Fully Distributed Bayesian Optimization with Stochastic Policies

Javier Garcia-Barcos[1] and **Ruben Martinez-Cantin**[1,2]
[1]Instituto de Investigacion en Ingenieria de Aragon, University of Zaragoza
[2]Centro Universitario de la Defensa, Zaragoza
jgbarcos, rmcantin@unizar.es

Abstract

Bayesian optimization has become a popular method for high-throughput computing, like the design of computer experiments or hyperparameter tuning of expensive models, where sample efficiency is mandatory. In these applications, distributed and scalable architectures are a necessity. However, Bayesian optimization is mostly sequential. Even parallel variants require certain computations between samples, limiting the parallelization bandwidth. Thompson sampling has been previously applied for distributed Bayesian optimization. But, when compared with other acquisition functions in the sequential setting, Thompson sampling is known to perform suboptimally. In this paper, we present a new method for fully distributed Bayesian optimization, which can be combined with any acquisition function. Our approach considers Bayesian optimization as a partially observable Markov decision process. In this context, stochastic policies, such as the Boltzmann policy, have some interesting properties which can also be studied for Bayesian optimization. Furthermore, the Boltzmann policy trivially allows a distributed Bayesian optimization implementation with high level of parallelism and scalability. We present results in several benchmarks and applications that show the performance of our method.

1 Introduction

Many engineering problems and scientific phenomena are being studied in high-throughput computing facilities, through complex computer models or simulators. Due to the large amount of resources that are needed, experiments should be carefully selected and studied. This is known as the design and analysis of computer experiments [Sacks *et al.*, 1989]. Bayesian optimization (BO) can be used for designing computer experiments which require the search of an optimum value [Jones *et al.*, 1998; Mockus *et al.*, 1978]. BO carefully selects the next experiments to perform in order to find the optimum value as efficiently as possible. For example, consider the problem of finding the optimal shape of a wing profile to reduce the drag force. Instead of dealing with the infamous equations of Navier-Stokes, we can experiment with a Computational Fluid Dynamics (CFD) simulator and check the outcome [Forrester *et al.*, 2006; Martinez-Cantin, 2019]. Thus, for the optimization algorithm, the simulator becomes a black-box where we only care about the resulting drag force. For scientific experiments, we may want to adjust the parameters of a computational model of cell migration to mimic the behaviour of *in vivo* or *in vitro* experiments [Merino-Casallo and others, 2018], or we may want to find new drugs through virtual screening [Hernandez-Lobato *et al.*, 2017]. Finally, one of the most popular applications for BO is the tuning of hyperparameters of complex machine learning models, such as deep neural networks [Snoek *et al.*, 2012; Klein *et al.*, 2017].

BO achieves sample efficiency by learning a probabilistic surrogate model of the the target function. Then, using several heuristics, called acquisition functions, we can select the next point to be evaluated or experimented. Therefore, BO is intrinsically a sequential process. Each new observation is incorporated into the surrogate model, which at the same time, modifies the acquisition function for the selection of future experiments. There are certain algorithms that allow parallel or batch queries of experiments [Snoek *et al.*, 2012; González *et al.*, 2016a; Desautels *et al.*, 2014], but they still require the queries to be computed in a sequential manner. In high-throughput systems, these methods require a central node that computes and dispatches the queries. Furthermore, as pointed out by [Kandasamy *et al.*, 2018], many of those methods do not allow for asynchronous execution. To the authors knowledge, only the Thompson sampling approach from [Hernandez-Lobato *et al.*, 2017] can be fully distributed. However, it is well-known that there are many other acquisition functions that performs better than Thompson sampling in practice [Shahriari *et al.*, 2016]. In this work, we introduce a new method to allow a fully distributed BO, which can be combined with any acquisition function.

However, the contribution of our paper is twofold. Before introducing our distributed BO algorithm, we present a new portrayal of BO in the Markov decision process framework, following the analysis of [Toussaint, 2014]. We show how this new framework allows a greater understanding of the features and capabilities of BO. Then, we introduce the idea of stochastic policies for BO, which is the ingredient necessary to perform fully distributed BO in combination with any acquisition function. Furthermore, we analyze the advantages of the stochastic policies beyond parallelization, from a theoretical and practical point of view.

2 The Optimization Agent

We consider the optimization algorithm as an agent that is interacting with the environment, which is the space of functions. The current state of the environment is the target function f. The agent does not have full observability of the state and can only perform partial observations by querying the function $y_t = f(\mathbf{x}_t)$. As pointed out in [Toussaint, 2014], the future decisions made by the agent can be modelled as a *partially observable Markov decision process* (POMDP). A similar analogy, although from a control theory perspective is also presented in [Lam *et al.*, 2016].

2.1 POMDPs

A (PO)MDP is a stochastic model where the agent and the environment are fully represented by state variables $s_t \in \mathcal{S}$ and the agent can change that state by performing actions $a_t \in \mathcal{A}$ in a Markovian way $p(s_{t+1}|a_t, s_t)$. The (PO)MDP model also assumes that the agent is rational, acting to maximize the future expected reward $\mathbb{E}[\sum_{t=0}^{N} R(s_t, a_t)]$. The behavior of the agent is encoded in the policy which maps states to actions $a_t = \pi(s_t)$. In order to rank the possible actions at one state, we can compute the Q-function which represents the *quality* of taking a certain action considering the future reward. The optimal Q-function Q^* can be computed by doing *full backups* of future rewards and actions recursively. Then, we can obtain the optimal greedy function by:

$$\pi^*(s_t) = \arg\max_a Q^*(s_t, a_t) \tag{1}$$

In the POMDP setting, the agent does not have full observability of the state and must rely on partial observations y_t following an observation model $p(y_t|s_t, a_t)$. Thus, the agent relies on *beliefs*, which are the distributions over possible states, given the known observations and actions $b_t = p(s_t|a_{0:t}, y_{0:t})$. Because, the belief is a sufficient statistic, it can be shown that a POMDP on state space, is equivalent to a MDP on belief space [Kaelbling *et al.*, 1996]. In this case, the transition model becomes $p(b_{t+1}|a_t, b_t) = \int_y p(b_{t+1}|a_t, b_t, y_t)p(y_t|a_t, b_t)dy$. Then, the reward and the policy become functions of the belief $r(b_t, a_t) = \int_s b_t(s)R(s_t, a_t)ds$ and $\pi(b_t) = \int_s b_t(s)\pi(s_t)ds$. In this case, the optimal policy and Q-function can also be mapped to the belief space $\pi^*(b_t) = \arg\max_a Q^*(b_t, a_t)$.

2.2 Bayesian Optimization

Bayesian optimization is a set of optimization methods [Shahriari *et al.*, 2016] with two important distinct features: a *probabilistic surrogate model* $p(f)$ to learn the properties and features of the target function that we are trying to optimize and, an *acquisition function* $\alpha(x, p(f))$ that, based on the surrogate model, rate the potential interest of subsequent queries.

More formally, BO tries to optimize a function $f : \mathcal{X} \to \mathbb{R}$ over some domain $\mathcal{X} \subset \mathbb{R}^d$, by carefully selecting the queries

POMDP / belief MDP	Bayesian optimization
State: s_t	Target function: f
Action: a_t	Next query: x_{t+1}
Observation: y_t	Response value: $y_t = f(x_t)$
Belief: $b_t = p(s_t)$	Surrogate model: $p(f)$
Q-function: $Q^*(b_t, a_t)$	Acquisition function: $\alpha(x, p(f))$
Reward: $R(s_t, a_t)$	Improvement: $\max(0, y_{t+1} - \rho_t)$

Table 1: Comparison of POMDP and belief MPD terms with respect to the corresponding elements in BO

of the function to reduce the number of evaluations of f before finding the optimum x^*. At iteration t, all previously observed values $\mathbf{y} = y_{1:t}$ at queried points $\mathbf{X} = \mathbf{x}_{1:t}$ are used to construct a probabilistic surrogate model $p(f|y_{1:t}, \mathbf{x}_{1:t})$. Typically, the next query location $\mathbf{x}_{t+1}$ is determined by greedily optimizing the acquisition function in $\mathcal{X}$:

$$\mathbf{x}_{t+1} = \arg\max_{\mathbf{x}\in\mathcal{X}} \alpha\left(\mathbf{x}, p(f \mid y_{1:t}, \mathbf{x}_{1:t})\right) \tag{2}$$

For example, we can use the expected improvement (EI) as the acquisition function [Mockus *et al.*, 1978]:

$$EI_t(\mathbf{x}) = \mathbb{E}_{p(y_{t+1} \mid y_{1:t}, \mathbf{x}_{1:t})}\left[\max(0, y_{t+1} - \rho_t)\right], \tag{3}$$

where $\rho_t = \max(y_1, \ldots, y_t)$ is the incumbent optimum at that iteration. EI is still one of the most popular choices, although there are multiple alternatives depending on the criteria selected, such as *optimism in the face of uncertainty* [Srinivas *et al.*, 2010], information about the optimum [Hennig and Schuler, 2012; Hernandez-Lobato *et al.*, 2014; Wang and Jegelka, 2017], etc.

It has been found that EI might be unstable in the first iterations due to the lack of information [Jones *et al.*, 1998; Bull, 2011]. Therefore, the optimization is initialized with p evaluations by sampling from low discrepancy sequences.

Surrogate Model

Most frequently, this takes the form of a Gaussian process (GP), although other alternatives have been presented, such as Bayesian neural networks [Hernandez-Lobato *et al.*, 2017]. For the remainder of the paper we consider a GP with zero mean and kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ as the surrogate model. The kernel is chosen to be the Matérn kernel with smoothness parameter ν and hyperparameters $\boldsymbol{\theta}$. The GP posterior model gives predictions at a query point $\mathbf{x}_q$ which are normally distributed $y_q \sim \mathcal{N}(\mu(\mathbf{x}_q), \sigma^2(\mathbf{x}_q))$, such that $\mu(\mathbf{x}_q) = \mathbf{k}(\mathbf{x}_q)^T\mathbf{K}^{-1}\mathbf{y}$, and $\sigma^2(\mathbf{x}_q) = k(\mathbf{x}_q, \mathbf{x}_q) - \mathbf{k}(\mathbf{x}_q)^T\mathbf{K}^{-1}\mathbf{k}(\mathbf{x}_q)$ where $\mathbf{k}(\mathbf{x}_q) = [k(\mathbf{x}_q, \mathbf{x}_i)]_{\mathbf{x}_i\in\mathbf{X}}$ and $\mathbf{K} = [\mathbf{k}(\mathbf{x}_i, \mathbf{x}_j)]_{\mathbf{x}_i,\mathbf{x}_j\in\mathbf{X}} + \mathbf{I}\sigma_n^2$.

2.3 BO as a POMDP

As can be seen, Equation (1) is analogous to Equation (2). The POMDP framework, as a rational Bayesian model, can also be applied to BO. Table 1 summarizes the connections between those frameworks. If we use EI as the acquisition function, then we assume that the improvement function $I = \max(0, y_{t+1} - \rho_t)$ is the reward. In this case, the EI is the optimal myopic policy for POMDP, as it maximizes the

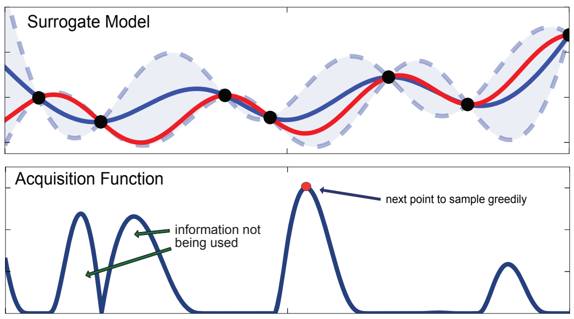

Figure 1: Example of a surrogate model (blue) on top of the target function (red). The greedy policy selects the maximum of the acquisition function (red dot), but completely ignores the rest of the regions which are almost as valuable.

expected reward one step ahead. More interestingly, entropy based acquisition functions can be interpreted as an active learning problem, for which the POMDP can also be applied as a framework [Lopes and Montesano, 2014].

3 Stochastic Policies for BO

BO policies are typically greedy in two ways. First, they are temporally greedy, that is, they look for the maximum immediate reward, although some lookahead alternatives have been proposed in the past [González *et al.*, 2016b; Lam *et al.*, 2016]. Second, they are spatially greedy, that is, they only select the action or next query that maximizes the acquisition function (or Q-function). However, one might want to explore suboptimal actions to gather knowledge about the world. In fact, Q-learning algorithms may not converge to the solution using a greedy policy due to the lack of infinite exploration.

Many acquisition functions, such as those mentioned in Section 2.2 are already designed to trade-off exploration and exploitation. However, this trade-off is based on the assumption that the surrogate model *is good enough* to predict both the expected function value and its uncertainty. In practice, even if the model is chosen carefully, we still need to learn the hyperparameters. Thus, for the first few iterations, the model is inaccurate. Interestingly enough, theoretical results rely on bounding the length-scales of the GP kernels to artificially increase the exploration, or directly using ϵ-greedy strategies to guarantee near-optimal convergence rates with unknown hyperparameters [Wang and de Freitas, 2014; Bull, 2011]. For inaccurate models, the acquisition function is still able to provide some information, but by greedily selecting a single value we are wasting some of that information. Figure 1 shows how the greedy policy only cares about the central mode of the acquisition function, while the two modes on the left are almost as interesting to be explored.

Instead, we propose to use a stochastic policy such as the following Boltzmann policy (also known as Gibbs or softmax policy):

$$p(\mathbf{x}_{t+1} \mid y_{1:t}, \mathbf{x}_{1:t}) = \frac{e^{\beta_t \alpha(x_{t+1}, p(f \mid y_{1:t}, \mathbf{x}_{1:t}))}}{\int_{x \in \mathcal{X}} e^{\beta_t \alpha(x, p(f \mid y_{1:t}, \mathbf{x}_{1:t}))} dx} \tag{4}$$

This policy defines a probability distribution for the next query or action. Thus, the actual next query is selected by sampling that distribution $\mathbf{x}_{t+1} \sim p(\mathbf{x}_{t+1} \mid y_{1:t}, \mathbf{x}_{1:t})$. This policy allows exploration even if the model is completely biased. Furthermore, it has some interesting properties for BO convergence as we will discuss in Section 3.1. The main result of this work is that the sampling process of $\mathbf{x}_{t+1}$ can be done **in parallel and fully distributed**, as will be discussed in Section 3.2. This approach can be applied to any acquisition function or surrogate model that can be found in the literature. Nevertheless, the theoretical analysis and posterior experimentation focuses on GP and EI as previously discussed in Section 2.2.

3.1 Theoretical Analysis

Convergence of BO algorithms has been extensively studied in terms of convergence rates [Bull, 2011] or regret bounds in a bandit setting [Srinivas *et al.*, 2010; Wang and de Freitas, 2014]. In this paper we are going to follow upon the analysis by [Bull, 2011] for the expected improvement. We show how the stochastic policy from equation 4 has the same rates as the greedy policy in the limiting case. We also show that the stochastic policy does not need to rely on the ϵ-greedy strategy for near-optimal rates.

Let $\mathcal{X} \subset \mathbb{R}^d$ be compact with non-empty interior. For a function $f : \mathcal{X} \to \mathbb{R}$, let $\mathbb{E}_f^u$ denote the expectation when minimizing the fixed function f with strategy u, as u can be random. We assume a prior π for $p(f)$, following a Gaussian process with a Matérn kernel with smoothness parameter ν and length-scales $\boldsymbol{\theta}$, having the Square exponential kernel as the limiting case of $\nu \to \infty$. Each kernel K_θ is associated with a space of functions $\mathcal{H}_\theta(X)$, its reproducing-kernel Hilbert space (RKHS).

Definition 1. *An $EI(\pi, \beta_t)$ strategy chooses:*

1. *initial design points $\mathbf{x}_1, \ldots, \mathbf{x}_k$ independently of f; and*
2. *further design points $\mathbf{x}_{t+1}$ $(t \geq k)$ sampled from* (4).

This is analogous to [Bull, 2011, Def. 1], but replacing the greedy selection by the stochastic selection. The $EI(\pi, \beta_t)$ strategy can also be adapted to consider estimated parameters [Bull, 2011, Def. 3]. Note that, for the stochastic policy, there is no need to enforce that the selection is dense for constant values of the acquisition function.

Theorem 2. *Let $\mathcal{X}$ be a finite space. If $\beta_t = \ln t / C_t$ and $C_t = \max_x |\max_z \alpha(z, p(f)) - \alpha(x, p(f))|$, then* (4) *is a greedy in the limit with infinite exploration (GLIE) policy. Therefore:*

1. *each point $\mathbf{x}$ is queried infinitely often if we use the $EI(\pi, \beta_t)$ strategy infinitely often, and*
2. *in the limit, the $EI(\pi, \beta_t)$ policy is greedy with respect to the acquisition function $\alpha(\cdot)$ with probability 1.*

The proof can be found in [Singh *et al.*, 2000, Ap. B] following the relations from Table 1. In order to generalize the previous result to $\mathcal{X} \in \mathbb{R}^d$, we can partition $\mathcal{X}$ in n regions of size $\mathcal{O}(t^{1/d})$. Following $EI(\pi, \beta_t)$ and assuming a large t each region will be sampled with high probability, thus, the mesh norm is small, which is the requirement for near-optimal rates [Bull, 2011, Lemma 12]. In fact, with β_1 the sampling distribution is exactly uniform.

Theorem 3. *Let $EI(\pi, \beta_t)$ be the strategy in Definition 1. If $\nu < \infty$, then for any $R > 0$,*

$$\sup_{\|f\|_{\mathcal{H}_\theta(X)} \leq R} \mathbb{E}_f^u[f(x_t^*) - \min f] = O((t/\log t)^{-\nu/d}(\log t)^\alpha),$$

with probability 1, while if $\nu = \infty$, the statement holds for all $\nu < \infty$.

The sketch of the proof is based on [Bull, 2011, Theorem 5], which uses greedy EI for optimality combined with ϵ-greedy to guarantee the reduced mesh norm. Following Theorem 2, for large t, the mesh norm will be small and the strategy will be greedy with probability 1. We encourage the reader to follow up the discussion in [Singh *et al.*, 2000] where it is shown that, for certain values of ϵ, the strategy followed by [Bull, 2011] is also GLIE.

3.2 Distributed BO

Now we are ready to present the main contribution of this work. The stochastic policy presented in Section 3 allow us to implement BO in a fully distributed setup which can be easily deployed and scaled. Contrary to most parallel or batch BO methods which require of a central node to keep track of the queries computed and deployed, our approach does not need a centralized node. All the computation can be done in each node of the distributed system. Furthermore, for optimal results, the nodes only need to broadcast their queries and observed values $\{\mathbf{x}_t, y_t\}$, requiring minimal communication bandwidth. In addition to that, communication can be asynchronous and be even robust to failures in the network, as the order of the queries and observations is irrelevant.

Algorithm 1 summarizes the code to be deployed in each node of the computing cluster or distributed system. The initialization part requires sampling from a low discrepancy sequence. This can be easily distributed by setting the precomputed sequence on a lookup table where each node accesses it based on their id. Once the initialization phase is done, each node builds its own surrogate model (e.g.: a GP) with all the data that is available to them. There is no requirement for the models to be synchronized or updated, although each node behaviour will be optimal if it has access to all the observations available as soon as possible.

One advantage of this setup is that it can be easily scaled by deploying new nodes using Algorithm 1, even halfway through the optimization process, as seen in Figure 2. In that case, the first operation is to collect all the data that has been broadcasted in the network instead of using the low discrepancy sequence. Another advantage is that we can play with

Algorithm 1 BO-NODE

```
Input: Budget $T$, low discrepancy sequence $LD$.
1: Query $LD$ for $p$ initialization points based on node id.
2: Broadcast $\mathbf{x}_{1:p}, y_{1:p}$
3: for $t = p \dots T$ do
4:     Collect $\mathbf{x}, y$ from other nodes when available.
5:     Update surrogate model $p(f|\mathbf{x}_{1:t}, y_{1:t})$
6:     Sample $\mathbf{x}_{t+1}$ with Equation (4).
7:     Broadcast $\mathbf{x}_{t+1}$ and $y_{t+1} = f(x_{t+1})$
8: end for
```

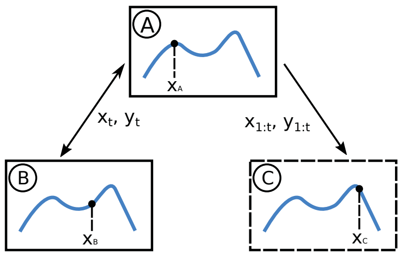

Figure 2: Visualization of the interaction between BO nodes. We have 3 nodes: A and B, which are already up and working; and C, a new node that we want to spin up mid-optimization. A and B only need to broadcast their new queries and observations. C needs to be given all the previous queries and observations up to the current instant t. Then it can resume its work in the same way as A and B.

the β_t parameter from equation (4) and deploy nodes more exploitative $\beta_t \to \infty$ or more exploratory $\beta_t \to 0$, or we can combine different kinds of node configurations, resulting in an overall behaviour analogous to *parallel tempering* [Neal, 1996].

Many parallel BO methods have been proposed in the past few years. The main idea of all those methods is to ensure that the parallel experiments are well-distributed among the search space, which is problematic when we use a greedy approach without adding new information between queries. Thus, some authors include artificially augmented data by hallucinated observations [Ginsbourger *et al.*, 2010; Snoek *et al.*, 2012] or by combining optimization with some degree of active learning in order maximize the knowledge about the target function [Desautels *et al.*, 2014; Contal *et al.*, 2013; Shah and Ghahramani, 2015] or by enforcing diversity through heuristics [González *et al.*, 2016a].

As noted on [Kandasamy *et al.*, 2018], the majority of parallel methods are synchronous. Recently, a Thompson Sampling approach [Hernandez-Lobato *et al.*, 2017; Kandasamy *et al.*, 2018] has been applied to achieve fully distributed BO. To the authors knowledge, this is the only method comparable to our proposed distributed BO method.

3.3 Sampling Strategies

Sampling from equation (4) is not trivial. Most acquisition functions are highly multimodal with many large areas of low probability between modes. This is known to be problematic for MCMC methods. Tempering methods, such as simulated or parallel tempering [Neal, 1996] usually perform better in these situations by allowing higher temperatures and, therefore, higher mixing of particles, in combination with lower temperatures to better represent the distribution. One advantage of our distribution is that it is already in Boltzmann form, allowing us to change the temperature by altering the β_t parameter. In our case, since we have a target temperature that we want to use for sequential convergence (as seen in Theorem 2) or for distributed exploration-exploitation (as seen in Section 3.2), we can select the temperature profile in such a way that the lower temperature matches the target. Alternatively, we can select another profile and use importance sampling on the target temperature. In practice, we found

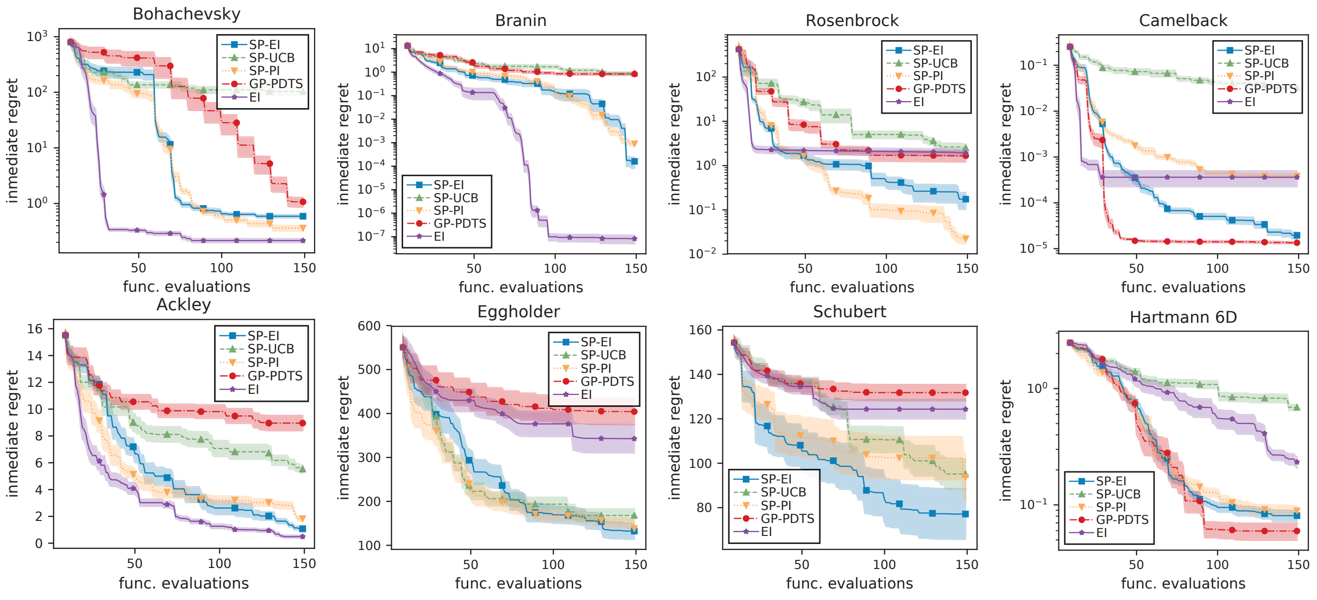

Figure 3: Inmediate regret on benchmark functions with the stochastic policy in combination with EI, UCB and PI. We include PDTS for comparison and sequential EI as baseline.

that, for small dimensional problems as those typically found for BO, a mixture of Gaussians with different variance as the proposal function and a fixed temperature using Metropolis-Hasting can be enough to get a good distribution.

4 Discussion

Although the main result of this work is on distributed BO, the POMDP framework can introduce new interesting ideas to explore in the context of BO. For example, multitasking or multifidelity systems have a great potential for high-throughput computing with BO, where we are able to improve the convergence of our method by introducing new information from other sources, either from related problems where we already have previous experiments or from less expensive sources like a smaller training datasets, incomplete executions or a simpler simulator. In both scenarios the target function f now belongs to a family of functions from other tasks or fidelities f_k. In the bandit setup, this is equivalent to contextual bandits. In the POMDP formulation, the function f is analogous to the state s (see Table 1). The POMDP model includes a transition function $p(s_{t+1} \mid a_t, s_t)$, which can be known a priori or learned, that defines a probability distribution of transitions between states. This transition function allows more flexibility than contextual variables.

Recent results on lookahead policies try to avoid the temporal greediness of BO. One interesting approach is based on dynamic programming [Lam *et al.*, 2016]. However, as pointed out by the authors, the dynamic programming approach is challenging due to the nested maximizations and expectations, requiring heuristics to relax the maximization steps. By using the stochastic policy from equation (4), we can provide a full Bayesian treatment of the dynamic programming, similar to value iteration [Kaelbling *et al.*, 1996].

5 Results

We show the performance of our stochastic policy for distributed BO with different acquisition functions: Expected Improvement (SP-EI), Probability of Improvement (SP-PI) and Upper Confidence Bounds (SP-UCB). We also include the parallel and distributed Thompson sampling (PDTS) [Hernandez-Lobato *et al.*, 2017] as an alternative distributed method and the sequential expected improvement (EI) as a baseline. Note that [Hernandez-Lobato *et al.*, 2017] already compares PDTS with parallel EI and ϵ-greedy methods. In order to simplify the comparison, we use a GP as the surrogate model for all the algorithms. However, both PDTS and our stochastic policy methods allow other surrogate models such as Bayesian neural networks. In all the experiments, we assume a network of 10 nodes synchronized, that is, function evaluations are performed in batches of 10 for all distributed methods. Note that EI has an unfair advantage, as it has access to all the data for each iteration while the distributed methods only update their GP model once every 10 observations. For all the plots, we display the average of each method over 10 trials with a 95% confidence interval. We use common random numbers to reduce the variance in the comparison and use the same initial samples among all methods.

5.1 Benchmark Functions

First, we start with a set of test problems for global optimization[1]. We have selected the functions to have a mixture of behaviours (smooth/sharp, single/multiple minima, etc.). The results on these functions are showcased in Figure 3. First, we can see how the performance of SP-EI is fairly consistent among the different functions, achieving better or similar results than the alternatives. Even for functions that are difficult

[1]From: https://www.sfu.ca/~ssurjano/optimization.html

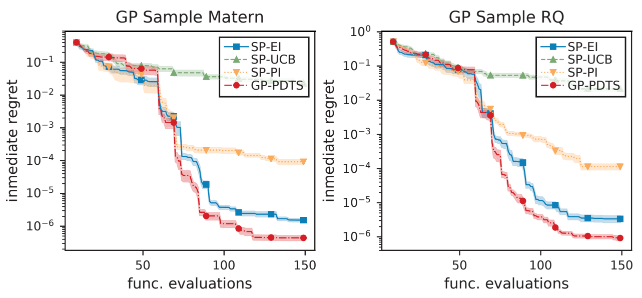

Figure 4: Optimizing random functions from a GP. A Matérn kernel (left) and rational quadratic kernel (right) are used in the GP. As a Matérn kernel is used in optimization, we have a *within-model* (left) and *out-of-model* (right) experiment. It is important to note that different kernels can generate different possible random functions, making the vertical axis not comparable between both problems.

to optimize with BO, such as Schubert, the stochastic policy is able to perform well thanks to the extra exploration induced by the sampling process. However, this exploration does not interfere with *easier* functions where exploitation is more important, such as Branin or Bohachevsky. Interestingly, SP-PI performs reasonably well. In the case of PI, the stochastic policy prevents excessive exploitation, a known behaviour of PI in the sequential BO setting [Shahriari *et al.*, 2016].

We have also followed the methodology from [Hennig and Schuler, 2012] and generated random functions from a known GP. We have studied two situations: a) we have studied the *within-model* problem, where the GP sample uses the same kernel as the optimization algorithm (Matérn with $\nu = 5/2$) and, b) the *out-of-model* problem where the GP sample is generated with a different kernel (a Rational Quadratic). The results in Figure 4 show how SP-EI is comparable to PDTS both for the *within-model* and the *out-of-model* problems.

5.2 Robot Pushing

In the next experiment, we use the *active learning for robot pushing* setup and code from [Wang and Jegelka, 2017]. It consists of performing active policy search on the task of selecting a pushing action of an object towards a designated goal location. The function has a 3-dimensional input: the robot location (r_x, r_y) and the pushing duration t_r. In a second experiment, we add the robot angle r_θ to have a 4-dimensional version. In this experiment, the repetitions are increased to 40, as each repetition is a different goal location. Figure 5 shows the results of both problems in which we can see how our methods SP-EI and SP-PI have faster convergence than GP-PDTS.

5.3 Hyperparameter Tuning of Neural Networks

Finally, we use the set of problems for hyperparameter tuning of neural networks from [Martinez-Cantin *et al.*, 2018], as these are good examples of the advantage of distributed BO for training expensive models in the cloud. We can see the results of both networks in Figure 6, where SP-EI and SP-PI consistently outperform PDTS.

Variational Autoencoder (VAE) on MNIST. A VAE is a generative method that learns a low dimensional representation of high dimensional data, such as images. We train a VAE for the MNIST dataset, and tune the following hyperparameters: number of nodes in the hidden layer, learning rate, learning rate decay and ϵ constant for the ADAM optimizer.

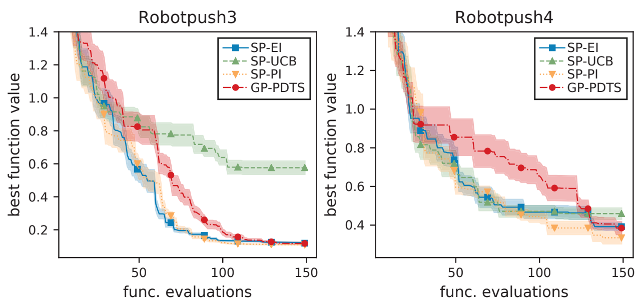

Figure 5: Robot pushing policy results, showing the 3-dimensional (left) and the 4-dimensional (right) problems.

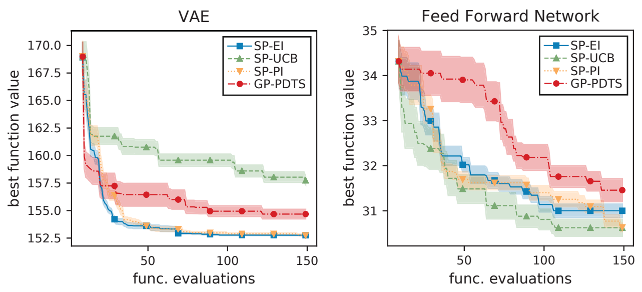

Figure 6: Hyperparameter tuning of neural network problems. A Variational Autoencoder trained with the MNIST dataset (left) and a Feedforward network trained with the Boston housing dataset (right).

Feedforward Network on Boston Housing. We fit a single layer feedforward network on the Boston housing dataset. The hyperparameters tuned are: number of nodes in the hidden layer, learning rate, learning rate decay and ρ parameter for the exponential decay rate from RMSprop.

6 Conclusion

We have introduced several implications and advantages of viewing Bayesian optimization as a Markov decision process. We also have shown that this approach can be interesting for further developments of BO, both in theory and practice. As the main contribution of the paper, we have presented a new method for fully distributed BO based on stochastic policies which can be easily integrated in any setup, independent of the surrogate model or acquisition function of choice. This distributed BO allows high scalability, even by adding new resources on demand and reducing the communication between nodes. We show how, in most cases, the stochastic policy outperforms the state of the art on distributed BO (PDTS) and even to the sequential expected improvement.

Acknowledgments

This work has been partly supported by projects DPI2015-65962-R, RTI2018-096903-B-I00 and DGA_T45-17R.

References

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

[Contal *et al.*, 2013] E. Contal, D. Buffoni, A. Robicquet, and N. Vayatis. Parallel Gaussian process optimization with upper confidence bound and pure exploration. In *ECMLKDD*, pages 225–240. Springer, 2013.

[Desautels *et al.*, 2014] T. Desautels, A. Krause, and J.W. Burdick. Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization. *JMLR*, 15(1):3873–3923, 2014.

[Forrester *et al.*, 2006] A. Forrester, N. Bressloff, and A. Keane. Optimization using surrogate models and partially converged computational fluid dynamics simulations. *Proceedings of the Royal Society of London A*, 462(2071):2177–2204, 2006.

[Ginsbourger *et al.*, 2010] D. Ginsbourger, R. Le Riche, and L. Carraro. Kriging is well-suited to parallelize optimization. In *Computational intelligence in expensive optimization problems*, pages 131–162. Springer, 2010.

[González *et al.*, 2016a] J. González, Z. Dai, P. Hennig, and N. Lawrence. Batch bayesian optimization via local penalization. In *AISTATS*, pages 648–657, 2016.

[González *et al.*, 2016b] J. González, M. Osborne, and N. Lawrence. Glasses: Relieving the myopia of bayesian optimisation. In *AISTATS*, pages 790–799, 2016.

[Hennig and Schuler, 2012] P. Hennig and C.J. Schuler. Entropy search for information efficient global optimization. *JMLR*, 13:1809–1837, 2012.

[Hernandez-Lobato *et al.*, 2014] J.M. Hernandez-Lobato, M.W. Hoffman, and Z. Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. In *NIPS*, pages 918–926, 2014.

[Hernandez-Lobato *et al.*, 2017] J.M. Hernandez-Lobato, J. Requeima, E.O. Pyzer-Knapp, and A. Aspuru-Guzik. Parallel and distributed thompson sampling for large-scale accelerated exploration of chemical space. In *ICML*, pages 1470–1479, 2017.

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

[Kaelbling *et al.*, 1996] L.P. Kaelbling, M.L. Littman, and A.W. Moore. Reinforcement learning: A survey. *JMLR*, 4:237–285, 1996.

[Kandasamy *et al.*, 2018] K. Kandasamy, A. Krishnamurthy, J. Schneider, and B. Poczos. Parallelised Bayesian optimisation via thompson sampling. In *AISTATS*, 2018.

[Klein *et al.*, 2017] A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter. Fast Bayesian optimization of machine learning hyperparameters on large datasets. In *AISTATS*, 2017.

[Lam *et al.*, 2016] R. Lam, K. Willcox, and D.H. Wolpert. Bayesian optimization with a finite budget: An approximate dynamic programming approach. In *NIPS*, pages 883–891, 2016.

[Lopes and Montesano, 2014] M. Lopes and L. Montesano. Active learning for autonomous intelligent agents: Exploration, curiosity, and interaction. *arXiv:1403.1497*, 2014.

[Martinez-Cantin *et al.*, 2018] R. Martinez-Cantin, K. Tee, and M. McCourt. Practical bayesian optimization in the presence of outliers. In *AISTATS*, pages 1722–1731, 2018.

[Martinez-Cantin, 2019] R. Martinez-Cantin. Funneled Bayesian optimization for design, tuning and control of autonomous systems. *IEEE Trans Cybern*, 49(4):1489–1500, 2019.

[Merino-Casallo and others, 2018] F. Merino-Casallo et al. Integration of in vitro and in silico models using Bayesian optimization with an application to stochastic modeling of mesenchymal 3d cell migration. *Front. Psychol.*, 9, 2018.

[Mockus *et al.*, 1978] J. Mockus, V. Tiesis, and A. Zilinskas. The application of Bayesian methods for seeking the extremum. In *Towards Global Optimisation 2*, pages 117–129. Elsevier, 1978.

[Neal, 1996] R.M. Neal. Sampling from multimodal distributions using tempered transitions. *Statistics and computing*, 6(4):353–366, 1996.

[Sacks *et al.*, 1989] J. Sacks, W.J. Welch, T.J. Mitchell, and H.P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–423, 1989.

[Shah and Ghahramani, 2015] A. Shah and Z. Ghahramani. Parallel predictive entropy search for batch global optimization of expensive objective functions. In *NIPS*, 2015.

[Shahriari *et al.*, 2016] B. Shahriari, K. Swersky, Z. Wang, R.P. Adams, and N. de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[Singh *et al.*, 2000] S. Singh, T. Jaakkola, M.L. Littman, and C. Szepesvári. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine learning*, 38(3):287–308, 2000.

[Snoek *et al.*, 2012] J. Snoek, H. Larochelle, and R. Adams. Practical Bayesian optimization of machine learning algorithms. In *NIPS*, pages 2960–2968, 2012.

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

[Toussaint, 2014] M. Toussaint. The Bayesian search game. In *Theory and Principled Methods for Designing Metaheuristics*. Springer, 2014.

[Wang and de Freitas, 2014] Z. Wang and N. de Freitas. Theoretical analysis of Bayesian optimisation with unknown Gaussian process hyperparameters. *arXiv*, 2014.

[Wang and Jegelka, 2017] Z. Wang and S. Jegelka. Max-value entropy search for efficient Bayesian optimization. In *ICML*, volume 70, pages 3627–3635, 2017.