Sampling Acquisition Functions for Batch Bayesian Optimization

Alessandro De Palma 1  Celestine Mendler-Dünnen 2  Thomas Parnell 2  Andreea Anghel 2  Haralampos Pozidis 2

Abstract
This paper presents Acquisition Thompson Sampling (ATS), a novel algorithm for batch Bayesian Optimization (BO) based on the idea of sampling multiple acquisition functions from a stochastic process. We define this process through the dependency of the acquisition functions on a set of model parameters. ATS is conceptually simple, straightforward to implement and, unlike other batch BO methods, it can be employed to parallelize any sequential acquisition function. In order to improve performance for multi-modal tasks, we show that ATS can be combined with existing techniques in order to realize different explore-exploit trade-offs and take into account pending function evaluations. We present experiments on a variety of benchmark functions and on the hyper-parameter optimization of a popular gradient boosting tree algorithm. These demonstrate the competitiveness of our algorithm with two state-of-the-art batch BO methods, and its advantages to classical parallel Thompson Sampling for BO.

1. Introduction
Bayesian optimization (BO) methods deal with the optimization of expensive, black-box functions, i.e., functions with no closed-form expression or derivative information. Many such problems arise in practice in fields such as: hyper-parameter optimization of machine learning algorithms (Snoek et al., 2012), synthesis of short polymer fiber materials (Li et al., 2017), and analog circuit design (Lyu et al., 2018). Therefore, in recent years, BO gained widespread popularity to the point that its versatility was showcased on toy problems like optimizing chocolate cookie recipes (Solnik et al., 2017).

From a high-level perspective, BO works by training a probabilistic regression model (typically, Gaussian Processes) on the available function evaluations. This model is then employed to decide where to evaluate the function next, maximizing a trade-off between exploration and exploitation that is formalized through an acquisition function \( a(x) \). Many sequential BO algorithms exist, differing by the choice of the regression model, the acquisition function, or properties of the input domain (Hutter et al., 2011; Srinivas et al., 2012; Shahriari et al., 2016; Springenberg et al., 2016; Hernandez-Lobato et al., 2015). Recently, parallel BO techniques have gained popularity. These methods select multiple points to be evaluated in parallel in order to better exploit modern hardware capabilities, and to speed up the overall optimization process in the case of costly evaluations like training machine learning algorithms. In the parallel case, typically, the acquisition function must return \( q \) points at once (batch BO), these being greedily selected one after the other.

Most successful batch BO algorithms focus on parallelizing a specific acquisition function (Kandasamy et al., 2018; Wu & Frazier, 2016; Daxberger & Low, 2017; Shah & Ghahramani, 2015; Gupta et al., 2018). In this paper, instead, we will focus on methods that can be applied to any sequential acquisition function, as in general no single one consistently outperforms the others (Hoffman et al., 2011). Amongst these general methods, the prevalent approach is to modify the landscape of the acquisition function \( a(x) \) after a point is scheduled so as to avoid returning another point in its neighborhood. For instance, Local Penalization (LP) (González et al., 2016) accomplishes this by suppressing the value of \( a(x) \) in the neighborhood of any scheduled \( x \) using an approximation of the Lipschitz constant of the unknown function. A family of algorithms including, e.g., B-LCB (Desautels et al., 2014), instead, obtains the same effect by hallucinating the results of the points that have been scheduled but not yet completed.

1.1. Our Intuition
In general, a sequential acquisition function depends on a vector of parameters mostly belonging to the employed regression model. We will denote the parameter vector by \( \theta \), and write \( a(x) = a(x, \theta) \) to refer to a member of the family of acquisition functions. The \( \theta \) vector exerts a strong effect on the practical performance of BO (Snoek et al., 2012). Hence, in practice, it is either set to its maximum likelihood value, or its influence is mitigated by using an
acquisition function averaged over a fixed number of possible values of $\theta$. Our intuition is that, rather than focusing on modifying the landscape of a fixed $a(x, \{\theta_i\})$, where $\{\theta_i\}$ possibly denotes a set of vectors, such a dependency can be exploited to select each point in a batch from a different member of the family. In order to do so, inspired by Parallel Thompson Sampling (P-TS) (Kandasamy et al., 2018), we interpret a given sequential acquisition function $a(x, \{\theta_i\})$ as being sampled from a stochastic process. It then suffices to draw samples from the process to obtain as many different acquisition functions as the number of batch points to evaluate.

1.2. Contributions

- We present ATS, a novel batch Bayesian optimization algorithm based on the idea of sampling acquisition functions from a stochastic process. ATS is conceptually simple, easy to implement and different from the previously published randomized approach (Kandasamy et al., 2018), in that it can parallelize any arbitrary sequential acquisition function.

- We show how existing techniques to diversify points within a batch, i.e., jitter and hallucination, can be incorporated into our randomized framework to improve performance of ATS on highly multi-modal functions.

- We present experimental results on five different well-known benchmark functions and on hyper-parameter tuning for XGBoost (Chen & Guestrin, 2016). We demonstrate and explain the advantages of ATS and its variants compared to P-TS, B-LCB and LP, three popular state-of-the-art batch BO methods.

2. Problem Statement and Background

Our goal is to find the minimum of a black-box function $f : \mathcal{X} \to \mathbb{R}$, defined on a bounded set $\mathcal{X} \subset \mathbb{R}^d$. While the function itself is unknown, we have access to the results of its evaluations $f(x)$ at arbitrary domain points $x$. We are interested in scenarios where these evaluations are costly and we seek to minimize $f$ using as few evaluations as possible. For instance, $f$ might represent the loss incurred by a machine learning algorithm trained with the hyper-parameter vector $x$.

2.1. Bayesian Optimization

Bayesian Optimization (BO) approaches this black-box optimization problem by placing a prior on $f$ and updating the model by conditioning on the evaluations as they are collected. Once the surrogate model is updated, BO employs it to select the next single point to evaluate according to a criterion referred to as the acquisition function. This function should not only "exploit" the regions that are likely to contain the minimum, but also "explore" those where uncertainty is significant.

Gaussian Processes (GP) are a common and versatile choice for the function prior. Under a GP, any finite number of function outputs is jointly distributed according to:

$$f(x_1), f(x_2), \ldots, f(x_n) \sim \mathcal{N}(\mu, K)$$

where the covariance $K$ of the Gaussian process is defined through a kernel function $k$, parametrized by $\theta_k$ as $K_{i,j} = k(x_i, x_j|\theta_k)$. The choice of the kernel function $k$ determines the family of functions that a GP can represent. In the following, we will use the Matérn 5/2 kernel due to its widespread usage and its good empirical performance on tuning machine learning algorithms (Snoek et al., 2012).

Moreover, we will assume the prior mean $\mu_0$ to be constant (yet possibly non-zero) on the input domain. In general, while the function $f$ is deterministic, its evaluations may be noisy. It is, therefore, common practice to assume Gaussian noise in addition to the GP, so that $y_i = f(x_i) + \mathcal{N}(0, \sigma^2)$. We will denote the set of collected observations up to evaluation $t$ with $D_t = \{(x_i, y_i) \mid i \in \{1, \ldots, t\}$. A desirable property of GP’s in this context is that $\mathcal{N}(\mu, K|D_t)$ is again a GP, and its posterior mean and variance at point $x_*$ are respectively (Rasmussen & Williams, 2006):

$$\mu(x_*)|D_t) = \mu_p + k_*(K + \sigma^2 I)^{-1}(y - \mu_p)$$

$$\sigma^2(x_*)|D_t) = k(x_*, x_*) + k_*(K + \sigma^2 I)^{-1}k_*$$

where $k_*$ denotes the vector with elements $k_{*,i} = k(x_*, x_i)$ and $K_{i,j} = k(x_i, x_j)$. In the remainder of the paper, we will ignore the evaluation noise and set $y_i = f(x_i)$ to facilitate the presentation. In order to generalize to the noisy case, it suffices to substitute $f(x_*)$ with $y_i$ and to include the noise in the kernel matrix.

Once $t$ evaluations are available, the next point is selected by maximizing the acquisition function: $x_{t+1} = \arg \max_{x} a(x|D_t)$, where the conditioning explicitly denotes the dependence on the updated surrogate model taking evaluations up to $t$ into account. Two widespread sequential acquisition functions are Expected Improvement (EI) (Jones et al., 1998) and Lower Confidence Bound (LCB) (Srinivas et al., 2012):

$$EI(x|D_t) = E_{f(x|D_t)}[\max(f(x^*) - f(x), 0)]$$

$$LCB(x|D_t) = \sigma(x|D_t) - \mu(x|D_t)$$

where $f(x^*)$ denotes the best (smallest) evaluation result recorded until evaluation $t$.

2.2. Parallel Bayesian Optimization

Bayesian Optimization as presented in the previous section is an inherently sequential algorithm: points are evaluated
one by one, and the model is updated to reflect the evaluation results. Many works have explored the possibility of evaluating more than one point at once, either asynchronously (Snoek et al., 2012; Kandasamy et al., 2018) or synchronously (González et al., 2016; Desautels et al., 2014; Gupta et al., 2018; Kathuria et al., 2016). Next, we will focus on the latter, more studied, case, as asynchronous algorithms can also be employed in a synchronous fashion.

Some approaches, like q-EI by Wang et al. (2016), theoretically generalize sequential acquisition functions to the case in which q points must be returned at once. Unfortunately, such approaches are not computationally feasible for q > 4, as the batch of points needs to be jointly optimized (Daxberger & Low, 2017). Therefore, most of the attention has been devoted to greedy methods, where points are selected one after the other. In this context, we will call pending evaluations, the points for which the evaluation has been scheduled but not yet completed. We will denote all pending evaluations with the prime sign: if evaluation \( i \) is pending, we will write \( x'_i \), meaning that \( x'_i \) is available, but not \( f(x'_i) \).

A plethora of approaches for batch BO exists; here, we will describe those that form the background to our algorithms. A well-known heuristic is to hallucinate the results of the pending evaluations, i.e., to approximate them with known quantities. This way, the acquisition function is updated from \( a(x|D_t) \) to \( a(x|D_t \cup \{ (x'_i, h_i) \ \forall x'_i \}) \), where \( h_i \) denotes the hallucinated evaluation result for the pending evaluation \( x'_i \). For instance, the very effective batch LCB (B-LCB) approach updates the surrogate model after each batch point has been scheduled by setting \( f(x'_i) = \mu(x'_i|D_t) \). The batch BO algorithm by Gupta et al. (2018), instead, parametrizes the LCB acquisition function to express different explore-exploit trade-offs and, arguing that different trade-offs are associated to different maxima of \( a(x|D_t) \), assigns each batch point a new trade-off.

2.3. Thompson Sampling

Thompson Sampling (TS) is a rather old heuristic to address the explore-exploit dilemma that gained popularity in the context of multi-armed bandits (Chapelle & Li, 2011). The general idea is to choose an action according to the probability that it is optimal. Applied to BO, this corresponds to sampling a function \( g \) from the Gaussian Process posterior and selecting \( x_{t+1} = \arg\max_X g \) (Kandasamy et al., 2018). In fact, let \( p_x(x) \) be the probability that point (action) \( x \) optimizes the GP posterior, then:

\[
p_x(x) = \int p_x(x|g) \, p(g|D_t) \, dg
= \int \delta(x - \arg\min_X g(x)) p(g|D_t) \, dg.
\]

TS is therefore equivalent to a randomized sequential acquisition function, one that is easy to parallelize. In fact, it suffices to sample multiple functions from the GP posterior to get a batch of points to be evaluated in parallel (Kandasamy et al., 2018). In this paper, we generalize the reasoning in (4) so as to parallelize an arbitrary acquisition function, and additionally incorporate ideas from hallucinations and intra-batch trade-off differentiation presented in the previous section.

3. Sampling Acquisition Functions

We first introduce Acquisition Thompson Sampling (ATS), our novel algorithm for both batch and asynchronous parallel Bayesian Optimization (§3.1) and then discuss two extensions thereof, j-ATS (§3.2) and h-ATS (§3.3). The common, novel idea behind all of these methods is to sample an acquisition function from a stochastic process for each parallel evaluation to perform.

3.1. Acquisition Thompson Sampling (ATS)

In general, any sequential acquisition function \( a(x) \) explicitly depends on the surrogate model’s hyper-parameters \( \theta \) and can be therefore written as \( a(x, \theta) \). The most common way to choose \( \theta \) is to optimize the marginal likelihood under the Gaussian process. The acquisition functions are very sensitive to these hyper-parameters and it is therefore advisable to adopt a fully Bayesian approach by marginalizing over them, as suggested by Snoek et al. (2012). Specifically:

\[
a(x|D_t) = \int a(x, \theta|D_t) \, p(\theta|D_t) \, d\theta
\]

This sensitivity is further confirmed by their empirical evaluations. The evaluation of the integral in (5), in practice, is approximated by \( \bar{a}_s(x) \), the empirical average of \( s \) samples from the data posterior of \( \theta \) obtained via Markov Chain Monte Carlo (MCMC) sampling:

\[
\bar{a}_s(x|D_t) = \frac{1}{s} \sum_{q=1}^{s} a(x, \theta_q|D_t) \text{ s.t. } \theta_q \sim p(\theta|D_t)
\]

In this paper, however, we use a different approach: we do not seek to fully marginalize over the hyper-parameters, but instead, to leverage the sensitivity to \( \theta \) for selecting a diverse batch of points. The main intuition is to recognize that \( \bar{a}_s(x) \) represents a sample from a probability distribution over functions that is implicitly defined by the \( \theta \) data posterior, i.e., a stochastic process. We will denote such a process by \( a_s(x|D_t) \), and write \( \bar{a}_s \sim p(\bar{a}_s|D_t) \) to refer to the sampling in (6). We can then argue that the sequential criterion for determining the next point to evaluate,

\[
x_{t+1} = \arg\max_X \bar{a}_s(x|D_t)
\]
can be seen as sampling from $p_{a_s^*}(x)$, the probability that $x$ maximizes the stochastic process. In fact, we can write:

$$p_{a_s^*}(x) = \int p_{a_s^*}(x|\tilde{a}_s) p(\tilde{a}_s|D_t) \, d\tilde{a}_s$$

$$= \int \delta(x - \arg \max_{x} \tilde{a}_s(x|\tilde{a}_s))p(\tilde{a}_s|D_t) \, d\tilde{a}_s$$

(7)

where $\delta$ denotes the Dirac delta distribution. In light of this interpretation, a sequential acquisition function can be employed in the parallel setting by sampling as many different acquisition functions from $a_s(x|D_t)$ (i.e., as many Monte Carlo approximations of the marginalized function $a(x|D_t)$) as the parallel evaluations to perform. We call this procedure ATS. Our algorithm closely resembles Thompson Sampling for Bayesian Optimization (cf. §2.3): in fact, (7) can be interpreted as an adaptation of (4) to a case in which the optimal action is defined with respect to a stochastic acquisition function rather than the GP posterior. While parallel TS is an acquisition function of its own, this adaptation allows us to obtain a general approach, applicable for the parallelization of any sequential acquisition function by means of its parametrization $\theta$. Both ATS and TS easily work in the asynchronous setting, in addition to batch setting. Moreover, ATS can be also employed on top of an existing greedy parallel approach to make it scale further. Algorithm 1 outlines the pseudo-code for a single iteration of the batch version of ATS.

In order to link theory to practice, it remains to fully specify the data posterior of $\theta$. Making use of the Bayes rule, we can write: $p(\theta|D_t) = \frac{1}{Z} p(D_t|\theta) p(\theta)$, where $\frac{1}{Z}$ is a normalization constant. $p(D_t|\theta)$ is the marginal likelihood of the collected data under the Gaussian process (Rasmussen & Williams, 2006), while $p(\theta)$ is the prior over the model hyper-parameters. We found the performance on high-dimensional functions (e.g., $d = 6$) to be strongly dependent on the choice of the prior and on the output range of $f$. To mitigate the effect of the latter, we employ $z$-normalization of the outputs $\{y_i\}$ after each model update. For what concerns the prior, we explored different options and eventually chose a modification of that employed by the GPyOpt authors (2016), as it was the one that performed the best in a variety of cases. Specifically: $p(\theta_k) = \Gamma(\alpha = 1, \beta = 6)$ and $p(\mu) = \text{Uniform}(-3, 3)$, where $\theta_k$ are the GP kernel hyper-parameters, and $\mu$ is the constant prior mean of the GP. The priors are independent as in Snoek et al. (2012).

3.2. Sampling Explore-Exploit Trade-Offs (j-ATS)

If $f$ exhibits multiple minima, ATS might over-exploit the current posterior of the function, producing a batch of points centered around a local minimum. In the hope of ending up in different valleys of $f$, we encourage the exploration of regions of the input domain with varying degrees of posterior uncertainty by pursuing different explore-exploit trade-offs within a batch. This approach, devised by Gupta et al. (2018), was originally implemented by explicitly solving a multi-objective optimization problem on $\text{LCB}_1(x|D_t) = j\sigma(x|D_t) - \mu(x|D_t)$, a parametrized version of the LCB acquisition function, and taking $M$ different trade-offs from the resulting Pareto curve (one per batch point). In this context, the parameter $j$ is commonly called jitter: the larger the jitter, the more exploitative the acquisition function.

Rather than explicitly solving the multi-objective optimization as Gupta et al. (2018), we translate this idea into our randomized framework by sampling different trade-offs from a distribution. In order to do this, we associate a new process $a_{s,j}(x|D_t)$ to the parametrization, and place an independent prior on $j$ so as to sample jitter alongside the $\theta$ vectors. In addition to LCB, such a parametrization is commonly employed for many popular acquisition functions; e.g., EI (the GPyOpt authors, 2016):

$$\text{EI}_j(x|D_t) = E_f(x|D_t)[\max(f(x^*)) - j - f(x), 0)]$$

As a consequence, sampling from $a_{s,j}(x|D_t)$ implies obtaining acquisition functions having different explore-exploit trade-offs. A batch iteration of this jittered variant (j-ATS) of the ATS algorithm is outlined in Algorithm 2.

It now remains to choose $p(\theta, j|D_t)$, the data posterior defining the process. As the data likelihood is independent of the parameters that do not belong to the surrogate model, we only need to define the prior of $j$. Let $C$ be a Bernoulli random variable with $p = 0.5$. Then:

$$\text{EI}: j|C = 1 \sim \text{logUniform}(-3, 0)$$
$$\text{LCB}: j|C = 1 \sim \text{Beta}(1, 12)$$

(8)

while we revert to ATS, as presented in Section 3.1 (without jitter), if $C = 0$. 

---

**Algorithm 1** Batch iteration, Acquisition Thompson Sampling (ATS).

Require: dataset $D_t := \{(x_k, f(x_k)) \forall k \leq t\}$, sequential acquisition function $a(x, \theta|D_t)$, batch size $M$.

1: for $i = 1, \ldots, M$ in parallel do
2:   sample $s$ new GP hyper-parameter vectors according to $\theta_q \sim p(\theta|D_t)$.
3:   $\tilde{a}_s(x|D_t) = \frac{1}{s} \sum_{q=1}^{s} a(x, \theta_q|D_t)$.
4:   $x_{t+i} \leftarrow \arg \max_x \tilde{a}_s(x|D_t)$, the $i$-th batch point.
5:   evaluate $f(x_{t+i})$.
6: $D_{t+M} \leftarrow D_t \cup \{(x_k, f(x_k)) \forall t \leq t \leq t + M\}$. 

---

Sampling Acquisition Functions for Batch Bayesian Optimization
Algorithm 2 Batch iteration, jittered Acquisition Thompson Sampling (j-ATS).

Require: dataset $D_t := \{(x_k, f(x_k)) \forall k \leq t\}$, sequential acquisition function $a(x, \theta | D_t)$, batch size $M$.

1: for $i = 1, \ldots, M$ in parallel do
2: sample $s$ new GP hyper-parameter vectors according to $\theta_q \sim p(\theta | D_t)$.
3: sample the jitter according to $j \sim p(j)$.
4: $\tilde{a}_{s,j}(x | D_t) \leftarrow \frac{1}{s} \sum_{q=1}^{s} a(x, \theta_{q,j} | D_t)$.
5: $x_{t+i} \leftarrow \arg \max_x \tilde{a}_{s,j}(x | D_t)$, the $i$-th batch point.
6: evaluate $f(x_{t+i})$.
7: $D_{t+M} \leftarrow D_t \cup \{(x_k, f(x_k)) \forall t < k \leq t + M\}.$

Algorithm 3 Batch iteration, hallucinated Acquisition Thompson Sampling (h-ATS).

Require: dataset $D_t := \{(x_k, f(x_k)) \forall k \leq t\}$, sequential acquisition function $a(x, \theta | D_t)$, batch size $M$.

1: $D_t := D_t$.
2: for $i = 1, \ldots, M$ do
3: sample $s$ new GP hyper-parameter vectors according to $\theta_q \sim p(\theta | D_{t+i-1})$, the hallucinated data posterior.
4: $\tilde{a}_{s,j}^{h}(x | D_t) \leftarrow \frac{1}{s} \sum_{q=1}^{s} a(x, \theta_{q} | D_t)$.
5: $x_{t+i} \leftarrow \arg \max_x \tilde{a}_{s,j}^{h}(x | D_t)$, the $i$-th batch point.
6: $h_{t+i} = \frac{1}{s} \sum_{q=1}^{s} \mu(x_{t+i}, \theta_q | D_t)$.
7: $D_{t+i} \leftarrow D_{t+i-1} \cup \{(x_{t+i}, h_{t+i})\}$.
8: for $i = 1, \ldots, M$ in parallel do
9: evaluate $f(x_{t+i})$.
10: $D_{t+M} \leftarrow D_t \cup \{(x_k, f(x_k)) \forall t < k \leq t + M\}.$

make use of the $\theta_q$ samples:

$$h_i = \frac{1}{s} \sum_{q=1}^{s} \mu(x'_i, \theta_q | D_t)$$

where the $\theta_q$s are those sampled using the data posterior on $\tilde{D}_{t-1}$, as in (9), but the posterior is on the last updated model (without hallucinations). We summarize a batch iteration of this hallucinated ATS (h-ATS) in Algorithm 3.

### 3.3. Hallucinating Pending Evaluations (h-ATS)

As explained in Section 2.2, approaches for parallel BO like (Desautels et al., 2014) work by augmenting the dataset with hallucinations of the pending evaluations. Evaluation $i$ is then selected by having $a(x)$ make use of the hallucinated surrogate model $\mathcal{N}(\mu, K | \tilde{D}_{t-1})$. $\tilde{D}_{t-1} = D_t \cup \{(x'_k, h_k) \forall t < k \leq t-1\}$, where $t$ is the time of the last completed evaluation and $h_k$ denotes a hallucination of $f(x'_k)$. As the $h_k$’s are approximations resulting from the last updated surrogate model, these approaches might be over-confident on the collected observations (Kathuria et al., 2016).

If we perform the sampling of the acquisition functions in a sequential manner, we can exploit the same technique in our randomized framework by hallucinating the data posterior, rather than the surrogate model (which remains independent of the pending evaluations). By doing that, we can make use of the knowledge of the locations of the pending evaluations in order to “update” the stochastic process to $a_s(x | D_{t-1})$. This means that, at iteration $i$, we sample:

$$\tilde{a}_{s,j}^{h}(x) = \frac{1}{s} \sum_{q=1}^{s} a(x; \theta_q) \text{ s.t. } \theta_q \sim p(\theta | \tilde{D}_{t-1}).$$

From the high-level perspective, sampling from the hallucinated data posterior means sampling models that should explain the hallucinations. In fact, the difference between $p(\theta | \tilde{D}_{t+i-1})$ and $p(\theta | D_t)$ lies in that the former takes the hallucinations into account in its data likelihood $p(D_t \cup \{(x'_k, h_k) \forall t < k \leq t-1\})$. The practical effect, analogously to j-ATS, is to increase the diversity across the batch points. In this case, though, no acquisition function-specific jitter prior needs to be devised; the approach is therefore more readily applicable to new acquisition functions. For what concerns hallucinations, we adapt the approximation employed by Desautels et al. (2014) to

### 4. Experimental Results

This section presents the results of our empirical evaluations for the three variants of the proposed ATS algorithm. In Section 4.1 we benchmark on well-known synthetic functions from the global optimization literature and in Section 4.2 we present the results on the hyper-parameter tuning of a gradient boosting tree algorithm on a binary classification task.

**Methodology and Baselines.** We compare against two state-of-the-art, well-known batch Bayesian optimization algorithms: Local Penalization (LP) (González et al., 2016) and batch LCB (B-LCB) (Desautels et al., 2014), and against Parallel “classical” Thompson Sampling (P-TS) (Kandasamy et al., 2018). LP was taken from the implementation available within the open-source GPyOpt package (the GPyOpt authors, 2016). We instead re-implemented B-LCB and P-TS within our framework. While LP is a general batch BO approach (applicable to any acquisition function), B-LCB was presented and analyzed only for LCB (therefore, even though their hallucination technique might generalize, we adhere to the common experimental practice) and P-TS parallelizes the TS acqui-
sition function only. Both B-LCB and LP are very competitive against many of the more complex, non-general algorithms presented in the last two years (Daxberger & Low, 2017; Gupta et al., 2018; Lyu et al., 2018; Kathuria et al., 2016). Moreover, their performance is superior to other algorithms commonly employed as baselines, such as q-KG (Wu & Frazier, 2016) and q-EI (Chevalier & Ginsbourger, 2016). Moreover, their performance is superior to other algorithms presented in the last two years (Daxberger & Low, 2017; Gupta et al., 2018; Kathuria et al., 2016; Lyu et al., 2018; Kathuria et al., 2017; Gupta et al., 2018; Kandasamy et al., 2018).

We employ two widely used acquisition functions, LCB and EI (cf. Section 2.1) for ATS and LP. For all algorithms, we employ a GP surrogate model with Matérn 5/2 kernel and EI (cf. Section 2.1) for ATS and LP. For all algorithms, we hence forgo such a comparison.

We employ two widely used acquisition functions, LCB and EI (cf. Section 2.1) for ATS and LP. For all algorithms, we employ a GP surrogate model with Matérn 5/2 kernel (Snoek et al., 2012). For LP, B-LCB and P-TS, consistently with common practice (Lyu et al., 2018), the acquisition function parameter vector \( \theta \) is set to its maximum likelihood value after each non-hallucinated model update\(^1\). For ATS, we instead sample multiple \( \theta \)s from its data posterior, as explained in Section 3.1. We perform the sampling by employing the Affine Invariant MCMC Ensemble sampler provided by the *emcee* python library (Foreman-Mackey et al., 2013). We decided to perform all the experiments in the batch BO setting, rather than asynchronous, to keep the uncertainty (i.e., the number of pending evaluations) constant and fixed across the algorithms and time. We remark that in practice all the three employed algorithms can be easily adapted to work in the asynchronous setting. In all cases, before starting the BO algorithms, the surrogate models are initialized with 5 random evaluations.

### 4.1. Synthetic Functions

Table 1 summarizes the characteristics of the synthetic functions we benchmark on. These functions are notoriously hard to optimize due to the presence of multiple local or global minima (Eggholder, Cosines, Hartmann), or deep valley-like regions (Brannin, Rosenbrock). Therefore, they are well-known in the global optimization community, and have been previously employed as benchmarks in the context of batch BO (Kathuria et al., 2016; González et al., 2016; Gupta et al., 2018; Kandasamy et al., 2018).

**Table 1. Synthetic functions characteristics.**

| Function     | Dimension | Domain            | Minimum       |
|--------------|-----------|-------------------|---------------|
| Brannin      | 2         | \([-5, 10]^2\times[0, 15]\] | 0.3979        |
| Cosines      | 2         | \([0, 1]^d\)      | -1.773        |
| Hartmann     | 6         | \([0, 1]^6\)      | -3.322        |
| Eggholder    | 2         | \([-512, 512]^2\] | -959.64       |
| Rosenbrock   | 4         | \([-5, 10]^4\]    | 0             |

\[^1\]Desautels et al., 2014\ keep \( \theta \) constant throughout the BO iterations.

\[^2\]in the case of Hartmann6, where EI and LCB show analogous sequential performance, we selected EI, as it exhibited a better

**Figure 1.** Parallel vs. sequential BO on three synthetic functions for varying batch size \( M \) and acquisition function.
that the relative performance of acquisition functions depends on the optimization problem at hand (Hoffman et al., 2011). The plots report mean values of regret (distance from the function minimum, clipped at $1e-6$) over at least 10 repetitions of the experiments; the ATS variant with the best performance on the given task is highlighted with a thicker line. In all plots, plain ATS eventually improves on P-TS, highlighting the benefits of focusing on the probability of maximizing an acquisition function rather than the model posterior. We separate the benchmark functions into two categories:

Figure 3 presents results for valley-like functions. B-LCB seems to excel on low-dimensional, relatively easy to optimize functions like Branin. The ATS variants do not show significantly different behavior, with one exception. On the Rosenbrock function, as the optimization proceeds, plain ATS gets below the current minimum found by all the other parallel algorithms: this suggests that the more explorative functions are not fully capable of descending the deep valley region; pure exploitation eventually pays off.

On the strongly multi-modal functions shown in Figure 2 (Cosines and Eggholder exhibit sinusoidal oscillations), instead, differences become more noticeable and the ATS variants perform particularly well. The simplest variant of our algorithm, plain ATS, is slower than $j$-ATS and $h$-ATS in the first iterations. By closely looking within the first batches, we indeed noticed a lack of diversity between the points. In fact, the posterior is particularly uninformative at the beginning of the optimization loop, so over-exploiting it, as ATS does, might be myopic. This is precisely the intuition that led us to the addition of jitter in the formulation of $j$-ATS in Section 3.2. $j$-ATS on average is the variant showing the best performance and clearly beats (Hart-
4.2. Hyper-Parameter Optimization

| Hyper-parameter                          | Domain          |
|------------------------------------------|-----------------|
| Number of boosting rounds                | [16, 256]       |
| Step shrinkage (η)                       | [0.1, 1]        |
| Maximum depth of a tree                  | [2, 8]          |
| Regularization coefficient (λ)           | \(10^{-2}, 10^7\) |
| Subsample ratio of columns               | [0.1, 1]        |

Table 3. Hyper-parameter domains for the tuning of XGBoost.

We perform a similar comparison on the real-world problem of Hyper-Parameter Optimization (HPO). We focus on gradient boosting decision trees, as they enjoy widespread adoption in academia, industry and competitive data science due to their state-of-the-art performance in a wide variety of machine learning tasks (Ke et al., 2017). Specifically, our goal is to tune the hyper-parameters of the popular XGBoost (Chen & Guestrin, 2016), so as to minimize the logistic loss incurred on a binary classification problem. We utilized the well-known Higgs (P. Baldi and P. Sadowski and D. Whiteson, 2014) dataset, freely available within the UCI ML repository, whose task is to predict whether a given process will produce Higgs bosons or not. As the overall number of signals in the original dataset is larger than ten millions, we uniformly subsampled \(1'200'000\) examples for the training, and \(400'000\) for the validation set to make the experiments feasible on our infrastructure. Table 3 reports the domains for the 5 hyper-parameters we tuned; we kept the maximum number of bins equal to 64. Figure 4 shows the minimum loss incurred on the fixed validation set after \(t\) batch iterations, averaged over 5 repetitions. We opted for a large batch size \((M = 20)\) to mimic a realistic scenario where the significant time cost of HPO needs to be amortized through parallelism. Moreover, we chose LCB over EI due to its better sequential performance, as in the previous Section. The results show that ATS-based algorithms are competitive to B-LCB also on HPO, while LP shows inferior performance. The ATS variant exhibiting the best behavior is, again, \(j\)-ATS, which achieves almost ideal speedup to sequential BO. This is to be expected, as in larger batches the diversity provided by sampling \(\theta\) alone might not be sufficient, and employing hallucinations might be short-sighted with respect to the future model updates.

5. Conclusions

We presented Acquisition Thompson Sampling (ATS), a novel algorithm for parallel Bayesian Optimization (BO) based on sampling acquisition functions from a stochastic process through their dependency on the surrogate model hyper-parameters. Moreover, we enhanced this conceptually simple algorithm by translating ideas from the batch BO literature into our randomized framework.

First \((j\text{-ATS})\), by adding jitter to the acquisition functions, we sample different explore-exploit trade-offs from a prior distribution embedded in the definition of the process, as opposed to obtaining these trade-offs by solving a multi-objective optimization problem (Gupta et al., 2018).

Second \((h\text{-ATS})\), we employ hallucinations of the pending evaluations to diversify the data posterior of the surrogate model hyper-parameters, rather than to update the surrogate model itself (Desautels et al., 2014). We obtain a versatile and multi-faceted algorithm, usable to parallelize any sequential acquisition function.

Finally, we benchmark against two state-of-the-art batch BO algorithms (González et al., 2016; Desautels et al., 2014) and against parallel Thompson Sampling (Kandasamy et al., 2018) on five synthetic functions from the global optimization literature, and on tuning a popular gradient boosting tree algorithm. It is shown that variants of ATS are competitive with the employed baselines and, in particular, out-perform them on strongly multi-modal functions.
References

Chapelle, O. and Li, L. An empirical evaluation of thompson sampling. In Proceedings of the 24th International Conference on Neural Information Processing Systems, NIPS’11, pp. 2249–2257, USA, 2011. Curran Associates Inc. ISBN 978-1-61839-599-3. URL http://dl.acm.org/citation.cfm?id=2986459.2986710.

Chen, T. and Guestrin, C. Xgboost: A scalable tree boosting system. In Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’16, pp. 785–794, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939785. URL http://doi.acm.org/10.1145/2939672.2939785.

Chevalier, C. and Ginsbourger, D. Fast computation of the multi-points expected improvement with applications in batch selection. In Revised Selected Papers of the 7th International Conference on Learning and Intelligent Optimization - Volume 7997, LION 7, pp. 59–69, New York, NY, USA, 2013. Springer-Verlag New York, Inc. ISBN 978-3-642-44972-7.

Daxberger, E. A. and Low, B. K. H. Distributed batch Gaussian process optimization. In Proceedings of the 34th International Conference on Machine Learning, volume 70 of Proceedings of Machine Learning Research, pp. 951–960, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

Desautels, T., Krause, A., and Burdick, J. Parallelizing exploration-exploitation tradeoffs with gaussian process bandit optimization. In Proc. International Conference on Machine Learning (ICML), 2014.

Foreman-Mackey, D., Hogg, D. W., Lang, D., and Goodman, J. emcee: The mcmc hammer. PASP, 125:306–312, 2013.

González, J., Dai, Z., Hennig, P., and Lawrence, N. Batch bayesian optimization via local penalization. In Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS 2016), volume 51 of JMLR Workshop and Conference Proceedings, pp. 648–657, 2016.

Gupta, S., Shilton, A., Rana, S., and Venkatesh, S. Exploiting strategy-space diversity for batch bayesian optimization. In Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics, volume 84 of Proceedings of Machine Learning Research, pp. 538–547, Playa Blanca, Lanzarote, Canary Islands, 09–11 Apr 2018. PMLR.

Hernandez-Lobato, J. M., Gelbart, M., Hoffman, M., Adams, R., and Ghahramani, Z. Predictive entropy search for bayesian optimization with unknown constraints. In Proceedings of the 32nd International Conference on Machine Learning, volume 37 of Proceedings of Machine Learning Research, pp. 1699–1707, Lille, France, 07–09 Jul 2015. PMLR.

Hoffman, M., Brochu, E., and de Freitas, N. Portfolio allocation for bayesian optimization. In Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence, UAI’11, pp. 327–336, Arlington, Virginia, United States, 2011. AUAI Press. ISBN 978-0-9749039-7-2.

Hutter, F., Hoos, H. H., and Leyton-Brown, K. Sequential model-based optimization for general algorithm configuration. In Proceedings of the 5th International Conference on Learning and Intelligent Optimization, LION’05, pp. 507–523, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-25565-6.

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

Kandasamy, K., Krishnamurthy, A., Schneider, J., and Poczos, B. Parallelised bayesian optimisation via thompson sampling. In Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics, volume 84 of Proceedings of Machine Learning Research, pp. 133–142, Playa Blanca, Lanzarote, Canary Islands, 09–11 Apr 2018. PMLR.

Kathuria, T., Deshpande, A., and Kohli, P. Batched gaussian process bandit optimization via determinantal point processes. In Advances in Neural Information Processing Systems 29, pp. 4206–4214. Curran Associates, Inc., 2016.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. LightGBM: A highly efficient gradient boosting decision tree. In NIPS, pp. 3149–3157, 2017.

Li, C., Rubín de Celis Leal, D., Rana, S., Gupta, S., Sutti, A., Greenhill, S., Slezak, T., Height, M., and Venkatesh, S. Rapid bayesian optimisation for synthesis of short polymer fiber materials. Scientific Reports, 7(1):5683, 2017. ISSN 2045-2322.

Lyu, W., Yang, F., Yan, C., Zhou, D., and Zeng, X. Batch Bayesian optimization via multi-objective acquisition ensemble for automated analog circuit design. In Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pp. 3312–3320, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR.
P. Baldi and P. Sadowski and D. Whiteson. Searching for Exotic Particles in High-energy Physics with Deep Learning. *Nature Communications* 5:4308 doi: 10.1038/ncomms5308, 2014.

Rasmussen, C. and Williams, C. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, USA, January 2006.

Shah, A. and Ghahramani, Z. Parallel predictive entropy search for batch global optimization of expensive objective functions. In *Advances in Neural Information Processing Systems* 28, pp. 3330–3338. Curran Associates, Inc., 2015.

Shahriari, B., Bouchard-Cote, A., and Freitas, N. Unbounded bayesian optimization via regularization. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pp. 1168–1176, Cadiz, Spain, 09–11 May 2016. PMLR.

Snoek, J., Larochelle, H., and Adams, R. P. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems* 25, pp. 2951–2959. Curran Associates, Inc., 2012.

Solnik, B., Golovin, D., Kochanski, G., Karro, J. E., Moitra, S., and Sculley, D. Bayesian optimization for a better dessert. In *Proceedings of the 2017 NIPS Workshop on Bayesian Optimization*, December 9, 2017, Long Beach, USA, 2017. The workshop is BayesOpt 2017 NIPS Workshop on Bayesian Optimization December 9, 2017, Long Beach, USA.

Springenberg, J. T., Klein, A., Falkner, S., and Hutter, F. Bayesian optimization with robust bayesian neural networks. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems* 29, pp. 4134–4142. Curran Associates, Inc., 2016.

Srinivas, N., Krause, A., Kakade, S. M., and Seeger, M. W. Information-theoretic regret bounds for gaussian process optimization in the bandit setting. *Information Theory. IEEE Transactions on*, pp. 3265, 2012.

the GPyOpt authors. GPyOpt: A bayesian optimization framework in python. [http://github.com/SheffieldML/GPyOpt](http://github.com/SheffieldML/GPyOpt), 2016.

Wang, J., Clark, S. C., Liu, E., and Frazier, P. I. Parallel Bayesian Global Optimization of Expensive Functions. *ArXiv e-prints*, February 2016.

Wu, J. and Frazier, P. The parallel knowledge gradient method for batch bayesian optimization. In *Advances in Neural Information Processing Systems* 29, pp. 3126–3134. Curran Associates, Inc., 2016.