Lookahead Bayesian Optimization with Inequality Constraints

Authored by:

Remi Lam
Karen Willcox

Abstract

We consider the task of optimizing an objective function subject to inequality constraints when both the objective and the constraints are expensive to evaluate. Bayesian optimization (BO) is a popular way to tackle optimization problems with expensive objective function evaluations, but has mostly been applied to unconstrained problems. Several BO approaches have been proposed to address expensive constraints but are limited to greedy strategies maximizing immediate reward. To address this limitation, we propose a lookahead approach that selects the next evaluation in order to maximize the long-term feasible reduction of the objective function. We present numerical experiments demonstrating the performance improvements of such a lookahead approach compared to several greedy BO algorithms, including constrained expected improvement (EIC) and predictive entropy search with constraint (PESC).

1 Paper Body

Constrained optimization problems are often challenging to solve, due to complex interactions between the goals of minimizing (or maximizing) the objective function while satisfying the constraints. In particular, non-linear constraints can result in complicated feasible spaces, sometimes partitioned in disconnected regions. Such feasible spaces can be difficult to explore for a local optimizer, potentially preventing the algorithm from converging to a global solution. Global optimizers, on the other hand, are designed to tackle disconnected feasible spaces and optimization of multi-modal objective functions. Such algorithms typically require a large number of evaluations to converge. This can be prohibitive when the evaluation of the objective function or the constraints is expensive, or when there is a finite budget of evaluations allocated for the optimization, as it is often the case with expensive models. This evaluation budget typically results from resource scarcity such as the restricted availability of a high-performance computer, finite financial resources to build prototypes, or even time when working

on a paper submission deadline. Bayesian optimization (BO) [19] is a global optimization technique designed to address problems with expensive function evaluations. Its constrained extension, constrained Bayesian optimization (CBO), iteratively builds a statistical model for the objective function and the constraints. Based on this model that leverages all the past evaluations, a utility function quantifies the merit of evaluating any design under consideration. At each iteration, a CBO algorithm evaluates the expensive objective function and constraints at the design which maximizes this utility function. In most existing methods, the utility function only quantifies the reward obtained over the immediate next step, and ignores the gains that could be collected at future steps. This results in greedy CBO algorithms. However, quantifying long-term rewards may be beneficial. For instance, in the presence of constraints, it could be valuable to learn the boundaries of the feasible space. In order to do so, it is likely that an infeasible design would need to be evaluated, bringing no immediate improvement, 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

but leading to long-term benefits. Such strategy requires planning over several steps. Planning is also required to balance the so-called exploration-exploitation trade-off. Intuitively, in order to improve the statistical model, the beginning of the optimization should mainly be dedicated to exploring the design space, while the end of the optimization should focus on exploiting that statistical model to find the best design. To balance this trade-off in a principled way, the optimizer needs to plan ahead and be aware of the remaining evaluation budget. To address the shortcomings of greedy algorithms, we propose a new lookahead formulation for CBO with a finite budget. This approach is aware of the remaining budget and can balance the exploration-exploitation trade-off in a principled way. In this formulation, the best optimization policy sequentially evaluates the design yielding the maximum cumulated reward over multiple steps. This optimal policy is the solution of an intractable dynamic programming (DP) problem. We circumvent this issue by employing an approximate dynamic programming (ADP) algorithm: rollout, building on the unconstrained BO algorithm in [17]. Numerical examples illustrate the benefits of the proposed lookahead algorithm over several greedy ones, especially when the objective function is multi-modal and the feasible space has a complex topology. The next section gives an overview of CBO and discusses some of the related work (Sec. 2). Then, we formulate the lookahead approach to CBO as a dynamic programming problem and demonstrate how to approximately solve it by adapting the rollout algorithm (Sec. 3). Numerical results are provided in Sec. 4. Finally, we present our conclusions in Sec. 5.

2

Constrained Bayesian Optimization

We consider the following optimization problem: (OPc) x? = argmin f (x) x?X

s.t.

(1)

gi (x) ? 0, ?i ? {1, . . . , I},

where x is a d-dimensional vector of design variables. The design space X is a bounded subset of Rd , f : X 7? R is an objective function, I is the number of inequality constraints and gi : X 7? R is the ith constraint function. The functions f and gi are considered expensive to evaluate. We are interested in finding the minimizer x? of the objective function f subject to the non-linear constraints gi ? 0 with a finite budget of N evaluations. We refer to this problem as the original constrained problem (OPc). Constrained Bayesian optimization (CBO) addresses the original constrained problem (OPc) by modeling the objective function f and the constraints gi as realizations of stochastic processes. Typically, each expensive-to-evaluate function is modeled with an independent Gaussian process (GP). At every iteration n, new evaluations of f and gi become available and augment a training set Sn = {(xj , f (xj), g1 (xj), ? ? ? , gI (xj))}nj=1 . Using Bayes rule, the statistical model is updated and the posterior quantities of the GP, conditioned on Sn , reflect the current representation of the unknown expensive functions. In particular, for any design x, the posterior mean ?n (x; ?) and the posterior variance ? 2n (x; ?) of the GP associated with the expensive function ? ? {f, g1 , ? ? ? , gI } can be computed cheaply using a closed-form expression (see [24] for an overview of GP). CBO leverages this statistical model to quantify, in a cheap-to-evaluate utility function Un , the usefulness of any design under consideration. The next design to evaluate is then selected by solving the following auxiliary problem (AP): (AP)

xn+1 = argmax Un (x; Sn).

(2)

x?X

The vanilla CBO algorithm is summarized in Algorithm 1. Many utility functions have been proposed in the literature. To decide which design to evaluate next, [27] proposed the use of constrained expected improvement EIc , which, in the case of independent GPs, can be computed in closed-form as the product of the expected improvement (obtained by considering the GP associated with the objective function) and the probability of feasibility associated with each constraint. This approach was later applied to machine learning applications [6] and extended to the multi-objective case [5]. Note that this method transforms an original constrained optimization problem into an unconstrained auxiliary problem by modifying the utility function. Other attempts to cast the constrained problem into an unconstrained one include [3]. That work uses 2

Algorithm 1 Constrained Bayesian Optimization Input: Initial training set S1 , budget N for n = 1 to N do Construct GPs using Sn Update hyperparameters Solve AP for xn+1 = argmaxx?X Un (x; Sn) Evaluate f (xn+1), g1 (xn+1), ? ? ? , gI (xn+1) Sn+1 = Sn ? {(xn+1 , f (xn+1), g1 (xn+1), ? ? ? , gI (xn+1))} end for

a penalty method to transform the original constrained problem into an unconstrained problem, to which they apply a radial basis functions (RBF) method for global optimization (constrained RBF methods exist as well [25]). Other techniques from local constrained optimization have been leveraged in [10] where the utility function is constructed based on an augmented Lagrangian formulation. This technique was recently extended in [22] where a slack-variables

formulation allows the handling of equality and mixed constraints. Another approach is proposed by [1]: at each iteration, a finite set of candidate designs is first generated from a Latin hypercube, second, candidate designs with expected constraint violation higher than a user-defined threshold are rejected. Finally, among the remaining candidates, the ones achieving the best expected improvement are evaluated (several designs can be selected simultaneously at each iteration in this formulation). Another method [26] solves a constrained auxiliary optimization problem: the next design is selected to maximize the expected improvement subject to approximated constraints (the posterior mean of the GP associated with a constraint is used in lieu of the constraint itself). Note that the two previous methods solve a constrained auxiliary problem. Another method to address constrained BO is proposed by [11], who develop an integrated conditional expected improvement criterion. Given a candidate design, this criterion quantifies the expected improvement point-wise (conditioned on the fact that the candidate will be evaluated). This pointwise improvement is then integrated over the entire design space. In the unconstrained case, in the integration phase, equal weight is given to designs throughout the design space. The constrained case is addressed by defining a weight function that depends on the feasible probability of a design: improvement at designs that are likely to be infeasible have low weight. The probability of a design being feasible is calculated using a classification GP. The computation of this criterion is more involved as there is no closed-form formulation available for the integration and techniques such as Monte Carlo or Markov chain Monte Carlo must be employed. In a similar spirit, [21] introduces a utility function which quantifies the benefit of evaluating a design by integrating its effect over the design space. The proposed utility function computes the expected reduction of the feasible domain below the best feasible value evaluated so far. This results in the expected volume of excursion criteria which also requires approximation techniques to be computed. The former approaches revolve around computing a quantity based on improvement and require having at least one feasible design. Other strategies use information gain as the key element to drive the optimization strategy. [7] proposed a two-step approach for constrained BO when the objective and the constraints can be evaluated independently. The first step chooses the next location by maximizing the constrained EI [27], the second step chooses whether to evaluate the objective or a constraint using an information gain metric (i.e., entropy search [12]). [13, 14] developed a strategy that simultaneously selects the design to be evaluated and the model to query (the objective or a constraint). The criterion used, predictive entropy search with constraints (PESC), is an extension of predictive entropy search (PES) [15]. One of the advantages of information gain-based methods stems from the fact that one does not need to start with a feasible design. All aforementioned methods use myopic utilities to select the next design to evaluate, leading to suboptimal optimization strategies. In the unconstrained BO setting, multiple-steps lookahead algorithms have been explored [20, 8, 18, 9, 17] and were shown to improve the performance of BO. To our knowledge, such lookahead strategies for constrained optimization have not yet been addressed in the literature and

also have the potential to improve the performance of CBO algorithms. 3

3

Lookahead Formulation of CBO

In this section, we formulate CBO with a finite budget as a dynamic programming (DP) problem (Sec. 3.1). This leads to an optimal but computationally challenging optimization policy. To mitigate the cost of computing such a policy, we employ an approximate dynamic programming algorithm, rollout, and demonstrate how it can be adapted to CBO with a finite budget (Sec. 3.2). 3.1

Dynamic Programming Formulation

We seek an optimization policy which leads, after consumption of the evaluation budget, to the maximum feasible decrease of the objective function. Because the value of the expensive objective function and constraints are not known before their evaluations, it is impossible to quantify such long-term reward within a cheap-to-evaluate utility function Un . However, CBO endows the objective function and the constraints with a statistical model that can be interrogated to inform the optimizer of the likely values of f and gi at a given design. This statistical model can be leveraged to simulate optimization scenarios over multiple steps and quantify their probabilities. Using this simulation mechanism, it is possible to quantify, in an average sense, the long-term reward achieved under a given optimization policy. The optimal policy is the solution of the DP problem that we formalize now. Let n be the current iteration number of the CBO algorithm, and N the total budget of evaluations, or horizon. We refer to the future iterations of the optimization generated by simulation as stages. For any stage k ? {n, ? ? ? , N }, all the information collected is contained in the training set Sk . The function f and the I functions gi are modeled with independent GPs. Their posterior quantities, conditioned on Sk , fully characterize our knowledge of f and gi . Thus, we define the state of our knowledge at stage k to be the training set Sk ? Zk . Based on the training set Sk , the simulation makes a decision regarding the next design xk+1 ? X to evaluate using an optimization policy. An optimization policy ? = {?1 , ? ? ? , ?N } is a sequence of rules, ?k : Zk 7? X for k ? {1, ? ? ? , N }, mapping a training set Sk to a design xk+1 = ?k (Sk).

In the simulations, the values f (xk+1) and gi (xk+1) are unknown and are treated as uncertainties. f We model those I + 1 uncertain quantities with I + 1 independent Gaussian random variables Wk+1 gi and Wk+1 based on the GPs: f Wk+1 ? N (?k (xk+1 ; f), ? 2k (xk+1 ; f)),

gi Wk+1

? N (?k (xk+1 ; gi), ? 2k (xk+1 ; gi)), and ? 2k (xk+1 ; ?) are the posterior mean

(3) (4)

where we recall that ?k (xk+1 ; ?) and variance of the GP associated with any expensive function ? ? {f, g1 , ? ? ? , gI }, conditioned on Sk , at xk+1 . Then, the 1 I simulation generates an outcome. A simulated outcome wk+1 = (fk+1 , gk+1 , ? ? ? , gk+1)?W? f g gI 1 RI+1 is a sample of the (I + 1)-dimensional random variable Wk+1 = [Wk+1 , Wk+1 , ? ? ? , Wk+1].

Note that simulating an outcome does not require evaluating the expensive f and gi . In particular, i fk+1 and gk+1 are not f (xk+1) and gi (xk+1). 1 I Once an outcome wk+1 = (fk+1 , gk+1 , ? ? ? , gk+1) is simulated, the system transitions to a new state Sk+1 , governed by the system dynamic Fk : Zk ? X ? W 7? Zk+1 given by:

1 I Sk+1 = Fk (Sk , xk+1 , wk+1) = Sk ? {(xk+1 , fk+1 , gk+1 , ? ? ? , gk+1)). (5) Now that the simulation mechanism is defined, one needs a metric to assert the quality of a given optimization policy. At stage k, a stage-reward function rk : Zk ? X ? W 7? R quantifies the merit 1 I of querying a design if the outcome wk = (fk+1 , gk+1 , ? ? ? , gk+1) occurs. This stage-reward is defined as the reduction of the objective function satisfying the constraints: n o Sk rk (Sk , xk+1 , wk+1) = max 0, fbest ? fk+1 , (6) Sk i if gk+1 ? 0 for all i ? {1, ? ? ? , I}, and rk (?, ?, ?) = 0 otherwise, where fbest is the best feasible value at stage k. Thus, the expected (long-term) reward starting from training set Sn under optimization policy ? is: " N # X J? (Sn) = E rk (Sk , ?k (Sk), wk+1) , (7) k=n

4

where the expectation is taken with respect to the (correlated) simulated values (wn+1 , ? ? ? , wN +1), and the state evolution is governed by Eq. 5. An optimal policy, ? ? , is a policy maximizing this long-term expected reward in the space of admissible policies ?: J?? (Sn) = max J? (Sn).

(8)

???

The optimal reward J?? (Sn) is given by Bellman?s principle of optimality and can be computed using the DP recursive algorithm, working backward from k = N ? 1 to k = n, JN (SN) =

max E[rN (SN , xN +1 , wN +1)] =

xN +1 ?X

max EIc (xN +1 ; SN)

xN +1 ?X

Jk (Sk) = max E[rk (Sk , xk+1 , wk+1) + Jk+1 (Fk (Sk , xk+1 , wk+1))],

(9)

xk+1 ?X

where each expectation is taken with respect to one simulated outcome vector wk+1 , and we have used the fact that E[rk (Sk , xk+1 , wk+1)] = EIc (xk+1 ; Sk) is the constrained expected improvement known in closed-form [27]. The optimal reward is given by J?? (Sn) = Jn (Sn). Thus, at iteration n of the CBO algorithm, the optimal policy select the next design xn+1 that maximizes Jn (Sn) given by Eqs. 9. In other words, the best decision to make at iteration n maximizes, on average, the sum of the immediate reward rn and the future long-term reward Jn+1 (Sn+1) obtained by making optimal subsequent decisions. This is illustrated in Fig. 1, left panel.

Sk wk+1 Sk+1 xk+1

xk+2

wk+2 Sk+2

xk+3
Sk wk+1 Sk+1 ?k+1 (Sk+1) xk+1
wk+2 Sk+2 ?k+2 (Sk+2)
???
???
???
???
???
???
???
???

Figure 1: Left: Tree illustrating the intractable DP formulation. Each black circle represents a training set and a design, each white circle is a training set. Dashed lines represent simulated outcomes resulting in expectations. The double arrows represent designs selected with the (unknown) optimal policy, leading to nested maximizations. Double arrows depict the bidirectional way information propagates when the optimal policy is built: each optimal decision depends on the previous steps and relies on the optimality of the future decisions. Right: Single arrows represent designs selected using a heuristic. This illustrates the unidirectional propagation of information when a known heuristic drives the simulations: each decision depends on the previous steps but is independent of the future ones. The absence of nested maximization leads to a tractable formulation.

3.2

Rollout for Constrained Bayesian Optimization

The best optimization policy evaluates, at each iteration n of the CBO algorithm, the design xn+1 maximizing the optimal reward J?? (Sn) (Eq. 8). This requires solving a problem with several nested maximizations and expectations (Eqs. 9), which is computationally intractable. To mitigate the cost of solving the DP algorithm, we employ an approximate dynamic programming (ADP) technique: rollout (see [2, 23] for an overview). Rollout selects the next design by maximizing a (suboptimal) long-term reward J? . The reward is computed by simulating optimization scenarios over several future steps. However, the simulated steps are not controlled by the optimal policy ? ? . Instead, rollout uses a suboptimal policy ?, i.e. a heuristic, to drive the simulation. This circumvents the need for nested maximizations (as illustrated in Fig. 1, right panel) and simplifies the computation of J? compared to J?? . We now formalize the rollout algorithm, propose a heuristic ? adapted to the context of CBO with a finite budget, and detail further numerical approximations. Let us consider the iteration n of the CBO algorithm. The long-term reward J? (Sn) induced by a (known) heuristic ? = {?1 , ? ? ? , ?N }, starting from state Sn , is defined by Eq. 7. This can be rewritten as J? (Sn) = Hn , where Hn is recursively defined, from k = N back to k = n, by: HN +1 (SN +1) = 0 Hk (Sk) = E[rk (Sk , ?k (Sk), wk+1) + ?Hk+1 (Fk (Sk , ?k (Sk), wk+1))], 5

(10)

where each expectation is taken with respect to one simulated outcome vector wk+1 , and ? ? [0, 1] is a discount factor encouraging the early collection of reward. A discount factor ? = 0 leads to a greedy policy, focusing on immediate reward. In that case, the reward J? simplifies to the constrained expected improvement EIc . A discount factor ? = 1, on the other hand, is indifferent to when the reward is collected. The fundamental simplification introduced by the rollout algorithm lies in the absence of nested maximizations in Eqs. 10. This is illustrated in Fig. 1, right panel. By applying a known heuristic, information only propagates forward: every simulated step depends on the previous steps, but is independent from the future simulated steps. This is in contrast to the DP algorithm, illustrated in Fig. 1. Because the optimal policy is not known, it needs to be built by solving a sequence of nested problems. Thus, information propagates both forward and backward. While Hn is simpler to compute than Jn , it still requires computing nested expectations for which there is no closed-form expression. To further alleviate the cost of computing the long-term reward, we introduce two numerical simplifications. First, we use a rolling horizon h ? N to decrease the ? = min{N, n+h}. number of future steps simulated. A rolling horizon h replaces the horizon N by N Second, the expectations with respect to the (I + 1)-dimensional Gaussian random variables are numerically approximated using Gauss-Hermite quadrature. We obtain the following formulation: ? ? (S ?) = 0 H N +1 N +1 ? k (Sk) = EIc (?k (Sk); Sk) + ? H

Nq X

(q)

? k+1 (Fk (Sk , ?k (Sk), w ?(q) [H k+1))],

(11)

q=1 (q)

where Nq is the number of quadrature weights ?(q) ? R and points wk+1 ? RI+1 . For all iteration n ? {1, ? ? ? , N } and for all xn+1 ? X , we define the utility function of our rollout algorithm for CBO with finite budget to be: Un (xn+1 ; Sn) = EIc (xn+1 ; Sn) + ?

Nq X

(q)

? n+1 (Fn (Sn , xn+1 , w ?(q) [H n+1))].

(12)

q=1

The heuristic ? is problem-dependent. A desirable heuristic combines two properties: (1) it is cheap to compute, (2) it is a good approximation of the optimal policy ? ? . In the case of CBO with a finite budget, the heuristic ? ought to mimic the exploration-exploitation trade-off balanced by the optimal policy ? ? . To do so, we propose using a combination of greedy CBO algorithms: maximization of the constrained expected improvement (which has an exploratory behavior) and a constrained optimization based on the posterior means of the GPs (which has an exploitative behavior). For a given iteration n, we define the heuristic ? = {?n+1 , ? ? ? , ?N? } such that for stages ? ? 1},

the policy component ?k : Zk 7? X , maps a state Sk to the design xk+1 k ? {n + 1, ? ? ? , N satisfying: xk+1 = argmax EIc (x; Sk).

(13)

x?X

The last policy component, ?N? : ZN? 7? X , maps a state SN? to xN? +1 such that: xN? +1 = argmin ?N? (x; f) s.t. P F (x; SN?) ? 0.99,

(14)

x?X

where P F is the probability of feasibility known in closed-form. Every evaluation of the utility function Un requires O Nqh applications of a heuristic component ?k . The heuristic that we propose

optimizes a quantity that requires O —Sk —2 of work. To summarize, the proposed approach sequentially selects the next design to evaluate by maximizing the long-term reward induced by a heuristic. This rollout algorithm is a one-step lookahead formulation (one maximization) and is easier to solve than the N -steps lookahead approach (N nested maximizations) presented in Sec. 3.1. Rollout is a closed-loop approach where the information collected at a given stage of the simulation is used to simulate the next stages. The heuristic used in the rollout is problem-dependent, and we proposed using a combination of greedy CBO algorithms to construct such a heuristic. The computation of the utility function is detailed in Algorithm 2. 6

Algorithm 2 Rollout Utility Function Function: utility(x, h, S) Construct GPs using S if h = 0 then U ? EIc (x; S) else U ? EIc (x; S) Generate Nq Gauss-Hermite quadrature weights ?(q) and points w(q) associated with x for q = 1 to Nq do S 0 ? S ? {(x, w(q))} if h ¿ 1 then x0 ? ?(S 0) using Eq. 13 else x0 ? ?(S 0) using Eq. 14 end if U ? U + ??(q) utility(x0 , h ? 1, S 0) end for end if Output: U

4

Results

In this section, we numerically investigate the proposed algorithm and demonstrate its performance on classic test functions and a reacting flow problem. To compare the performance of the different CBO algorithms tested, we use the utility gap metric [14]. At iteration n, the utility gap en measures the error between the optimum feasible value f ? and the value of the objective function at a recommended design x?n :

—f (x?n) ? f ? — if x?n is feasible, en = (15) —? ? f ? — else, where ? is a user-defined penalty punishing infeasible recommendations. The recommended design, x?n , differs from the design selected for evaluation xn . It is the design that the algorithm would recommend to evaluate if the optimization were to be stopped at iteration n, without early notice. We use the same system of recommendation as [14]: x?n = argmin ?n (x; f) s.t. P F (x; Sn) ? 0.975. (16) x?X

Note that the utility gap en is not guaranteed to decrease because recommendations x?n are not necessarily better with iterations. In particular, en is not the best error achieved in the training set Sn . In the following numerical experiments, for the rollout algorithm, we use independent zero-mean GPs with

automatic relevance determination (ARD) square-exponential kernel to model each expensive-toevaluate function. In Algorithm. 1, when the GPs are constructed, the vector of hyper-parameters ? i associated with the ith GP kernel is estimated by maximization of the marginal likelihood. However, to reduce the cost of computing Un , the hyper-parameters are kept constant in the simulated steps (i.e., in Algorithm. 2). To compute the expectations of Eqs. 11-12, we employ Nq = 3I+1 Gauss-Hermite quadrature weights and points and we set the discount factor to ? = 0.9. Finally, at iteration n, the Sn best value fbest is set to the minimum posterior mean ?n (x; f) over the designs x in the training set Sn , such that the posterior mean of each constraint is feasible. If no such point can be found, Sn 2 then fbest is set to the maximum of {?n (x; f) + 3?m } over the designs x in Sn , where ?m is the maximum variance of the GP associated with f . The EIC algorithm is computed as a special case of the rollout with rolling horizon h = 0, and we use the Spearmint package1 to run the PESC algorithm. We additionally run a CBO algorithm that selects the next design to evaluate based on the posterior means of the GPs2 : xn+1 = argmin ?n (x; f) s.t. ?n (x; gi) ? 0, ?i ? {1, . . . , I}. (17) x?X

1 2

https://github.com/HIPS/Spearmint/tree/PESC As suggested by a reviewer.

7

0.0 log10 Median Utility Gap en

log10 Median Utility Gap en

1 0 ?1 ?2 ?3

P ESC PM EIc

?4

Rollout, h = 1 Rollout, h = 2 Rollout, h = 3

?5 0

5

10

15

20 25 Iteration n

30

35

40

P ESC PM EIc

?0.5

Rollout, h = 1 Rollout, h = 2 Rollout, h = 3

?1.0 ?1.5 ?2.0 ?2.5 ?3.0 0

5

10

15

20 25 Iteration n

30

35

40

Figure 2: Left: Multi-modal objective and single constraint (P1). Right: Linear objective and multiple non-linear constraints (P2). Shaded region indicates 95% confidence interval of the median statistic. We refer to this algorithm as PM. We also compare the CBO algorithms to three local algorithms (SLSQP, MMA and COBYLA) and to one global evolutionary algorithm (ISRES). We now consider four problems with different design space dimensions d, several numbers of constraints I, and various topologies of the feasible space. The three first problems, P1-3, are analytic functions while the last one, P4, uses a reacting flow model that requires solving a set of partial differential equations (PDEs) [4]. For P1 and P2, we use N = 40 evaluations (as in [6, 10]). For P3 and P4, we use a small number of iterations N = 60, which corresponds to situations where the functions are very expensive to evaluate (e.g. solving large systems of PDEs can take over a day on a supercomputer). The full description of the problems is available in the appendix. In Figs. 2-3, we show the median of the utility gap, the shadings represent the 95% confidence interval of the median computed by bootstrap. Other statistics of the utility gap are shown in the appendix. For P1, the median utility gap for EIC, PESC, PM and the rollout algorithm with h ? {1, 2, 3} is shown in Fig. 2 (left panel). The PM algorithm does not improve its recommendations. This is not surprising because PM focuses on exploitation (PM does not depends on posterior variance) which can result in the algorithm failing to make further progress. Such behavior has already been reported in [16] (Sec. 3). The three other CBO algorithms perform similarly in the first 10 iterations. PESC is the first to converge to a utility gap ? 10?2.7 . The rollout performs better or similarly than EIC. In the 15 first iterations, longer rolling horizons lead to slightly lower utility gaps. This is likely to be due to the more exploratory behavior associated with lookahead, which helps differentiating the global solution from the local ones. For the remaining iterations, the shorter rolling horizons reduce the utility gap faster than longer rolling horizons before reaching a plateau. EIC and rollout outperform PESC after 25 iterations. We note that EIC and rollout have essentially converged. For P2, the median performance of EIC, PESC, PM and rollout with rolling horizon h ? {1, 2, 3} is shown in Fig. 2 (right panel). The PM algorithm reduces the utility gap in the first 10 iterations, but reaches a plateau at 10?1.7 . The three other CBO algorithms perform similarly up to iteration 15, where PESC reaches a plateau 3 . This similarity may be explained by the fact that the local solutions are easily differentiable from the global one, leading to no advantage for exploratory behavior. In this example, the rollout algorithms reached the same plateau at 10?3 , with longer horizons h taking more iterations to converge. EIC performs better than rollout h = 2 before its performance slightly decreases, reaching a plateau at a larger utility gap 10?2.6 (note that the utility gap is not computed with the best value observed so far and thus is not guaranteed to decrease). This increase of the median utility gap can be explained by the fact that a few runs change their recommendation from one local minimum to another one, resulting in the change in median utility function. This is also reflected in the 95% confidence interval of the median, which further indicates that the statistic is sensitive to a few runs. For P3, the median utility gap for the four CBO algorithms is shown

in Fig. 3 (left panel). PM is rapidly outperformed by the other algorithms. The PESC algorithm is outperformed by EIC and rollout after 25 iterations. Again, we note that rollout with h = 1 obtains a lower utility gap that EIC at every iteration. The rollout with h ? {2, 3} exhibits a different behavior: it starts decreasing the utility gap later in the optimization but achieves a better performance when the evaluation budget 3

Results obtained for PESC mean utility gap are consistent with [13].

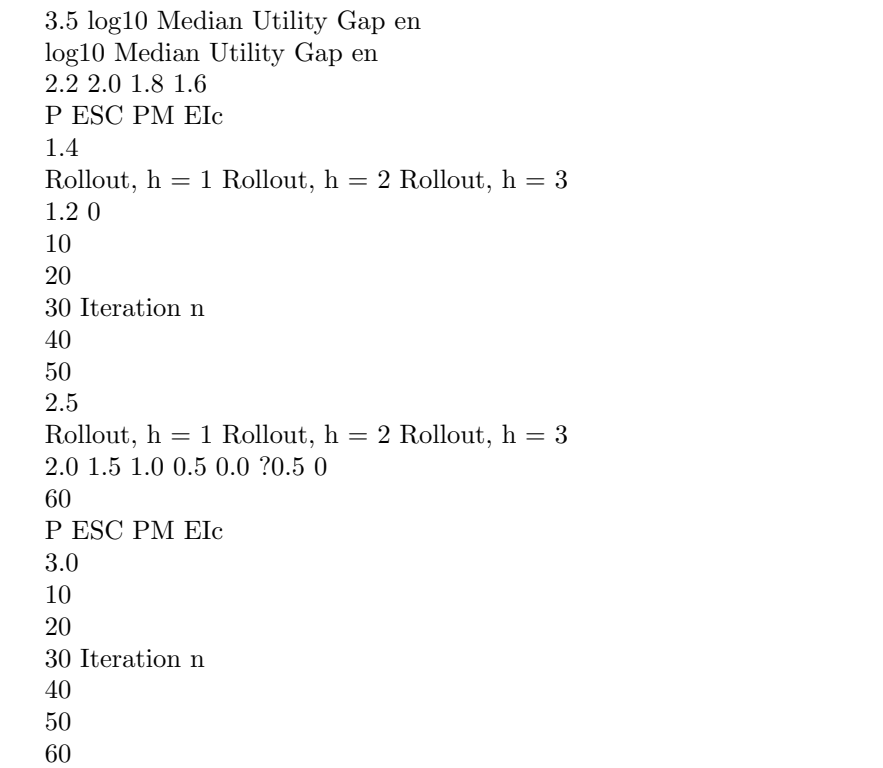

Figure 3: Left: Multi-modal 4-d objective and constraint (P3). Right: Reacting flow problem (P4). The awareness of the remaining budget explains the sharp decrease in the last iterations for the rollout. is consumed. Note that none of the algorithms has converged to the global solution, and the strong multi-modality of the objective and constraint function seems to favor exploratory behaviors. For the reacting flow problem P4, the median performances are shown in Fig. 3 (right panel). PM rapidly reaches a plateau at en ? 101.3 . PESC reduces rapidly the utility gap, outperforming the other algorithms after 15 iterations. EIC and rollout perform similarly and slowly decrease the utility gap up to iteration 40, where EIC reaches a plateau and rollout continues to improve performance, slightly outperforming PESC at the end of the optimization. The results are summarized in Table. 1, and show that the rollout algorithm with different rolling horizons h (R-h) performs similarly or favorably compared

to the other algorithms. Table 1: Log median utility gap log10 (eN). Statistics computed over m independent runs. Prob

d

N

I

m

SLSQP

MMA

COBYLA

ISRES

PESC

PM

EIC

R-1

R-2

R-3

P1 P2 P3 P4

2 2 4 4

40 40 60 60

1 2 1 1

500 500 500 50

0.59 -0.40 2.15 0.80

0.59 -0.40 3.06 0.80

-0.05 -0.82 3.06 0.80

-0.19 -0.70 1.68 0.13

-2.68 -2.43 1.66 0.09

0.30 -1.76 1.79 1.26

-4.45 -2.62 1.60 0.57

-4.59 -2.99 1.48 -0.10

-4.52 -2.99 1.31 -0.10

-4.42 -2.994 1.35 0.19

Based on the four previous examples, we notice that increasing the rolling horizon h does not necessarily improve the performance of the rollout algorithm. One possible reason stems from the fact that lookahead algorithms rely more on the statistical model that greedy algorithms. Because this model is learned as the optimization unfolds, it is an imperfect model (in particular the hyperparameters of the GPs are updated after each iteration, but not after each stage of a simulated scenario). By simulating too many steps with the GPs, one may be over-confidently using the model. In some sense, the rolling horizon h, as well as the discount factor ?, can be interpreted as a form of regularization. The effect of a larger rolling horizon is problem-dependent, and experiment P3 suggests that multimodal problems in higher dimension may benefits from longer rolling horizons.

5

Conclusions

We proposed a new formulation for constrained Bayesian optimization with a finite budget of evaluations. The best optimization policy is defined as the one maximizing, in average, the cumulative feasible decrease of the objective function over multiple steps. This optimal policy is the solution of a dynamic programming problem that is intractable due to the presence of nested maximizations. To circumvent this difficulty, we employed the rollout algorithm. Rollout uses a heuristic to simulate optimization scenarios over several step, thereby computing an approximation of the long-term reward. This heuristic is problem-dependent and, in this paper, we proposed to use a combination of cheap-to-evaluate greedy CBO algorithms to construct such heuristic. The proposed algorithm was numerically investigated and performed similarly or favorably compared to constrained expected improvement (EIC) and predictive entropy search with constraint (PESC). This work was supported in part by the AFOSR MURI on multi-information sources of multi-physics systems under Award Number FA9550-15-1-0038, program manager Dr. Jean-Luc Cambier. 4

For cost reasons, the median for h = 3 was computed with m = 100 independent runs instead of 500.

9

2 References

[1] C. Audet, A. J. Booker, J. E. Dennis Jr, P. D. Frank, and D. W. Moore. A surrogate-model-based method for constrained optimization. AIAA paper, 4891, 2000. [2] D. P. Bertsekas. Dynamic programming and optimal control, volume 1. Athena Scientific, 1995. [3] M. Bj?rkman and K. Holmstr?m. Global optimization of costly nonconvex functions using radial basis functions. Optimization and Engineering, 4(1):373?397, 2000. [4] M. Buffoni and K. E. Willcox. Projection-based model reduction for reacting flows. In 40th Fluid Dynamics Conference and Exhibit, page 5008, 2010. [5] P. Feliot, J. Bect, and E. Vazquez. A Bayesian approach to constrained single-and multi-objective optimization. Journal of Global Optimization, 67(1-2):97?133, 2017. [6] J. Gardner, M. Kusner, K. Q. Weinberger, J. Cunningham, and Z. Xu. Bayesian optimization with inequality constraints. In T. Jebara and E. P. Xing, editors, Proceedings of the 31st International Conference on Machine Learning (ICML-14), pages 937?945. JMLR Workshop and Conference Proceedings, 2014. [7] M. A. Gelbart, J. Snoek, and R. P. Adams. Bayesian optimization with unknown constraints. arXiv preprint arXiv:1403.5607, 2014. [8] D. Ginsbourger and R. Le Riche. Towards Gaussian process-based optimization with finite time horizon. In mODa 9?Advances in Model-Oriented Design and Analysis, pages 89?96. Springer, 2010. [9] J. Gonz?lez, M. Osborne, and N. D. Lawrence. GLASSES: Relieving the myopia of Bayesian optimisation. In Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, pages 790?799, 2016. [10] R. B. Gramacy, G. A. Gray, S. Le Digabel, H. K. H. Lee, P. Ranjan, G. Wells, and S. M. Wild. Modeling an augmented Lagrangian for blackbox constrained optimization. Technometrics, 58(1):1?11, 2016. [11] R. B. Gramacy

and H. K. H. Lee. Optimization under unknown constraints. arXiv preprint arXiv:1004.4027, 2010. [12] P. Hennig and C. J. Schuler. Entropy search for information-efficient global optimization. The Journal of Machine Learning Research, 13(1):1809?1837, 2012. [13] J. M. Hern?ndez-Lobato, M. A. Gelbart, R. P. Adams, M. W. Hoffman, and Z. Ghahramani. A general framework for constrained bayesian optimization using information-based search. arXiv preprint arXiv:1511.09422, 2015. [14] J. M. Hern?ndez-Lobato, M. A. Gelbart, M. W. Hoffman, R. P. Adams, and Z. Ghahramani. Predictive entropy search for bayesian optimization with unknown constraints. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 2015. [15] J. M. Hern?ndez-Lobato, M. W. Hoffman, and Z. Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. In Advances in Neural Information Processing Systems, pages 918?926, 2014. [16] D. R. Jones. A taxonomy of global optimization methods based on response surfaces. Journal of Global Optimization, 21(4):345?383, 2001. [17] R. R. Lam, K. E. Willcox, and D. H. Wolpert. Bayesian optimization with a finite budget: An approximate dynamic programming approach. In Advances in Neural Information Processing Systems, pages 883?891, 2016. [18] C. K. Ling, K. H. Low, and P. Jaillet. Gaussian process planning with lipschitz continuous reward functions: Towards unifying Bayesian optimization, active learning, and beyond. arXiv preprint arXiv:1511.06890, 2015. [19] J. Mockus, V. Tiesis, and A. Zilinskas. The application of bayesian methods for seeking the extremum. Towards Global Optimization, 2(117-129):2, 1978. [20] M. A. Osborne, R. Garnett, and S. J. Roberts. Gaussian processes for global optimization. In 3rd International Conference on Learning and Intelligent Optimization (LION3), pages 1?15, 2009. [21] V. Picheny. A stepwise uncertainty reduction approach to constrained global optimization. In AISTATS, pages 787?795, 2014.

10

[22] V. Picheny, R. B. Gramacy, S. Wild, and S. Le Digabel. Bayesian optimization under mixed constraints with a slack-variable augmented Lagrangian. In Advances in Neural Information Processing Systems, pages 1435?1443, 2016. [23] W. B. Powell. Approximate Dynamic Programming: Solving the Curses of Dimensionality, volume 842. John Wiley & Sons, 2011. [24] C. E. Rasmussen and C. K. I. Williams. Gaussian Processes for Machine Learning. MIT Press, Cambridge, MA, 2006. [25] R. G. Regis. Constrained optimization by radial basis function interpolation for high-dimensional expensive black-box problems with infeasible initial points. Engineering Optimization, 46(2):218?243, 2014. [26] M. J. Sasena, P. Y. Papalambros, and P. Goovaerts. The use of surrogate modeling algorithms to exploit disparities in function computation time within simulation-based optimization. Constraints, 2:5, 2001. [27] M. Schonlau, W. J. Welch, and D. R. Jones. Global versus local search in constrained optimization of computer models. Lecture Notes-Monograph Series, pages 11?25, 1998.

11