Parallelised Bayesian Optimisation via Thompson Sampling

Kirthevasan Kandasamy
Carnegie Mellon University
kandasamy@cs.cmu.edu

Akshay Krishnamurthy
University of Massachusetts, Amherst
akshay@cs.umass.edu

Jeff Schneider, Barnabás Póczos
Carnegie Mellon University
{schneide, bapoczos}@cs.cmu.edu

Abstract

We design and analyse variations of the classical Thompson sampling (TS) procedure for Bayesian optimisation (BO) in settings where function evaluations are expensive but can be performed in parallel. Our theoretical analysis shows that a direct application of the sequential Thompson sampling algorithm in either synchronous or asynchronous parallel settings yields a surprisingly powerful result: making n evaluations distributed among M workers is essentially equivalent to performing n evaluations in sequence. Further, by modelling the time taken to complete a function evaluation, we show that, under a time constraint, asynchronous parallel TS achieves asymptotically lower regret than both the synchronous and sequential versions. These results are complemented by an experimental analysis, showing that asynchronous TS outperforms a suite of existing parallel BO algorithms in simulations and in an application involving tuning hyper-parameters of a convolutional neural network. In addition to these, the proposed procedure is conceptually much simpler than existing work for parallel BO.

1 Introduction

Many real world problems require maximising an expensive unknown function f from noisy evaluations. As evaluations are typically expensive in such applications, we would like to optimise the function with a minimal number of evaluations. For example, consider the task of tuning the hyper-parameters of a machine learning model, which can be framed as a black-box optimisation problem where an evaluation to f at x trains the model on the hyper-parameters x, and returns the cross validation accuracy $f(x)$ on the validation set. The evaluation of $f(x)$ is noisy due to sources of randomness in the training procedure and is typically expensive, especially with large models. In drug discovery, each x characterises a candidate drug and $f(x)$ measures various qualities such as the potency, specificity, and solubility of the drug via an expensive *in vitro* or *in vivo* test. Bayesian optimisation (BO) refers to a suite of methods for black-box optimisation under Bayesian assumptions on f that has been successfully applied in hyper-parameter tuning, drug discovery and other applications in policy search, online advertising, and scientific experimentation [12, 17, 30, 32, 40].

Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS) 2018, Lanzarote, Spain. PMLR: Volume 84. Copyright 2018 by the author(s).

In this paper, we develop new algorithms for *parallel Bayesian optimisation.* For example, in hyper-parameter tuning, with modern computing infrastructures, we have the ability to evaluate several hundred hyper-parameters in parallel. The training time for each hyper-parameter is influenced by a myriad of factors, including contention on shared compute resources and the actual hyper-parameter choices, so it typically exhibits significant variability. Our goal is to find a set of hyper-parameters that achieve low validation error, in a short amount of time. Similarly, in drug discovery, high throughput screening equipment can test several thousand candidate drugs at the same time.

Addressing this problem in the above and several other applications with parallel function evaluations, we design and analyse new algorithms for parallel Bayesian optimisation. Our algorithms are synchronous and asynchronous parallel versions of Thompson Sampling (TS), which we call synTS and asyTS, respectively. These algorithms are conceptually simple, easy to implement, and also scale to large number of parallel evaluations. In a departure from prior work on parallel BO, we explicitly model evaluation times and study the relationship between optimisation performance and time, in addition to the more standard relationship between optimisation and the number of function evaluations. Our main contributions are:

1. A theoretical analysis demonstrating that both synTS and asyTS making n evaluations distributed among M workers is almost as good as if the n evaluations were made in sequence.
2. We introduce and analyse simple regret with time as a resource in parallel settings. Under this definition, asyTS outperforms the synchronous and sequential versions up to constant factors.
3. Empirically, we demonstrate that TS significantly

outperforms existing methods for parallel BO in both the synchronous and asynchronous settings on several synthetic problems and a hyperparameter tuning task. A python implementation of our algorithm and experiments is available at `github.com/kirthevasank/gp-parallel-ts`.

Related Work

Bayesian optimisation methods start with a prior belief distribution for f and incorporate function evaluations into updated beliefs in the form of a posterior. Popular algorithms choose points to evaluate f via deterministic query rules such as expected improvement (EI) [21] or upper confidence bounds (UCB) [41]. We however, will focus on a randomised selection procedure known as Thompson sampling [42], which selects a point by maximising a random sample from the posterior. TS has been explored for sequential BO [4, 39] and some recent theoretical advances have characterised the performance of TS in sequential settings [3, 7, 27, 35, 36].

There has been a flurry of recent activity in parallelising BO [8, 9, 11, 13, 19, 26, 38, 43–46]. In comparison to this prior work, our approach enjoys one or more of the following advantages.

1. **Asynchronicity:** The majority of work on parallel BO are in the synchronous (batch) setting. To our knowledge, only [11, 19, 43] can handle asynchronous parallelisation.
2. **Theoretical underpinnings:** Most methods for parallel BO do not come with theoretical guarantees, with the exception of some work using UCB techniques [8, 9, 26]. Crucially, to the best of our knowledge, no theoretical guarantees are available for asynchronous methods.
3. **Conceptual simplicity:** *All* of the above methods either introduce additional hyper-parameters and/or ancillary computational subroutines. Some methods become computationally prohibitive when there are a large number of workers and must resort to approximations [19, 38, 43, 46]. In contrast, our approach is conceptually simple – a direct adaptation of the sequential TS algorithm to the parallel setting. Hence, it is robust in practice, especially with a large number of workers. Further, unlike existing methods, its computationally complexity does not increase with M and is exactly the same as the sequential version.

We mention that parallelised versions of TS have been explored to varying degrees in some applied domains of bandit and reinforcement learning research [15, 18, 31]. However, to our knowledge, we are the first to theoretically analyse parallel TS. More importantly, we are also the first to develop and study TS in an asynchronous parallel setting. Besides BO, there has been a line of work on online learning with delayed feedback (as we have in the parallel setting) [22, 33]. In addition, Jun et al. [23] study a best-arm identification problem when queries are issued in batches. But these papers only consider finite decision sets and do not model evaluation times to study trade-offs when time is viewed as the primary resource.

2 Preliminaries

We wish to maximise an unknown function $f : \mathcal{X} \to \mathbb{R}$ defined on a compact domain $\mathcal{X} \subset \mathbb{R}^d$, by repeatedly obtaining noisy evaluations of f; when we evaluate f at $x \in \mathcal{X}$, we observe $y = f(x)+\epsilon$ where the noise ϵ satisfies $\mathbb{E}[\epsilon] = 0$. We work in the Bayesian paradigm, modeling f itself as a random quantity. Following the plurality of Bayesian optimisation literature, we assume that f is a sample from a Gaussian process [34] and that the noise, $\epsilon \sim \mathcal{N}(0, \eta^2)$, is i.i.d normal. A Gaussian process (GP) is characterised by a mean function $\mu : \mathcal{X} \to \mathbb{R}$ and prior (covariance) kernel $\kappa : \mathcal{X}^2 \to \mathbb{R}$. If $f \sim \mathcal{GP}(\mu, \kappa)$, then $f(x)$ is distributed normally as $\mathcal{N}(\mu(x), \kappa(x, x))$ for all $x \in \mathcal{X}$. Additionally, given n observations $A = \{(x_i, y_i)\}_{i=1}^n$ from this GP, where $x_i \in \mathcal{X}$, $y_i = f(x_i) + \epsilon_i \in \mathbb{R}$, the posterior for f is also a GP with mean μ_A and covariance κ_A given by,

$$\begin{aligned} \mu_A(x) &= k^\top (K + \eta^2 I_n)^{-1} Y, \\ \kappa_A(x, \tilde{x}) &= \kappa(x, \tilde{x}) - k^\top (K + \eta^2 I_n)^{-1} \tilde{k}. \end{aligned} \tag{1}$$

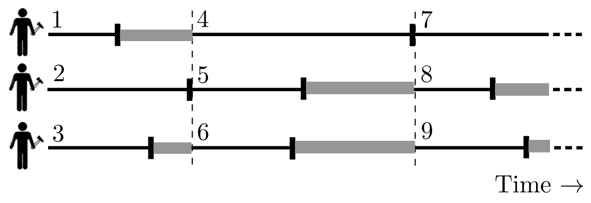
Here $Y \in \mathbb{R}^n$ such that $Y_i = y_i$, and $k, \tilde{k} \in \mathbb{R}^n$ are such that $k_i = \kappa(x, x_i), \tilde{k}_i = \kappa(\tilde{x}, x_i)$. The Gram matrix $K \in \mathbb{R}^{n\times n}$ is given by $K_{i,j} = \kappa(x_i, x_j)$, and $I_n \in \mathbb{R}^{n\times n}$ is the identity matrix. Some common choices for the kernel are the squared exponential (SE) kernel and the Matérn kernel. We refer the reader to Rasmussen and Williams [34] for more background on GPs.

Our goal is to find the maximiser $x_\star = \operatorname{argmax}_{x\in\mathcal{X}} f(x)$ of f through repeated evaluations. In the BO literature, this is typically framed as minimising the *simple regret*, which is the difference between the optimal value $f(x_\star)$ and the best evaluation of the algorithm. Since f is a random quantity, so is its optimal value and hence the simple regret. This motivates studying the *Bayes simple regret*, which is the expectation of the simple regret. Formally, we define the simple regret, SR(n), and Bayes simple regret, BSR(n), of an algorithm after n evaluations as,

$$\begin{aligned} \mathrm{SR}(n) &= f(x_\star) - \max_{j=1,\dots,n} f(x_j), \\ \mathrm{BSR}(n) &= \mathbb{E}[\mathrm{SR}(n)]. \end{aligned} \tag{2}$$

The expectation in BSR(n) is with respect to the prior $f \sim \mathcal{GP}(\mathbf{0}, \kappa)$, the noise in the observations $\epsilon_j \sim \mathcal{N}(0, \eta^2)$, and any randomness of the algorithm. We focus on simple regret here mostly to simplify exposition; our proof also applies for *cumulative regret*, which may be more familiar.

In many applications of BO, including hyperparameter tuning, the time required to evaluate the function is the dominant cost, and we are most interested in maximising f in



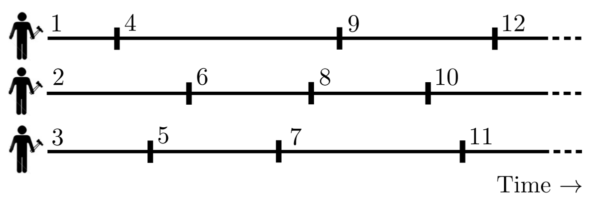


Figure 1: An illustration of the synchronous (left) and asynchronous (right) settings using $M = 3$ workers. The short vertical lines indicate when a worker finished its last evaluation. In the synchronous setting the grey shaded regions indicate idle time after a worker finishes its job. The horizontal location of a number indicates when the worker started its next evaluation while the number itself denotes the order in which the evaluation was dispatched by the algorithm.

a short period of time. Moreover, there is often considerable variability in the time required for different evaluations, caused by inherent differences between points in the domain, randomness of the environment, or other factors. For example, in the hyperparameter tuning application, unpredictable factors such as resource contention, initialisation, etc., may induce significant variability in evaluation times.

To adequately capture these settings, we model the time to complete an evaluation as a random variable, and measure performance in terms of the simple regret within a time budget, T. Specifically, letting $N = N(T)$ denote the (random) number of evaluations performed by an algorithm within time T, we define the simple regret $\mathrm{SR}'(T)$ and the Bayes simple regret $\mathrm{BSR}'(T)$ as

$$\mathrm{SR}'(T) = \begin{cases} f(x_\star) - \max_{j\leq N} f(x_j) & \text{if } N \geq 1 \\ \max_{x\in\mathcal{X}} |f(x_\star) - f(x)| & \text{otherwise} \end{cases},$$
$$\mathrm{BSR}'(T) = \mathbb{E}[\mathrm{SR}'(T)]. \tag{3}$$

This definition is similar to (2), except, when an algorithm has not completed an evaluation yet, its simple regret is the worst possible value. In $\mathrm{BSR}'(T)$, the expectation now also includes the randomness in the evaluation times in addition to the three sources of randomness in $\mathrm{BSR}(n)$. In this work, we will model evaluation times as random variables independent from f; specifically we consider uniform, half-normal, or exponential random variables. While the model does not precisely capture all aspects of evaluation times observed in practice, we prefer it because (a) it is fairly general, (b) it leads to a clean algorithm and analysis, and (c) the resulting algorithm has good performance on real applications, as we demonstrate in Section 4. Studying other models for the evaluation time is an intriguing question for future work and is discussed further in Section 5.

To our knowledge, all prior theoretical work for parallel BO [8, 9, 26], measures regret in terms of the total number of evaluations, i.e. $\mathrm{SR}(n), \mathrm{BSR}(n)$. However, explicitly modeling evaluation times and treating time as the main resource in the definition of regret is a better fit for applications and leads to new conclusions in the parallel setting.

Parallel BO: We are interested in parallel approaches for BO, where the algorithm has access to M workers that can evaluate f at different points in parallel. In this setup, we wish to differentiate between the synchronous and asynchronous settings, illustrated in Fig. 1. In the former, the algorithm issues a batch of M queries simultaneously, one per worker, and waits for all M evaluations to be completed before issuing the next batch. In contrast, in the asynchronous setting, a new evaluation may be issued as soon as a worker finishes its last job and becomes available. In the parallel setting, N in (3) will refer to the number of evaluations completed by *all* M workers.

One of our goals in the theoretical analysis will be to quantify the trade-offs between information accumulation and worker utilisation in the sequential, synchronous parallel and asynchronous parallel settings. When comparing the three settings purely in terms of the number of evaluations, i.e. $\mathrm{BSR}(n)$, the parallel settings are naturally at a disadvantage: the sequential algorithm makes use of feedback from all its previous evaluations when issuing a query, whereas a parallel algorithm could be missing up to $M-1$ of them. As we will see however, for our TS algorithms, this difference is fairly small - the bounds for the parallel algorithms are only slightly worse than for sequential variants. The advantage in the parallel setting however, is that we will be able to complete more evaluations than a sequential version within an allotted time. One can make a similar argument to compare the synchronous and asynchronous settings. When issuing queries, a synchronous algorithm has more information about f, since all previous evaluations complete before a batch is selected, whereas asynchronous algorithms always issue queries with $M-1$ missing evaluations. For example, in Fig. 1, when dispatching the fourth job, the synchronous version uses results from the first three evaluations whereas the asynchronous version uses just the result of the first evaluation. However, in the synchronous setting, workers may sit idle for some time waiting for the other workers to finish. Foreshadowing our results in Theorem 5, when there is significant variability in evaluation times, worker utilisation is more important than information accumulation, and hence the asynchronous setting will enable better bounds on $\mathrm{BSR}'(T)$. Next, we present our algorithms.

3 Thompson Sampling for Parallel BO

A review of sequential TS: Thompson sampling [42] is a randomised strategy for sequential decision making under uncertainty. At step j, TS samples x_j according to the posterior probability that it is the optimum. That is, x_j is drawn from the posterior density $p_{x_\star}(\cdot|\mathcal{D}_j)$ where $\mathcal{D}_j = \{(x_i, y_i)\}_{i=1}^{j-1}$ is the history of query-observation pairs up to step j. For GPs, this allows for a very simple and elegant algorithm. Observe that we can write $p_{x_\star}(x|\mathcal{D}_j) = \int p_{x_\star}(x|g)\, p(g|\mathcal{D}_j)\mathrm{d}g$, and that $p_{x_\star}(\cdot|g)$ puts all its mass at the maximiser $\mathrm{argmax}_x\, g(x)$ of g. Therefore, at step j, we draw a sample g from the posterior for f conditioned on $\mathcal{D}_j$ and set $x_j = \mathrm{argmax}_x\, g(x)$ to be the maximiser of g. We then evaluate f at x_j. The resulting procedure, called seqTS, is displayed in Algorithm 1.

Algorithm 1: seqTS Thompson [42]

```
Require: Prior GP  $\mathcal{GP}(\mathbf{0}, \kappa)$.
1: $\mathcal{D}_1 \leftarrow \varnothing$,  $\mathcal{GP}_1 \leftarrow \mathcal{GP}(\mathbf{0}, \kappa)$.
2: for $j = 1, 2, \ldots$ do
3:     Sample $g \sim \mathcal{GP}_j$.
4:     $x_j \leftarrow \mathrm{argmax}_{x\in\mathcal{X}}\, g(x)$.
5:     $y_j \leftarrow$ Query $f$ at $x_j$.
6:     $\mathcal{D}_{j+1} \leftarrow \mathcal{D}_j \cup \{(x_j, y_j)\}$.
7:     Compute posterior $\mathcal{GP}_{j+1} = \mathcal{GP}(\mu_{\mathcal{D}_{j+1}}, \kappa_{\mathcal{D}_{j+1}})$
       conditioned on $\mathcal{D}_{j+1}$. See (1).
8: end for
```

Asynchronous Parallel TS: For the asynchronously parallel setting, we propose a natural adaptation of the above algorithm. Precisely, when a worker finishes an evaluation, we update the posterior with the query-feedback pair, sample g from the posterior, and re-deploy the worker with an evaluation at $x_j = \mathrm{argmax}_x\, g(x)$. The procedure, called asyTS, is displayed in Algorithm 2. In the first M steps, when at least one of the workers have not been assigned a job yet, the algorithm skips lines 3–5 and samples g from the prior GP, $\mathcal{GP}_1$, in line 6.

Algorithm 2: asyTS

```
Require: Prior GP  $\mathcal{GP}(\mathbf{0}, \kappa)$.
1: $\mathcal{D}_1 \leftarrow \varnothing$,  $\mathcal{GP}_1 \leftarrow \mathcal{GP}(\mathbf{0}, \kappa)$.
2: for $j = 1, 2, \ldots$ do
3:     Wait for a worker to finish.
4:     $\mathcal{D}_j \leftarrow \mathcal{D}_{j-1} \cup \{(x', y')\}$ where $(x', y')$ are the
       worker's previous query/observation.
5:     Compute posterior $\mathcal{GP}_j = \mathcal{GP}(\mu_{\mathcal{D}_j}, \kappa_{\mathcal{D}_j})$.
6:     Sample $g \sim \mathcal{GP}_j$,  $x_j \leftarrow \mathrm{argmax}\, g(x)$.
7:     Re-deploy worker to evaluate $f$ at $x_j$.
8: end for
```

Synchronous Parallel TS: To illustrate comparisons, we also introduce a synchronous parallel version, synTS, which makes the following changes to Algorithm 2. In line 3 we wait for all M workers to finish and compute the GP posterior with all M evaluations in lines 4–5. In line 6 we draw M samples and re-deploy all workers with evaluations at their maxima in line 7.

We wish to highlight the main methodological differences of our algorithms with prior work for parallel BO. Since existing methods select points using deterministic criteria such as UCB or EI, they need to *explicitly* enforce diversity of query points so as to prevent the algorithm from picking the same or similar points for all M workers. Consequently, such methods introduce additional hyperparameters and/or potentially expensive computational routines. In contrast, asyTS and synTS are essentially the same as their sequential counterpart and their computational complexity does not increase with M. In addition to this computational advantage, this conceptual simplicity results in robust empirical performance in practice. Our theoretical analysis shows that a straightforward application of TS works because its inherent randomness is sufficient to avoid redundant function evaluations when managing M workers in parallel. This phenomenon is confirmed by our experiments in Section 4, where we see that explicitly encouraging diversity does not improve the performance of asyTS. We demonstrate this empirically by constructing a variant asyHTS of asyTS which employs one such diversity scheme found in the literature. asyHTS performs either about the same as or slightly worse than asyTS in the many experiments we study in Section 4.

3.1 Theoretical Results

We now present our theoretical contributions. We analyse the performance of parallelised TS both with the number of evaluations n and the time budget T as the resource. In particular, we study how these rates change with the number of workers M and demonstrate that as M increases, while BSR(n) worsens slightly for the parallel settings when compared to the sequential setting, BSR$'(T)$ can improve dramatically. We provide theorem statements here to convey key intuitions, with all formal statements and proofs deferred to Appendices A and B. We use $\asymp, \lesssim$ to denote equality/inequality up to constant factors that are common across all theorem statements.

Maximum Information Gain (MIG): As in prior work, our regret bounds involve the MIG [41], which captures the statistical difficulty of the BO problem. It quantifies the maximum information a set of n observations provide about f. To define the MIG, and for subsequent convenience, we introduce one notational convention. For a finite subset $A \subset \mathcal{X}$, we use $y_A = \{(x, f(x) + \epsilon) \mid x \in A\}$ to denote the query-observation pairs corresponding to the set A. The MIG is then defined as $\Psi_n = \max_{A\subset\mathcal{X}, |A|=n} I(f; y_A)$ where I denotes the Shannon Mutual Information. Srinivas et al. [41] show that Ψ_n is sublinear in n for different classes of kernels; e.g. for the SE kernel, $\Psi_n \propto \log(n)^{d+1}$ and for the Matérn kernel with smoothness parameter ν, $\Psi_n \propto n^{1-\frac{\nu}{2\nu+d(d+1)}}$.

Our first goal is to compare the simple regret $\mathrm{BSR}(n)$ after n evaluations for synTS and asyTS with that of seqTS. To this end, we prove the following theorem for seqTS which is a straightforward extension of a result in [35].

Theorem 1 (Informal. $\mathrm{BSR}(n)$ for seqTS)**.** *Let $f \sim \mathcal{GP}(\mathbf{0}, \kappa)$. Then, for* seqTS*,* $\mathrm{BSR}(n) \lesssim \sqrt{\Psi_n \log(n)/n}$.

The first theoretical result of this paper, presented below in Theorem 2, bounds $\mathrm{BSR}(n)$ for synTS.

Theorem 2 (Informal. $\mathrm{BSR}(n)$ for synTS)**.** *Let $f \sim \mathcal{GP}(\mathbf{0}, \kappa)$. Then, for* synTS*,*

$$\mathrm{BSR}(n) \lesssim \frac{M\sqrt{\log(M)}}{n} + \sqrt{\frac{\Psi_n \log(n+M)}{n}}.$$

A comparison of the bounds in Theorems 1 and 2 reveals that, for reasons explained before in Section 2, seqTS outperforms synTS purely in terms of the number of evaluations n. However, for large n, the first term in the bound for synTS vanishes faster than the latter, and the dependence on M in the latter term is insignificant when $n \gg M$. Hence, the difference between the sequential and synchronous parallel algorithms is small and negligible for large n. We also note that the leading constant for the second term is the same as that in Theorem 1. This implies a powerful conclusion: synTS with M parallel workers is almost as good as the sequential version with as many evaluations.

To present the results for the asynchronous setting, we introduce the following quantity ξ_M, which bounds the information we can gain about f from the evaluations in progress. Assume that we have completed n evaluations to f at the points in $\mathcal{D}_n$ and that there are q evaluations in process at points in A_q. That is $\mathcal{D}_n, A_q \subset \mathcal{X}$, $|\mathcal{D}_n| = n$ and $|A_q| = q < M$. Then $\xi_M > 0$ satisfies the following for all $n \geq 1$,

$$\max_{A_q \subset \mathcal{X}, |A_q|<M} I(f; y_{A_q} | y_{\mathcal{D}_n}) \;\leq\; \frac{1}{2}\log(\xi_M). \qquad (4)$$

Our next result is a bound on $\mathrm{BSR}(n)$ for asyTS.

Theorem 3 (Informal. $\mathrm{BSR}(n)$ for asyTS)**.** *Let $f \sim \mathcal{GP}(\mathbf{0}, \kappa)$. Then, for* asyTS*,*

$$\mathrm{BSR}(n) \lesssim \sqrt{\frac{\xi_M \Psi_n \log(n)}{n}}.$$

Unfortunately, this bound depends on ξ_M which can be quite large under general conditions. However, ξ_M is a well studied quantity in the GP literature; precisely, Desautels et al. [9], Krause et al. [28] show that ξ_M can be bounded by a kernel dependent constant C_κ by initially querying f using an uncertainty sampling procedure for γ_M samples. This sampling procedure, which iteratively samples the points with the largest variance in the GP, is asynchronously parallelisable. Desautels et al. [9] shows that for the Matérn kernel, with $\gamma_M \asymp \mathrm{poly}(M)$, this procedure guarantees $C_\kappa \leq e^e$, and for the SE kernel, with $\gamma_M \asymp M\mathrm{polylog}(M)$, we can achieve any constant $C_\kappa > 1$, depending on the order of the polylog term. These values for C_κ, γ_M are not absolute – by picking a larger γ_M we can achieve smaller C_κ. In all cases however, γ_M is at most polynomial in M and does not depend on n. We provide more details on the initialisation scheme in Appendix A.4. By initialising asyTS with this sampling scheme, we obtain the bound below.

Corollary 4 (Informal. $\mathrm{BSR}(n)$ for asyTS after initialisation)**.** *Let $f \sim \mathcal{GP}(\mathbf{0}, \kappa)$. By first initialising* asyTS *with an uncertainty sampling scheme [9, 28], we have,*

$$\mathrm{BSR}(n) \lesssim \frac{\gamma_M}{n} + \sqrt{\frac{C_\kappa \Psi_n \log(n)}{n}}.$$

Despite the dependence on the initialisation scheme and the constant term C_κ, Corollary 4 is encouraging: since the first term in the bound vanishes faster than the latter, up to constant factors, asyTS with M parallel workers is almost as good as seqTS.

That said, we believe that bounds similar to Theorem 2 should be obtainable for asyTS without the additional constant C_κ and without the initialisation scheme. For instance, asyTS performs very well in our experiments even though we do not use this initialisation scheme. We leave it to future work to resolve this gap.

Now that we have bounds on the regret as a function of the number of evaluations, we can turn to bounding $\mathrm{BSR}'(T)$, the simple regret with time as the main resource. For this, we consider three different random distribution models for the time to complete a function evaluation: uniform, half-normal, and exponential. We choose these three distributions since they exhibit three different notions of tail decay, namely bounded, sub-Gaussian, and sub-exponential[1]. Table 1 describes these distributions and states the expected number of evaluations $n_{\mathrm{seq}}, n_{\mathrm{syn}}, n_{\mathrm{asy}}$ for seqTS, synTS, asyTS respectively with M workers in time T. The final theoretical result of this paper, presented below, bounds $\mathrm{BSR}'(T)$ for the Thompson sampling variants.

Theorem 5 (Informal, Simple regret with time for TS)**.** *Let $f \sim \mathcal{GP}(0, \kappa)$ and assume that for* asyTS*, ξ_M is bounded by C_κ after suitable initialisation. Assume that the times taken for an evaluation are i.i.d random variables with either uniform, half-normal or exponential distributions. Let $n_{seq}, n_{syn}, n_{asy}$ be as given in Table 1. Then $n_{seq} \leq n_{syn} \leq n_{asy}$ and $\mathrm{BSR}'(T)$ can be upper bounded by the following terms for* seqTS*,* synTS*, and* asyTS*.*

$$\text{seqTS:} \quad \sqrt{\frac{\Psi_{n_{\mathrm{seq}}} \log(n_{\mathrm{seq}})}{n_{\mathrm{seq}}}},$$

[1]While we study uniform, half-normal and exponential, analogous results for other distributions with similar tail behaviour are possible with the appropriate concentration inequalities. See Appendix B.

Distribution	pdf $p(x)$	seqTS	synTS	asyTS
$\text{Unif}(a,b)$	$\frac{1}{b-a}$ for $x \in (a,b)$	$n_{\text{seq}} = \frac{2T}{b+a}$	$n_{\text{syn}} = M\frac{T(M+1)}{a+bM}$	$n_{\text{asy}} = Mn_{\text{seq}} \;\; (> n_{\text{syn}})$
$\mathcal{HN}(\zeta^2)$	$\frac{\sqrt{2}}{\zeta\sqrt{\pi}}e^{-\frac{x^2}{2\zeta^2}}$ for $x > 0$	$n_{\text{seq}} = \frac{T\sqrt{\pi}}{\zeta\sqrt{2}}$	$n_{\text{syn}} \asymp \frac{Mn_{\text{seq}}}{\sqrt{\log(M)}}$	$n_{\text{asy}} = Mn_{\text{seq}}$
$\text{Exp}(\lambda)$	$\lambda e^{-\lambda x}$ for $x > 0$	$n_{\text{seq}} = \lambda T$	$n_{\text{syn}} \asymp \frac{Mn_{\text{seq}}}{\log(M)}$	$n_{\text{asy}} = Mn_{\text{seq}}$

Table 1: The second column shows the probability density functions $p(x)$ for the uniform Unif(a, b), half-normal $\mathcal{HN}(\zeta^2)$, and exponential Exp(λ) distributions. The subsequent columns show the expected number of evaluations $n_{\text{seq}}, n_{\text{syn}}, n_{\text{asy}}$ for seqTS, synTS, and asyTS respectively with M workers. synTS always completes fewer evaluations than asyTS; e.g., in the exponential case, the difference could be a $\log(M)$ factor.

$$\text{synTS:} \quad \frac{M\sqrt{\log(M)}}{n_{\text{syn}}} + \sqrt{\frac{\Psi_{n_{\text{syn}}}\log(n_{\text{syn}}+M)}{n_{\text{syn}}}},$$

$$\text{asyTS:} \quad \frac{\text{poly}(M)}{n_{\text{asy}}} + \sqrt{\frac{C_\kappa\Psi_{n_{\text{asy}}}\log(n_{\text{asy}})}{n_{\text{asy}}}}.$$

As the above bounds are decreasing with the number of evaluations and since $n_{\text{seq}} < n_{\text{syn}} < n_{\text{asy}}$, the bound for BSR$'(T)$ shows the opposite trend to BSR(n): asyTS is better than synTS which is better than seqTS. While the difference between synTS and asyTS is only a constant factor for the uniform distribution, it grows with the number of workers M for heavier tailed distributions; $\sqrt{\log(M)}$ for the half-normal and $\log(M)$ for the exponential. Hence, as the number of workers M increases, asyTS becomes increasingly attractive when compared to synTS. Intuitively, when there is more variability in evaluations, workers may sit idle for longer in the synchronous setting and hence synTS will complete fewer evaluations than asyTS.

Synopsis: The take-aways of our theoretical analysis can be summarised as follows. Theorems 2 and 3 show that since the synchronous setting has more information than the asynchronous setting, it achieves a better bound for BSR(n). Therefore, if function evaluations deterministically take the same amount of time, the synchronous algorithm may be preferred. Further, in some applications, we are necessarily in the synchronous setting. For example, in pre-clinical drug discovery, high throughput screening equipment can test a few thousand compounds in parallel, but only in batches [16]. However, Theorem 5 contends that if there is significant variability in evaluation times, then it is prudent to be asynchronous despite the lack of information when compared to the synchronous setting.

A note on the proofs: To lift the sequential TS proof of Russo and Van Roy [35] to the synchronous case we exploit several properties of TS in this setting; for example, the distribution of $x_j|\mathcal{D}_j$ is the same as $x_\star|\mathcal{D}_j$, all jobs in one batch are completed before the next, and that conditioned on the randomness of the algorithm, the jobs in one batch are deterministic. These properties, along with a careful decomposition of terms in the instantaneous regret yields the bound. Unfortunately these properties do not hold in the asynchronous case, and we resort to techniques from Desautels et al. [9] to bound the MIG via uncertainty sampling. For Theorem 5, we establish concentration results for sums of random variables and sums of their maxima. The proofs of the uniform and half-normal cases uses standard sub-Gaussianity arguments whereas the proof for the exponential distribution uses a logarithmic Sobolev inequality and Herbst's argument [6].

4 Experiments

We compare parallelised TS with a comprehensive suite of parallel BO methods from the literature on a series of synthetic experiments and a hyper-parameter tuning task on the CIFAR-10 dataset.

Methods: *Synchronous Methods:* synRAND: synchronous random sampling, synTS: synchronous TS, synBUCB from [9], synUCBPE from [8]. *Aynchronous Methods:* asyRAND: asynchronous random sampling, asyHUCB: an asynchronous version of UCB with hallucinated observations [9, 11], asyUCB: asynchronous upper confidence bound [41], asyEI: asynchronous expected improvement [21], asyTS: asynchronous TS, asyHTS: asynchronous TS with hallucinated observations to explicitly encourage diversity. This last method is based on asyTS but bases the posterior on $\mathcal{D}_j \cup \{(x, \mu_{\mathcal{D}_j}(x))\}_{x\in F_j}$ in line 5 of Algorithm 2, where F_j are the points in evaluation by other workers at step j and $\mu_{\mathcal{D}_j}$ is the posterior mean conditioned on just $\mathcal{D}_j$; this preserves the mean of the GP, but shrinks the variance around the points in F_j. This method is inspired by [9, 11], who use such hallucinations for UCB/EI-type strategies so as to discourage picking points close to those that are already in evaluation. asyUCB and asyEI directly use the sequential UCB and EI criteria, since the asynchronous versions do not repeatedly pick the same point for all workers. asyHUCB adds hallucinated observations to encourage diversity and is similar to [11] (who use EI instead) and is also an asynchronous version of [9]. While there are other methods for parallel BO, many of them are either computationally quite expensive and/or require tuning several hyperparameters which might affect performance; They are not straightforward to implement and their implementations are not publicly available. Appendix C describes

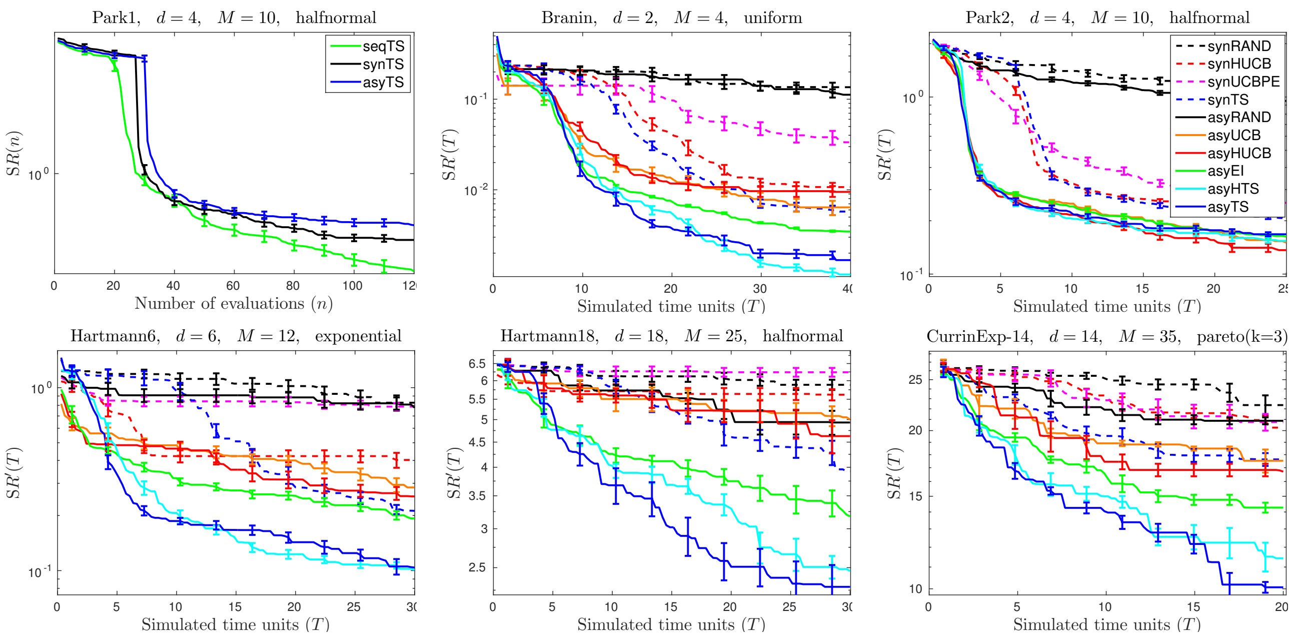


Figure 2: Results on the synthetic experiments. The title states the function used, its dimensionality d, the number of workers M and the distribution used for the time. All distributions were constructed so that the expected time for one evaluation was one time unit. The dotted lines depict synchronous methods while the solid lines are for asynchronous methods. The error bars indicate one standard error. All figures were averaged over at least 15 experiments.

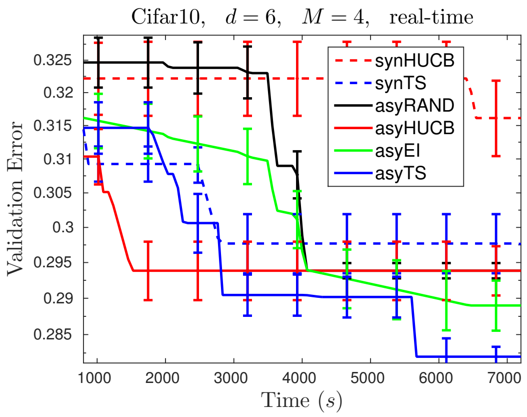
implementation details for all methods.

Synthetic Experiments: We first present some results on a suite of benchmarks for global optimisation. To better align with our theoretical analysis, we add Gaussian noise to the function value when querying. This makes the problem more challenging than standard global optimisation where evaluations are not noisy. In our first experiment, we corroborate the claims in Theorems 1, 2, and 3 by comparing the performance of seqTS, synTS, and asyTS in terms of the number of evaluations n on the Park1 function. The results, displayed in the first panel of Fig. 2, confirm that when comparing solely in terms of n, the sequential version outperforms the parallel versions while synchronous does marginally better than asynchronous.

Next, we present results on a series of global optimisation benchmarks with different values for the number of parallel workers M. We model the evaluation "time" as a random variable that is drawn from either a uniform, half-normal, exponential, or Pareto[2] distribution. Each time a worker makes an evaluation, we also draw a sample from this time distribution and maintain a queue to simulate the different start and finish times for each evaluation. The results are presented in Fig. 2 where we plot the simple regret $\mathrm{SR}'(T)$ against (simulated) time T. In the Park2 experiment, all asynchronous methods perform roughly the same and outperform the synchronous methods. On all other the other problems, asyTS performs best among the asynchronous methods and synTS among the synchronous methods. asyHTS, which also uses hallucinated observations, performs about the same or slightly worse than asyTS, demonstrating that there is no need to explicitly encourage diversity in TS. It is worth emphasizing that the improvement of TS over other methods become larger as M increases (e.g. $M > 20$). We believe that the ability to scale robustly with the number of workers is primarily due to the conceptual simplicity of our approach. Appendix C provides more details on these functions and additional synthetic experiments.

Image Classification on Cifar-10: We experiment with tuning hyperparameters of a 6 layer convolutional neural network on an image classification task on the Cifar-10 dataset [29]. We tune the number of filters/neurons at each layer in the range $(16, 256)$. Here, each function evaluation trains the model on 10K images for 20 epochs and computes the validation accuracy on a validation set of 10K images. Our implementation uses Tensorflow [1] and we use a parallel set up of $M = 4$ Titan X GPUs. The number of filters influences the training time which varied between ~ 4 to ~ 16 minutes depending on the size of the model. Note that this deviates from our theoretical analysis which treats function evaluation times as independent random variables, but it still introduces variability to evaluation times and demonstrates the robustness of our approach. Each method is given a budget of 2 hours to find the best model by optimising accuracy on a validation set. These evaluations are noisy since the result of each training procedure depends on the initial parameters of the network and other stochasticity in the training procedure. Since the true value of this function is unknown, we simply report the best validation error achieved by each method. Due to the expensive nature of

[2] A Pareto distribution with parameter k has a pdf which decays $p(x) \propto x^{-(k+1)}$.



synBUCB	synTS	asyRAND
25.63 ± 0.002	22.83 ± 1.01	23.93 ± 1.78

asyEI	asyHUCB	asyTS
$\mathbf{19.49 \pm 0.21}$	22.14 ± 1.12	$\mathbf{19.53 \pm 0.11}$

Figure 3: Results on the Cifar-10 experiment. Left: The best validation set error vs time for each method (lower is better). Top: Test set error after training the best model chosen by each method for 80 epochs. The results presented are averaged over 9 experiments.

this experiment we only compare 6 of the above methods. The results are presented in Fig. 3. asyTS performs best on the validation error. The following are ranges for the number of evaluations for each method over 9 experiments; *synchronous:* synBUCB: 56 - 68, synTS: 56 - 68. *asynchronous:* asyRAND: 93 - 105, asyEI: 83 - 92, asyHUCB: 85 - 92, asyTS: 80 - 88.

While 20 epochs is insufficient to completely train a model, the validation error gives a good indication of how well the model would perform after sufficient training. In Fig. 3, we also give the error on a test set of 10K images after training the best model chosen by each algorithm to completion, i.e. for 80 epochs. asyTS and asyEI are able to recover the best models which achieve an accuracy of about 80%. While this falls short of state of the art results on Cifar-10 (for e.g. [14]), it is worth noting that we use only a small subset of the Cifar-10 dataset and a relatively small model. Nonetheless, it demonstrates the superiority of our approach over other baselines.

5 Conclusion

We study parallelised versions of TS for synchronous and asynchronous BO. Theoretically, we demonstrate that the algorithms synTS and asyTS perform as well as their purely sequential counterpart in terms of number of evaluations. However, when we factor in time, asyTS outperforms the other two versions. Practically, the main advantage of the proposed methods over prior approaches are conceptual simplicity and straightforward implementation, which enables us to scale robustly to a large number of workers.

We close with some avenues for future research. One challenge that we have already mentioned, is bounding the regret for asyTS without the initialisation procedure. Second, we are interested in other models for evaluation times, for example to capture correlations between the evaluation time and the query point $x_j \in \mathcal{X}$ that arise in practice, such as in our CNN experiment. We believe that in such problems, even sequential algorithms might be different from the conventional BO methods. One could also consider models where some workers are slower than the rest. Third, in the asynchronous setting, there might be instances where the algorithm might choose to kill an evaluation in progress based on the result of a completed job. The algorithm might also choose to wait for another evaluation to finish without immediately deploying a free worker, so as to incorporate additional information. Finally, an open challenge for TS in GPs is maximising the sample g (step 4, Algorithm 1) which can be computationally challenging as g is a random quantity (See Appendix C). We look forward to pursuing these directions.

Acknowledgements

This research is partly funded by DOE grant DESC0011114, NSF grant IIS1563887, the Darpa D3M program, and AFRL. KK is supported by a Facebook fellowship and a Siebel scholarship.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv:1603.04467*, 2016.

[2] Robert J Adler. *An Introduction to Continuity, Extrema, and Related Topics for General Gaussian Processes*. IMS, 1990.

[3] Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *Conference on Learning Theory (COLT)*, 2012.

[4] Hildo Bijl, Thomas B Schön, Jan-Willem van Wingerden, and Michel Verhaegen. A Sequential Monte Carlo approach to Thompson sampling for Bayesian optimization. *arXiv preprint arXiv:1604.00169*, 2016.

[5] Stéphane Boucheron and Maud Thomas. Concentration inequalities for order statistics. *Electronic Communications in Probability*, 2012.

[6] Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration inequalities: A nonasymptotic theory of independence*. Oxford University Press, 2013.

[7] Sayak Ray Chowdhury and Aditya Gopalan. On kernelized multi-armed bandits. *arXiv:1704.00445*, 2017.

[8] Emile Contal, David Buffoni, Alexandre Robicquet, and Nicolas Vayatis. Parallel Gaussian process optimization with upper confidence bound and pure exploration. In *European Conference on Machine Learning (ECML/PKDD)*, 2013.

[9] Thomas Desautels, Andreas Krause, and Joel W Burdick. Parallelizing exploration-exploitation tradeoffs in Gaussian process bandit optimization. *Journal of Machine Learning Research (JMLR)*, 2014.

[10] Subhashis Ghosal and Anindya Roy. Posterior consistency of Gaussian process prior for nonparametric binary regression". *Annals of Statistics*, 2006.

[11] David Ginsbourger, Janis Janusevskis, and Rodolphe Le Riche. Dealing with asynchronicity in parallel gaussian process based global optimization. In *Conference of the ERCIM WG on Computing and Statistics*, 2011.

[12] Javier Gonzalez, Joseph Longworth, David James, and Neil Lawrence. Bayesian Optimization for Synthetic Gene Design. In *NIPS Workshop on Bayesian Optimization in Academia and Industry*, 2014.

[13] Javier González, Zhenwen Dai, Philipp Hennig, and Neil D Lawrence. Batch Bayesian Optimization via Local Penalization. *arXiv:1505.08052*, 2015.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[15] José Miguel Hernández-Lobato, James Requeima, Edward O Pyzer-Knapp, and Alán Aspuru-Guzik. Parallel and distributed thompson sampling for large-scale accelerated exploration of chemical space. *arXiv preprint arXiv:1706.01825*, 2017.

[16] James P Hughes, Stephen Rees, S Barrett Kalindjian, and Karen L Philpott. Principles of early drug discovery. *British journal of pharmacology*, 162(6):1239–1249, 2011.

[17] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *LION*, 2011.

[18] Brett W Israelsen, Nisar R Ahmed, Kenneth Center, Roderick Green, and Winston Bennett. Towards adaptive training of agent-based sparring partners for fighter pilots. In *AIAA Information Systems-AIAA Infotech@ Aerospace*, page 0343, 2017.

[19] Janis Janusevskis, Rodolphe Le Riche, David Ginsbourger, and Ramunas Girdziusas. Expected Improvements for the Asynchronous Parallel Global Optimization of Expensive Functions: Potentials and Challenges. In *Learning and Intelligent Optimization*, 2012.

[20] D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian Optimization Without the Lipschitz Constant. *J. Optim. Theory Appl.*, 1993.

[21] Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *J. of Global Optimization*, 1998.

[22] Pooria Joulani, Andras Gyorgy, and Csaba Szepesvári. Online learning under delayed feedback. In *International Conference on Machine Learning (ICML)*, 2013.

[23] Kwang-Sung Jun, Kevin Jamieson, Robert Nowak, and Xiaojin Zhu. Top arm identification in multi-armed bandits with batch arm pulls. In *Artificial Intelligence and Statistics (AISTATS)*, 2016.

[24] Kirthevasan Kandasamy, Jeff Schenider, and Barnabás Póczos. High Dimensional Bayesian Optimisation and Bandits via Additive Models. In *International Conference on Machine Learning*, 2015.

[25] Kirthevasan Kandasamy, Gautam Dasarathy, Junier Oliva, Jeff Schenider, and Barnabás Póczos. Gaussian Process Bandit Optimisation with Multi-fidelity Evaluations. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[26] Tarun Kathuria, Amit Deshpande, and Pushmeet Kohli. Batched gaussian process bandit optimization via determinantal point processes. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[27] Emilie Kaufmann, Nathaniel Korda, and Rémi Munos. Thompson sampling: An asymptotically optimal finite-time analysis. In *ALT*, volume 12, pages 199–213. Springer, 2012.

[28] Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9(Feb):235–284, 2008.

[29] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images, 2009.

[30] R. Martinez-Cantin, N. de Freitas, A. Doucet, and J. Castellanos. Active Policy Learning for Robot Plan-

ning and Exploration under Uncertainty. In *Proceedings of Robotics: Science and Systems*, 2007.

[31] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[32] David Parkinson, Pia Mukherjee, and Andrew R Liddle. A Bayesian model selection analysis of WMAP3. *Physical Review*, 2006.

[33] Kent Quanrud and Daniel Khashabi. Online learning with adversarial delays. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[34] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. Adaptative computation and machine learning series. University Press Group Limited, 2006.

[35] Daniel Russo and Benjamin Van Roy. Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 39(4):1221–1243, 2014.

[36] Daniel Russo and Benjamin Van Roy. An Information-theoretic analysis of Thompson sampling. *Journal of Machine Learning Research (JMLR)*, 2016.

[37] MW. Seeger, SM. Kakade, and DP. Foster. Information Consistency of Nonparametric Gaussian Process Methods. *IEEE Transactions on Information Theory*, 2008.

[38] Amar Shah and Zoubin Ghahramani. Parallel predictive entropy search for batch global optimization of expensive objective functions. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[39] Bobak Shahriari, Ziyu Wang, Matthew W Hoffman, Alexandre Bouchard-Côté, and Nando de Freitas. An Entropy Search Portfolio for bayesian Optimization. *arXiv preprint arXiv:1406.4625*, 2014.

[40] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian Optimization of Machine Learning Algorithms. In *Advances in Neural Information Processing Systems*, 2012.

[41] Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. In *ICML*, 2010.

[42] W. R. Thompson. On the Likelihood that one Unknown Probability Exceeds Another in View of the Evidence of Two Samples. *Biometrika*, 1933.

[43] Jialei Wang, Scott C Clark, Eric Liu, and Peter I Frazier. Parallel Bayesian Global Optimization of Expensive Functions. *arXiv:1602.05149*, 2016.

[44] Zi Wang, Chengtao Li, Stefanie Jegelka, and Pushmeet Kohli. Batched high-dimensional bayesian optimization via structural kernel learning. *arXiv:1703.01973*, 2017.

[45] Zi Wang, Clement Gehring, Pushmeet Kohli, and Stefanie Jegelka. Batched large-scale bayesian optimization in high-dimensional spaces. In *AISTATS*, 2018.

[46] Jian Wu and Peter Frazier. The parallel knowledge gradient method for batch bayesian optimization. In *Advances In Neural Information Processing Systems*, 2016.