Parallel and Distributed Thompson Sampling for Large-scale Accelerated Exploration of Chemical Space

José Miguel Hernández-Lobato [*1] **James Requeima** [*1 2] **Edward O. Pyzer-Knapp** [3 4] **Alán Aspuru-Guzik** [3]

Abstract

Chemical space is so large that brute force searches for new interesting molecules are infeasible. *High-throughput virtual screening* via computer cluster simulations can speed up the discovery process by collecting very large amounts of data in parallel, e.g., up to hundreds or thousands of parallel measurements. Bayesian optimization (BO) can produce additional acceleration by sequentially identifying the most useful simulations or experiments to be performed next. However, current BO methods cannot scale to the large numbers of parallel measurements and the massive libraries of molecules currently used in high-throughput screening. Here, we propose a scalable solution based on a parallel and distributed implementation of Thompson sampling (PDTS). We show that, in small scale problems, PDTS performs similarly as parallel expected improvement (EI), a batch version of the most widely used BO heuristic. Additionally, in settings where parallel EI does not scale, PDTS outperforms other scalable baselines such as a greedy search, ϵ-greedy approaches and a random search method. These results show that PDTS is a successful solution for large-scale parallel BO.

1. Introduction

Chemical space is huge: it is estimated to contain over 10^{60} molecules. Among these, fewer than 100 million compounds can be found in public repositories or databases (Reymond et al., 2012). This discrepancy between *known* compounds and *possible* compounds indicates the potential for discovering many new compounds with highly desirable functionality (e.g., new energy materials, pharmaceuticals, dyes, etc.). While the vast size of chemical space makes this an enormous opportunity, it also presents a significant difficulty in the identification of new relevant compounds among the many unimportant ones. This challenge is so great that any discovery process relying purely on the combination of scientific intuition with trial and error experimentation is slow, tedious and in many cases infeasible.

To accelerate the search, high-throughput approaches can be used in a combinatorial exploration of small specific areas of chemical space (Rajan, 2008). These have led to the development of high-throughput virtual screening (Pyzer-Knapp et al., 2015; Gómez-Bombarelli et al., 2016) in which large libraries of molecules are created and then analyzed using theoretical and computational techniques, typically by running a large number of parallel simulations in a computer cluster. The objective is to reduce an initially very large library of molecules to a small set of promising leads for which expensive experimental evaluation is justified. However, even though these techniques only search a tiny drop in the ocean of chemical space, they can result in massive libraries whose magnitude exceeds traditional computational capabilities. As a result, at present, there is an urgent need to accelerate high-throughput screening approaches.

Bayesian optimization (BO) (Jones et al., 1998) can speed up the discovery process by using machine learning to guide the search and make improved decisions about what molecules to analyze next given the data collected so far. However, current BO methods cannot scale to the large number of parallel measurements and the massive libraries of candidate molecules currently used in high-throughput screening (Pyzer-Knapp et al., 2015). While there are BO methods that allow parallel data collection, these methods have typically been limited to tens of data points per batch (Snoek et al., 2012; Shahriari et al., 2014; Gonzlez et al., 2016). In contrast, high-throughput screening may allow the simultaneous collection of thousands of data points via large-scale parallel computation. This creates a need for new scalable methods for parallel Bayesian optimization.

[*]Equal contribution [1]University of Cambridge, Cambridge, UK [2]Invenia Labs, Cambridge, UK [3]Harvard University, Cambridge, USA [4]IBM Research, UK. Correspondence to: José Miguel Hernández-Lobato <jmh233@cam.ac.uk>, Edward O. Pyzer-Knapp <epyzerk3@uk.ibm.com>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

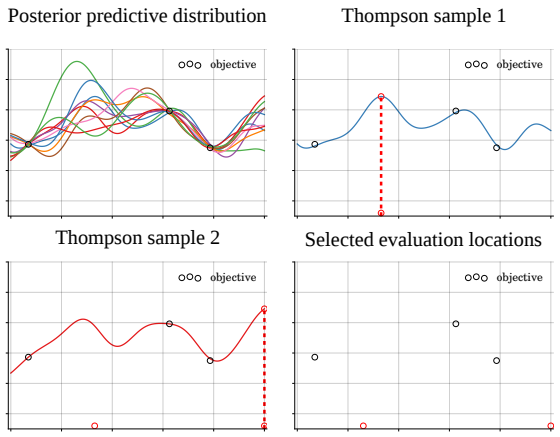

Figure 1. Illustration of Thompson sampling and PDTS.

To address the above difficulty, we present here a scalable solution for parallel Bayesian optimization based on a distributed implementation of the Thompson sampling heuristic (Thompson, 1933; Chapelle & Li, 2011). We show that, for the case of small batch sizes, the proposed parallel and distributed Thompson sampling (PDTS) method performs as well as a parallel implementation of expected improvement (EI) (Snoek et al., 2012; Ginsbourger et al., 2011), the most widely used Bayesian optimization heuristic. Parallel EI selects the batch entries sequentially and so EI proposals can't be parallelized, which limits its scalability properties. PDTS generates each batch of evaluation locations by selecting the different batch entries independently and in parallel. Consequently, PDTS is highly scalable and applicable to large batch sizes. We also evaluate the performance of PDTS in several real-world high-throughput screening experiments for material and drug discovery, where parallel EI is infeasible. In these problems, PDTS outperforms other scalable baselines such as a greedy search strategy, ϵ-greedy approaches and a random search method. These results indicate that PDTS is a successful solution for large-scale parallel Bayesian optimization.

Algorithm 1 Sequential Thompson sampling

Input: initial data $\mathcal{D}_{\mathcal{I}(1)} = \{(\mathbf{x}_i, y_i)\}_{i \in \mathcal{I}(1)}$
for $t = 1$ **to** T **do**
  Compute current posterior $p(\boldsymbol{\theta}|\mathcal{D}_{\mathcal{I}(t)})$
  Sample $\boldsymbol{\theta}$ from $p(\boldsymbol{\theta}|\mathcal{D}_{\mathcal{I}(t)})$
  Select $k \leftarrow \text{argmax}_{j \notin \mathcal{I}(t)} \mathbf{E}[y_j|\mathbf{x}_j, \boldsymbol{\theta}]$
  Collect y_k by evaluating f at $\mathbf{x}_k$
  $\mathcal{D}_{\mathcal{I}(t+1)} \leftarrow \mathcal{D}_{\mathcal{I}(t)} \cup \{(\mathbf{x}_k, y_k)\}$
end for

2. BO and Thompson Sampling

Let us assume we have a large library of candidate molecules $\mathcal{M} = \{m_1, \ldots, m_{|\mathcal{M}|}\}$. Our goal is to identify a small subset of elements $\{m_i\} \subset \mathcal{M}$ for which the $f(m_i)$ are as high as possible, with f being an expensive-to-evaluate objective function. The objective f could be, for example, an estimate of the power-conversion efficiency of organic photovoltaics, as given by expensive quantum mechanical simulations (Scharber et al., 2006), and we may want to identify the top 1% elements in $\mathcal{M}$ according to this score.

Bayesian optimization methods can be used to identify the inputs that maximize an expensive objective function f by performing only a reduced number of function evaluations. For this, BO uses a model to make predictions for the value of f at new inputs given data from previous evaluations. The next point to evaluate is then chosen by maximizing an acquisition function that quantifies the benefit of evaluating the objective at a particular location.

Let $\mathbf{x}_1, \ldots, \mathbf{x}_{|\mathcal{M}|}$ be D-dimensional feature vectors for the molecules in $\mathcal{M}$ and let $\mathcal{D}_{\mathcal{I}} = \{(\mathbf{x}_i, y_i) : i \in I\}$ be a dataset with information about past evaluations, where I is a set with the indices of the molecules already evaluated, $\mathbf{x}_i$ is the feature vector for the i-th molecule in $\mathcal{M}$ and $y_i = f(m_i)$ is the result of evaluating the objective function f on that molecule. We assume that the evaluations of f are noise free, however, the methods described here can be applied to the case in which the objective evaluations are corrupted with additive Gaussian noise. BO typically uses a probabilistic model to describe how the y_i in $\mathcal{D}_{\mathcal{I}}$ are generated as a function of the corresponding features $\mathbf{x}_i$ and some model parameters $\boldsymbol{\theta}$, that is, the model specifies $p(y_i|\mathbf{x}_i, \boldsymbol{\theta})$. Given the data $\mathcal{D}_{\mathcal{I}}$ and a prior distribution $p(\boldsymbol{\theta})$, the model also specifies a posterior distribution $p(\boldsymbol{\theta}|\mathcal{D}_{\mathcal{I}}) \propto p(\boldsymbol{\theta}) \prod_{i \in I} p(y_i|\mathbf{x}_i, \boldsymbol{\theta})$. The predictive distribution for any $m_j \in \mathcal{M} \setminus \{m_i : i \in I\}$ is then given by $p(y_j|\mathbf{x}_j, \mathcal{D}_{\mathcal{I}}) = \int p(y_j|\mathbf{x}_j, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}_{\mathcal{I}})\, d\boldsymbol{\theta}$. BO methods use this predictive distribution to compute an acquisition function (AF) given by

$$\alpha(\mathbf{x}_j|\mathcal{D}_{\mathcal{I}}) = \mathbf{E}_{p(y_j|\mathbf{x}_j, \mathcal{D}_{\mathcal{I}})} \left[U(y_j|\mathbf{x}_j, \mathcal{D}_{\mathcal{I}}) \right] , \qquad (1)$$

where $U(y_j|\mathbf{x}_j, \mathcal{D}_{\mathcal{I}})$ is the utility of obtaining value y_j when evaluating f at m_j. Eq. (1) is then maximized with respect to $j \notin I$ to select the next molecule m_j on which to evaluate f. The most common choice for the utility is the improvement: $U(y_j|\mathbf{x}_j, \mathcal{D}_{\mathcal{I}}) = \max(0, y_j - y_\star)$, where $y_\star$ is equal to the best y_i in $\mathcal{D}_{\mathcal{I}}$. In this case, Eq. (1) is called the expected improvement (EI) (Jones et al., 1998). Ideally, the AF should encourage both exploration and exploitation. For this, the expected utility should increase when y_j takes high values on average (to exploit), but also when there is high uncertainty about y_j (to explore). The EI utility function satisfies these two requirements.

Thompson sampling (TS) (Thompson, 1933) can be understood as a version of the previous framework in which the utility function is defined as $U(y_j|\mathbf{x}_j, \mathcal{D}_{\mathcal{I}}) = y_j$ and the expectation in (1) is taken with respect to $p(y_j|\mathbf{x}_j, \boldsymbol{\theta})$

instead of $p(y_j|\mathbf{x}_j, \mathcal{D}_\mathcal{I})$, with $\boldsymbol{\theta}$ being a sample from the posterior $p(\boldsymbol{\theta}|\mathcal{D}_\mathcal{I})$. That is, when computing the AF, TS approximates the integral in $p(y_j|\mathbf{x}_j, \mathcal{D}_\mathcal{I}) = \int p(y_j|\mathbf{x}_j, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}_\mathcal{I})\, d\boldsymbol{\theta}$ by Monte Carlo, using a single sample from $p(\boldsymbol{\theta}|\mathcal{D}_\mathcal{I})$ in the approximation. The TS utility function enforces only exploitation because the expected utility is insensitive to any variance in y_j. Despite this, TS still enforces exploration because of the variance produced by the Monte Carlo approximation to $p(y_j|\mathbf{x}_j, \mathcal{D}_\mathcal{I})$. Under TS, the probability of evaluating the objective at a particular location matches the probability of that location being the maximizer of the objective, given the model assumptions and the data from past evaluations. Algorithm 1 contains the pseudocode for TS. The plots in the top of Figure 1 illustrate how TS works. The top-left plot shows several samples from a posterior distribution on f induced by $p(\boldsymbol{\theta}|\mathcal{D}_\mathcal{I})$ since each value of the parameters $\boldsymbol{\theta}$ corresponds to an associated value of f. Sampling from $p(\boldsymbol{\theta}|\mathcal{D}_\mathcal{I})$ is then equivalent to selecting one of these samples for f. The selected sample represents the current AF, which is optimized in the top-right plot in Figure 1 to select the next evaluation.

2.1. Parallel BO

So far we have considered the sequential evaluation setting, where BO methods collect just a single data point in each iteration. However, BO can also be applied in the parallel setting, which involves choosing a batch of multiple points to evaluate next in each iteration. For example, when we run S parallel simulations in a computer cluster and each simulation performs one evaluation of f.

Snoek et al. (2012) describe how to extend sequential BO methods to the parallel setting. The idea is to select the first evaluation location in the batch in the same way as in the sequential setting. However, the next evaluation location is then selected while the previous one is still pending. In particular, given a set K with indexes of pending evaluation locations, we choose a new location in the batch based on the expectation of the AF under all possible outcomes of the pending evaluations according to the predictions of the model. Therefore, at any point, the next evaluation location is obtained by optimizing the AF

$$\alpha_{\text{parallel}}(\mathbf{x}_j|\mathcal{D}_\mathcal{I}, \mathcal{K}) = \mathbf{E}_{p(\{y_k\}_{k\in\mathcal{K}}|\{\mathbf{x}_k\}_{k\in\mathcal{K}}, \mathcal{D}_\mathcal{I})} \left[\alpha(\mathbf{x}_j|\mathcal{D}_\mathcal{I} \cup \mathcal{D}_\mathcal{K})\right], \quad (2)$$

where $\mathcal{D}_\mathcal{K} = \{(y_k, \mathbf{x}_k)\}_{k\in\mathcal{K}}$ and $\alpha(\mathbf{x}_j|\mathcal{D}_\mathcal{I} \cup \mathcal{D}_\mathcal{K})$ is given by (1). Computing this expression exactly is infeasible in most cases. Snoek et al. (2012) propose a Monte Carlo approximation in which the expectation in the second line is approximated by averaging across a few samples from the predictive distribution at the pending evaluations, that is, $p(\{y_k\}_{k\in\mathcal{K}}|\{\mathbf{x}_k\}_{k\in\mathcal{K}}, \mathcal{D}_\mathcal{I})$. These samples are referred to as *fantasized* data.

This approach for parallel BO has been successfully used to collect small batches of data (about 10 elements in size), with EI as utility function and with a Gaussian process as the model for the data (Snoek et al., 2012). However, it lacks scalability to large batch sizes, failing when we need to collect thousands of simultaneous measurements. The reason for this is the high computational cost of adding a new evaluation to the current batch. The corresponding cost includes: ① sampling the fantasized data, ② updating the posterior predictive distribution to $p(y_j|\mathbf{x}_j, \mathcal{D}_\mathcal{I} \cup \mathcal{D}_\mathcal{K})$, which is required for evaluating $\alpha(\mathbf{x}_j|\mathcal{D}_\mathcal{I} \cup \mathcal{D}_\mathcal{K})$, and ③ optimizing the Monte Carlo approximation to (2). Step ② can be very expensive when the number of training points in $\mathcal{D}_\mathcal{I}$ is very large. This step is also considerably challenging when the model does not allow for exact inference, as it is often the case with Bayesian neural networks. Step ③ can also take a very long time when the library of candidate molecules $\mathcal{M}$ is very large (e.g., when it contains millions of elements) and among all the remaining molecules we have to find one that maximizes the AF.

Despite these difficulties, the biggest disadvantage in this approach for parallel BO is that it cannot be parallelized since it is a sequential process in which (2) needs to be iteratively optimized, with each optimization step having a direct effect on the next one. This prevents this method from fully exploiting the acceleration provided by multiple processors in a computer cluster. The sequential nature of the algorithm is illustrated by the plot in the left of Figure 2. In this plot computer node 1 is controlling the BO process and decides the batch evaluation locations. Nodes $2, \ldots, 5$ then perform the evaluations in parallel. Note that steps ② and ③ from the above description have been highlighted in green and magenta colors.

In the following section we describe an algorithm for batch BO which can be implemented in a fully parallel and distributed manner and which, consequently, can take full advantage of multiple processors in a computer cluster. This novel method is based on a parallel implementation of the Thompson sampling heuristic.

Algorithm 2 Parallel and distributed Thompson sampling

Input: initial data $\mathcal{D}_{\mathcal{I}(1)} = \{\mathbf{x}_i, y_i\}_{i\in\mathcal{I}(1)}$, batch size S
for $t = 1$ **to** T **do**
  Compute current posterior $p(\boldsymbol{\theta}|\mathcal{D}_{\mathcal{I}(t)})$
  for $s = 1$ **to** S **do**
    Sample $\boldsymbol{\theta}$ from $p(\boldsymbol{\theta}|\mathcal{D}_{\mathcal{I}(t)})$
    Select $k(s) \leftarrow \text{argmax}_{j\notin\mathcal{I}(t)} \mathbf{E}[y_j|\mathbf{x}_j, \boldsymbol{\theta}]$
    Collect $y_{k(s)}$ by evaluating f at $\mathbf{x}_{k(s)}$
  end for
  Executed in parallel in node s
  $\mathcal{D}_{\mathcal{I}(t+1)} = \mathcal{D}_{\mathcal{I}(t)} \cup \{\mathbf{x}_{k(s)}, y_{k(s)}\}_{s=1}^{S}$
end for

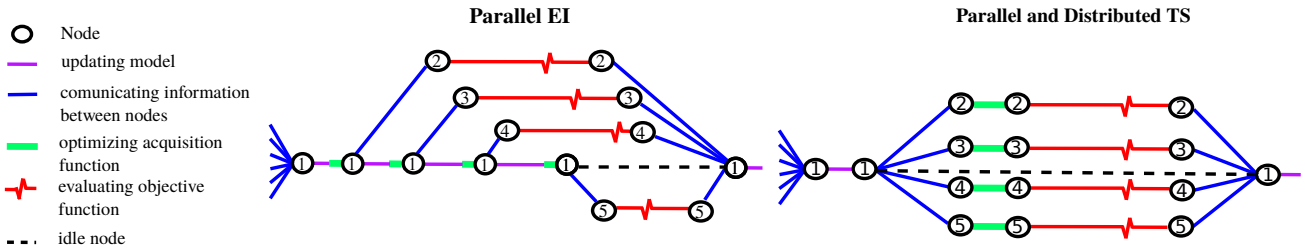

Figure 2. A visualization of one iteration of BO using parallel EI as implemented in (Snoek et al., 2012) and PDTS. Note that in PDTS the model is updated once and sample points are acquired independently by the nodes. With parallel EI, the the location of the next sample points is dependent on the location of previous sample points in the batch so these are computed sequentially.

3. Parallel and Distributed Thompson Sampling

We present an implementation of the parallel BO method from Section 2.1 based on the Thompson sampling (TS) heuristic. In particular, we propose to apply to (2) the same approximation that TS applied to (1). For this, we choose in (2) the same utility function used by TS in the sequential setting, that is, $U(y_j|\mathbf{x}_j, \mathcal{D}_\mathcal{I} \cup \mathcal{D}_\mathcal{K}) = y_j$. Then, we approximate the expectation with respect to $\{y_k\}_{k\in\mathcal{K}}$ in (2) by Monte Carlo, averaging across just one sample of $\{y_k\}_{k\in\mathcal{K}}$ drawn from $p(\{y_k\}_{k\in\mathcal{K}}|\{\mathbf{x}_k\}_{k\in\mathcal{K}}, \mathcal{D}_\mathcal{I})$. After that, $\alpha(\mathbf{x}_j|\mathcal{D}_\mathcal{I} \cup \mathcal{D}_\mathcal{K})$ in (2) is approximated in the same way as in the sequential setting by first sampling $\boldsymbol{\theta}$ from $p(\boldsymbol{\theta}|\mathcal{D}_\mathcal{I} \cup \mathcal{D}_\mathcal{K})$ and then approximating $p(y_j|\mathbf{x}_j, \mathcal{D}_\mathcal{I} \cup \mathcal{D}_\mathcal{K})$ with $p(y_j|\mathbf{x}_j, \boldsymbol{\theta})$. Importantly, in this process, sampling first $\{y_k\}_{k\in\mathcal{K}}$ from $p(\{y_k\}_{k\in\mathcal{K}}|\{\mathbf{x}_k\}_{k\in\mathcal{K}}, \mathcal{D}_\mathcal{I})$ and then $\boldsymbol{\theta}$ from $p(\boldsymbol{\theta}|\mathcal{D}_\mathcal{I} \cup \mathcal{D}_\mathcal{K})$ is equivalent to sampling $\boldsymbol{\theta}$ from just $p(\boldsymbol{\theta}|\mathcal{D}_\mathcal{I})$. The reason for this is that updating a posterior distribution with synthetic data sampled from the model's predictive distribution produces on average the same initial posterior distribution. The result is that parallel TS with batch size S is the same as running sequential TS S times without updating the current posterior $p(\boldsymbol{\theta}|\mathcal{D}_\mathcal{I})$, where each execution of sequential TS produces one of the evaluation locations in the batch. Importantly, these executions can be done in distributed manner, with each one running in parallel in a different node.

The resulting parallel and distributed TS (PDTS) method is highly scalable and can be applied to very large batch sizes by running each execution of sequential TS on the same computer node that will then later evaluate f at the selected evaluation location. Algorithm 2 contains the pseudocode for PDTS. The parallel nature of the algorithm is illustrated by the plot in the right of Figure 2. In this plot computer node 1 is controlling the BO process. To collect four new function evaluations in parallel, computer node 1 sends the current posterior $p(\boldsymbol{\theta}|\mathcal{D}_\mathcal{I})$ and $\mathcal{I}$ to nodes $2, \ldots, 5$. Each of them samples then a value for $\boldsymbol{\theta}$ from the posterior and optimizes its own AF given by $\mathbf{E}[y_j|\mathbf{x}_j, \boldsymbol{\theta}]$, with $j \notin \mathcal{I}$. The objective function is evaluated at the selected input and the resulting data is sent back to node 1. Figure 1 illustrates how PDTS selects two parallel evaluation locations. For this, sequential TS is run twice.

The scalability of PDTS makes it a promising method for parallel BO in high-throughput screening. However, in this type of problem, the optimization of the AF is done over a discrete set of molecules. Therefore, whenever we collect a batch of data in parallel with PDTS, several of the simultaneous executions of sequential TS may choose to evaluate the same molecule. A central computer node (e.g. the node controlling the BO process) maintaining a list of molecules currently selected for evaluation can be used to avoid this problem. In this case, each sequential TS node sends to the central node a ranked list with the top S (the batch size) molecules according to its AF. From this list, the central node then selects the highest ranked molecule that has not been selected for evaluation before.

4. Related Work

Ginsbourger et al. (Ginsbourger et al., 2010) proposed the following framework for parallel BO: given a set of current observations $\mathcal{D}_\mathcal{I}$ and pending experiments $\{\mathbf{x}_k\}_{k=1}^\mathcal{K}$, an additional set of fantasies $\mathcal{D}_\mathcal{K} = \{(\mathbf{x}_k, y_k)\}_{k=1}^\mathcal{K}$ can be assumed to be the result of those pending experiments. A step of Bayesian optimization can then be performed using the augmented dataset $\mathcal{D}_\mathcal{I} \cup \mathcal{D}_\mathcal{K}$ and the acquisition function $\alpha(\mathbf{x}|\mathcal{D}_\mathcal{I} \cup \mathcal{D}_\mathcal{K})$. Two different values are proposed for the fantasies: the *constant liar*, where $y_k = L$ for some constant L and all $k = 1 \ldots \mathcal{K}$, and the *Kriging believer*, where y_k is given by the GP predictive mean at $\mathbf{x}_k$.

Snoek et al. (2012) compute a Monte Carlo approximation of the expected acquisition function over potential fantasies sampled from the model's predictive distribution. Recent methods have been proposed to modify the parallel EI procedure to recommend points jointly (Chevalier & Ginsbourger, 2013; Marmin et al., 2015; Wang et al., 2016).

Azimi et al. (2010) describe a procedure called *simulated matching* whose goal is to propose a batch $\mathcal{D}_\mathcal{K}$ of points which is a good match for the set of samples that a sequential BO policy π would recommend. The authors consider a batch "good" if it contains a sample that yields, with high

probability, an objective value close to that of the best sample produced by a sequential execution of π.

Several authors have proposed to extend the *upper confidence bound* (UCB) heuristic to the parallel setting. Since the GP predictive variance depends only on the input location of the observations, Desautels et al. (2014) propose GP-BUCP acquisition which uses the UCB acquisition with this updated variance. Contal et al. (2013) introduce the Gaussian Process Upper Confidence Bound with Pure Exploration (GP-UCB-PE). Under this procedure, the first point is obtained using the standard UCB acquisition function while the remaining points are sequentially selected to be the ones yielding the highest predictive variance, while still lying in a region that contains the maximizer with high probability.

Shah & Ghahramani (2015) extend the Predictive Entropy Search (PES) heuristic to the parallel setting (PPES). PPES seeks to recommend a collection of samples $\mathcal{D}_\mathcal{K}$ that yields the greatest reduction in entropy for the posterior distribution of $\mathbf{x}^\star$, the latent objective maximizer. Wu & Frazier (2016) propose the Parallel Knowledge Gradient Method which optimizes an acquisition function called the parallel knowledge gradient (q-KG), a measure of the expected incremental solution quality after q samples.

An advantage of PDTS over parallel EI and other related methods is that the approximate marginalization of potential experimental outcomes adds no extra computational cost to our procedure and so PDTS is highly parallelizable. Finally, unlike other approaches, PDTS can be applied to a wide variety of models, such as GPs and Bayesian neural networks, since it only requires samples from an exact or approximate posterior distribution.

5. Bayesian Neural Networks for High-throughput Screening

Neural networks are well-suited for implementing BO on molecules. They produce state-of-the-art predictions of chemical properties (Ma et al., 2015; Mayr et al., 2016; Ramsundar et al., 2015) and can be applied to large data sets by using stochastic optimization (Bousquet & Bottou, 2008). Typical applications of neural networks focus on the deterministic prediction scenario. However, in large search spaces with multiple local optima (which is the case when navigating chemical space), it is desirable to use a probabilistic approach that can produce accurate estimates of uncertainty for efficient exploration and so, we use *probabilistic back-propagation* (PBP), a recently-developed technique for the scalable training of Bayesian neural networks (Hernández-Lobato & Adams, 2015). Note that other methods for approximate inference in Bayesian neural networks could have been chosen as well (Blundell et al., 2015; Snoek et al., 2015; Gal & Ghahramani, 2016). We prefer PBP because it is fast and it does not require the tuning of hyper-parameters such as learning rates or regularization constants (Hernández-Lobato & Adams, 2015).

Given a dataset $\mathcal{D}_\mathcal{I} = \{(\mathbf{x}_i, y_i)\}_{i\in\mathcal{I}}$, we assume that $y_i = f(\mathbf{x}_i; \mathcal{W}) + \epsilon_i$, where $f(\cdot; \mathcal{W})$ is the output of a neural network with weights $\mathcal{W}$. The network output is corrupted with additive noise variables $\epsilon_i \sim \mathcal{N}(0, \gamma^{-1})$. The network has L layers, with V_l hidden units in layer l, and $\mathcal{W} = \{\mathbf{W}_l\}_{l=1}^L$ is the collection of $V_l \times (V_{l-1}+1)$ synaptic weight matrices. The $+1$ is introduced here to account for the additional per-layer biases. The activation functions for the hidden layers are rectifiers: $\varphi(x) = \max(x, 0)$.

The likelihood for the network weights $\mathcal{W}$ and the noise precision γ is

$$p(\{y_i\}_{i\in\mathcal{I}}|\mathcal{W}, \{\mathbf{x}_i\}_{i\in\mathcal{I}}, \gamma) = \prod_{i\in\mathcal{I}} \mathcal{N}(y_i|f(\mathbf{x}_i; \mathcal{W}), \gamma^{-1}) .$$

We specify a Gaussian prior distribution for each entry in each of the weight matrices in $\mathcal{W}$:

$$p(\mathcal{W}|\lambda) = \prod_{l=1}^{L} \prod_{k=1}^{V_l} \prod_{j=1}^{V_{l-1}+1} \mathcal{N}(w_{kj,l}|0, \lambda^{-1}) , \qquad (3)$$

where $w_{kj,l}$ is the entry in the k-th row and j-th column of $\mathbf{W}_l$ and λ is a precision parameter. The hyper-prior for λ is gamma: $p(\lambda) = \text{Gam}(\lambda|\alpha_0^\lambda, \beta_0^\lambda)$ with shape $\alpha_0^\lambda = 6$ and inverse scale $\beta_0^\lambda = 6$. This relatively low value for the shape and inverse scale parameters makes this prior weakly-informative. The prior for the noise precision γ is also gamma: $p(\gamma) = \text{Gam}(\gamma|\alpha_0^\gamma, \beta_0^\gamma)$. We assume that the y_i have been normalized to have unit variance and, as above, we fix $\alpha_0^\gamma = 6$ and $\beta_0^\gamma = 6$.

The exact computation of the posterior distribution for the model parameters $p(\mathcal{W}, \gamma, \lambda|\mathcal{D}_\mathcal{I})$ is not tractable in most cases. PBP approximates the intractable posterior on $\mathcal{W}$, γ and λ with the tractable approximation

$$q(\mathcal{W}, \gamma, \lambda) = \left[\prod_{l=1}^{L} \prod_{k=1}^{V_l} \prod_{j=1}^{V_{l-1}+1} \mathcal{N}(w_{kj,l}|m_{kj,l}, v_{kj,l})\right] \text{Gam}(\gamma \,|\, \alpha^\gamma, \beta^\gamma)\text{Gam}(\lambda \,|\, \alpha^\lambda, \beta^\lambda) , \qquad (4)$$

whose parameters are tuned by iteratively running an assumed density filtering (ADF) algorithm over the training data (Opper, 1998). The main operation in PBP is the update of the mean and variance parameters of q, that is, the $m_{kj,l}$ and $v_{kj,l}$ in (4), after processing each data point $\{(\mathbf{x}_i, y_i)\}$. For this, PBP matches moments between the new q and the product of the old q with the corresponding likelihood factor $\mathcal{N}(y_i \,|\, f(\mathbf{x}_i; \mathcal{W}), \gamma^{-1})$. The matching of moments for the distributions on the weights is achieved

by using well-known Gaussian ADF updates, see equations 5.12 and 5.1 in (Minka, 2001).

To compute the ADF updates, PBP finds a Gaussian approximation to the distribution of the network output $f(\mathbf{x}_i; \mathcal{W})$ when $\mathcal{W} \sim q$. This is achieved by doing a forward pass of $\mathbf{x}_i$ through the network, with the weights $\mathcal{W}$ being randomly sampled from q. In this forward pass the non-Gaussian distributions followed by the output of the neurons are approximated with Gaussians that have the same means and variances as the original distributions. This is a Gaussian approximation by moment matching. We refer the reader to Hernández-Lobato & Adams (2015) for full details on PBP.

After several ADF iterations over the data by PBP, we can then make predictions for the unknown target variable $y_\star$ associated with a new feature vector $\mathbf{x}_\star$. For this, we obtain a Gaussian approximation to $f(\mathbf{x}_\star; \mathcal{W})$ when $\mathcal{W} \sim q$ by applying the forward pass process described above.

To implement TS, as described in Algorithm 1, we first sample the model parameters $\boldsymbol{\theta}$ from the posterior $p(\boldsymbol{\theta}|\mathcal{D}_\mathcal{I})$ and then optimize the AF given by $\mathbf{E}[y_j|\mathbf{x}_j, \boldsymbol{\theta}]$, with $j \notin \mathcal{I}$. When the model is a Bayesian neural network trained with PBP, the corresponding operations are sampling $\mathcal{W}$ from q and then optimizing the AF given by $f(\mathbf{x}_j; \mathcal{W})$, with $j \notin \mathcal{I}$. This last step requires the use of a deterministic neural network, with weight values given by the posterior sample from q, to make predictions on all the molecules that have not been evaluated yet. Then, the molecule with highest predictive value is selected for the next evaluation.

6. Experiments with GPs and Parallel EI

We first compare the performance of our parallel and distributed Thompson sampling (PDTS) algorithm with the most popular approach for parallel BO: the parallel EI method from Section 2.1. Existing implementations of parallel EI such as spearmint[1] use a Gaussian process (GP) model for the objective function. To compare with these methods, we also adopt a GP as the model in PDTS. Note that parallel EI cannot scale to the large batch sizes used in high-throughput screening. Therefore, we consider here only parallel optimization problems with small batch sizes and synthetic objective functions. Besides PDTS and parallel EI, we also analyze the performance of the sequential versions of these algorithms: TS and EI.

To implement Thompson sampling (TS) with a GP model, we approximate the non-parametric GP with a parametric approximation based on random features, as described in the supplementary material of (Hernández-Lobato et al., 2014). For the experiments, we consider a cluster with 11 nodes: one central node for controlling the BO process and 10 additional nodes for parallel evaluations. We assume that all objective evaluations take a very large amount of time and that the cost of training the GPs and recomputing and optimizing the AF is negligible in comparison. Thus, in practice, we perform these experiments in a sequential (non-parallel) fashion with the GP model being updated only in blocks of 10 consecutive data points at a time.

As objective functions we consider the two dimensional Bohachevsky and Branin-Hoo functions and the six dimensional Hartmann function, all available in Benchfunk[2]. We also consider the optimization of functions sampled from the GP prior over the 2D unit square using a squared exponential covariance function with fixed 0.1 length scale. After each objective evaluation, we compute the immediate regret (IR), which we define as the difference between the best objective value obtained so far and the minimum value of the objective function. The measurement noise is zero in these experiments.

Figure 3 reports mean and standard errors for the logarithm of the best IR seen so far, averaged across 50 repetitions of the experiments. In the plots, the horizontal axis shows the number of function evaluations performed so far. Note that in these experiments TS and EI update their GPs once per sample, while PDTS and parallel EI update only every 10 samples. Figure 3 shows that EI is better than TS in most cases, although the differences between these two methods are small in the Branin-Hoo function. However, EI is considerably much better than TS in Hartmann. The reason for this is that in Hartmann there are multiple equivalent global minima and TS tends to explore all of them. EI is by contrast more exploitative and focuses on evaluating the objective around only one of the minima. The differences between parallel EI and PDTS are much smaller, with both obtaining very similar results. The exception is again Hartmann, where parallel EI is much better than PDTS, probably because PDTS is more explorative than parallel EI. Interestingly, PDTS performs better than parallel EI on the random samples from the GP prior, although parallel EI eventually catches up.

These results indicate that PDTS performs in practice very similarly to parallel EI, one of the most popular methods for parallel BO.

7. Experiments with Molecule Data Sets

We describe the molecule data sets used in our experiments. The input features for all molecules are 512-bit Morgan circular fingerprints (Rogers & Hahn, 2010), calculated with a bond radius of 2, and derived from the canonical SMILES as implemented in the RDkit package (Landrum).

[1] https://github.com/HIPS/Spearmint

[2] https://github.com/mwhoffman/benchfunk

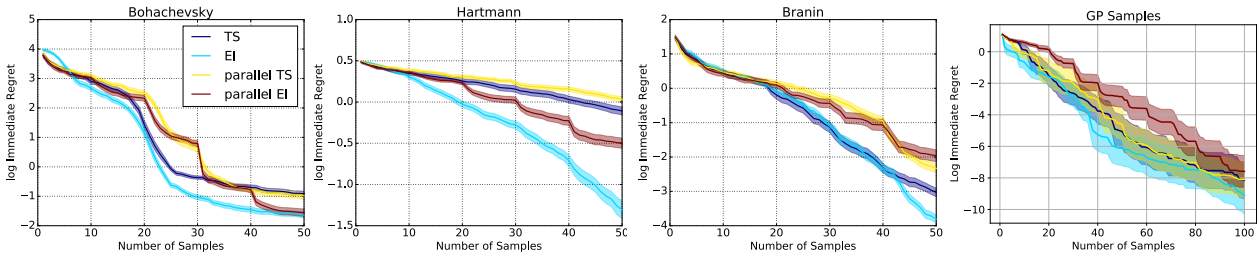

Figure 3. Immediate regret in experiments with GPs, using TS, EI, PDTS and parallel EI for optimizing synthetic functions (first 3 plots) and functions sampled from a GP prior (fourth plot).

Harvard Clean Energy Project: The Clean Energy Project is the world's largest materials high-throughput virtual screening effort (Hachmann et al., 2014; 2011), and has scanned more than 3.5 million molecules to find those with high power conversion efficiency (PCE) using quantum-chemical techniques, taking over 30,000 years of CPU time. The target value within this data set is the power conversion efficiency (PCE), which is calculated for the 2.3 million publicly released molecules, using the Scharber model (Dennler et al., 2008) and frontier orbitals calculated at the BP86 (Perdew, 1986; Becke, 1993) def2-SVP (Weigend & Ahlrichs, 2005) level of theory.

Dose-Response Data Set: These data sets were obtained from the NCI-cancer database (Many authors). The dose-response target value has a potential range of -100 to 100, and reports a percentage cell growth relative to a no-drug control. Thus, a value of +40 would correspond to a 60% growth inhibition and a value of -40 would correspond to 40% lethality. Molecules with a positive value for the dose-response are known as inhibitors, molecules with a score less than 0 have a cytotoxic effect. Results against the NCI-H23 cell line were taken against a constant log-concentration of -8.00M and where multiple identical conditions were present in the data an average was used for the target variables. In this data set we are interested in finding molecules with smaller values of the target variable.

Malaria Data Set: The Malaria data set was taken from the *P. falciparum* whole cell screening derived by combining the GSK TCAMS data set, the Novatis-GNF Malaria Box data set and the St Jude's Research Hospital data set, as released through the Medicines for Malaria Venture website (Spangenberg et al., 2013). The target variable is the EC50 value, which is defined as the concentration of the drug which gives half maximal response. Much like the Dose response data set, the focus here is on minimization: the lower the concentration, the stronger the drug.

7.1. Results

We evaluate the gains produced by PDTS in experiments simulating a high throughput virtual screening setting. In these experiments, we sequentially sample molecules from libraries of candidate molecules given by the data sets from Section 7. After each sampling step, we calculate the 1% recall, that is, the fraction of the top 1% of molecules from the original library that are found among the sampled ones. For the CEP data, we compute recall by focusing on molecules with PCE larger than 10%. In all data sets, each sampling step involves selecting a batch of molecules among those that have not been sampled so far. In the Malaria and One-dose data sets we use batches of size 200. These data sets each contain about 20,000 molecules. By contrast, the CEP data set contains 2 million molecules. In this latter case, we use batches of size 500. We use Bayesian neural networks with one hidden layer and 100 hidden units.

We compare the performance of PDTS with two baselines. The first one, *greedy*, is a sampling strategy that only considers exploitation and does not perform any exploration. We implement this approach by selecting molecules according to the average of the probabilistic predictions generated by PBP. That is, the greedy approach ignores any variance in the predictions of the Bayesian neural network and generates batches by just ranking molecules according to the mean of the predictive distribution given by PBP. The second baseline is a Monte Carlo approach in which the batches of molecules are selected uniformly at random. These two baselines are comparable to PDTS in that they can be easily implemented in a large scale setting in which the library of candidate molecules contains millions of elements and data is sampled using large batch sizes.

In the Malaria and One-dose data sets, we average across 50 different realizations of the experiments. This is not possible in the CEP data set, which is 100 times larger than the two other data sets. In the CEP case, we report results for a single realization of the experiment (in a second realization we obtained similar results). Figure 4 shows the recall obtained by each method in the molecule data sets. PDTS significantly outperforms the Monte Carlo approach, and also offers better performance than greedy sampling. This shows the importance of building in exploration into the sampling strategy, rather than relying on purely exploitative methods. The greedy approach performs best in the

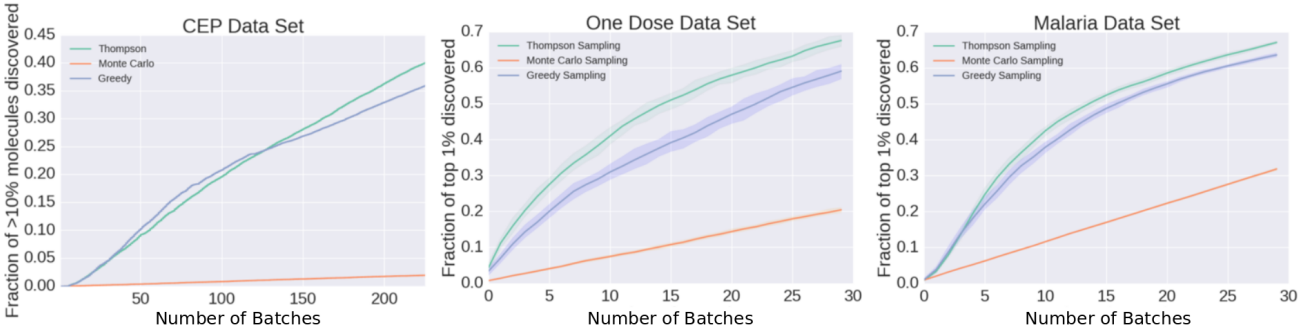

Figure 4. Recall obtained by PDTS on each data set. For the CEP data, the recall for molecules with a PCE $> 10\%$ is reported, whilst for One-dose and Malaria we report the recall for the molecules in the top 1%. In addition to the Monte Carlo sampling baseline, we also include results for a greedy sampling approach, in which there is no exploration, and the molecules are chosen according to the mean of the predictive distribution given by PBP. The overall lower performance of this greedy strategy illustrates the importance of exploration in this type of problems.

CEP data set. In this case, the greedy strategy initially finds better molecules than PDTS, but after a while PDTS overtakes, probably because a promising area of chemical space initially discovered by the greedy approach starts to become exhausted.

The previous results allow us to consider the savings produced by BO. In the CEP data set, PDTS achieves about 20 times higher recall values than the Monte Carlo approach, which is comparable to the exhaustive enumeration that was used to collect the CEP data. We estimate that, with BO, the CEP virtual screening process would have taken 1,500 CPU years instead of the 30,000 that were actually used. Regarding the One-dose and Malaria data sets, PDTS can locate in both sets about 70% of the top 1% molecules by sampling approximately 6,000 molecules. By contrast, the Monte Carlo approach would require sampling 14,000 molecules. This represents a significant reduction in the discovery time for new therapeutic molecules and savings in the economic costs associated with molecule synthesis and testing.

7.2. Comparison with ϵ-greedy Approaches

We can easily modify the greedy baseline from the previous section to include some amount of exploration by replacing a small fraction of the molecules in each batch with molecules chosen uniformly at random. This approach is often called ϵ-greedy (Watkins, 1989), where the variable ϵ indicates the fraction of molecules that are sampled uniformly at random. The disadvantage of the ϵ-greedy approach is that it requires the tuning of ϵ to the problem of interest whereas the amount of exploration is automatically set by PDTS.

We compared PDTS with different versions of ϵ-greedy in the same way as above, using $\epsilon = 0.01, 0.025, 0.05$ and 0.075. The experiments with the One-dose and the Malaria data sets are similar to the ones done before. However, we now sub-sample the CEP data set to be able to average across 50 different realizations of the experiment: we choose 4,000 molecules uniformly at random and then collect data in batches of size 50 across 50 different repetitions of the screening process. We compute the average rank obtained by each method across the $3 \times 50 = 150$ simulated screening experiments. A ranking equal to 1 indicates that the method always obtains the highest recall at the end of the experiment, while a ranking equal to 5 indicates that the method always obtains the worst recall value. Table 1 shows that the lowest average rank is obtained by PDTS, which achieves better exploration-exploitation trade-offs than the ϵ-greedy approaches.

Table 1. Average rank and standard errors by each method.

Method	Rank
$\epsilon = 0.01$	3.42 ± 0.28
$\epsilon = 0.025$	3.02 ± 0.25
$\epsilon = 0.05$	2.86 ± 0.23
$\epsilon = 0.075$	3.20 ± 0.26
PDTS	$\mathbf{2.51 \pm 0.20}$

8. Conclusions

We have presented a Parallel and Distributed implementation of Thompson Sampling (PDTS), a highly scalable method for parallel Bayesian optimization. PDTS can be applied when scalability limits the applicability of competing approaches. We have evaluated the performance of PDTS in experiments with both Gaussian process and probabilistic neural networks. We show that PDTS compares favorably with parallel EI in problems with small batch sizes. We also demonstrate the effectiveness of PDTS on large scale real world applications that involve searching chemical space for new molecules wit improved properties. We show that PDTS outperforms other scalable approaches on these applications, in particular, a greedy search strategy, ϵ-greedy approaches and a random search method.

Acknowledgements

J.M.H.L. acknowledges support from the Rafael del Pino Foundation. The authors thank Ryan P. Adams for useful discussions. A.A.-G. and E.O.P.-K. acknowledge the Department of Energy Program on Theory and modeling through grant DE-SC0008733.

References

Azimi, Javad, Fern, Alan, and Fern, Xiaoli Z. Batch Bayesian optimization via simulation matching. In *NIPS*, pp. 109–117, 2010.

Becke, Axel D. Density-functional thermochemistry. III. The role of exact exchange. *The Journal of Chemical Physics*, 98(7): 5648, 1993.

Blundell, Charles, Cornebise, Julien, Kavukcuoglu, Koray, and Wierstra, Daan. Weight uncertainty in neural networks. In *ICML*, pp. 1613–1622, 2015.

Bousquet, Olivier and Bottou, Léon. The tradeoffs of large scale learning. In *NIPS*, pp. 161–168, 2008.

Chapelle, Olivier and Li, Lihong. An empirical evaluation of Thompson sampling. In *NIPS*, pp. 2249–2257. 2011.

Chevalier, Clément and Ginsbourger, David. Fast computation of the multi-points expected improvement with applications in batch selection. In *International Conference on Learning and Intelligent Optimization*, pp. 59–69. Springer, 2013.

Contal, Emile, Buffoni, David, Robicquet, Alexandre, and Vayatis, Nicolas. Parallel Gaussian process optimization with upper confidence bound and pure exploration. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 225–240. Springer, 2013.

Dennler, G., Scharber, M. C., Ameri, T., Denk, P., Forberich, K., Waldauf, C., and Brabec, C. J. Design rules for donors in bulk-heterojunction tandem solar cells? towards 15% energy-conversion efficiency. *Adv. Mater.*, 20(3):579–583, feb 2008.

Desautels, Thomas, Krause, Andreas, and Burdick, Joel W. Parallelizing exploration-exploitation tradeoffs in Gaussian process bandit optimization. *Journal of Machine Learning Research*, 15(1):3873–3923, 2014.

Gal, Yarin and Ghahramani, Zoubin. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, pp. 1050–1059, 2016.

Ginsbourger, David, Le Riche, Rodolphe, and Carraro, Laurent. Kriging is well-suited to parallelize optimization. In *Computational Intelligence in Expensive Optimization Problems*, pp. 131–162. Springer, 2010.

Ginsbourger, David, Janusevskis, Janis, and Le Riche, Rodolphe. Dealing with asynchronicity in parallel Gaussian process based global optimization. In *4th International Conference of the ERCIM WG on computing & statistics (ERCIM'11)*, 2011.

Gómez-Bombarelli, Rafael, Aguilera-Iparraguirre, Jorge, Hirzel, Timothy D., Duvenaud, David, Maclaurin, Dougal, Blood-Forsythe, Martin A., Chae, Hyun Sik, Einzinger, Markus, Ha, Dong-Gwang, Wu, Tony, Markopoulos, Georgios, Jeon, Soonok, Kang, Hosuk, Miyazaki, Hiroshi, Numata, Masaki, Kim, Sunghan, Huang, Wenliang, Hong, Seong Ik, Baldo, Marc, Adams, Ryan P., and Aspuru-Guzik, Alán. Design of efficient molecular organic light-emitting diodes by a high-throughput virtual screening and experimental approach. *Nature Materials*, aug 2016.

Gonzlez, J., Dai, Z., Hennig, P., and Lawrence, N. Batch Bayesian optimization via local penalization. In *AISTATS*, pp. 648–657, 2016.

Hachmann, Johannes, Olivares-Amaya, Roberto, Atahan-Evrenk, Sule, Amador-Bedolla, Carlos, Sanchez-Carrera, Roel S., Gold-Parker, Aryeh, Vogt, Leslie, Brockway, Anna M., and Aspuru-Guzik, Alan. The Harvard Clean Energy Project: Large-Scale Computational Screening and Design of Organic Photovoltaics on the World Community Grid. *J. Phys. Chem. Lett.*, 2(17):2241–2251, sep 2011.

Hachmann, Johannes, Olivares-Amaya, Roberto, Jinich, Adrian, Appleton, Anthony L., Blood-Forsythe, Martin A., Seress, László R., Román-Salgado, Carolina, Trepte, Kai, Atahan-Evrenk, Sule, Er, Sleyman, Shrestha, Supriya, Mondal, Rajib, Sokolov, Anatoliy, Bao, Zhenan, and Aspuru-Guzik, Alán. Lead candidates for high-performance organic photovoltaics from high-throughput quantum chemistry – the Harvard Clean Energy Project. *Energy Environ. Sci.*, 7(2):698–704, 2014.

Hernández-Lobato, José Miguel and Adams, Ryan P. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *ICML*, pp. 1861–1869, 2015.

Hernández-Lobato, José Miguel, Hoffman, Matthew W, and Ghahramani, Zoubin. Predictive entropy search for efficient global optimization of black-box functions. In *NIPS*, pp. 918–926, 2014.

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

Landrum, Greg. RDKit: Open-source cheminformatics.

Ma, Junshui, Sheridan, Robert P., Liaw, Andy, Dahl, George E., and Svetnik, Vladimir. Deep neural nets as a method for quantitative structure–activity relationships. *Journal of Chemical Information and Modeling*, 55(2):263–274, feb 2015.

Many authors. NCI Database Download Page. URL http://cactus.nci.nih.gov/download/nci/.

Marmin, Sébastien, Chevalier, Clément, and Ginsbourger, David. Differentiating the multipoint expected improvement for optimal batch design. In *International Workshop on Machine Learning, Optimization and Big Data*, pp. 37–48. Springer, 2015.

Mayr, Andreas, Klambauer, Gnter, Unterthiner, Thomas, and Hochreiter, Sepp. Deeptox: Toxicity prediction using deep learning. *Front. Environ. Sci.*, 3, feb 2016.

Minka, Thomas P. *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology, 2001.

Opper, Manfred. A Bayesian approach to on-line learning. In Saad, David (ed.), *On-Line Learning in Neural Networks*, pp. 363–378. Cambridge University Press (CUP), 1998.

Perdew, John P. Density-functional approximation for the correlation energy of the inhomogeneous electron gas. *Phys. Rev. B*, 33(12):8822–8824, jun 1986.

Pyzer-Knapp, Edward O., Suh, Changwon, Gómez-Bombarelli, Rafael, Aguilera-Iparraguirre, Jorge, and Aspuru-Guzik, Alán. What is high-throughput virtual screening? A perspective from organic materials discovery. *Annu. Rev. Mater. Res.*, 45(1): 195–216, jul 2015.

Rajan, Krishna. Combinatorial Materials Sciences: Experimental Strategies for Accelerated Knowledge Discovery. *Annu. Rev. Mater. Res.*, 38(1):299–322, aug 2008.

Ramsundar, Bharath, Kearnes, Steven, Riley, Patrick, Webster, Dale, Konerding, David, and Pande, Vijay. Massively multitask networks for drug discovery. *arXiv preprint arXiv:1502.02072*, 2015.

Reymond, Jean-Louis, Ruddigkeit, Lars, Blum, Lorenz, and van Deursen, Ruud. The enumeration of chemical space. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 2(5):717–733, apr 2012.

Rogers, David and Hahn, Mathew. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, 50(5): 742–754, may 2010.

Scharber, M.C., Mhlbacher, D., Koppe, M., Denk, P., Waldauf, C., Heeger, A.J., and Brabec, C.J. Design rules for donors in bulk-heterojunction solar cells–towards 10% energy-conversion efficiency. *Advanced Materials*, 18(6):789–794, 2006.

Shah, Amar and Ghahramani, Zoubin. Parallel predictive entropy search for batch global optimization of expensive objective functions. In *NIPS*, pp. 3330–3338, 2015.

Shahriari, Bobak, Wang, Ziyu, Hoffman, Matthew W, Bouchard-Côté, Alexandre, and de Freitas, Nando. An entropy search portfolio for Bayesian optimization. *arXiv preprint arXiv:1406.4625*, 2014.

Snoek, Jasper, Larochelle, Hugo, and Adams, Ryan P. Practical Bayesian optimization of machine learning algorithms. In *NIPS*, pp. 2951–2959, 2012.

Snoek, Jasper, Rippel, Oren, Swersky, Kevin, Kiros, Ryan, Satish, Nadathur, Sundaram, Narayanan, Patwary, Md. Mostofa Ali, Prabhat, and Adams, Ryan P. Scalable Bayesian optimization using deep neural networks. In *ICML*, pp. 2171–2180, 2015.

Spangenberg, Thomas, Burrows, Jeremy N., Kowalczyk, Paul, McDonald, Simon, Wells, Timothy N. C., and Willis, Paul. The Open Access Malaria Box: A Drug Discovery Catalyst for Neglected Diseases. *PLoS ONE*, 8(6):e62906, jun 2013.

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

Wang, Jialei, Clark, Scott C, Liu, Eric, and Frazier, Peter I. Parallel Bayesian global optimization of expensive functions. *arXiv preprint arXiv:1602.05149*, 2016.

Watkins, Christopher John Cornish Hellaby. *Learning from delayed rewards*. PhD thesis, University of Cambridge England, 1989.

Weigend, Florian and Ahlrichs, Reinhart. Balanced basis sets of split valence triple zeta valence and quadruple zeta valence quality for H to Rn: Design and assessment of accuracy. *Phys. Chem. Chem. Phys.*, 7(18):3297, 2005.

Wu, Jian and Frazier, Peter. The parallel knowledge gradient method for batch Bayesian optimization. In *NIPS*, pp. 3126–3134, 2016.