasonable thing to do.

First we test the global optimization schemes on 3 test functions of low dimensionality. The Branin function is a common benchmark for Bayesian optimization that is 2 dimensional [9]. The Rosenbrock function is a commonly used non-convex test function for local optimization algorithms. Although unimodal, the optimum of the Rosenbrock function lie in a long narrow valley which make the function hard to optimize. Finally, the Hartmann3 function is 3 dimensional and has four local optima. As we can see from Figure 1, SOO-UCB performs competitively against GP-UCB on the these low dimensional test functions. Both SOO-UCB and GP-UCB achieve very high accuracy up to $10^{-8}$ in terms of the distance to the optimal objective value. In comparison, SOO, due to the lack of a strong prior assumption, cannot take advantage of the points sampled thus lagging behind. It appears that at least in this scenario, GP-UCB is a highly competitive algorithm.

In the experiments shown in Figure 2, we compare the approaches in consideration on the Shekel function and the Hartmann6 function. The Shekel function is 4-dimensional and has 10 local optima. The Hartmann6 function is 6-dimensional as the name suggests and has 6 local optima. On these higher dimensional problems, the performance of GP-UCB begins to dwindle. Despite the increase in dimensionality, SOO-UCB is still able to optimize the test functions to relatively high precision. SOO does not perform as well as SOO-UCB again because of its weak assumptions. The demise of GP-UCB on these two test functions may due in part to the inability of an global optimizer to optimize the acquisition function exactly in each iteration. As the dimensionality increases, so is the difficulty of optimizing a non-convex function globally as the cost of covering the space grows exponentially. The problem is compounded by the existence of many local optimum for each of the test functions considered here. For SOO-UCB, the cost of optimization also grow with dimensionality. But since SOO-UCB refines the partition of the space in each iteration, it will

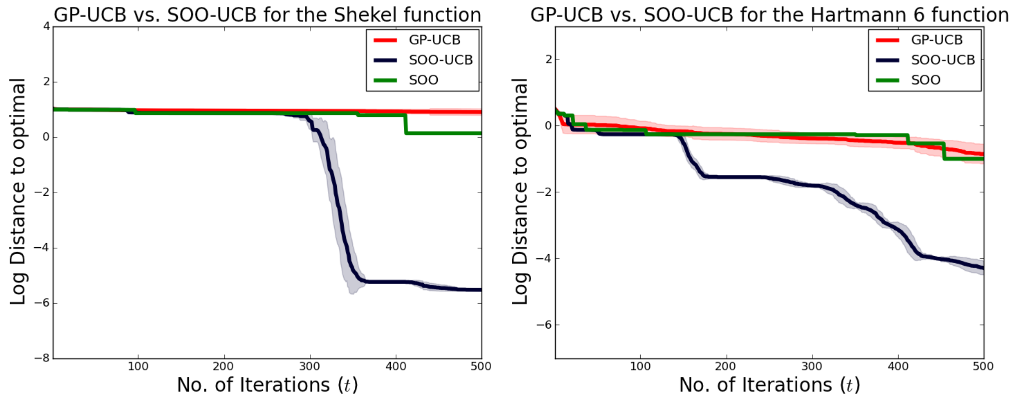


Figure 2: Comparison of GP-UCB, SOO, and SOO-UCB on multi-modal test functions of moderate dimensionality. The Shekel function is of dimensionality 4 and the Hartmann 6 function is 6 dimensional. In this set of experiments, GP-UCB performs poorly. This may due in part to the hardness of optimizing the acquisition function.

Table 1: Time required for the test functions measured in seconds. SOO is very fast as it does not maintain a GP. SOO-UCB maintains uses a GP to produce more accurate posterior estimates which also makes it slower. Also the frequent rejection of proposals would also result in much bigger trees which further slows the algorithm. GP-UCB tends is quite slow compared to the other two algorithms as it not only maintains a GP to but also optimizes its acquisition function in each iteration.

| Algorithm | Branin | Rosenbrock | Hartmann3 | Hartmann6 | Shekel |
|---|---|---|---|---|---|
| GP-UCB | 29.9438 | 29.5716 | 34.0311 | 115.2402 | 100.7770 |
| SOO-UCB | 3.0680 | 3.4693 | 3.9722 | 2.0918 | 3.8951 |
| SOO | 0.1810 | 0.1835 | 0.1871 | 0.4313 | 0.4350 |

eventually be fine enough such that it attains high precision. The optimization of the acquisition function through algorithms like DIRECT demands the repartitioning of the space in each iteration. To reach a finer granularity, we either have to sacrifice speed by building very fine partitions in each iteration or accuracy by using coarser partitions.

The proposed approach is not only competitive against GP-UCB in terms of effectiveness, it is also more computationally efficient. As we can see in Table 1, SOO-UCB is about 10-40 times faster than GP-UCB on the test functions we have experimented with. This is because instead of optimize the acquisition function in each iteration, the SOO algorithm that sits inside only optimizes once. SOO-UCB, however, is much slower than SOO. This is because SOO-UCB also employs a GP to reject points proposed by SOO. To sample one point, SOO may have to propose many points before one is accepted. For this reason, SOO-UCB would build much bigger trees compared to SOO and thus slower.

## 4 Conclusion and Future work

In this report, we present a new global optimization algorithm SOO-UCB. By using SOO and a GP at the same time, the proposed approach is capable of global optimization without an acquisition function. We are able outperform the existing BO algorithms on a broad class of test functions while requiring less computational time. In addition, we discussed the consistency of the proposed approach. Despite the attractive properties, the convergence rate of this algorithm remains elusive. In the future, we would like to be able to prove convergence properties of this algorithm.

## References


[1] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In *NIPS*, pages 2546–2554, 2011.

[2] E. Brochu, V. M. Cora, and N. de Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. Technical Report UBC TR-2009-23 and arXiv:1012.2599v1, Dept. of Computer Science, University of British Columbia, 2009.

[3] S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvari. X-armed bandits. *JMLR*, 12:1655–1695, 2011.

[4] A. D. Bull. Convergence rates of efficient global optimization algorithms. *JMLR*, 12:2879–2904, 2011.

[5] N. de Freitas, A. Smola, and M. Zoghi. Exponential regret bounds for Gaussian process bandits with deterministic observations. In *ICML*, 2012.

[6] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.*, 9(2):159–195, 2001.

[7] M. Hoffman, E. Brochu, and N. de Freitas. Portfolio allocation for Bayesian optimization. In *UAI*, pages 327–336, 2011.

[8] David R Jones, C D Perttunen, and B E Stuckman. Lipschitzian optimization without the Lipschitz constant. *J. of Optimization Theory and Applications*, 79(1):157–181, 1993.

[9] D.R. Jones. A taxonomy of global optimization methods based on response surfaces. *J. of Global Optimization*, 21(4):345–383, 2001.

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

[11] Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *Machine Learning: ECML 2006*, pages 282–293. Springer, 2006.

[12] D. Lizotte. *Practical Bayesian Optimization*. PhD thesis, University of Alberta, Canada, 2008.

[13] D. Lizotte, R. Greiner, and D. Schuurmans. An experimental methodology for response surface optimization methods. *J. of Global Optimization*, pages 1–38, 2011.

[14] Daniel Lizotte, Tao Wang, Michael Bowling, and Dale Schuurmans. Automatic gait optimization with Gaussian process regression. In *Proc. of IJCAI*, pages 944–949, 2007.

[15] N. Mahendran, Z. Wang, F. Hamze, and N. de Freitas. Adaptive MCMC with Bayesian optimization. *Journal of Machine Learning Research - Proceedings Track*, 22:751–760, 2012.

[16] J. Močkus. The Bayesian approach to global optimization. In *Systems Modeling and Optimization*, volume 38, pages 473–481. Springer, 1982.

[17] R. Munos. Optimistic optimization of a deterministic function without the knowledge of its smoothness. In *NIPS*, 2011.

[18] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[19] J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian optimization of machine learning algorithms. In *NIPS*, 2012.

[20] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *ICML*, 2010.

[21] E. Vazquez and J. Bect. Convergence properties of the expected improvement algorithm with fixed mean and covariance functions. *J. of Statistical Planning and Inference*, 140:3088–3095, 2010.