A Nonmyopic Approach to Cost-Constrained Bayesian Optimization

Eric Hans Lee[1] **David Eriksson**[2] **Valerio Perrone**[3] **Matthias Seeger**[3]

[1]SigOpt: An Intel Company; eric.lee@intel.com
[2]Facebook; deriksson@fb.com
[3]Amazon; [vperrone, matthis]@amazon.de

Abstract

Bayesian optimization (BO) is a popular method for optimizing expensive-to-evaluate black-box functions. BO budgets are typically given in iterations, which implicitly assumes each evaluation has the same cost. In fact, in many BO applications, evaluation costs vary significantly in different regions of the search space. In hyperparameter optimization, the time spent on neural network training increases with layer size; in clinical trials, the monetary cost of drug compounds vary; and in optimal control, control actions have differing complexities. *Cost-constrained BO* measures convergence with alternative cost metrics such as time, money, or energy, for which the sample efficiency of standard BO methods is ill-suited. For cost-constrained BO, *cost efficiency* is far more important than sample efficiency. In this paper, we formulate cost-constrained BO as a constrained Markov decision process (CMDP), and develop an efficient rollout approximation to the optimal CMDP policy that takes both the cost and future iterations into account. We validate our method on a collection of hyperparameter optimization problems as well as a sensor set selection application.

1 INTRODUCTION

Bayesian optimization (BO) is a class of methods for global optimization of expensive black-box functions. In BO, a probabilistic surrogate model is used to approximate the objective and future evaluations are selected via an acquisition function. BO has been successfully applied to applications such as robotic gait control [Calandra et al., 2016], sensor set selection [Garnett et al., 2010], and neural network hyperparameter tuning [Snoek et al., 2012]. BO is favored for these tasks because of its sample-efficient nature. Achieving this sample-efficiency requires BO to balance exploration and exploitation. However, standard acquisition functions such as expected improvement (EI) are often too greedy in practice. As a result, they perform poorly on multimodal problems [Hernández-Lobato et al., 2014] and have provably sub-optimal performance in certain settings, e.g., bandit problems [Srinivas et al., 2010]. A key research goal in BO is developing less greedy acquisition functions [Shahriari et al., 2016]. Examples include predictive entropy search (PES) [Hernández-Lobato et al., 2014] or knowledge gradient (KG) [Frazier et al., 2008]. Lam et al. [2016] frame the exploration-exploitation trade-off as a balance between immediate and future rewards in a continuous state and action space Markov decision process (MDP). In this framework, *non-myopic* acquisition functions are optimal MDP policies, and promise better performance by considering the impact of future evaluations up to a given BO budget (also referred to as the *horizon*).

While BO budgets are typically given in iterations, this implicitly measures convergence in terms of iteration count and assumes uniform evaluation cost. For many practical BO applications, evaluation costs may vary in different regions of the search space. For example, the time spent on neural network training increases with layer size; the cost of different drug compounds vary; and control actions in optimal control have differing complexities. In all these cases, standard BO is often unable to achieve fast convergence in terms of unit cost. Motivated by these examples, we develop methods that improve convergence when measured by cost. This cost may be time, energy, or money, and the goal is to minimize the objective given a cost budget.

Cost-constrained BO measures convergence with these alternative cost metrics for which standard BO methods are unsuited. We extend non-myopic BO to handle the

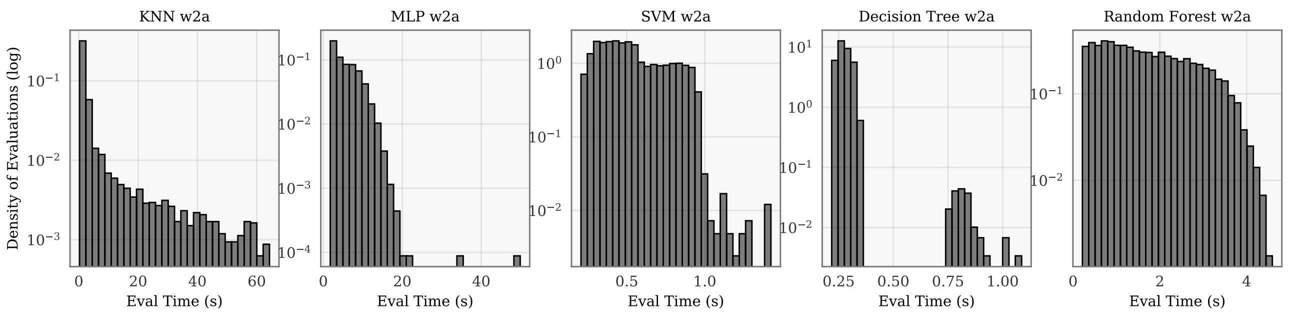

Figure 1: Runtime distribution, log-scaled, of 5000 randomly selected points for the k-nearest-neighbors (KNN), Multi-layer Perceptron (MLP), Support Vector Machine (SVM), Decision Tree (DT), and Random Forest (RF) hyperparameter optimization problems, each trained on the OpenML w2a dataset [Vanschoren et al., 2013]. The runtimes vary, often by an order of magnitude or more.

cost-constrained setting. Our contributions follow:

- We analyze failure modes of common approaches to cost-constrained BO, in which greedy behavior results in poor per-cost performance.
- To avoid overly greedy behavior, we formulate cost-constrained BO as an instance of a constrained Markov decision process (CMDP). This formulation is a novel extension of recent research on non-myopic BO which uses a simpler Markov decision process (MDP) formulation.
- We introduce an approximation to the optimal constrained MDP policy based on rollout of feasible trajectories. Rollout is a popular class of approximate MDP solutions in which future BO realizations and their corresponding values are simulated using the surrogate and then averaged.
- We validate the performance of our methods on a set of practical hyperparameter optimization problems and a sensor set selection problem.

2 RELATED WORK

Most prior approaches to cost-constrained BO occur in the *grey-box* setting, in which additional information about the objective is available. *Multi-fidelity* BO is a widely studied setting in which fidelity parameters $s \in [0,1]^m$, such as iteration count or grid size, are assumed to be a low-accuracy approximation of high-fidelity evaluations [Forrester et al., 2007, Kandasamy et al., 2017, Poloczek et al., 2017, Wu and Frazier, 2019]. Increasing s increases the accuracy at the expense of run time. In addition, Multi-fidelity methods are often application-specific. For example, Hyperband [Li et al., 2017] and its BO variants [Falkner et al., 2018, Klein et al., 2017a,b] cheaply train many neural network configurations for a few epochs, and then prunes unpromising configurations. In *multi-task* BO [Swersky et al., 2013], hyperparameter optimization is first run on cheaper instances before considering more expensive ones. Swersky et al. [2013] introduce a cost-constrained, multi-task variant of entropy search to speed-up optimization of logistic regression and latent Dirichlet allocation. Cost information is input as a set of cost preferences (e.g., a parameter $\mathbf{x}_1$ is more expensive than a parameter $\mathbf{x}_2$) by Abdolshah et al. [2019], who develop a multi-objective, constrained BO method that evaluates cheap points before expensive ones, as determined by the cost preferences, to find feasible, low-cost solutions. These methods outperform their black-box counterparts by evaluating cheap proxies or cheap points before selecting expensive evaluations, which is accomplished by leveraging additional cost information inside the optimization routine. While all these methods demonstrate strong performance, they sacrifice generality and do not apply to black-box BO. Moreover, by relying on parallel resources, these techniques target time efficiency rather than compute time, cost, or energy efficiency.

3 BACKGROUND AND MOTIVATION

Gaussian process regression and BO: The goal in BO is to find a global minimizer of a continuous function $f(\mathbf{x})$ over a compact set $\Omega \subseteq \mathbb{R}^d$. If $f(\mathbf{x})$ is expensive to evaluate, we want to rely on a sample-efficient optimization method. BO uses a Gaussian process (GP) to model $f(\mathbf{x})$ from the data $\mathcal{D}_t = \{(\mathbf{x}_i, y_i)\}_{i=1}^t$. We write this as $f(\mathbf{x}) \sim \mathcal{GP}(\mu_t(\mathbf{x}), \sigma_t^2(\mathbf{x}))$, where $\mu_t(\mathbf{x}), \sigma_t^2(\mathbf{x})$ are the GP mean and variance at $\mathbf{x}$, respectively (see A in the supplementary materials for more details). The next evaluation location $\mathbf{x}_{t+1}$ is determined by maximizing an acquisition function $\Lambda(\mathbf{x} \mid \mathcal{D}_t)$: $\mathbf{x}_{t+1} = \arg\max_\Omega \Lambda(\mathbf{x} \mid \mathcal{D}_t)$.

Cost-constrained BO: BO's sample efficiency leads to fast convergence only if evaluations have similar costs, an assumption that is often not true in practice. Cost-constrained BO is an important problem, and we argue that many BO problems in machine learning are, in fact, cost-constrained. Figure 1 illustrates this by randomly evaluating 5000 hyperparameter configurations for five common hyperparameter optimization problems (HPO). The resulting evaluation times vary, sometimes by more than two orders of magnitude. Moreover, the majority of evaluations are cheap, suggesting that significant cost savings may be achieved by *using a cost-efficient instead of a sample-efficient optimizer*.

The de-facto cost-constrained method in the black-box setting is to normalize the acquisition by cost model $c(\mathbf{x})$. This extends EI to *EI per unit cost (EIpu)*:

$$\text{EIpu}(\mathbf{x}) := \frac{\text{EI}(\mathbf{x})}{c(\mathbf{x})},$$

which is designed to balance the objective's cost and evaluation quality. Snoek et al. [2012] showed that EIpu can boost performance on a variety of HPO problems.

However, EIpu often demonstrates underwhelming performance. We examine why this may occur in Figure 2, in which EIpu (green) is slower than EI (red) at HPO of a k-nearest-neighbor model. The empirical optimum,

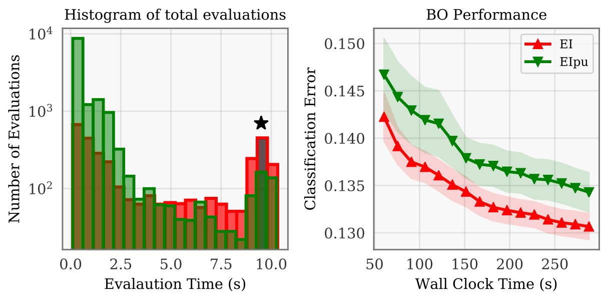

Figure 2: We run EI and EIpu on KNN. *Left:* EIpu evaluates many more cheap points than EI, which evaluates more expensive points. The optimum's cost, one of the most expensive points, is a black star. *Right:* EIpu performs poorly as a result.

namely the best point over all trials (black star), has high cost —thus, dividing by the cost penalizes EIpu *away* from the optimum and diminishes its performance. This is evident from the evaluation time histograms: EIpu evaluates many cheap points while EI evaluates fewer but more expensive points. Due to its bias towards cheap points, EIpu is likely to only display strong results when optima are relatively cheap, which is problematic in the general black-box setting. Indeed, one can adversarially increase the cost at the optimum to make EIpu perform poorly.

Markov decision processes: Nonmyopic BO frames the exploration-exploitation trade-off as a balance of immediate and future rewards in a finite horizon Markov decision process (MDP). We use standard notation from [Puterman, 2014]: an MDP is the collection $< T, \mathbb{S}, \mathbb{A}, P, R >$. Here, $T = \{0, 1, \ldots, h-1\}$, $h < \infty$ is the set of decision epochs, assumed finite for our problem. The state space $\mathbb{S}$ encapsulates the information needed to model the system from time $t \in T$. $\mathbb{A}$ is the action space. Given a state $s \in \mathbb{S}$ and an action $a \in \mathbb{A}$, $P(s'|s, a)$ is the transition probability of the next state being s'. $R(s, a, s')$ is the reward received for choosing action a from state s, and ending in state s'.

A decision rule, $\pi_t : \mathbb{S} \to \mathbb{A}$, maps states to actions at time t. A policy π is a series of decision rules $\pi = (\pi_0, \pi_1, \ldots, \pi_{h-1})$, one at each decision epoch. Given a policy π, a starting state s_0, and horizon h, we can define the expected total reward $V_h^\pi(s_0)$ as:

$$V_h^\pi(s_0) = \mathbb{E}\left[\sum_{t=0}^{h-1} R(s_t, \pi_t(s_t), s_{t+1})\right].$$

In phrasing a sequence of decisions as an MDP, our goal is to find the optimal policy π^* that maximizes the expected total reward, i.e., $\sup_{\pi \in \Pi} V_h^\pi(s_0)$, where Π is the space of all admissible policies.

Constrained Markov decision processes: A constrained Markov decision process (CMDP) is an MDP with an additional set of cost constraints [Altman, 1999, Piunovskiy, 2006, Bertsekas, 2005]. These costs, like MDP rewards, are accumulated through state by action until a certain horizon. A CMDP extends an MDP, and is the collection $< T, \mathbb{S}, \mathbb{A}, P, R, C, \tau >$. Here, $C(s, a, s') : \mathbb{S} \times \mathbb{A} \times \mathbb{S} \to \mathbb{R}$ is a cost function measuring the cost of choosing action a from state s, and ending in state s'. τ is the cost constraint, and we assume without loss of generality that it is a positive scalar.

The cost function C in a CMDP induces a cumulative cost function $C_h^\pi(s_0)$, which is analogous to a value function that replaces the reward with the cost:

$$C_h^\pi(s_0) = \mathbb{E}\left[\sum_{t=0}^{h-1} C(s_t, \pi_t(s_t), s_{t+1})\right].$$

$C_h^\pi(s_0)$ measures total expected cost given a policy π, starting state s_0, and horizon h. The goal in a CMDP is to find the optimal policy, defined as:

$$\pi^* = \arg\max_\pi V_h^\pi(s_0),$$
$$\text{subject to } \; C_h^\pi(s_0) \leq \tau.$$

In other words, we want to determine the policy that maximizes the expected reward subject to having cost less than τ. We refer readers to the standard CMDP

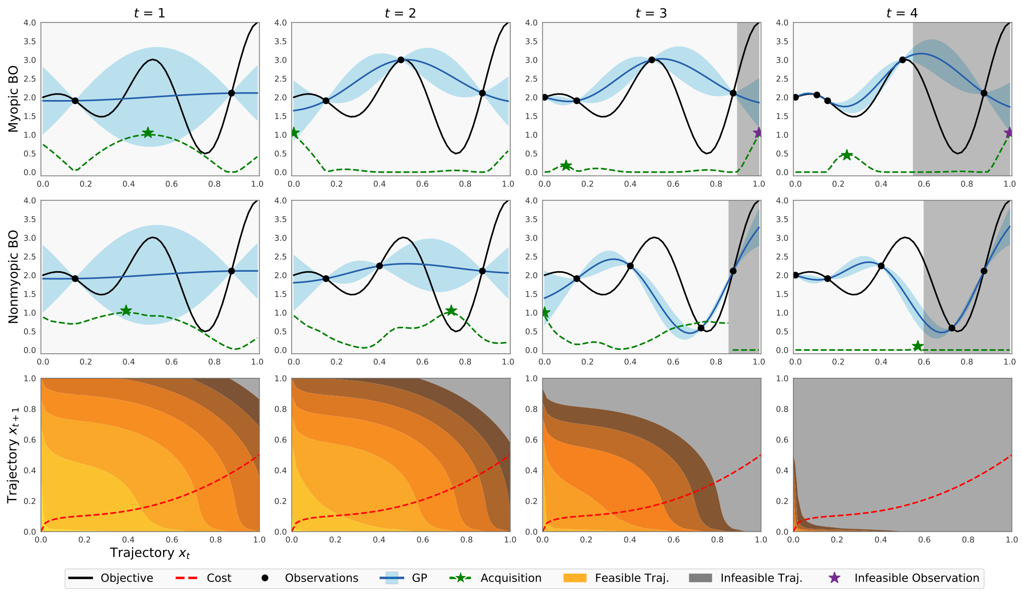

Figure 3: We illustrate the importance of nonmyopia on a carefully chosen toy problem. The top row depicts BO using a myopic acquisition; the middle row depicts nonmyopic, cost-constrained BO; the bottom row depicts feasible trajectories of length two. We run BO until the objective's minimum is infeasible. Myopic BO exhausts its budget before getting close to the global minimum. A nonmyopic approach accounts for the cost (and infeasibility) of future evaluations, and is able to sample the global minimum early. To save space, we only plot feasible trajectories for the myopic approach in the top row. The trajectory contour for nonmyopic BO is very similar.

treatment [Altman, 1999] for more information. The notion of feasible trajectories is important when discussing both exact and approximate CMDP solutions. A CMDP trajectory is a sequence of states and actions:

$$(s_0, a_0), (s_1, a_1), \ldots, (s_{h-1}, a_{h-1}).$$

A CMDP trajectory is said to be *feasible* if it doesn't violate its cost constraint τ for any non-negative integer $0 \leq \ell \leq h-1$ less than horizon h:

$$\sum_{t=0}^{\ell} C(s_t, a_t, s_{t+1}) < \tau.$$

For consistency, we can extend all feasible trajectories to have length h by introducing an intermediate state and action which produce zero reward and cost (the formal equivalent of "standing still"). The set of all feasible trajectories is known as G.

4 NONMYOPIC, COST-CONSTRAINED BO

We might think of cost-constrained BO as the following constrained optimization problem:

$$\min_{\mathbf{X} \in 2^{\Omega}} \min_{\mathbf{x} \in \mathbf{X}} f(\mathbf{x})$$
$$\text{subject to } \sum_{\mathbf{x} \in \mathbf{X}} c(\mathbf{x}) \leq \tau.$$

Because our optimization domain is 2^{Ω}, i.e., the power set of Ω, we have a nested minimization problem. We assume $f(\mathbf{x})$ outputs not only its value, but also its evaluation cost determined by a cost function $c(\mathbf{x})$—which is also black-box. Our goal is to minimize $f(\mathbf{x})$ subject to the total evaluation cost not exceeding τ. This is a pre-specified upper bound on total cost, such as compute time, dollars, or energy consumption.

One recent research direction in BO has been the development of nonmyopic BO [Frazier, 2018, Lam et al., 2016, Lam and Willcox, 2017, Yue and Kontar, 2019, Lee et al., 2020a], which account for the impact of future evaluations and are thus able to make better decisions. These decisions are computed by modeling BO as an MDP and then approximating its optimal policy. We aim to leverage this MDP framework to make similarly principled, nonmyopic decisions in the cost-constrained setting. We do this by extending the MDP to a CMDP, which takes into account variable evaluation costs. The next decision in this setting is a approximation to the optimal CMDP policy.

Figure 3 illustrates the advantage of accounting for the cost of future evaluations with CMDP. The objective and cost, which have been carefully chosen, are plotted in black and red respectively, and the bottom row indicates feasible trajectories of length two as the optimization continues. The optimization domains are similarly shaded when evaluations become infeasible.

A greedy approach[1], seen in the first row, does not account for the remaining budget, and is therefore unable to evaluate the global minimum before its becomes infeasible due to an insufficient budget. A nonmyopic policy, seen in the second row, better accounts for the cost of future evaluations; it sees that by the fourth iteration, the right side of the domain becomes infeasible, and decides to evaluate there earlier. As a result, it gets much closer to the global minimum.

In the next section, we formalize our CMDP framework. We note that our framework is vaguely related to BO with resources (BOR) [Dolatnia et al., 2016], who consider a partially observable MDP (POMDP) framework for BO when resource consumption of the objective varies, and when there might be multiple agents that can evaluate the acquisition function in parallel.

4.1 BO AS A CONSTRAINED MARKOV DECISION PROCESS

Given a deterministic cost function $c(\mathbf{x}) : \Omega \to \mathbb{R}^+$, a cost budget τ, and a GP prior over the observation set $\mathcal{D}_t$ with mean μ_t and kernel k_t, we model h steps of cost-constrained BO as the following CMDP: $< T, \mathbb{S}, \mathbb{A}, P, R, C, \tau >$.

Here, T is the set of decision epochs $\{0, 1, \dots, h-1\}$ representing h steps of BO. While we might want to use an infinite horizon, e.g., $h = \infty$ to continue optimization until our cost budget is exhausted, we assume a finite horizon for tractability. Our state space is the set of observations reachable from starting state $\mathcal{D}_t$ with h BO steps, and the action space is Ω; actions correspond to sampling a point in Ω.

The transition probabilities from state $\mathcal{D}_t$ to state $\mathcal{D}_{t+1}$, where $\mathcal{D}_{t+1} = \mathcal{D}_t \cup \{(\mathbf{x}_{t+1}, y_{t+1})\}$, given an action $\mathbf{x}_{t+1}$, are defined as:

$$\begin{aligned}&P(\mathcal{D}_{t+1} \mid \mathcal{D}_t, \mathbf{x}_{t+1}) \\ &\sim \mathcal{N}(\mu^{(t)}(\mathbf{x}_{t+1}; \mathcal{D}_t), K^{(t)}(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}; \mathcal{D}_t)).\end{aligned}$$

In other words, the probability of transitioning from $\mathcal{D}_t$ to $\mathcal{D}_{t+1}$ is the probability of sampling y_{t+1} from the posterior of $\mathcal{GP}(\mu_t, \sigma_t^2)$ at $\mathbf{x}_{t+1}$.

Given an action and transition to a new state $\mathcal{D}_{t+1}$, our reward function is derived from the the EI criterion [Jones et al., 1998]. Let y_t^* be the minimum observed value in the observed set $\mathcal{D}_t$, i.e., $y_t^* = \min\{y_0, \dots, y_t\}$. Then our reward is expressed as:

$$\begin{aligned}R(\mathcal{D}_t, \mathbf{x}_{t+1}, \mathcal{D}_{t+1}) &= (y_t^* - y_{t+1})^+ \\ &\equiv \max(y_t^* - y_{t+1}, 0).\end{aligned}$$

Our CMDP cost is given by $c(\mathbf{x})$. We assume that this cost is deterministic and state-independent; it only depends on the action. In practice, the cost function may be learned as well. We emphasize that we assume a *deterministic* cost function; the algorithms and theory we establish do not extend trivially to stochastic cost functions. Finally, we assume a positive scalar constraint τ. However, we could extend this to a vector-valued constraint. For example, in materials design, there might be a finite amount of each constituent component, each with its own budget [Abdolshah et al., 2019].

The expected total reward and cost of a policy π are

$$\begin{aligned}V_h^\pi(\mathcal{D}_k) &= \mathbb{E}\Bigg[\sum_{t=k}^{k+h-1} R(\mathcal{D}_t, \pi_t(\mathcal{D}_t), \mathcal{D}_{t+1})\Bigg] \\ &= \mathbb{E}\Bigg[\sum_{t=k}^{k+h-1} (y_t^* - y_{t+1})^+\Bigg], \\ C_h^\pi(\mathcal{D}_k) &= \mathbb{E}\Bigg[\sum_{t=k}^{k+h-1} c(\pi_t(\mathcal{D}_t))\Bigg].\end{aligned}$$

More intuitively, $V_h^\pi(\mathcal{D}_k)$ is the expected reduction in the objective function using policy π, and $C_h^\pi(\mathcal{D}_k)$ is the accompanying expected cost. We can represent a trajectory though this CMDP as the sequence:

$$(\mathbf{x}_k, y_k), (\mathbf{x}_{k+1}, y_{k+1}), \dots, (\mathbf{x}_{k+1}, y_{k+h}).$$

As our cost is strictly positive, a trajectory $(\mathbf{x}_k, y_k), (\mathbf{x}_{k+1}, y_{k+1}), \dots, (\mathbf{x}_{k+1}, y_{k+\ell})$ is feasible if $\sum_{i=k}^{k+\ell} c(\mathbf{x}_i) \leq \tau$ for some $\ell \leq h$.

5 METHODS

CMDPs are considered far more difficult to solve than MDPs [Altman, 1999], and the standard dynamic programming approach of Bertsekas [2017] does not extend trivially —Bellman's principle of optimality no longer applies. Indeed, unlike the MDP case, the existence of an optimal policy is not guaranteed. The standard CMDP solution is to solve a large linear program in the state and action spaces, but this is computationally intractable for all but the smallest problems. The difficulty of solving CMDPs in the BO setting is made more difficult by the exponentially growing infinite state space, which consequently excludes standard solutions such as an exact solve on a discretized problem.

In this paper, we approximate the optimal CMDP policy through rollouts, which has been used successfully

[1]We show EI over EIpu for space's sake; the former performs better than the latter. The low cost on the left part of the domain causes EIpu to evaluate exclusively there.

in the standard BO setting to improve performance over myopic acquisition functions [Lam et al., 2016].

5.1 ROLLOUT

MDP Rollout Rollouts forward-simulate the value function of a fixed policy, and select the action yielding the maximal simulated reward. We make this more precise as follows. For a given current state $\mathcal{D}_k$, we denote our base rollout policy $\tilde{\pi} = (\tilde{\pi}_0, \tilde{\pi}_1, \ldots, \tilde{\pi}_{h-1})$. We introduce the notation $\mathcal{D}_{k,0} \equiv \mathcal{D}_k$ to define the initial state of our MDP and $\mathcal{D}_{k,t}$ for $1 \leq t \leq h$ to denote the random variable that is the state at each decision epoch. In the case of BO, each individual decision rule $\tilde{\pi}_t$ consists of maximizing the base acquisition function Λ given the current state $s_t = \mathcal{D}_{k,t}$,

$$\tilde{\pi}_t = \arg\max_{\mathbf{x}\in\Omega} \Lambda_t(\mathbf{x} \mid \mathcal{D}_{k,t}).$$

Using this policy, we define the non-myopic acquisition function $\Lambda_h(\mathbf{x})$ as the rollout of $\tilde{\pi}$ to horizon h i.e., the expected reward of $\tilde{\pi}$ starting with the action $\tilde{\pi}_0 = \mathbf{x}$:

$$\Lambda_h(\mathbf{x}_{k+1}) := \mathbb{E}\left[V_h^{\tilde{\pi}}(\mathcal{D}_k \cup \{(\mathbf{x}_{k+1}, y_{k+1})\})\right],$$

where y_{k+1} is the noisy observed value of f at $\mathbf{x}_{k+1}$. Λ_h is better than Λ in expectation for a correctly specified GP prior and for any acquisition function. This follows from standard results in the MDP literature [Bertsekas, 2017]. If we can sample from the transition probability P, we can estimate the expected reward of $\tilde{\pi}$ through policy evaluation, i.e., Monte-Carlo integration:

$$V_h^{\tilde{\pi}}(s_0) \approx \frac{1}{N}\sum_{i=1}^{N}\left[\sum_{t=0}^{h-1} R(s_t^i, \tilde{\pi}_t(s_t^i), s_{t+1}^i)\right].$$

CMDP Rollout In the CMDP setting, rollout is also forward simulation of action and reward given a fixed base policy, except that CMDP rollout only forward simulates feasible trajectories in G and discards infeasible trajectories [Bertsekas, 2005]. In practice, this means that as we roll out a base policy $\tilde{\pi}$, we terminate either once we reach the horizon or violate the cost constraint. There remains the question of what base policy we might use; the performance of rollout depends on its base policy. We develop a base policy by considering the following two cases:

$\mathbf{h = 1}$: Assume the argmax of EI has cost $c(\mathbf{x}^*) \leq \tau$. The following policy π is CMDP optimal:

$$\pi(\mathcal{D}_t) = \mathbf{x}^* = \arg\max_{\mathbf{x}\in\Omega} \mathrm{EI}(\mathbf{x} \mid \mathcal{D}_t).$$

$\mathbf{h > 1}$: Assume the argmax of EI has cost $c(\mathbf{x}^*) = \tau^*$ and there exists a point of small cost $c(\mathbf{x}_\epsilon) = \epsilon$. If $\tau = \tau^* + \epsilon$, then $\mathbf{x}_\epsilon$ should be evaluated before $\mathbf{x}^*$. In the limit, a point that is free to evaluate should be evaluated first.

A reasonable base policy should, at the minimum, satisfy these two cases. For the first case, maximizing EI must necessarily be the last step in our base policy. In the second case, we note that maximizing EIpu for the first rollout iteration will result in the desired behavior. For simplicity's sake, we extend EIpu until the last iteration. The base rollout policy $\tilde{\pi} = (\tilde{\pi}_0, \ldots, \tilde{\pi}_{h-1})$ that we consider is therefore

$$\tilde{\pi}_t(\mathcal{D}_t) = \begin{cases} \arg\max_{\mathbf{x}\in\Omega} \mathrm{EIpu}(\mathbf{x} \mid \mathcal{D}_t), & t < h-1, \\ \arg\max_{\mathbf{x}\in\Omega} \mathrm{EI}(\mathbf{x} \mid \mathcal{D}_t), & t = h-1. \end{cases}$$

In other words, $\tilde{\pi}$ rolls out $h-1$ steps of EIpu followed by a last step of EI.

This base policy has a few advantages. If $h = 1$ and the budget is sufficient, it is CMDP optimal. If the cost is uniform, this is equivalent to rollout of EI, which has been shown to improve performance in standard BO [Wu and Frazier, 2019, Lee et al., 2020a]. Lastly, this base policy is consistent with an early exploration, late exploitation strategy, which is a common heuristic in multifidelity and multitask settings; EIpu tends to select cheaper points. Therefore, $\tilde{\pi}$ starts by trying to select cheaper points and then ends with selecting a point that is likely more expensive.

5.2 THEORETICAL ANALYSIS

If a base policy $\tilde{\pi}$ is deterministic, rollout in the MDP setting will perform better in expectation than the base policy itself. The same holds true in the CMDP setting if $c(\mathbf{x})$ is also deterministic.

Definition 1 *[Bertsekas, 2017]: A policy π is sequentially consistent if, for every trajectory from any s_0:*

$$(s_0, a_0), (s_1, a_1), \ldots, (s_{h-1}, a_{h-1}),$$

π generates the following trajectory starting at s_1:

$$(s_1, a_1), (s_2, a_2) \ldots, (s_{h-1}, a_{h-1}).$$

Theorem 1 *[Bertsekas, 2005]: In the CMDP setting, a rollout policy π_{roll} does no worse than its base policy $\tilde{\pi}$ in expectation if $\tilde{\pi}$ is sequentially consistent i.e.,*

$$V_h^{\pi_{roll}}(s_0) \geq V_h^{\tilde{\pi}}(s_0).$$

Thus, the value function of a rollout policy is always greater than the value function of the base policy.

To guarantee sequential consistency of our acquisition function, we need only consistently break ties if the acquisition function has multiple maxima.

6 EXPERIMENTS

We compare CMDP rollout, which we compute via quasi-Monte Carlo integration, to EI and EIpu. We use a GP with the Matérn-5/2 ARD kernel to model both the objective and the cost function[2], and learn hyperparameters via maximum likelihood estimation. When rolling out acquisition functions, we use L-BFGS-B using 5 restarts, selected by evaluating the acquisition on a Latin hypercube of $10d$ points and picking the five best as starting points. When comparing different replications we first need to interpolate the objective function values onto a set of discrete costs. Given these interpolated value, we plot the mean with one standard deviation. Code to reproduce our experiments is found at `https://github.com/ericlee0803/lookahead_release`.

6.1 SYNTHETIC PROBLEM

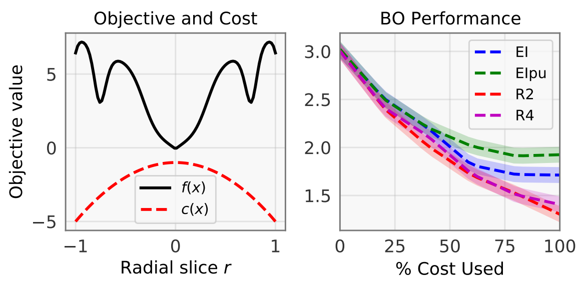

Figure 4: In this example, we examine a carefully chosen example showcasing the strength of the rollout approach. We consider the a multimodal objective whose most expensive point is the global minimum. EIpu performs worse than EI, and both tend to get stuck in cheaper, local minimum. Our rollout policy for horizons 2 and 4 performs better than both EI and EIpu

In Figure 4, we examine a carefully chosen synthetic example showcasing the strength of the rollout approach. We consider the cost-constrained optimization problem

$$f(\mathbf{x}) = 10\|\mathbf{x}\|_2 \sin(2\pi\|\mathbf{x}\|_2),$$
$$c(\mathbf{x}) = 10 - 5\|\mathbf{x}\|_2,$$

in the domain $[-1, 1]^2$, and a budget of 150. The cost function has been designed so that its maximum aligns with the minimum of the objective. As we motivated earlier, EIpu struggles with these types of problems. We run BO with EI, EIpu, and rollout with our base policy 50 times and plot the results. This is seen on the right, in which EIpu (green) performs worse than EI (blue). However, rollout of our base policy, for horizons two and four in pink and red respectively, performs much better than both.

[2]We use a log-warped GP to model positive cost.

6.2 HYPERPARAMETER OPTIMIZATION

We compare rollout performance to EI and EIpu on HPO of three different models: k-nearest neighbors (KNN), decision trees, and random forests, with budgets of 800, 15, and 15 seconds respectively. These are relatively small problems, chosen due to the number of replications required to show statistical significance. All models are trained for 50 replications on the OpenML w2a dataset [Vanschoren et al., 2013]. We compare the competing algorithms in terms of the best classification error achieved on the validation set.

k-nearest neighbors: The k-nearest neighbors algorithm is a class of methods used for classification of either spatially-orientated data or data with a known distance metric (i.e., data embedded in a Hilbert space). We consider a $5d$ search space: dimensionality reduction percentage in $[1\text{e}{-6}, 1.0]$ (log-scaled), type in {Gaussian, Random}, neighbor count in $\{1, 2, \ldots, 256\}$, weight function in {Uniform, Distance}, and distance in {Minkowski, Cityblock, Cosine, Euclidean, L1, L2, Manhattan}. We one-hot encode categorical variables.

Decision Trees: Decision trees are popular predictive models used in statistics, data mining, and machine learning. In the case of classification, leaves represent class labels and paths represent sets of features that lead to those class labels. During training, a tree is built by splitting the source set into subsets which constitute the successor children. The splitting is based on a threshold that maximizes some notion of information gain such as entropy. The depth of a decision tree is pre-specified. We consider a $3d$ search space: tree depth in $\{1, 2, \ldots, 64\}$, tree split threshold in $[0.1, 1.0]$ log-scaled, and split feature size in $[1\text{e}{-3}, 0.5]$ (log-scaled).

Random Forests: A random forest is a set k of decision trees, and classifies based off the plurality decision generated form all its trees —this technique is known as *bagging*, and improves robustness in the classification algorithm. We consider a $3d$ search space: number of trees in $\{1, 2, \ldots, 256\}$, tree depth in $\{1, 2, \ldots, 64\}$, and tree split threshold in $[0.1, 1.0]$ (log-scaled).

6.3 SENSOR SET SELECTION

The sensor set selection problem [Garnett et al., 2010] seeks to improve the predictive accuracy, as measured by the root mean squared error (RMSE), of a physical sensor network. We denote a sensor network's configuration of m sensors in d-dimensional space as $\mathbf{X} \in \mathbb{R}^{m \times d}$.

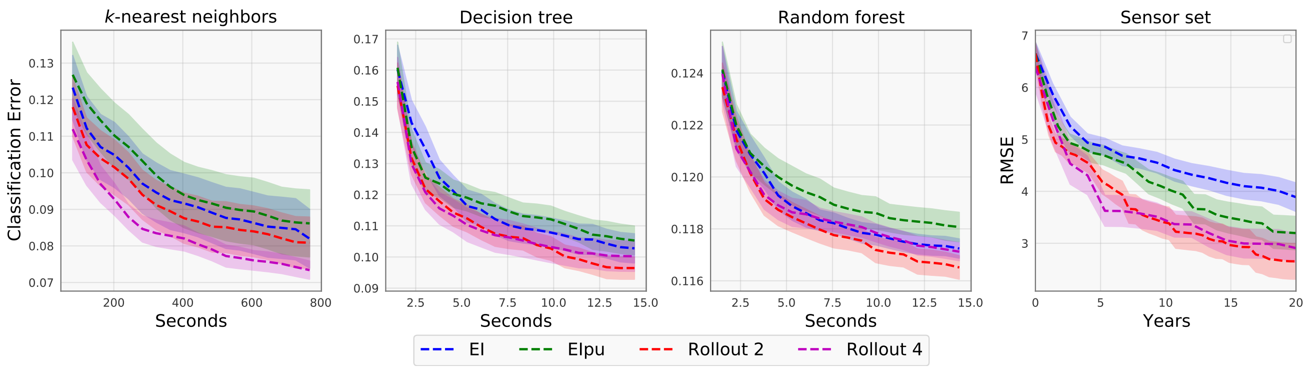

Figure 5: We compare the classification error among EI, EIpu, and our cost-constrained rollout for horizons 2 and 4. Rollout performs better than both EI and EIpu. Shaded areas represent one standard deviation around the mean.

This configuration must be manually adjusted each time it is updated. This is typically assumed to have uniform cost; we modify the problem to consider a *sensor adjustment cost*. We assume this cost is correlated with the distance each sensor in the network has to move; for simplicity, we assume the cost of any new sensor configuration $\mathbf{X}'$ is proportional to the straight-line distance $d(\mathbf{X}, \mathbf{X}') = \|\mathbf{X} - \mathbf{X}'\|_F$. This is an example of a CMDP whose cost is not state-independent; our cost now depends on the prior state.

We consider a small sensor set selection problem using ten sensors. Our objective is RMSE of the sensor predictions against ground truth weather data taken from UK Meteorological Office MIDAS surface stations [Centre, 2012]. The time budget is twenty years.

6.4 ANALYSIS

We plot the resuts of our HPO and sensor set selection experiments in Figure 5, and find that CMDP rollout generally outperforms both EI and EIpu. In this section we discuss key insights gained over the course of experimentation.

Cost Modeling: We found the cost function to be simpler to model than the objective function. In practice, the cost may only depend on a few key parameters (e.g., tree depth). Thus, using a vanilla GP to model the cost is inefficient —a tailored (parametric) cost model or a GP that incorporates parameter importance into its lengthscale priors will likely lead to better results [Lee et al., 2020b, Guinet et al., 2020].

Search Space Sensitivity: EIpu's performance depends on the correlation between objective value and cost. Unsurprisingly, this correlation often depends on the search space in practice. For example, assume a decision tree of depth d achieves maximal classification error and that its training cost increases with depth. (**i**) If the search space is $[1, d]$, the maximum will be the most expensive point and EIpu will perform poorly; (**ii**) if the search space is $[1, 2d]$, the maximum will be have middling cost and EIpu will perform moderately well; (**iii**) if the search space is $[1, 10d]$, the maximum will have cheap cost and EIpu will perform very well. In our experiments, we found CMDP rollout to be more robust to the shape of the cost surface. This is expected, as a CMDP optimal policy selects the point that maximally reduces the objective function given the cost constraint.

7 CONCLUSION

In this paper, we have shown the importance of cost-constrained BO and formulated it as an instance of a constrained Markov decision process (CMDP). We developed a rollout algorithm using a cheap exploration, expensive exploitation base policy that performed better than EI and EIpu on three hyperparameter optimization problems and a sensor set selection problem.

These investigations into cost-constrained BO are promising and we believe there are many interesting directions for future work. First, the overhead of the optimizer itself should be taken into account, especially in the context of HPO. While the overhead is negligible when the cost of evaluating the black-box is high (e.g., when training deep neural networks), future work could explore simpler heuristics to lower the overhead of using rollouts. Second, we believe approximate solutions to CMDPs other than rollout are worth investigating. State aggregation and state truncation are classic methods in the MDP setting that reduce the state space according to the transition probabilities and Altman [1999] extends them to the CMDP setting. Consequently, we may approximate our model through state aggregation and state truncation and then compute an exact solution via linear programming.

Finally, we have limited our discussion to the sequential BO setting. However, cost-constrained BO becomes significantly more complex in the batch setting, when evaluations are performed in parallel. This is another interesting topic for future work.

References

Majid Abdolshah, Alistair Shilton, Santu Rana, Sunil Gupta, and Svetha Venkatesh. Cost-aware multi-objective Bayesian optimisation. *arXiv preprint arXiv:1909.03600*, 2019.

Eitan Altman. *Constrained Markov decision processes*, volume 7. CRC Press, 1999.

D Bertsekas. Rollout algorithms for constrained dynamic programming. *Lab. for Information and Decision Systems Report*, 2646, 2005.

Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*, volume I. Athena scientific Belmont, MA, 4th edition, 2017.

Russel E Caflisch. Monte Carlo and quasi-Monte Carlo methods. *Acta numerica*, 7:1–49, 1998.

Roberto Calandra, André Seyfarth, Jan Peters, and Marc Peter Deisenroth. Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence*, 76(1-2):5–23, 2016.

NCAS British Atmospheric Data Centre. Met office integrated data archive system (midas) land and marine surface stations data, 2012. URL `http://badc.nerc.ac.uk/data/ukmo-midas`.

Nima Dolatnia, Alan Fern, and Xiaoli Fern. Bayesian optimization with resource constraints and production. In *Twenty-Sixth International Conference on Automated Planning and Scheduling*, 2016.

Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: robust and efficient hyperparameter optimization at scale. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1436–1445. PMLR, 2018.

Alexander IJ Forrester, András Sóbester, and Andy J Keane. Multi-fidelity optimization via surrogate modelling. *Proceedings of the royal society a: mathematical, physical and engineering sciences*, 463(2088): 3251–3269, 2007.

Peter I. Frazier. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

Peter I. Frazier, Warren B. Powell, and Savas Dayanik. A knowledge-gradient policy for sequential information collection. *SIAM Journal on Control and Optimization*, 47(5):2410–2439, 2008.

Roman Garnett, Michael A. Osborne, and Stephen J. Roberts. Bayesian optimization for sensor set selection. In *Proceedings of the 9th ACM/IEEE Conference on Information Processing in Sensor Networks*, pages 209–219, 2010.

Gauthier Guinet, Valerio Perrone, and Cédric Archambeau. Pareto-efficient Acquisition Functions for Cost-Aware Bayesian Optimization. *NeurIPS Meta Learning Workshop*, 2020.

José Miguel Hernández-Lobato, Matthew W. Hoffman, and Zoubin Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. In *Advances in Neural Information Processing Systems 27*, pages 918–926, 2014.

Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998. ISSN 1573-2916.

Kirthevasan Kandasamy, Gautam Dasarathy, Jeff G. Schneider, and Barnabás Póczos. Multi-fidelity Bayesian optimisation with continuous approximations. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1799–1808. PMLR, 2017.

Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast Bayesian optimization of machine learning hyperparameters on large datasets. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 528–536. PMLR, 2017a.

Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast Bayesian optimization of machine learning hyperparameters on large datasets. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 528–536. PMLR, 2017b.

Rémi Lam and Karen Willcox. Lookahead Bayesian optimization with inequality constraints. In *Advances in Neural Information Processing Systems 30*, pages 1890–1900, 2017.

Rémi Lam, Karen Willcox, and David H. Wolpert. Bayesian optimization with a finite budget: An approximate dynamic programming approach. In *Ad-*

vances in Neural Information Processing Systems 29, pages 883–891, 2016.

Eric Hans Lee, David Eriksson, David Bindel, Bolong Cheng, and Mike Mccourt. Efficient rollout strategies for Bayesian optimization. In *Proceedings of the Thirty-Sixth Conference on Uncertainty in Artificial Intelligence*, volume 124 of *Proceedings of Machine Learning Research*, pages 260–269. AUAI Press, 2020a.

Eric Hans Lee, Valerio Perrone, Cedric Archambeau, and Matthias Seeger. Cost-aware Bayesian optimization. In *ICML AutoML Workshop*, 2020b.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816, 2017.

Alexey B Piunovskiy. Dynamic programming in constrained Markov decision processes. *Control and Cybernetics*, 35(3):645, 2006.

Matthias Poloczek, Jialei Wang, and Peter I. Frazier. Multi-information source optimization. In *Advances in Neural Information Processing Systems 30*, pages 4288–4298, 2017.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 2014.

Carl Rasmussen and Chris Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems 25*, pages 2960–2968, 2012.

Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias W. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on Machine Learning*, pages 1015–1022. Omnipress, 2010.

Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Multi-task Bayesian optimization. In *Advances in Neural Information Processing Systems 26*, pages 2004–2012, 2013.

Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. Openml: Networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60, 2013. doi: 10.1145/2641190.2641198. URL `http://doi.acm.org/10.1145/2641190.2641198`.

Jian Wu and Peter I. Frazier. Practical two-step lookahead Bayesian optimization. In *Advances in Neural Information Processing Systems 32*, pages 9810–9820, 2019.

Xubo Yue and Raed Al Kontar. Lookahead Bayesian optimization via rollout: Guarantees and sequential rolling horizons. *arXiv preprint arXiv:1911.01004*, 2019.

A GAUSSIAN PROCESS REGRESSION

We place a GP prior on $f(\mathbf{x})$, denoted by $f \sim \mathcal{GP}(\mu, K)$, where $\mu : \Omega \to \mathbb{R}$ and $k : \Omega \times \Omega \to \mathbb{R}$ are the mean function and covariance kernel, respectively. The kernel $k(\mathbf{x}, \mathbf{x}')$ correlates neighboring points, and may contain *hyperparameters*, such as lengthscales that are learned to improve the quality of approximation [Rasmussen and Williams, 2006]. For a given $\mathcal{D}_t = \{(\mathbf{x}_i, y_i)\}_{i=1}^t$, we define:

$$\mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_t \end{pmatrix}, \ \mathbf{k}(\mathbf{x}) = \begin{pmatrix} k(\mathbf{x}, \mathbf{x}_1) \\ \vdots \\ k(\mathbf{x}, \mathbf{x}_t) \end{pmatrix}, \ \mathsf{K} = \begin{pmatrix} \mathbf{k}(\mathbf{x}_1)^\top \\ \vdots \\ \mathbf{k}(\mathbf{x}_k)^\top \end{pmatrix}.$$

We assume y_i is observed with Gaussian white noise: $y_i = f(\mathbf{x}_i) + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. Given a GP prior and data $\mathcal{D}_t$, the resulting posterior distribution for function values at a location $\mathbf{x}$ is the Normal distribution $\mathcal{N}(\mu_t(\mathbf{x}; \mathcal{D}_t), \sigma_t^2(\mathbf{x}; \mathcal{D}_t))$:

$$\mu_t(\mathbf{x}; \mathcal{D}_t) = \mu(\mathbf{x}) + \mathbf{k}(\mathbf{x})^\top (\mathsf{K} + \sigma^2 \mathsf{I}_t)^{-1} (\mathbf{y} - \mu(\mathbf{x})),$$
$$\sigma_t^2(\mathbf{x}; \mathcal{D}_t) = k(\mathbf{x}, \mathbf{x}) - \mathbf{k}(\mathbf{x})^\top (\mathsf{K} + \sigma^2 \mathsf{I}_t)^{-1} \mathbf{k}(\mathbf{x}),$$

where I_t is the $t \times t$ identity matrix.

A.1 KERNELS

The kernel functions we use in this paper are the squared exponential (SE) kernel, Matérn 5/2 kernel, and Matérn 3/2 kernel, respectively

$$k_{\mathrm{SE}}(\mathbf{x}, \mathbf{x}') = \alpha^2 \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\ell^2}\right)$$

$$k_{5/2}(\mathbf{x}, \mathbf{x}') = \alpha^2 \left(1 + \frac{\sqrt{5}}{\ell} + \frac{5}{3\ell^2}\right) \exp\left(-\frac{\sqrt{5}\|\mathbf{x} - \mathbf{x}'\|}{\ell}\right)$$

$$k_{3/2}(\mathbf{x}, \mathbf{x}') = \alpha^2 \left(1 + \frac{\sqrt{3}}{\ell}\right) \exp\left(-\frac{\sqrt{3}\|\mathbf{x} - \mathbf{x}'\|}{\ell}\right)$$

B COMPUTATIONAL COST

The main computational subroutines in a point-wise calculation of $\Lambda_h(\mathbf{x} \mid \mathcal{D}_n)$ are h sequential updates of a Cholesky factorization to update the GP posterior, and h maximizations of $\Lambda(\mathbf{x} \mid \mathcal{D}_n)$.

Assume the GP's kernel matrix is $K_{11} \in \mathbb{R}^{n \times n} = L_{11} L_{11}^T$. Augmenting the Cholesky of K_{11} augmented with a single row and column is done by:

$$K = \begin{bmatrix} K_{11} & K_{12} \\ K_{12}^T & K_{22} \end{bmatrix}, \ K_{11} = L_{11} L_{11}^T,$$

$$L = \begin{bmatrix} L_{11} & 0 \\ L_{12}^T & L_{22} \end{bmatrix}$$

$$L_{12} = L_{11}^{-1} K_{12}, \ L_{22} = \mathrm{chol}(K_{22} - L_{12}^T L_{12}).$$

Sequentially updating the Cholesky will increase the size of the kernel matrix each time. Therefore, updating the Cholesky h times has total cost $\mathcal{O}(h(n+h)^2)$

Maximizing $\Lambda(\mathbf{x} \mid \mathcal{D}_n)$ is done with quasi-Newton methods, typically via BFGS or L-BFGS with r restarts. Obtaining the gradient of EI is equivalent to obtaining the gradient for $\mu^{(n)}(\mathbf{x}; \mathcal{D}_n)$ and $K^{(n)}(\mathbf{x}, \mathbf{x}'; \mathcal{D}_n)$. The cost of obtaining the gradient is $\mathcal{O}(n^2 d)$ (we treat the number of quasi-Newton iterations and restarts as a constant). Therefore, maximizing $\Lambda(\mathbf{x} \mid \mathcal{D}_n)$ sequentially h times has total complexity $\mathcal{O}(n^2 d)$.

B.1 EFFICIENT INTEGRATION

Monte Carlo (MC) is well-suited to high-dimensional integration and is the standard way to rollout a base policy. MC converges at a rate of $\sigma/\sqrt{N}$, the standard deviation of the MC estimator, where σ is the sample variance and N is the total number of samples. MC's primary drawback is slow convergence. Increasing precision by an order of magnitude requires two orders of magnitude more samples. If σ is high, many samples may be required to converge. We use two variance reduction techniques shown in Lee et al. [2020a] to significantly reduce the computational overhead of rollout: Quasi-Monte carlo and common random numbers. Variance reduction is a class of methods that improve convergence by decreasing the variance of the estimator.

Quasi-Monte Carlo (QMC) Instead of sampling directly from the probability distribution, QMC instead uses a low-discrepancy sequence as its sample set. QMC converges at a rate bounded above by $\log(N)^h/N$, where N is the number of samples and h is the integral's dimension. This bound stems from the well-known Koksma-Hlawka inequality Caflisch [1998], and is roughly linear for large N and moderate h. In practice, this bound is often loose and convergence proceeds faster.

Common random numbers (CRN) CRN is used when estimating a quantity to be optimized over parameter $\mathbf{x}$, and is implemented by using the same random number stream for all values of $\mathbf{x}$. CRN does not decrease the point-wise variance of an estimate, but rather decreases the covariance between two neighboring estimates, which smooths out the function.