MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

Efficient Multi-Step Lookahead Bayesian Optimization with Local Search Constraints

Paulson, Joel A.; Sorouifar, Farshud; Chakrabarty, Ankush

TR2022-161 December 09, 2022

Abstract

Bayesian optimization (BO) is a data-efficient approach for optimizing expensive-to-evaluate black-box functions that suffer from noisy evaluations. Traditional BO algorithms ignore the relationship between consecutive input values, which is known to lead to significant "jumps" in the search space that cannot be implemented in practice, especially in online experimental systems. For example, in performance-driven control applications, large changes in the chosen setpoint parameters may trigger fail-safe mechanisms or lead to violation of critical safety constraints. In such applications, it is necessary to limit the allowable search space at each BO iteration, which can be done by incorporating local search constraints into the original problem setting. In this paper, we show how this novel BO setting can be cast as a Markov decision process (MDP) for which the optimal policy is characterized by an intractable dynamic programming (DP) problem. To overcome this challenge, we take advantage of approximate DP methods, particularly rollout with fast policy search, to derive an efficient multi-step lookahead BO policy. We also propose a novel base policy needed for the rollout algorithm, which explicitly incorporates the local search restrictions in an efficient and intuitive manner. Lastly, we empirically show that our proposed multi-step lookahead BO policy outperforms existing methods on a well-known benchmark problem.

IEEE Conference on Decision and Control (CDC) 2022

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

Efficient Multi-Step Lookahead Bayesian Optimization with Local Search Constraints

Joel A. Paulson, Farshud Sorouifar, and Ankush Chakrabarty

Abstract— **Bayesian optimization (BO) is a data-efficient approach for optimizing expensive-to-evaluate black-box functions that suffer from noisy evaluations. Traditional BO algorithms ignore the relationship between consecutive input values, which is known to lead to significant "jumps" in the search space that cannot be implemented in practice, especially in online experimental systems. For example, in performance-driven control applications, large changes in the chosen setpoint parameters may trigger fail-safe mechanisms or lead to violation of critical safety constraints. In such applications, it is necessary to limit the allowable search space at each BO iteration, which can be done by incorporating *local search constraints* into the original problem setting. In this paper, we show how this novel BO setting can be cast as a Markov decision process (MDP) for which the optimal policy is characterized by an intractable dynamic programming (DP) problem. To overcome this challenge, we take advantage of approximate DP methods, particularly rollout with fast policy search, to derive an efficient multi-step lookahead BO policy. We also propose a novel base policy needed for the rollout algorithm, which explicitly incorporates the local search restrictions in an efficient and intuitive manner. Lastly, we empirically show that our proposed multi-step lookahead BO policy outperforms existing methods on a well-known benchmark problem.**

I. Introduction

In a variety of important real-world applications, we are tasked with derivative-free global optimization of expensive-to-evaluate black-box functions. In many cases, the objective function structure is unknown and function evaluations may be noisy. Relevant examples include hyperparameter tuning [1], experiment design [2], simulated-based parameter estimation [3]–[5], and automated calibration of complex multivariable control structures [6]–[8]. Bayesian optimization (BO) is a family of machine learning-based optimization algorithms specifically developed to address these challenging problems [9], [10]. The key to the success of BO compared to alternative derivative-free optimization methods is its sample-efficiency. The classical BO algorithm comprises two major steps: (i) the construction of a probabilistic surrogate model of the objective function from all currently available data and (ii) the optimization over some utility metric defined in terms of this surrogate model to decide where the objective function should be evaluated next. Through proper design of the utility (also called acquisition) function, one may achieve an adequate balance between *exploration* (i.e., querying in regions where the model predictions are most uncertain) and *exploitation* (i.e., querying in regions where the model is confident the solutions are good).

Classical BO algorithms are greedy in nature in the sense that they ignore how the design selected at the current iteration will affect future optimization steps. This type of strategy is well-known to lead to many aggressive "jumps" in the search space since BO may select a sample to explore an uncertain region and then immediately jump back to a more promising region with no regard for the distance between consecutive query points. This can become a major issue, especially in online experimental systems that have an associated cost with frequent input changes. For example, BO has recently proven useful for automatically assigning reference inputs for multiple actuators to minimize operational energy use [11]. However, when the reference inputs are suddenly changed to newly suggested values that are far apart in a norm-sense, this can lead to violation of heat pump states due to escalated temperature transients. Therefore, in such situations, it is important to modify the standard BO problem setting to include what we refer to in this work as *local search constraints*, which restrict the sequential selection of input points to lie within some specified region.

Although many constrained BO (CBO) methods have been developed, to the best of our knowledge, this work is the first to investigate local search constraints in BO. A comprehensive overview of CBO is provided in [3] – the focus of such methods is on black-box constraint functions that require the construction of additional surrogate models. `TuRBO` [12] is a trust region-based BO algorithm that restricts the movement of next sample in a given iteration; however, these constraints are added to limit the region over which the probabilistic surrogate model must be constructed. `TuRBO` is not directly applicable to problems with local search constraints, because these constraints can be violated when the diameter of the trust region exceeds that of the feasible region. `SnAKe` [13] is a recently proposed BO algorithm that aims to minimize input change costs by building an ordered optimization path from a set of randomly drawn Thompson samples of the probabilistic surrogate model. However, `SnAKe` is unable to ensure feasible solutions in our problem setting since the objective samples may not produce solutions that satisfy the local search constraints.

In this work, we propose a novel *multi-step lookahead* BO strategy that takes into account the remaining function evaluations while also guaranteeing satisfaction of the local search constraints. To construct such a strategy, we must first define the notion of an "optimal policy" that maximizes the long-term reward over multiple future steps. Building upon

J. Paulson and F. Sorouifar are with the Department of Chemical and Biomolecular Engineering, The Ohio State University, Columbus, OH 43210, USA. {sorourifar.1,paulson.82}@osu.edu

A. Chakrabarty is with Mitsubishi Electric Research Laboratories, Cambridge, MA 02139, USA. achakrabarty@ieee.org

the work in [14], we can characterize the optimal policy as the solution to a dynamic programming (DP) problem derived from Bellman's principle of optimality. However, in the context of lookahead BO problems, the DP procedure is intractable due to the uncountable state and action spaces, along with potentially long decision horizons. Approximate dynamic programming (ADP) methods are often used to overcome these limitations in practice [15]. A variety of non-myopic BO policies have been recently developed by applying different ADP methods to the lookahead BO problem [16]–[18]. A variant of policy iteration-based ADP uses the *rollout* algorithm [19, Chapter 2], which uses a *base policy* to evaluate the long-term benefits using several stochastic simulations over multiple future steps. The choice of the base policy can have a substantial impact on the performance of the rollout algorithm. Therefore, the key contribution of this work is to propose an efficient and effective base policy in the context of lookahead BO with local search constraints. As we show in our numerical example, this can provide substantial performance improvements over greedy BO methods.

In summary, our **major contributions** include: (i) we cast the BO with local search constraints as a Markov decision process (MDP), whose optimal policy we characterize exactly using DP; (ii) we show how to efficiently compute non-myopic rollouts for long decision horizons using a Monte Carlo integration algorithm developed by employing the re-parameterization trick for Gaussian random variables; and, (iii) we provide a novel, efficient acquisition function which trades off local and global search, and therefore, can be used as a powerful base policy required by some ADP methods.

The remainder of this paper is organized as follows: We formally present our problem setting in Section II. We summarize the optimal policy conditions and the rollout approximation in Section III. In Section IV, we introduce our new base policy and discuss its efficient implementation. A numerical example is presented in Section V. Finally, we conclude the paper and discuss some interesting directions for future work in Section VI.

II. Problem Formulation

We consider the following black-box optimization problem

$$\min_{x\in\mathcal{X}} \; f(x), \tag{1}$$

where $x \in \mathcal{X}$ are the decision variables that are restricted to a compact optimization domain $\mathcal{X} \subset \mathbb{R}^d$ and $f : \mathcal{X} \to \mathbb{R}$ is the unknown objective function that is defined in terms of an expensive-to-evaluate experiment (could be a physical experiment, a high-fidelity simulation, or a combination). We assume that the objective function can be evaluated, either by measurement or some estimation procedure, once the experiment is completed. Thus, our goal is to find a point $x \in \mathcal{X}$ with the smallest possible objective value by querying f at a *sequence* of points $\{x_k\}_{k=1}^N$ over a finite budget of N function evaluations (total experiments).

Since a mathematical representation of the objective is completely unknown, we endow f with a prior probability distribution, denoted by p, which is formally defined in Section III. This setting matches that of traditional Bayesian optimization (BO), which determines the next informative sampling point x_{k+1} given the complete set of current observations $\mathcal{D}_k$ by solving an auxiliary problem of the form

$$x_{k+1} = \operatorname*{argmax}_{x\in\mathcal{X}} \; \alpha_{k+1}(x \mid \mathcal{D}_k), \tag{2}$$

where $\alpha_{k+1} : \mathcal{X} \to \mathbb{R}$ denotes the utility (or acquisition) function, which is defined in terms of the current posterior distribution for $f \sim p(f \mid \mathcal{D}_k)$ conditioned on the observations $\mathcal{D}_k$. The main idea here is that, since the optimization runtime is dominated by the evaluations of f, we should be willing to dedicate time and effort to selecting the most informative points to evaluate. The exact definition of "informative" depends on the chosen auxiliary problem (see [9] for a comparison between different acquisition functions). An important issue that often arises in practice when sequentially solving (2) is that subsequent iterations x_k and x_{k+1} may be very far apart, which leads to significant practical challenges, especially when performing online optimization in experimental systems. Practical examples of such situations are discussed in the introduction.

We aim to address this challenge by assuming there is a restriction on the selection of the next sample that can be represented by a collection of constraints of the form

$$x_{k+1} \in \mathcal{T}_k(x_k), \quad \forall k \in \{0, \dots, N-1\}, \tag{3}$$

where

$$\mathcal{T}_k(x_k) = \{x_{k+1} \in \mathcal{X} : c_k(x_k, x_{k+1}) \leq 0\},$$

is a set that depends on the previous x_k and is defined in terms of a *known* function $c_k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^{n_{c_k}}$. At first glance, this issue may appear easy to handle by simply imposing the constraint $x_{k+1} \in \mathcal{T}_k(x_k)$ when solving (2); however, commonly-used acquisition functions lead to "greedy" optimization strategies in the sense that they are oblivious to the remaining number of objective function evaluations. In addition, as we demonstrate in Section V, there are cases where all feasible points have equal utility such that the resulting sampling policy is prone to getting stuck in regions of $\mathcal{X}$ that yield suboptimal objective values.

Therefore, our *objective* is to develop a method that can systematically account for the remaining budget to plan out the sequence $\{x_k\}_{k=1}^N$ in a non-myopic fashion. We can devise such a method by first formulating the problem within a dynamic programming (DP) framework, which is discussed in the next section. Note that we assume that the initial data set $\mathcal{D}_0$ could potentially comprise samples from the global search space $\mathcal{X}$. This is because the initial dataset can be generated offline, and offline experiments can be reset, and therefore do not necessarily have to satisfy (3).

III. Approximate Dynamic Programming for Finite-Budget Bayesian Optimization

In this section, we first provide an overview of Gaussian process (GP) models that are used to as a probabilistic surrogate for the unknown objective function f. We then discuss

how the non-myopic BO framework can be interpreted as an instance of a Markov decision process (MDP) such that the optimal lookahead BO policy can be characterized as a DP instance. Lastly, since the optimal policy is intractable in practice due to the uncountable state and action spaces, we discuss approximate DP methods that can be used to develop more tractable, non-myopic policies.

A. Gaussian process regression

We model the black-box objective function f as the realization of a Gaussian process (GP), which is an infinite collection of random variables, any subset of which has a joint Gaussian distribution [20]. As such, a GP represents a distribution over functions $f(x) \sim \mathcal{GP}(m(x), \kappa(x, x'))$ specified by its mean $m(x)$ and covariance $\kappa(x, x')$ functions. The prior GP distribution is defined through an initial choice of $m(x)$ and $\kappa(x, x')$ that, in practice, depend on hyperparameters that must be calibrated to data by maximizing the log marginal likelihood (see, e.g., [20] for details).

Not only are GPs non-parametric models, they have simple analytic expressions for the posterior mean and covariance conditioned on the current training set $\mathcal{D}_k = \{(x_i, y_i)\}_{i=1}^k$ that consists of k potentially noisy observations $y_i = f(x_i) + \epsilon_i$ at a corresponding point x_i, with $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ being an i.i.d. Gaussian noise term for all $i = 1, \ldots, k$. In particular, the posterior $f(x) \mid \mathcal{D}_k \sim \mathcal{N}(\mu_k(x), \sigma_k^2(x))$ at any future test point $x \in \mathcal{X}$ is Gaussian with the following posterior mean $\mu_k(x)$ and posterior variance $\sigma_k^2(x)$

$$\begin{aligned} \mu_k(x) &= m(x) + \boldsymbol{k}_k^\top(x)(\boldsymbol{K}_k + \sigma^2 I_k)^{-1}(Y_k - M_k), \\ \sigma_k^2(x) &= \kappa(x, x) - \boldsymbol{k}_k^\top(x)(\boldsymbol{K}_k + \sigma^2 I_k)^{-1}\boldsymbol{k}_k(x), \end{aligned} \tag{4}$$

where $Y_k = [y_1, \ldots, y_k]^\top$, $M_k = [m(x_1), \ldots, m(x_k)]^\top$, $\boldsymbol{k}_k(x) = [\kappa(x_1, x), \ldots, \kappa(x_k, x)]^\top$, and $\boldsymbol{K}_k$ is a $k \times k$ matrix whose ij^{th} entry is $\kappa(x_i, x_j)$.

B. Markov decision process formulation

We can view the BO problem with local search constraints as an instance of N-stage DP by casting it as a finite-horizon MDP, which are expressed in terms of a chosen "state" space and "action" space. The state of our MDP at iteration k is $\mathcal{D}_k$. The action of our MDP at iteration k is x_{k+1}, which is the location of our next sample of the objective function. The state of our system is then defined recursively as follows

$$\mathcal{D}_{k+1} = \mathcal{D}_k \cup \{(x_{k+1}, y_{k+1})\}, \tag{5}$$

where y_{k+1} denotes a predicted value of the objective function at x_{k+1} generated according to the available probabilistic model $p(f(x_{k+1}) \mid \mathcal{D}_k)$. Based on our learned GP model for the objective function, we can represent the simulated value of y_{k+1} as follows

$$y_{k+1} \sim \mathcal{N}\left(\mu_k(x_{k+1}), \sigma_k^2(x_{k+1})\right). \tag{6}$$

Consequently, we can use the re-parameterization trick [21] to express y_{k+1} in terms of an effective stochastic disturbance $w_{k+1} \sim \mathcal{N}(0, 1)$ such that the simulated objective value only depends on the current system state $\mathcal{D}_k$ and the current action x_{k+1}; concretely,

$$y_{k+1} = \mu_k(x_{k+1}) + \sigma_k(x_{k+1})w_{k+1}.$$

Based on the structure of the imposed local constraints (3) and stochastic dynamics (5), the resulting system satisfies the *Markov property* [22, Section 3.6], i.e., the predicted state distribution and constraints are conditionally independent of previous state values given the current state.

C. Dynamic programming for BO

In ADP parlance, let $\boldsymbol{\pi} = \{\pi_1, \pi_2, \ldots, \pi_N\}$ denote a *policy* that represents a sequence of functions π_k. Each π_k maps from sets of available observations to feasible points in the search space that satisfy the local search constraints, i.e., $x_k = \pi_k(\mathcal{D}_{k-1}) \in \mathcal{T}_{k-1}(x_{k-1})$. To define an optimal policy, we need an overall reward function to evaluate the quality of different policies. In the context of the original problem (1), we define the *utility* generated by a state $\mathcal{D}_k$ to be the negative of the minimum observed objective function value

$$u(\mathcal{D}_k) = -\min_{(x,y)\in\mathcal{D}_k} y. \tag{7}$$

This utility function encodes the fact that points with small objective values are desirable regardless of where it is placed in the observed data sequence.

Given a utility function u, an initial set of observations $\mathcal{D}_0$, and a policy $\boldsymbol{\pi}$, we define the corresponding *value function* as the expected increase in utility

$$V_{\boldsymbol{\pi}}(\mathcal{D}_0) = \mathbb{E}\left[\textstyle\sum_{k=1}^N r(\mathcal{D}_{k-1}, \mathcal{D}_k)\right], \tag{8}$$

where the expectation operator is defined with respect to future observations $\mathcal{D}_1, \ldots, \mathcal{D}_N$ (or equivalently $w_1, \ldots, w_N$) and $r(\mathcal{D}_{k-1}, \mathcal{D}_k) = u(\mathcal{D}_k) - u(\mathcal{D}_{k-1})$ is the utility increase between stages. The optimal policy $\boldsymbol{\pi}^\star$ is the one that maximizes the expected utility over the set of feasible policies Π satisfying the local search constraints (3). Formally,

$$V^\star(\mathcal{D}_0) = V_{\boldsymbol{\pi}^\star}(\mathcal{D}_0) = \max_{\boldsymbol{\pi}\in\Pi} V_{\boldsymbol{\pi}}(\mathcal{D}_0), \tag{9}$$

where $V^\star$ denotes the optimal value function.

Using Bellman's principle of optimality [15], we can recursively compute $V^\star$ using the following DP algorithm

$$V_k(\mathcal{D}) = \max_{x^+\in\mathcal{T}_{N-k}(x)} \mathbb{E}_{\mathcal{D}^+}\left[r(\mathcal{D}, \mathcal{D}^+) + V_{k-1}(\mathcal{D}^+)\right], \tag{10}$$

for all $k = 1, \ldots, N$ starting from an initial value of $V_0(\mathcal{D}) = 0$, where $V_k(\mathcal{D})$ denotes the optimal value function for the tail subproblem defined in terms of k remaining stages and $\mathcal{D}^+ = \mathcal{D} \cup \{x^+, y^+\}$ is the successor set of observations. The optimal value function (9) then corresponds exactly to $V_N(\mathcal{D}_0)$. Furthermore, we can say that, if we always select the value of x^+ that maximizes the right-hand side of (10) for all k and all $\mathcal{D}$, then the resulting policy must be equal to that of the optimal policy $\boldsymbol{\pi}^\star$ [15].

It is useful to express the right-hand side of (10) in terms of the *optimal Q-factors* (also known as the state-action value

function in the reinforcement learning literature [22])

$$Q_k(x \mid \mathcal{D}) = \mathbb{E}_y\Big[r(\mathcal{D}, \mathcal{D} \cup \{(x,y)\}) + V_{k-1}(\mathcal{D} \cup \{(x,y)\})\Big], \tag{11}$$

which represents the optimal value of the tail sub-problem when starting at state $\mathcal{D}$ and taking action x at iteration $N-k$ and following an optimal policy for all remaining steps. From (10) and (11), we can infer the following relationship:

$$V_k(\mathcal{D}_{N-k}) = \max_{x \in \mathcal{T}_{N-k}(x_{N-k})} Q_k(x \mid \mathcal{D}_{N-k}), \tag{12}$$

between the value function and the Q-factor. We can now more straightforwardly define the optimal policy

$$\pi_k^\star(\mathcal{D}_{k-1}) = \operatorname*{argmax}_{x \in \mathcal{T}_{k-1}(x_{k-1})} Q_{N-k+1}(x \mid \mathcal{D}_{k-1}), \tag{13}$$

in terms of the Q-factor, for all $k = 1, \ldots, N$.

D. Rollout and direct policy search

Calculating the optimal non-myopic policy for BO requires knowledge of the Q-factors, which are recursively defined by the DP algorithm (10) and (11). However, (10) is intractable in our problem setting since it must be executed over an uncountable state space that grows in dimension by $d+1$ at every iteration. The action space is also uncountable, though defined over a fixed dimension, which makes handling the nested constrained maximization problems and expectation operators impossible to carry out exactly.

To find a tractable method of solving this problem, we rely on approximate DP (ADP) methods that have shown promising results on a variety of practical stochastic optimal control problems with similar characteristics to BO. Concretely, we employ the rollout algorithm (c.f. [19, Section 2.4]) and generate stochastic simulations to approximate Q_{k-1} or V_{k-1} over multiple future time steps [23]. The main computational advantage of the rollout algorithm is that it relaxes the requirement that future candidate points be selected optimally by using a suboptimal heuristic policy to decide which action to take in a given state. The choice of the heuristic is notoriously problem-dependent. Recent work has suggested to use existing greedy BO policies as the heuristic [14], [18]. However, as discussed in Section II, such greedy strategies do not adequately handle local search constraints (3). To address this challenge, we introduce a novel heuristic *base policy* in the next section. Before that, we summarize the rollout procedure and discuss how to further improve its computational efficiency using direct policy search.

To derive the rollout updates, we first define a heuristic base policy $\tilde{\boldsymbol{\pi}} = \{\tilde{\pi}_1, \ldots, \tilde{\pi}_N\}$ that may differ from the optimal policy $\boldsymbol{\pi}^\star$. Given such a base policy $\tilde{\boldsymbol{\pi}}$, we let $V_k^{\tilde{\boldsymbol{\pi}}}$ denote its corresponding value function for k remaining stages, which is described by the recursion for $k = 1, \ldots, N$

$$V_k^{\tilde{\boldsymbol{\pi}}}(\mathcal{D}) = \mathbb{E}_{\mathcal{D}^+}\left[r(\mathcal{D}, \mathcal{D}^+) + V_{k-1}^{\tilde{\boldsymbol{\pi}}}(\mathcal{D}^+)\right], \tag{14}$$

where $\mathcal{D}^+ = \mathcal{D} \cup \{(\tilde{\pi}_{N-k+1}(\mathcal{D}), y_{N-k+1})\}$ and $V_0^{\tilde{\boldsymbol{\pi}}}(\mathcal{D}) = 0$.

Although (14) bears resemblance to (10), there are some important differences. The biggest difference is that (14) can be evaluated in a forward fashion since the base policy can be evaluated for any given system state. Even though this has reduced some of the complexity of the DP algorithm, we still need to make two additional approximations to derive a tractable policy. First, we use a shortened horizon $h \leq N$ to limit the number of steps that we need to simulate forward in time. Second, we need to approximate the expectation with some type of numerical method such as quadrature or Monte Carlo (MC) integration. Due to the nested structure of the expectation, it would require an exponentially growing number of quadrature points at each stage such that we rely on MC to avoid this growth. We can then define the rollout policy in a similar manner to (13) as follows:

$$\pi_k^{\text{roll}}(\mathcal{D}_{k-1}) = \operatorname*{argmax}_{x_k \in \mathcal{T}_{k-1}(x_{k-1})} \tilde{Q}_h^{\tilde{\boldsymbol{\pi}}}(x_k \mid \mathcal{D}_{k-1}), \tag{15}$$

where $\tilde{Q}_h^{\tilde{\boldsymbol{\pi}}}$ is an approximate Q-factor (for only h steps ahead) constructed using MC. Although computing $\tilde{Q}_h^{\tilde{\boldsymbol{\pi}}}$ is tractable, it is not simple to numerically optimize.

A simpler alternative, often referred to as policy search, is to parameterize the base policy $\tilde{\boldsymbol{\pi}}_\theta$ in terms of some unknown parameters $\theta \in \Theta$ and then directly optimize over this newly defined space. We focus on time-invariant base polices of the form $\tilde{\boldsymbol{\pi}}_\theta = \{\tilde{\pi}_\theta, \ldots, \tilde{\pi}_\theta\}$ as this reduces the dimensionality of the policy search optimization problem, though, in principle, any parametrized base policy can be used. Using this idea, we define the following policy

$$\pi_k^{\text{ps}}(\mathcal{D}_{k-1}) = \operatorname*{argmax}_{\theta \in \Theta} \tilde{V}_h^{\tilde{\pi}_\theta}(\mathcal{D}_{k-1}), \tag{16}$$

where $\tilde{V}_h^{\tilde{\pi}_\theta}(\mathcal{D})$ is an approximated value function evaluated at a given state $\mathcal{D}$ and policy $\tilde{\pi}_\theta$ for an h-step problem. The MC evaluation procedure for $\tilde{V}_h^{\tilde{\pi}_\theta}(\mathcal{D})$ is summarized below

$$\tilde{V}_h^{\tilde{\pi}_\theta}(\mathcal{D}) = \frac{1}{S}\sum_{j=1}^{S}\sum_{i=0}^{h-1} r(\mathcal{D}_i^{(j)}, \mathcal{D}_{i+1}^{(j)}), \tag{17}$$

where

$$\mathcal{D}_{i+1}^{(j)} = \mathcal{D}_i^{(j)} \cup \{(x_{i+1}^{(j)}, y_{i+1}^{(j)})\}, \quad \mathcal{D}_0^{(j)} = \mathcal{D}, \tag{18a}$$

$$x_{i+1}^{(j)} = \tilde{\pi}_\theta(\mathcal{D}_i^{(j)}), \tag{18b}$$

$$y_{i+1}^{(j)} = \mu_i^{(j)}(x_{i+1}^{(j)}) + \sigma_i^{(j)}(x_{i+1}^{(j)}) w_{i+1}^{(j)}, \tag{18c}$$

S is the number of MC samples, and $\{w_1^{(j)}, \ldots, w_h^{(j)}\}_{j=1}^S$ are the complete set of uncertainty samples, with each element being drawn from a standard normal distribution. When Θ is selected such that we only need to consider a finite set of policies, then (16) can be easily evaluated by simulating (17) for each $\theta \in \Theta$ and selecting the one with the highest h-step value. As a result, our proposed approach is significantly faster than the DP method or traditional rollout ADP, though this may come with a reduction in performance.

IV. An Efficient Base Policy for Global Optimization with Local Search Constraints

We are now in a position to define our novel base policy as the solution to the following optimization problem

$$\tilde{\pi}_\theta(\mathcal{D}_k) = \operatorname*{argmax}_{x \in \mathcal{T}_k(x_k)} Q_1(x \mid \mathcal{D}_k) - \theta \|x - x_{k+1}^{\text{global}}(\mathcal{D}_k)\|, \tag{19}$$

Algorithm 1 Lookahead BO with Local Search Constraints

Input: The optimization domain $\mathcal{X}$, initial data $\mathcal{D}_0$, GP prior m and κ, total number of iterations N, rollout horizon h, number of MC samples S, and policy parameter set Θ.

1: **for** $k = 1$ to N **do**
2: Determine x_k by solving the policy search optimization problem (16) with approximate value function $\tilde{V}_h^{\tilde{\pi}_\theta}$ evaluated using the proposed base policy (19).
3: Evaluate the expensive objective function at the design suggested in the previous step to obtain a potentially noisy observation, i.e., $y_k = f(x_k) + \epsilon_k$.
4: Augment the dataset $\mathcal{D}_k = \mathcal{D}_{k-1} \cup \{(x_k, y_k)\}$, which can be used to update the GP posterior according to (4).
5: **end for**

where $\theta \geq 0$ is a non-negative scalar weight and $x_{k+1}^{\text{global}}(\mathcal{D}_k)$ corresponds to the predicted one-step optimal solution to the problem without local search constraints:

$$x_{k+1}^{\text{global}}(\mathcal{D}_k) = \underset{x \in \mathcal{X}}{\operatorname{argmax}}\, Q_1(x \mid \mathcal{D}_k). \tag{20}$$

Therefore, similarly to previous work [14], [18], we focus on a greedy policy in our rollout procedure for computational efficiency; however, our policy attempts to balance two disparate terms. The first term in (19), $Q_1(x \mid \mathcal{D}_k)$, is simply the one-step optimal acquisition function, which provides a one-step quantification of improvement in our local search region around the previous candidate x_k. The second term $\|x - x_{k+1}^{\text{global}}(\mathcal{D}_k)\|$, on the other hand, corresponds to the distance from the suggested one-step globally optimal solution (20), which indicates there is some inherent value to moving toward this point, which can potentially be sampled at some future iteration. Since we have two competing objectives, a natural way to balance them is to treat the problem as a multi-objective optimization problem that can be scalarized using a weight factor θ, as shown in (19). We propose using the policy search-based rollout algorithm (16) to adaptively select θ based on the predicted future performance.

For our approach to be computationally efficient, the optimization problems in (19) and (20) must be solved as fast as possible since they will be repeatedly called during the rollout procedure. To this end, we analytically reformulate the one-step Q-factor for the utility function in (7) as follows

$$\begin{aligned} Q_1(x_{k+1} \mid \mathcal{D}_k) &= \mathbb{E}_{y_{k+1}}\left[\max\{y_{k+1} - u(\mathcal{D}_k), 0\}\right], \\ &= \sigma_k(x_{k+1}) z_k(x_{k+1}) \Phi(z_k(x_{k+1})) + \phi(z_k(x_{k+1})), \end{aligned} \tag{21}$$

where $z_k(x) = (u(\mathcal{D}_k) - \mu_k(x))/\sigma_k(x)$ and Φ and ϕ denote the standard Gaussian cumulative density and probability density functions, respectively. It is interesting to note that the expression in (21) is equivalent to the classical expected improvement (EI) acquisition function proposed in [24], which is one of the most popular acquisition functions in traditional "greedy" BO. This is not very surprising since EI is known to be one-step optimal for the utility function (7).

We summarize the proposed method in Algorithm 1. We plan to formally analyze the convergence of this algorithm in future work; however, we provide some brief theoretical justification for its structure. In particular, if the solution to (20) also satisfies the local search constraints, i.e., $x_{k+1}^{\text{global}}(\mathcal{D}_k) \in \mathcal{T}_k(x_k)$, then the value of θ becomes irrelevant such that $x_{k+1}^{\text{global}}(\mathcal{D}_k) = \tilde{\pi}_\theta(\mathcal{D}_k)$ for all $\theta \in \Theta$. Additionally, as $\theta \to \infty$, then the solution (20) corresponds to the point in $\mathcal{T}_k(x_k)$ that is nearest to $x_{k+1}^{\text{global}}(\mathcal{D}_k)$ (in the norm sense). Assuming $x_{k+1}^{\text{global}}(\mathcal{D}_k)$ is roughly constant for i iterations into the future, then we would have that $x_{k+i} \approx x_{k+1}^{\text{global}}(\mathcal{D}_k)$ for some $i \geq 1$. Therefore, Algorithm 1 mimics (20) under certain conditions, which is important because the policy generated by (20) is guaranteed to converge to the global minima under suitable regularity conditions on the covariance function κ, as proven in [25, Theorem 7]. Even though this does not immediately imply convergence of our proposed approach, it does hint at a possible direction to achieve convergence. Furthermore, in practice, one cares about improved performance (lower objective values found in fewer iterations), which we demonstrate in a reproducible manner on a numerical example next.

V. Numerical Example

For our benchmark problem, we use a modified version of the Branin function over the domain $x \in [-5, 10] \times [0, 15]$. The true unknown function $f(x)$ is given by

$$f(x) = f_1(x) + f_2(x) + l_1(x) + l_2(x), \tag{22}$$

where

$$f_1(x) = a\left(x_2 - bx_1^2 + cx_1 - r\right)^2, \tag{23a}$$
$$f_2(x) = s(1-t)\cos(x_1) + s, \tag{23b}$$
$$l_1(x) = 5e^{-5\left((x_1+3.14))^2 + (x_2 - 12.27)^2\right)}, \tag{23c}$$
$$l_2(x) = 5e^{-5\left((x_1-3.14))^2 + (x_2 - 2.275)^2\right)}, \tag{23d}$$

are functions defined in terms of the following constants $a = 1$, $b = 5.1/(4\pi^2)$, $c = 5/\pi$, $r = 6$, $s = 10$, $t = 1/(8\pi)$. Note that $f_1(x) + f_2(x)$ corresponds to the standard Branin function (that has 3 global minima), while $l_1(x)$ and $l_2(x)$ are modifications developed to convert the 2 global minima at $(-3.14, 12.27)$ and $(3.14, 2.275)$ to local minima. Thus, our modified Branin function (22) has a unique global minimum at $(9.42, 2.475)$. To increase the difficult of the problem, we assume that the selected sequence of query points $\{x_k\}_{k=1}^N$ must satisfy local search constraints of the form (3)

$$-0.75 \leq x_{1,k+1} - x_{1,k} \leq +0.75, \tag{24a}$$
$$-1.5 \leq x_{2,k+1} - x_{2,k} \leq +1.5. \tag{24b}$$

To implement our proposed lookahead BO strategy, summarized in Algorithm 1, we use a GP with a Mateŕn 5/2 prior covariance function κ. We use maximum likelihood estimation to learn the corresponding hyperparameters at every iteration (including the length scale for each dimension, the scale, and the noise variance) [20]. We randomly draw 10 points in $\mathcal{X}$ to construct the initial dataset $\mathcal{D}_0$ and always select x_0 to be the point that leads to the minimum objective value within this set. We chose $S = 20$ samples using Latin

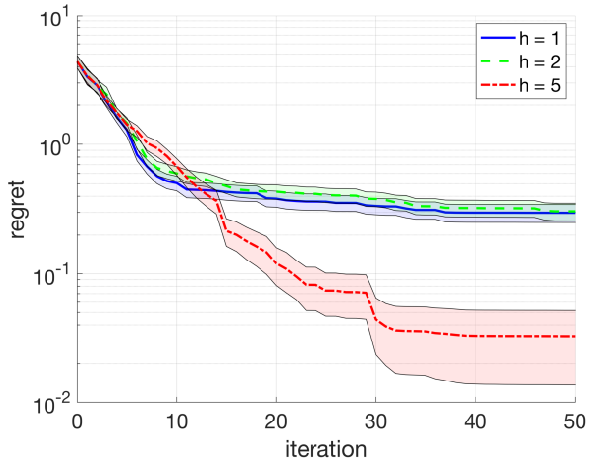


Fig. 1: Expected simple regret versus number of BO iterations for the modified Branin problem for different rollout horizons $h \in \{1, 2, 5\}$. Confidence intervals are shown in the form of error bars, which were estimated from 50 independent realizations of the initial random samples.

hypercube sampling and run the algorithm for a total of $N = 50$ iterations. To evaluate the proposed base policy in (19), we use a multi-start L-BFGS-B algorithm [26] with 10 restarts selected by evaluating Q_1 on a Latin hypercube of 1000 points and picking the best 10 as our initial conditions. We use `CasADi` [27] to efficiently compute the first-order derivatives required by L-BFGS-B. To perform the policy search step (16), we only consider two discrete values of θ that leads to either a pure local or global step for simplicity.

We compare Algorithm 1's performance on three different rollout horizons $h \in \{1, 2, 5\}$. The case $h = 1$ corresponds to traditional greedy BO using an EI acquisition function where we can only search over the feasible region defined by the local search constraints. We use simple regret as our performance metric, which is defined as follows

$$\text{Regret}_k(\mathcal{D}_0) = \min_{i=1,\dots,k} y_k - f^\star, \tag{25}$$

where $f^\star = \min_{x\in\mathcal{X}} f(x)$ is the true global solution. Since the initial dataset $\mathcal{D}_0$ is random, we repeat each algorithm 50 times to obtain an estimate of the average simple regret $\mathbb{E}_{\mathcal{D}_0}[\text{Regret}(\mathcal{D}_0)]$ along with confidence intervals estimated by one standard deviation divided by the square root of the number of replications. The results of the empirically estimated average simple regret versus the number of iterations N is shown in Fig. 1. We see the rollout case with $h = 5$ considerably outperforms the traditional greedy BO approach ($h = 1$) and the short-sighted BO approach ($h = 2$) after around 12 iterations. Furthermore, this long-term improvement in performance does not come at a large cost in short-term performance since $h = 5$ still shows a considerable reduction in (average) regret during the first 10 iterations. This is a direct consequence of the proposed base policy (19), which captures both local and global information.

To better understand the empirical performance improvements observed in Fig. 1, we have plotted the design policy sequences generated by the proposed algorithm with $h \in \{1, 2, 5\}$ in Fig. 2. The greedy BO policy (Fig. 2a) essentially gets stuck near $u(\mathcal{D}_0)$, which is the best observed sample in the initial dataset because there is no room for improvement in the one-step reachable zone (shown as magenta squares) once the neighboring local minimum is found. By increasing the rollout horizon $h = 2$ (Fig. 2b), we see the policy is able to pull away from the nearest local minimum by following a path of reasonable improvement. However, it quickly gets trapped by the other local minimum and shows a similar behavior to the greedy BO algorithm at that point. The $h = 5$ case (Fig 2c), on the other hand, shows a stronger preference for globally exploring the objective function surface such that it is able to quickly move toward the true global minimum after a few steps of local improvement. It is important to note that the optimization path shown in Fig 2c differs from that generated by (19) with $\theta = \infty$ since the rollout algorithm allows us to adaptively choose between different θ values at each step. In this way, we achieve a truly hybrid strategy that can more adeptly handle the exploration-exploitation tradeoff in the presence of local search constraints.

VI. Conclusions

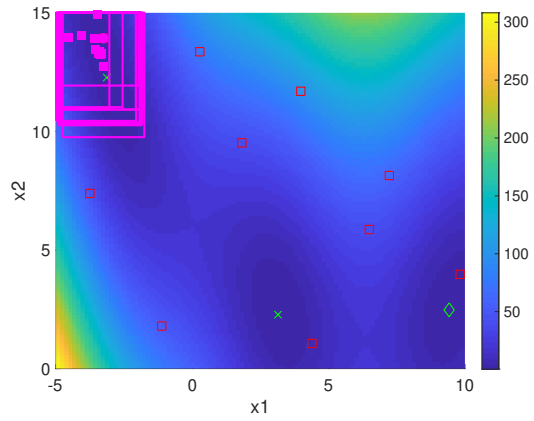
This paper develops a variant of Bayesian optimization (BO) for global optimization problems with expensive objective functions and local search constraints imposed on the chosen sequence of query points. These local search constraints are important for limiting the input cost of classical BO methods, which result in aggressive jumps in the search space that can cause problems in online experimental systems such as controller auto-tuning. We formulate a non-myopic BO procedure as a dynamic programming (DP) problem, which is intractable to solve in its natural form in part due to having an uncountable state space. We take advantage of rollout combined with fast policy search to overcome this computational challenge, which are well-known approximate DP methods. In addition, we propose a novel base policy that incorporates the local search constraints, which is needed to run the rollout algorithm in order to on-the-fly approximate the value function at the current state. In future work, we intend to implement this algorithm on an industrial controller auto-tuning problem related to energy-use minimization as well as develop improved base policy heuristics for which we can directly establish convergence.

References

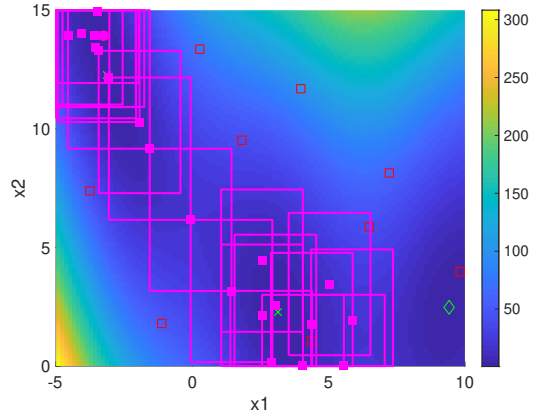
[1] J. Wu, S. Toscano-Palmerin, P. I. Frazier, and A. G. Wilson, "Practical multi-fidelity bayesian optimization for hyperparameter tuning," in *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, vol. 115 of *Proceedings of Machine Learning Research*, pp. 788–798, PMLR, 22–25 Jul 2020.

[2] A. M. Schweidtmann, A. D. Clayton, N. Holmes, E. Bradford, R. A. Bourne, and A. A. Lapkin, "Machine learning meets continuous flow chemistry: Automated optimization towards the Pareto front of multiple objectives," *Chemical Engineering Journal*, vol. 352, pp. 277–282, 2018.

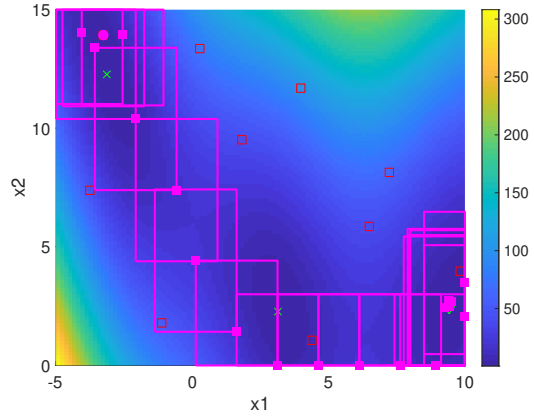
[3] J. A. Paulson and C. Lu, "COBALT: COnstrained Bayesian optimizAtion of computationaLly expensive grey-box models exploiting derivaTive information," *Computers & Chemical Engineering*, vol. 160, p. 107700, 2022.



(a) $h = 1$



(b) $h = 2$



(c) $h = 5$

Fig. 2: The selected samples over 20 iterations using the proposed non-myopic BO strategy as a function of the rollout horizon h. The contour plots correspond to the (unknown) true objective function in (22), the red $\square$ correspond to the initial set of samples, the green $\times$ correspond to the two local minima, the green $\diamond$ corresponds to the global minimum, the magenta $\square$ correspond to the samples selected by the BO policy, and the magenta box regions correspond to the local search constraints in (24).

[4] A. Chakrabarty, E. Maddalena, H. Qiao, and C. Laughman, "Scalable bayesian optimization for model calibration: Case study on coupled building and HVAC dynamics," *Energy and Buildings*, vol. 253, p. 111460, 2021.

[5] A. Chakrabarty, S. A. Bortoff, and C. R. Laughman, "Simulation failure robust bayesian optimization for estimating black-box model parameters," in *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 1533–1538, IEEE, 2021.

[6] D. Piga, M. Forgione, S. Formentin, and A. Bemporad, "Performance-oriented model learning for data-driven MPC design," *IEEE control systems letters*, vol. 3, no. 3, pp. 577–582, 2019.

[7] J. A. Paulson and A. Mesbah, "Data-driven scenario optimization for automated controller tuning with probabilistic performance guarantees," *IEEE Control Systems Letters*, vol. 5, no. 4, pp. 1477–1482, 2021.

[8] Q. Lu, L. D. González, R. Kumar, and V. M. Zavala, "Bayesian optimization with reference models: A case study in MPC for HVAC central plants," *Computers & Chemical Engineering*, vol. 154, p. 107491, 2021.

[9] 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*, vol. 104, no. 1, pp. 148–175, 2015.

[10] P. I. Frazier, "A tutorial on Bayesian optimization," *arXiv preprint arXiv:1807.02811*, 2018.

[11] A. Chakrabarty, C. Danielson, S. A. Bortoff, and C. R. Laughman, "Accelerating self-optimization control of refrigerant cycles with Bayesian optimization and adaptive moment estimation," *Applied Thermal Engineering*, vol. 197, p. 117335, 2021.

[12] D. Eriksson, M. Pearce, J. Gardner, R. D. Turner, and M. Poloczek, "Scalable global optimization via local Bayesian optimization," in *Advances in Neural Information Processing Systems*, pp. 5496–5507, 2019.

[13] J. P. Folch, S. Zhang, R. M. Lee, B. Shafei, D. Walz, C. Tsay, M. van der Wilk, and R. Misener, "SnAKe: Bayesian optimization with pathwise exploration," *arXiv preprint arXiv:2202.00060*, 2022.

[14] R. Lam, K. Willcox, and D. H. Wolpert, "Bayesian optimization with a finite budget: An approximate dynamic programming approach," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[15] D. Bertsekas, *Dynamic programming and optimal control: Volume I*, vol. 1. Athena Scientific, 2012.

[16] S. Jiang, H. Chai, J. Gonzalez, and R. Garnett, "BINOCULARS for efficient, nonmyopic sequential experimental design," in *International Conference on Machine Learning*, pp. 4794–4803, PMLR, 2020.

[17] S. Jiang, D. Jiang, M. Balandat, B. Karrer, J. Gardner, and R. Garnett, "Efficient nonmyopic Bayesian optimization via one-shot multi-step trees," *Advances in Neural Information Processing Systems*, vol. 33, pp. 18039–18049, 2020.

[18] E. Lee, D. Eriksson, D. Bindel, B. Cheng, and M. Mccourt, "Efficient rollout strategies for Bayesian optimization," in *Conference on Uncertainty in Artificial Intelligence*, pp. 260–269, PMLR, 2020.

[19] D. Bertsekas, *Reinforcement Learning and Optimal Control*. Athena Scientific Optimization and Computation Series, Athena Scientific, 2019.

[20] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*. Cambridge, MA: MIT Press, 2006.

[21] D. P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," in *Proc. of NeurIPS*, (Cambridge, MA, USA), p. 2575–2583, MIT Press, 2015.

[22] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT Press, 2018.

[23] D. P. Bertsekas, "Dynamic programming and suboptimal control: A survey from ADP to MPC," *European Journal of Control*, vol. 11, no. 4-5, pp. 310–334, 2005.

[24] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[25] E. Vazquez and J. Bect, "Convergence properties of the expected improvement algorithm with fixed mean and covariance functions," *Journal of Statistical Planning and inference*, vol. 140, no. 11, pp. 3088–3095, 2010.

[26] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Mathematical programming*, vol. 45, no. 1, pp. 503–528, 1989.

[27] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "Casadi: A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.