Bayesian Optimization: Which Constraints Matter

MAX BUTLER, Mathematics for Real-World Systems, University of Warwick, UK
ALEKSANDRA TOKAEVA, Mathematics for Real-World Systems, University of Warwick, UK
JAMES TOWN, Mathematics for Real-World Systems, University of Warwick, UK
XIETAO WANG LIN, Mathematics for Real-World Systems, University of Warwick, UK
JUERGEN BRANKE, Warwick Business School, University of Warwick, UK
JUAN UNGREDDA, ESTECO SpA, Italy

We present three distinct methods for solving global black-box optimization problems with decoupled black-box constraints, in which subsets of the objective and constraint functions may be evaluated independently. In particular, our methods aim to take into account that often only a handful of the constraints may be binding at the optimum, and hence evaluate only relevant constraints when trying to optimize a function. All of our methods are in the Bayesian Optimization paradigm, which relies on the assumption that all function evaluations are expensive, and hence we have only a very limited budget of evaluations. We propose two algorithms which extend two existing coupled approaches, and call them 'decoupled constrained Knowledge Gradient' and 'decoupled Expected Improvement', and we further propose a small modification to the latter as a separate algorithm. We benchmark these methods against existing methods and also discuss their scalability to higher dimensions.

CCS Concepts: • **Theory of computation → Nonconvex optimization**.

Additional Key Words and Phrases: Gaussian Processes, Bayesian Optimisation, Constrained Optimisation

ACM Reference Format:
Max Butler, Aleksandra Tokaeva, James Town, Xietao Wang Lin, Juergen Branke, and Juan Ungredda. 2024. Bayesian Optimization: Which Constraints Matter. *ACM Trans. Model. Comput. Simul.* 1, 1, Article 1 (June 2024), 19 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Black-box optimisation is at the heart of many real-world systems, possibly constrained by a set of black-box constraints. For instance, the MOPTA-2008 benchmark is a problem in the automotive industry, where the objective is to minimise the weight of the vehicle subject to 68 black-box constraints with 124 variables. Many of the black-box functions could involve expensive experiments or simulations [Jones 2008]. Hence, it is crucial to minimise the amount of function evaluations while optimizing the function.

A popular method for black-box optimisation is Bayesian optimisation (BO) which has seen use in environmental monitoring [Marchant and Ramos 2012] or hyperparameters tuning of machine

Authors' Contact Information: Max Butler, Mathematics for Real-World Systems, University of Warwick, Coventry, UK, max.butler@warwick.ac.uk; Aleksandra Tokaeva, Mathematics for Real-World Systems, University of Warwick, Coventry, UK, aleksandra.tokaeva@warwick.ac.uk; James Town, Mathematics for Real-World Systems, University of Warwick, Coventry, UK, james.p.town@warwick.ac.uk; Xietao Wang Lin, Mathematics for Real-World Systems, University of Warwick, Coventry, UK, xietao.wang-lin@warwick.ac.uk; Juergen Branke, Warwick Business School, University of Warwick, Coventry, UK, juergen.branke@wbs.ac.uk; Juan Ungredda, ESTECO SpA, Trieste, Italy, ungredda@esteco.com.

ACM 1558-1195/2024/6-ART1
https://doi.org/XXXXXXX.XXXXXXX

learning algorithms [Joy et al. 2016]. The underlying principle of BO is to build a surrogate model for our black-box functions and then devise an acquisition function which quantifies the value of evaluating a given design in determining the optimal design for our problem. The canonical choice of the surrogate model for our objective function and each constraint is a Gaussian Process (GP) model, where a GP is a multi-dimensional normal distribution fully specified by its mean and covariance [Rasmussen and Williams 2005]. GPs are used so frequently because they are a fairly simple and well-studied class of models, and at the same time, a GP provides an uncertainty estimate. We describe the procedure of fitting a GP more rigorously in the Appendix B. In general, most methods in BO are differentiated by their choice of acquisition function or optimisation procedure. We discuss several well-known acquisition functions in more detail in Section 3.

In the unconstrained setting, common acquisition functions include the Expected Improvement (EI) [ur Rehman et al. 2014] or the Knowledge Gradient (KG) [Scott et al. 2011]. In the former, one attempts to maximise the expectation of the amount by which one improves over the current best-sampled design. This is a probability-weighted average of the possible improvement values over the current best-sampled design, where we also do not penalise for the values which would fall below the current best sampled. This is to avoid always sampling at the current maximum mean of the GP, and also explore areas with high variance. This is known as the problem of balancing exploration with exploitation, which is a fundamental idea in BO. EI is sometimes criticised for being a 'myopic' metric [Jiang et al. 2020], meaning that it only focuses on the gain which is immediately derived from sampling a certain point. We might be more interested in determining the point to sample which will give desirable properties in the posterior distribution. This is a 'one-step look ahead' principle, and KG is an algorithm of this sort, where one seeks to sample the point which maximises the posterior mean of the GP [Ungredda and Branke 2024].

During the last decade, BO has been vastly extended to the constrained setting. For instance, constrained Expected Improvement (cEI) considers the EI acquisition function weighted by the probability of feasibility [Gardner et al. 2014]. More recently, constrained Knowledge Gradient (cKG) was explored by [Ungredda and Branke 2024] which extends KG to the constrained problem. Other methods make use of the Augmented Lagrangian [Picheny et al. 2016] which converts the constrained problem to an unconstrained one, or information-theoretic methods such as Predictive Entropy Search with constraints (PESC) [Miguel Hernández-Lobato et al. 2015].

A potential pitfall of these methods is that they evaluate all the black-box functions together (coupled evaluation) and cannot exploit decoupled evaluations. It might be interesting to consider the case where black-box functions are evaluated separately (decoupled evaluation); decoupled constrained BO was first considered before by [Gardner et al. 2014], where they describe the "chicken-and-egg pathology" of cEI for decoupled evaluations. PESC was also extended to decoupled evaluations quite trivially as its acquisition function can be decomposed into individual functions [Miguel Hernández-Lobato et al. 2016, 2015]. The latest addition is from [Nguyen et al. 2024] using the popular principle of "optimism in the face of uncertainty" from multi-armed bandits [Papini et al. 2019].

In this report, we propose three new constrained BO methods for decoupled evaluations: decoupled constrained Knowledge Gradient (dcKG), decoupled constrained Expected Improvement (dcEI) and a hybrid method between cEI and dcKG (EI+KG) and explore their performance on three different synthetic problems.

The remaining part of the work is structured as follows: in Section 2 we state the problem setting and explain how BO works; in Section 3 we give an overview of existing approaches (in particular, the fully coupled versions of EI and KG); in Section 4 we present original work focused on extending the aforementioned approaches to the decoupled setting; in Section 5 we present numerical experiments showing the performance of different methods on synthetic problems; and finally in Section 6 we discuss conclusions which can be drawn from this work and open questions which remain.

2 PROBLEM SETTING AND BAYESIAN OPTIMISATION

2.1 Problem Setting

We consider a sub-region $X \subset \mathbb{R}^d$, and a black-box function $f : X \to \mathbb{R}$ with constraints $c_k : X \to \mathbb{R}$. We wish to find the optimizer, x^*, of this function subject to each constraint function being not positive:

$$x^* = \arg\min_{x \in X} f(x), \quad \text{s.t.} \quad c_k(x) \leq 0 \quad \forall\, k = 1, \dots, K.$$

The objective function f takes as arguments a design vector $x \in X$ and returns an observation (which we assume to be noiseless) $y = f(x)$ or $y = c_k(x), k = 1, \dots, K$, where we assume that constraint and objective function observations are independent and may be evaluated individually. The latter assumption contrasts with the more common assumption in the literature that if we evaluate one constraint at a point x, then we must evaluate *all* of the constraints at that point, anf hence return the objective function value and a *vector* of constraint values. [Miguel Hernández-Lobato et al. 2015] gives the example of a financial simulator, in which every part of a simulation must be run at once.

We assume that we have a total budget of B samples, and after the budget has been consumed we should return a recommended design, x_r. Its quality is determined by the difference between the true optimal value, $f(x^*)$, and the value of the objective function at our recommended point, $f(x_r)$. We naturally come to the question of how to penalise infeasibility, because we are only interested in those points which are feasible, i.e. $x \in F$, where $F = \{x | c_k(x) \leq 0; \forall k = 1, \dots, K\}$. If x_r is not feasible then we simply set the objective function to return some penalty value, M. Finally, we measure the quality of a solution by an Opportunity Cost (OC) to be minimised:

$$\mathrm{OC}(x_r) = \begin{cases} f(x^*) - f(x_r), & x_r \in F \\ f(x^*) - M, & x_r \notin F \end{cases} \tag{1}$$

Without loss of generality, we assume that a penalty $M = 0$. However, M may be set by using the minimum surrogate model estimate of the objective function in the design space [Letham et al. 2019]. We also remark that the penalty value is often best set by an experienced practitioner in the area of the problem [Ungredda and Branke 2024].

2.2 Summary of Bayesian Optimisation

As stated earlier, we build a surrogate model and then use this model to quantify the added value in the task of determining the location of the optimiser given by evaluating each candidate point. We introduce here the notation and basic concepts that we use in our report. The basic procedure of BO is as follows.

Algorithm 1: Bayesian optimization procedure

```
1 Perform some space-filling initial data collection, so that is it possible to make informed
    decisions about sampling.
2 while budget $B$ of evaluations is not exhausted do
3     Fit independent GPs for each constraint and the objective.
4     Use the acquisition function to obtain the most promising location (and
        constraint/objective in the decoupled case) to evaluate next.
5     Evaluate the true quality of this design.
6     Update the surrogate models with the new evaluated point.
7 end
8 Return the recommended design.
```

Now we briefly summarise what is meant by using a GP surrogate model for the objective function. When fitting a GP, we 'approximate' the objective function f by a Gaussian process (GP) which is fully specified by its mean, $\mu(x)$, and covariance, $\sigma(x, x')$, functions. Thus at a point $x \in \mathcal{X}$, $y \sim \mathcal{N}(\mu(x), \sigma(x, x))$. We specify some prior belief about the mean (typically this is just set to 0) and the covariance, with the latter being encoded in a kernel $\sigma(x, x')$. For example, one could choose

$$\sigma(x, x') = \sigma_f^2 \exp\left(-\sum_{i=1}^{n} \frac{1}{2} \frac{(x_i - x_i')^2}{l_i^2}\right),$$

where σ_f and l_i are hyper-parameters typically chosen using maximum likelihood approach. Note that the exact correlation structure that we impose on design vectors is again problem-specific and should be set by an experienced practitioner. The correlation structure is an essential part of BO, because when searching for promising designs, we base on the assumption that close points have fairly similar values, while far-located points have unrelated values. That is why the requirement that the objective function varies smoothly and not too quickly is important for our model to work well.

We also need to take into account the fact that the conditional distribution of f will change based on data collected, $D_f = (x_i, f(x_i))_{i=1}^n$. Here subscript f shows that the dataset is for the objective function, not one of the constraints. Constraints have independent datasets with subscript c. The conditional distribution turns out to be fairly simple, in the sense that one can condition on data and a *fantasised* $x_{n+1} \in \mathcal{X}$ with different possible observations $y_{n+1} = \mu(x_{n+1}) + s$, where $s \in \mathcal{R}$. At any other point $z \in \mathcal{X}$, one finds that the mean and variance scale linearly in s, which is convenient behaviour. For further information on how conditioning works, one can consult the Appendix B which is based on [Rasmussen and Williams 2005].

The most straightforward freedom one has in the BO framework using GP models is to change the acquisition function, which quantifies how useful it is to evaluate the objective function at a given point. Acquisition functions depend solely on the parameters of the existing GP model. In Section 3 we will recap the principles of two most widespread existing acquisition functions (KG and EI). As with kernel and hyper-parameters choices, it is important to choose acquisition functions which are tailored to one's specific problem. For example, in high-dimensional problems, the time complexity of some approaches is prohibitive, whereas for others (such as Expected Improvement) it may well be slower to perform the fitting (which involves matrix inversion having complexity $O(n^3)$) than to optimise the acquisition function. So, given D_f as above (sometimes called the training dataset), we fit our GP model on the training dataset and then optimise the acquisition function over the domain. Then we evaluate the objective function at this chosen point and hence get another data point (x_{n+1}, y_{n+1}), which we add to our train dataset. We repeat the process until we have exhausted our budget of function evaluations. At this point, we must return a recommended solution x_r, which from a risk-neutral perspective is given by $x_r = \arg\max_x \mu_y^n(x)$.

Each constraint function is modelled similarly, with data for the k^{th} constraint of $D_c^k = \{(x_i, c_k(x_i))\}_{i=1}^{n_k}$. It is important to note the subscript in n_k, because we may have a different number of observations for each constraint and also observations may be taken at completely different points because we may be in the decoupled setting. We construct the total constraint dataset being the union of all constraint observation data as $\mathbf{D_c} = \cup_{k=1}^{K} D_c^k$. It will be useful to introduce the probability of feasibility of a given point as simply

$$\mathrm{PF} = \mathbb{P}[c_k(x) \leq 0;\;\; 1 \leq k \leq K \mid \mathbf{D_c}] = \prod_{k=1}^{K} \mathbb{P}[c_k(x) \leq 0 | D_c^k].$$

The latter expression may be evaluated simply through the product of univariate Gaussian CDFs because we assume constraints are independent. This fact gives the natural generalisation of x_r to the constrained setting as

$$x_r = \arg\max_{x \in X} \mu_y^B(x) \mathbb{P}[\mathbf{c}(x) \leq 0],$$

where $\mathbb{P}[\mathbf{c}(x) \leq 0]$ is the probability of feasibility of a design x. We remark at this point that evaluating a constraint does not immediately translate to better expected objective performance, but rather to more accurate feasibility information which may change our current belief about the location of the optimum.

3 EXISTING APPROACHES

Our proposed methods extend the use of two existing acquisition functions to the decoupled setting, and we therefore give the basic details of the existing work on the constrained Knowledge Gradient (cKG) acquisition function and the constrained Expected Improvement (cEI) acquisition function.

We also mention Constrained Predictive Entropy Search (PESC), constrained Thompson Sampling (TS) and Penalised Knowledge Gradient (pKG) as existing approaches, but we don't use them in our report.

3.1 Constrained Knowledge Gradient

We first introduce the principles of the *unconstrained* Knowledge Gradient (KG) acquisition function [Scott et al. 2011]. We initially introduce KG in the unconstrained setting, and will shown how this has been extended to the fully coupled constrained setting. The acquisition function selects the point which maximises the KG function,

$$\mathrm{KG}(x) = \mathbb{E}\left[\max_{x' \in X}\{\mu_y^{n+1}(x')\} - \max_{x' \in X}\{\mu_y^n(x')\} | x_{n+1} = x\right]. \tag{2}$$

This acquisition function computes the expectation of the difference between the current (meaning given $\mathbf{D_c}$) maximum of the posterior mean of the GP surrogate model and new maximum posterior mean given that we sampled at a point x. The reason that this is called a 'knowledge gradient' is that we are asking the question of how much knowledge about the maximum posterior mean we can gain from sampling a given point. It is worth noting that $KG(x) \geq 0$ due to Jensen's inequality and the convexity of the *max* function, so as one would expect it is impossible to lose knowledge by sampling a point (even if it turns out not to be very good).

In conceptual terms, the simplest way of evaluating the KG function (for which one cannot obtain a closed-form expression) is to simulate random variables $y \sim N(\mu(x), k(x,x))$ (either through quasi-random sampling or a quantile sampler), update the GP surrogate model, and maximise the mean. We then average over these maximum posterior mean values. This is the procedure described in [Frazier 2018], although there do exist other approaches for approximating $KG(x)$. One of the earliest works, [Scott et al. 2011], propose discretising the design space and computing the epigraph, then integrating over a piecewise linear function. The intuition for this approach relates to the fact mentioned earlier that the effect of varying values observed at x_{n+1} is linear on other points. However, increasing the number of dimensions requires more discretisation points and therefore this approach does not scale well computationally. In efforts to build on this, [Wu and Frazier 2016] used Monte Carlo sampling and thus avoided having to discretise the design space. Using Monte-Carlo samples improves how well the algorithm scales, but the trade-off is that it increases the complexity of the computations which must be done, so makes little sense to use in low dimensions [Ungredda Suárez 2022]. The state of the art was introduced by [Ungredda and Branke 2024], where a 'hybrid KG' was introduced, which aimed to combine the best of both approaches via a reparametrisation trick . It consists of obtaining high value points from the posterior GP mean, and then using these as the discretisation. The idea here is that the epigraph of the function will primarily be composed of those hyperplanes corresponding to points which already have a high value, so we can use this 'optimal discretisation' to approximate the epigraph quite accurately with relatively few samples. Combining both approaches allows one to leverage superior time complexity of the Monte-Carlo based acquisition functions and the computational performance of discretising the design space.

In [Ungredda and Branke 2024], KG was extended to the case of fully coupled constraints in the simple way of weighting the value of a design by the probability that it is feasible, and likewise for the current best candidate point.

Therefore, to quantify the benefit of a design vector, we first find the recommended design given by the sampled trained data D_c and D_f as,

$$x_r^n = \arg\max_{x \in X} \mu_y^n(x)\mathrm{PF}^n(x).$$

The difference in performance between the current recommended design and the new best performance presents an acquisition function for a design x,

$$\mathrm{cKG}(x) = \mathbb{E}\left[\max_{x' \in X}\{\mu_y^{n+1}(x')\mathrm{PF}^{n+1}(x')\} - \mu_y^{n+1}(x_r^n)\mathrm{PF}^{n+1}(x_r^n) | x_{n+1} = x\right]. \tag{3}$$

Again by Jensen's inquality, the last equation is positive for all the design space and in [Ungredda and Branke 2024] it is pointed out that $\mu_y^{n+1}(x_r^n)$ may be marginalised over y^{n+1}, such that

$$\mathrm{cKG}(x) = \mathbb{E}\left[\max_{x' \in X}\{\mu_y^{n+1}(x')\mathrm{PF}^{n+1}(x')\} - \mu_y^{n}(x_r^n)\mathrm{PF}^{n+1}(x_r^n) | x_{n+1} = x\right]. \tag{4}$$

It happens to be the case that when there are no constraints, the formula reduces to the standard formula for KG. The cKG value computes the maximum of the PF-weighted posterior distribution, and then subtracts from this the value of the PF-weighted current recommended design. This acquisition function quantifies the benefit of the next point at which we sample in terms not only of the mean weighted by probability of feasibility, but it also takes into account how new information about the constraints may affect our estimation of the probability that our existing x_r is feasible. In the original paper, several theoretical guarantees of convergence to the optimal design are given, which rely on the fact that in a finite search space X with an infinite sampling budget all points will be sampled infinitely often, hence guaranteeing the optimal point will be found.

The main drawback of the KG method is that it is very computationally intensive, for the reason that to determine $KG(x)$ for some $x \in \mathcal{X}$, we must generate several possible observations according to our current model. We then derive the posterior distributions for each of these observations, and must maximise each of them, for example, by L-BFGS, an optimization method used in Botorch.

Even L-BFGS scales poorly in this setting, because the landscape is typically very complex with many local maxima, so many different initialisations are needed to converge to the true global maximum of the posterior with high probability. This must be done for many different points in the design space to determine points with a high KG value, illustrating how costly this metric is to evaluate.

3.2 Constrained Expected Improvement

The second method which we will develop is based on work in [Schonlau et al. 1998]. The standard EI acquisition function is given by

$$\mathrm{EI}(x|f^*) = \mathbb{E}[\max(y - f^*, 0)].$$

This essentially asks for the expected amount by which a point improves over the current best-sampled location, f^*. We observe that we do not consider negative penalties for the failure to improve, else we would essentially just select the maximum GP mean every time which would be extremely myopic, meaning that it would fail to sufficiently explore the design space and always seek to maximise the current model. Since we are essentially asking for the tail probability of a normal distribution, it is not surprising that there is a closed-form expression for the EI value:

$$\mathrm{EI}(x|f^*) = (\mu_y^n(x) - f^*)\Phi(z) + k_y^n(x,x)\phi(z), \quad \text{where } x = \frac{\mu_y^n(x) - f^*}{k^n(x,x)}.$$

Here, Φ denotes the CDF of a Gaussian, ϕ is the PDF of a Gaussian, and μ^n is the GP mean. [Schonlau et al. 1998] then extended this acquisition function to the case of constrained problems simply by weighting the EI value by the probability of feasibility:

$$\mathrm{cEI}(x|f^*) = \mathrm{EI}(x|f^*)\mathrm{PF}^n(x),$$

where PF^n is again the probability of feasibility of x and $\mathrm{EI}(x)$ is as above. The main attraction of this method is its conceptual simplicity and the fact that it is very cheap to compute EI values.

3.3 Constrained Predicted Entropy Search (PESC)

[Miguel Hernández-Lobato et al. 2014] use as acquisition function to maximize the mutual information between y and x given the collected data, as,

$$PESC(x) = H(y|D_f, D_c) - \mathbb{E}_{x^*}\left[H[y|D_f, D_c, x, x^*]\right]$$

The first term on the right-hand side is computed as the entropy of a product of independent Gaussians. However, the second term in the right-hand side of has to be approximated. The expectation is approximated by averaging over samples of $\bar{x}^* \sim p(x^*|D_f, D_c)$. To sample x^*, first, samples from f and $c_1, \ldots, c_K$ are drawn from their GP posteriors. Then, a constrained optimisation problem is solved using the sampled functions to yield a sample $\bar{x}^*$.

PESC also was extended to constrained setting, where [Miguel Hernández-Lobato et al. 2016] extended predictive entropy search [Miguel Hernández-Lobato et al. 2014] to constraints and detailed how to make the sophisticated approximation of the entropy reduction computationally tractable in practice. Their PESC algorithm often achieves strong results and is widely considered the state-of-the-art for constrained BO despite its rather large computational costs.

Note that in the PESC article it is shown how to use PESC in the decoupled setting. We used repository Spearmint with decoupled PESC realization as a benchmark in our numerical experiments on artificial problems. However, PESC turned to be very slow and computationally demanding and hence we weren't able to run PESC the needed amount of iterations to get the desired accuracy.

3.4 Thompson Sapling with constraints (TS)

[D. Eriksson and Poloczek 2019] extended Thompson Sampling to constraints. Let $x_1, \ldots, x_n$ be candidate points. Then a realization is taken at the candidate points location $(\bar{f}(x_i), \bar{c}_1(x_i), \ldots, \bar{c}_m(x_i))$ for all x_i with $1 \leq i \leq r$ from the respective posterior distributions. Therefore, if $\bar{F} = \{x_i | \bar{c}_l(x_i) \leq 0 \text{ for } 1 \leq l \leq m\}$ is not empty, then the next design vector is selected by $\arg\max_{x \in \bar{F}} \bar{f}(x)$. Otherwise, a point is selected according to the minimum total violation $\sum_{l=1}^{m} \max\{\bar{c}_l(x), 0\}$.

[D. Eriksson and Poloczek 2019] further implements a strategy for high-dimensional design space problems based on the trust region that confines samples locally and study the effect of different transformations on the objective and constraints.

3.5 Penalised Knowledge Gradient (pKG)

Article [W. Chen and Tang 2021] extend KG to constrained problems by penalising any new sample by the

$$KG(x) = \mathbb{E}\left[\max_{x \in X}\{\mu_y^{n+1}(x)\} - \max_{x \in X}\{\mu_y^{n}(x)\} | x^{n+1} = x\right] PF^n(x^{n+1} = x) \tag{5}$$

This acquisition function immediately discourages exploration in regions of low probability of feasibility and the one-step-lookahead is only on the unpenalised objective function. In their work, they extend their formulation to batches and propose a discretisation-free Monte-Carlo approach based on [Wu and Frazier 2017].

4 OUR CONTRIBUTIONS

In this section, we introduce the original methods which we have developed to extend those in Section 3, which dealt with the fully coupled setting. The key point here is to leverage the freedom to independently evaluate constraints to try to build accurate models of the feasibility region for those constraints which are binding at the optimum. Intuitively, those constraints for which a small change would lead to a change on the location of the best feasible point are those which we should choose to evaluate, and moreover we should choose to evaluate them near the location where we believe the optimiser lies.

We extend cKG in a conceptually fairly simple way in Section 4.1, and introduce in Section 4.2 give a framework for incorporating EI into a decoupled setting. It is a framework in the sense that one could use a different acquisition function within the method we present. We have already introduced the methods which we will use as benchmarks, those being cKG and cEI.

4.1 Decoupled Constrained Knowledge Gradient

The modification which we make to the KG acquisition function is based on the observation that when we compute the KG value at a given point, we assume that we have sampled all of the constraints, and we thus update all of the GP models based on the random variables we draw from their current distribution. We could instead consider the KG value of a single constraint, which would mean that for a given point we would only simulate a realisation of a single constraint value or the objective function. We would then optimise the PF-weighted posterior mean of the objective function and subtract from this the PF-weighted current recommended location value, exactly as before. In order to present the acquisition function, we will introdue a variable m which counts the iteration number for the optimisation loop, rather than function evaluations for a given constraint, which may differ (m effectively counts total function evaluations). We then define $\kappa(m)$ to be the index of the constraint which we evaluate at a given iteration. Our acquisition function is thus

$$\text{dcKG}^k(x) = \mathbb{E}\left[\max_{x' \in X}\{\mu_y^{m+1}(x')\text{PF}^{m+1}(x')\} - \mu_y^{m+1}(x_r)\text{PF}^{m+1}(x_r^n)\middle| x_{m+1} = x, \kappa(m+1) = k\right], \quad (6)$$

where $k \in \{0, ..., K\}$ and we take $\text{dcKG}^0(x)$ to correspond to the KG value obtained from sampling the objective function. We then optimize each of these $K+1$ functions as before to find the optimisers, $\{x_*^k\}_{k=0}^K$, and corresponding dcKG values $\{\text{dcKG}^k(x_*^k)\}_{k=0}^K$. We then find constraint or objective with the highest dcKG value, $k_* = \max_k\{\text{dcKG}^k(x_*^k)\}$, and sample this constraint (or the objective) at the corresponding point x_*^k.

It is important to point out that we will in general have $k+1$ different optimisers, for the reason that the point which upon sampling will have the greatest information gain will differ for each constraint. We therefore are selecting the sampling points optimally for each constraint, rather than optimally across all of them at the same time, and it is this which allows us to learn relevant constraints and rapidly optimise the objective function. We also remark that if we sample the k^{th} constraint, then none of the other GP models are affected by this. For instance, in the product defining the quantity $\text{PF}(x)$, only one of the terms in this product will change upon sampling a given constraint. With this being said, dcKG is still a more expensive metric than KG for determining the utility of sampling a candidate design. The reason is that although we fit the same number $(K+1)$ of GP models per set of RVs we simulate in both methods, the optimisation in cKG is done all at once with the fitted GP models. This contrasts with dcKG, where we must optimise the PF-weighted posterior mean via L-BFGS $(K+1)$ times, because we need to at each candidate point optimise for the possibility of sampling each constraint.

The process of one loop iteration is summarised in the pseudocode below.

Algorithm 2: Decoupled Knowledge Gradient

```
1 Fit GP Models based on Current Data
2 Initialise KGVals = Empty List
3 for $k$ in range$(0, K+1)$ do
4     Optimise dcKG$^k$ acquisition function via L-BFGS
5     Add optimal value to KGVals
6 end
7 Select highest index in KGVals
8 Evaluate Corresponding Constraint/Objective
9 Return Data
```

4.2 Decoupled Expected Improvement

Suppose that in optimising a function, we have determined the point $x^* \in \mathcal{X}$ which optimises the cEI value. In light of the discussion at the beginning of Section 4, we would consider those constraints which are most relevant to be those which could have the greatest effect on the quantity $PF(x)$. Such constraints will be those for which $\mu^k(x^*)$ is small or $\sigma^k(x^*)$ is large because, in our GP model, the probability that these constraints will be violated will be higher than that for other constraints. These constraints are those which matter, because as stated in the Opportunity Cost from Section 1, if a point is infeasible, then we introduce a penalty value which aims to make the value of an infeasible point worse than that of any possible feasible point. Hence, an evaluation of these constraints is most likely to have a significant effect on the belief about the quality of the point.

To formalise this, we can consider the set $\{\mathbb{P}[c_k(x) \geq 0]\}_{k=1}^{K}$, which is the list of probabilities that a constraint is *infeasible*. A simple way of quickly trying to learn the constraint surface near x^* is then to place this list into descending order, and run through it evaluating each constraint in the corresponding order until we reach one which fails.

This is somewhat inefficient, because we may well find that the objective function is actually not any better than our current best location, and so running through all of the constraints prior to evaluating the objective function would be wasting our budget. The method we introduce therefore works as before but takes a parameter δ as an argument, which gives the minimum probability of infeasibility at which we should evaluate the objective function instead of a constraint. This means that the value point at which we evaluate the objective is when we reach $1 \leq k \leq K$ with $\mathbb{P}[c_k(x) \geq 0] < \delta$. If no point is infeasible, then as above we should finish by evaluating the objective function. In our simulations, we decided that when the probability of a given constraint being violated is less than 0.1, it becomes more informative to evaluate the objective function, so we set $\delta = 0.1$. We then take the point of view of a risk *averse* user, and if the objective function is an improvement over the current best sampled, we continue to run through the list of constraints in order to check feasibility. If a constraint fails or we have evaluated all of them, we then restart the loop by selecting a new point according to the cEI acquisition function until our budget has been exhausted.

It is important to point out that although we are risk averse in our selection of evaluation points in the BO process, we still recommend the PF-weighted maximum of the GP model to the user at the end in our simulations. One might reasonably ask whether we should be so risk averse in the BO process given that our final recommendation assumes a risk-neutral user. This was considered, and so some simulations were done using a minimum probability of infeasibility $\tilde{\delta} < \delta$ after which we would not evaluate any more of the constraints. We discovered that this led to the common difficulty in EI of a myopic approach which failed to properly explore the design space. It is why as our final algorithm we chose to use the seemingly less efficient approach of running through all constraints until violation (or success if the point turns out to be feasible).

The dEI procedure is summarised in the pseudocode below.

Algorithm 3: Decoupled Expected Improvement

```
Fit GP Models based on Current Data
Optimise cEI acquisition function via L-BFGS
InfeasibilityList = sorted list of probabilities of constraint violation
Initialise i = 0
while $i < K$ do
    if InfeasibilityList[$i$] > $\delta$ then
        yNew = Evaluate Constraint Corresponding to InfeasibilityList[$i$]
        Fit Constraint GP Model
        if yNew > 0 then
            end While
        else
            $i = i + 1$
        end
    end
    if Objective Not Yet Evaluated and InfeasibilityList[$i$] < $\delta$ then
        yNew = Evaluate Objective Function
        Fit Objective GP Model
        if yNew > Current Best then
            $i = i$
        else
            end While
        end
    end
    else
        yNew = Evaluate Constraint Corresponding to InfeasibilityList[$i$]
        Fit Constraint GP Model
        if yNew > 0 then
            end While
        else
            $i = i + 1$
        end
    end
end
if $i == K$ then
    Evaluate Objective
end
Return Data
```

4.3 EI+KG

We also developed a third related approach, that we called EI+KG. In this approach after selecting a candidate design through the cEI acquisition function, we evaluate only one function based on the dcKG value. The intuitive reason that one might hope this approach could perform at least at well as evaluating constraints based on constraint violation probability is that we spend fewer evaluations concentrated on one single design, but instead we can re-select the candidate design that we feel is most informative more often. Furthermore, the non-myopic nature of the dcKG metric may select a constraint to evaluate in a way which is more informative for future evaluations rather than quite pessimistically running through constraints in the order of constraint violation probability at the current iteration. One would not expect this method to outperform the full dcKG algorithm, because the point is not optimised for the constraint we end up evaluating. However, we have obtained promising numerical results for this method, suggesting it can compete with dcKG. EI+KG also has the benefit, as with the constraint violation formulation of our decoupled EI algorithm, of being much less computationally expensive than the dcKG method. This is because at each iteration we only need to calculate the dcKG values at a single point, rather than at many points to optimise it.

5 RESULTS AND DISCUSSION

In this section, we compare our three decoupled approaches, namely: decoupled Knowledge Gradient, decoupled Expected Improvement, and the hybrid method EI+KG against one another on synthetic test functions. We also compare them against the following existing coupled approaches: constrained Knowledge Gradient (cKG) [Ungredda and Branke 2024] and constrained Expected Improvement (cEI) [Gardner et al. 2014].

cKG was implemented using code provided in the GitHub of [Ungredda and Branke 2024], and cEI was implemented using BoTorch [Balandat et al. 2020].

We also tested the Mystery function with 8 additional redundant constraints (redundant in the sense they are positive at all times). In this setting we would expect decoupled methods to perform well against coupled approaches, making it a good test of our decoupled methods.

5.1 Both test functions: plot accuracy

We tested the algorithms on two synthetic decoupled constrained test functions: the Mystery Function and Test Function 2 from [Sasena 2002] (further details on these functions are available in the Appendix A). We did our tests without noise and with an initial design sample of 6, results were averaged over 50 replications each with random initial design points.

Our results are shown in Figures 1 and 2. Fig. 1 (**top row**) present shaded plots that show the landscape of the objective functions including the infeasible regions. Note that Mystery function has one non-linear constraint and Test Function 2 has a more complex feasible region created by three constraints.

Fig. 1 (**bottom row**) shows the convergence towards the optimal value for each method over iterations.

- These graphs show that our three **decoupled** methods dcKG, dcEI and EI + KG, outperform cKG on both test functions (the under-performance of cKG is surprising as in cEI paper it demonstrates improved performance compared to cEI but we weren't able to recreate this).
- dcEI suffers from high volatility due to sometimes predicting the best point to be unfeasible in some runs. This problem would need to be investigated in future work, but currently results in performance not being as strong as the other 2 of our methods in both test problems (dcEI is even performing worse than cEI).
- Then dcKG, cEI and EI + KG all perform very similarly on both test problems. This could be expected for the mystery function as the optimum is on the boundary of the only constraint, but slightly surprising for Test function 2 as one of the three constraints isn't active at the optimum. In future work we would run tests on more complex functions that would really benefit from a decoupled approach. We have done this with one function in the next sub section by adding redundant constraints to the mystery function.

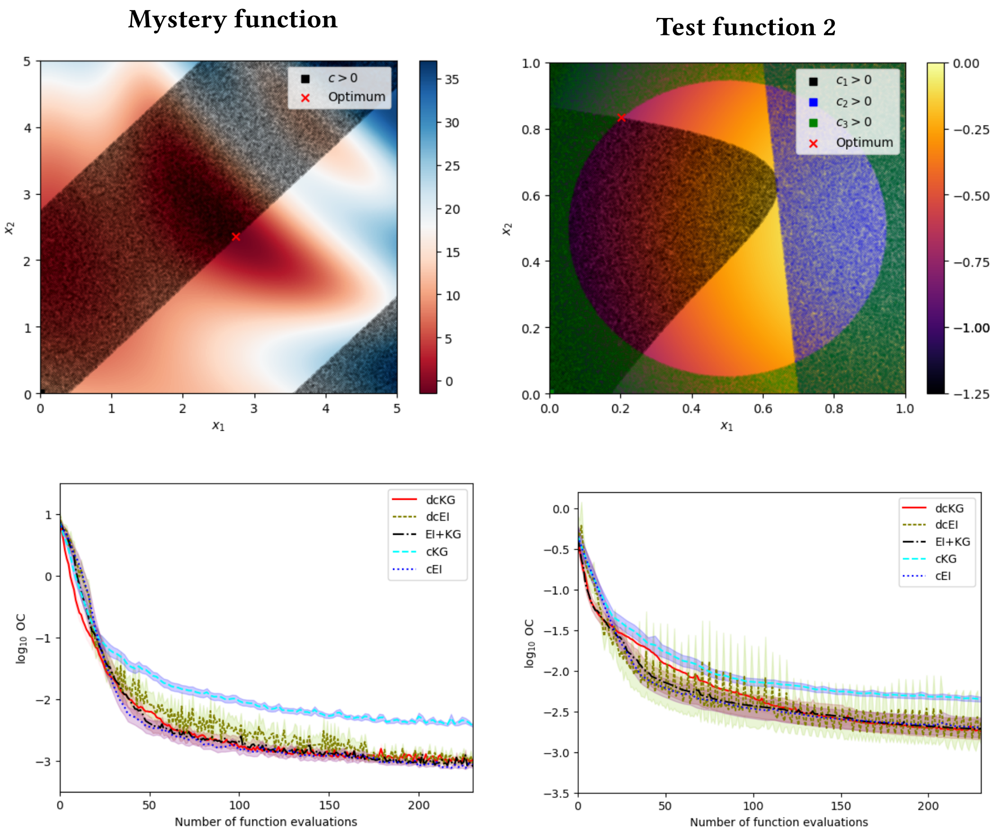

Fig. 1. The (**top row**) shows the feasible and infeasible (shaded) regions of the test functions. (**Bottom row**) shows the mean and mean squared error over runs of the difference between current value and the optimum.

5.2 Both test functions: plot proportion of each function evaluations

Fig. 2 shows plots of the proportion of evaluations at the objective function and the constraints at each iteration, across all runs for each method.

The (**top row**) shows this for the Mystery Function, and here we got results that were to be expected with equal proportion going to both the objective and the single constraint. This was expected because the optimum is on the constraint boundary so to find improvement will require sampling both the constraint and the objective function and any new point could easily be infeasible.

The (**bottom row**) shows results for Test Function 2.

- For Test Function 2, we expected to have equal levels of sampling of the objective function, constraint 1 and constraint 3 and less of constraint 2 (because it is positive in a large area around the optimum). We saw this behaviour with dcKG, however even after more than 100 iterations it still sampled constraint 2, which is slightly unexpected as it doesn't appear necessary.
- This could be due to the method still exploring the space as with a different function there could be a region in the infeasible area of which is feasible and of a higher value. In future

work this behaviour would be investigated further and with functions that do have these "feasible pockets".

- The other two methods do not exhibit the same behaviour, this makes sense for dcEI as this method evaluates all constraints when it finds a point with a higher value than the current best forcing it to approach an even proportion as it closes in on the optimum.
- The fact EI + KG isn't able to determine which constraint matters as well as dcKG is a feature needing further investigation, perhaps forcing KG values to be calculated at only one point impacts the selection quality more than anticipated.

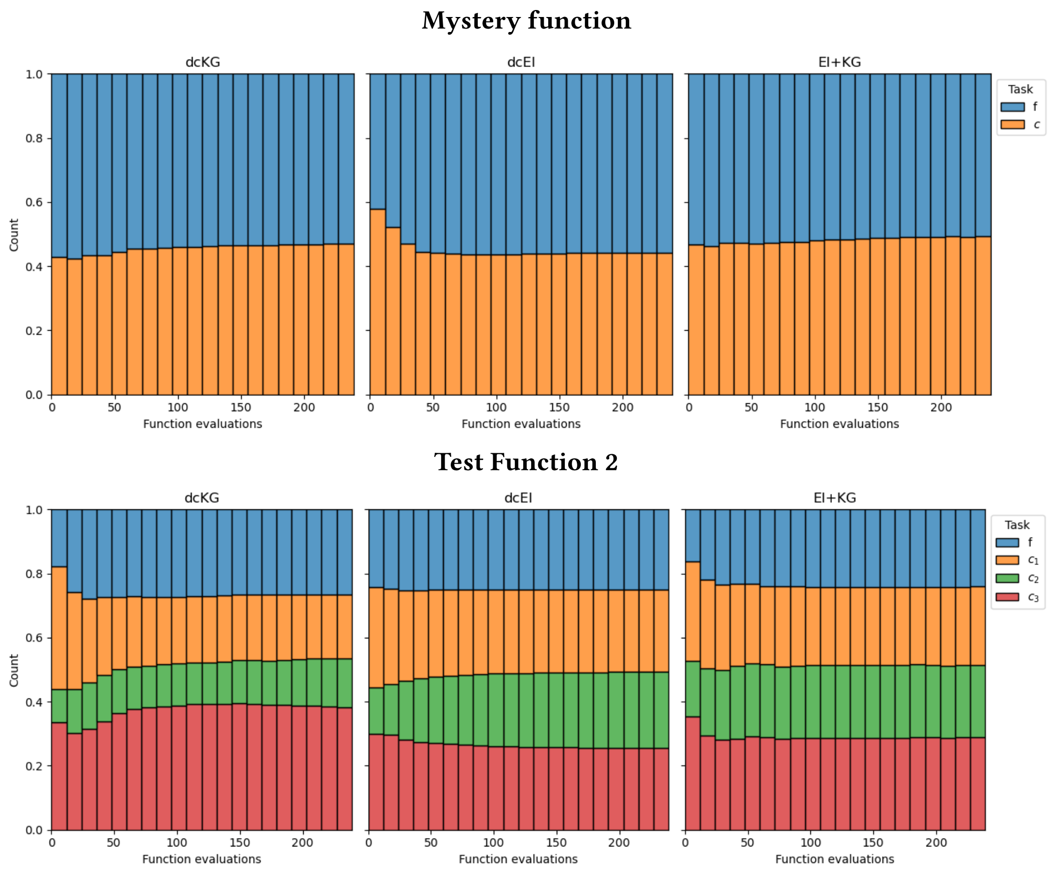

Fig. 2. Shows the proportion of function evaluations being of the objective function or the constraint on each iteration across the realisations

5.3 Mystery Function with redundant constraints

In this subsection we tested our methods on the Mystery Function with 8 additional redundant (always positive) constraints. This creates a situation where our decoupled methods should have a clear and obvious advantage against coupled approaches. Our results are shown in Figure 3.

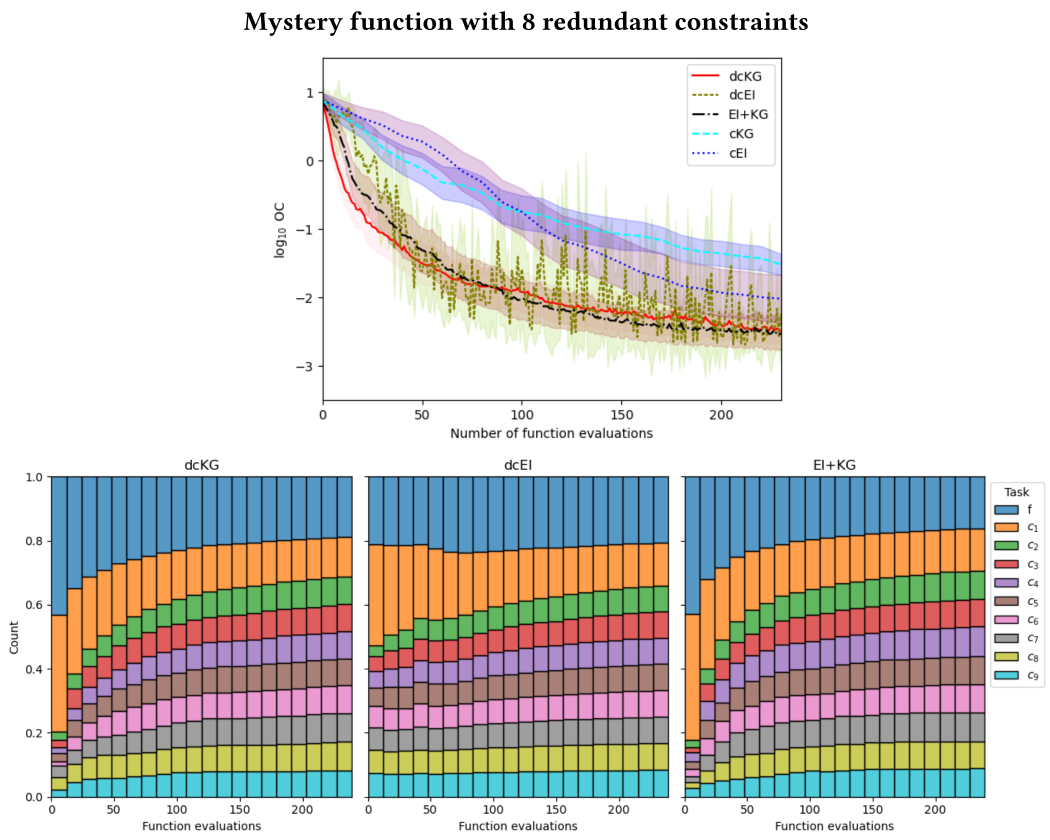

Fig. 3. (**top row**) shows the mean and mean squared error over runs of the difference between current value and the optimum. (**bottom row**) shows the proportion of function evaluations being of the objective function or the constraint on each iteration across the realisations

Fig. 3 (**top row**) shows the convergence towards the optimal value for each method over iterations.

- This plot shows that our three **decoupled** methods all perform much better than the two **coupled** methods. It is not surprising since decoupled methods are able to avoid sampling the 8 redundant constraints.
- No method in particular stands out as being the best.
- However, dcEI is still struggling with the volatility it was displaying on the previous two test functions.

The (**bottom row**) shows plots of the proportion of evaluations at the objective function and the constraints at each iteration, across all runs for each method.

- dcKG and EI + KG both perform similarly on this test function: both evaluate the objective function and the first constraint most, which is what we would expect.
- However, especially as the function evaluations increase, both methods start sampling the redundant constraints more and more. This is something that needs to be investigated in further work but could be caused by the algorithms turning to more exploration of the space since they are already close to the optimum and are now searching for "feasible pockets"

within the infeasible regions where it maybe possible to improve on the current maximum (of course our function has no such regions).
- dcEI evaluates the redundant constraints. This is expected because when dcEI finds a point that improves on its current best, it has to sample every constraint and this will happen frequently sampling at the optimum.

6 CONCLUSION AND FURTHER WORK

For the problem of Bayesian optimisations, we have proposed three new methods:

- decoupled constrained Knowledge Gradient, an extension to constrained Knowledge Gradient;
- decoupled Expected Improvement, an extension to constrained Expected Improvement;
- hybrid EI + KG, a combination of deEI and dcKG that hopes to combine the speed of EI and the accuracy of KG.

Empirically we are able to show that the decoupled methods perform well on a problem containing many constraints where only a few of them matter (see Section 5.3), and at least perform equally as well as coupled methods when all constraints are relevant at the optimum.

Despite promising results, the data does raise some questions that need to be addressed in future work. The poor performance of cKG compared to the results obtained in the original paper [Ungredda and Branke 2024], if we are able to recreate their result we may also see an increase in the performance of our dcKG method. The volatility of dcEI is holding back the performance of the method we believe this is a problem that can be solved in future work and would make dcEI then a strong performer whilst also being cheap to evaluate.

We would also like to propose some steps that were delayed for future work. The most important (and perhaps most difficult) aspect is the investigation of the extend to which our algorithms can be applied to high-dimensional problems (and the majority of real-world problems are very high-dimensional, consider MOPTA, for example). High dimensions impose the so-called "curse of dimensionality" restriction, meaning that in order to sample a population of points from some area we need amount of points, growing exponentially, which makes our even now computationally expensive algorithms even more heavy.

One approach that we believe to be promising is called TURBO [D. Eriksson and Poloczek 2019]. It was later extended to constrained problems in [Eriksson and Poloczek 2021]. We believe a good next step will be to compare the performance of the proposed decoupled and coupled algorithms on some high-dimensional problem, then choose the best and implement it in TURBO.

ACKNOWLEDGMENTS

The authors would like to thank Juergen Branke and Juan Ungredda for their tireless support throughout the work required to produce this report.

REFERENCES

Maximilian Balandat, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G Wilson, and Eytan Bakshy. 2020. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems*, H Larochelle, M Ranzato, R Hadsell, M F Balcan, and H Lin (Eds.), Vol. 33. Curran Associates, Inc., 21524–21538. https://proceedings.neurips.cc/paper_files/paper/2020/file/f5b1b89d98b7286673128a5fb112cb9a-Paper.pdf

J. Gardner R. D. Turner D. Eriksson, M. Pearce and M. Poloczek. 2019. Scalable global optimization via local Bayesian optimization. *Advances in Neural Information Processing Systems* 32 (2019), 5497–5508.

D. Eriksson and M. Poloczek. 2021. Scalable Constrained Bayesian Optimization. *arxiv.org* (2021). https://arxiv.org/pdf/2002.08526

P Frazier. 2018. A Tutorial on Bayesian Optimization. *ArXiv* abs/1807.02811 (2018). https://api.semanticscholar.org/CorpusID:49656213

Jacob R Gardner, Matt J Kusner, Zhixiang Xu, Kilian Q Weinberger, and John P Cunningham. 2014. Bayesian optimization with inequality constraints. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32 (ICML'14)*. JMLR.org, II–937–II–945.

Shali Jiang, Daniel Jiang, Maximilian Balandat, Brian Karrer, Jacob Gardner, and Roman Garnett. 2020. Efficient Nonmyopic Bayesian Optimization via One-Shot Multi-Step Trees. In *Advances in Neural Information Processing Systems*, H Larochelle, M Ranzato, R Hadsell, M F Balcan, and H Lin (Eds.), Vol. 33. Curran Associates, Inc., 18039–18049. https://proceedings.neurips.cc/paper_files/paper/2020/file/d1d5923fc822531bbfd9d87d4760914b-Paper.pdf

Donald R Jones. 2008. Large-scale multi-disciplinary mass optimization in the auto industry. In *MOPTA 2008 Conference*, Vol. 64.

T T Joy, S Rana, S Gupta, and S Venkatesh. 2016. Hyperparameter tuning for big data using Bayesian optimisation. In *2016 23rd International Conference on Pattern Recognition (ICPR)*. 2574–2579. https://doi.org/10.1109/ICPR.2016.7900023

Benjamin Letham, Brian Karrer, Guilherme Ottoni, and Eytan Bakshy. 2019. Constrained Bayesian optimization with noisy experiments. *Bayesian Analysis* 14, 2 (2019), 495–519. https://doi.org/10.1214/18-BA1110

R Marchant and F Ramos. 2012. Bayesian optimisation for Intelligent Environmental Monitoring. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2242–2249. https://doi.org/10.1109/IROS.2012.6385653

José Miguel Hernández-Lobato, Michael A Gelbart, Ryan P Adams, Matthew W Hoffman, Zoubin Ghahramani, J M Hernández-Lobato, M A Gelbart, R P Adams, M W Hoffman, and Z Ghahramani Hernández-Lobato. 2014. Predictive entropy search for efficient global optimization of black-box functions. *Advances in Neural Information Processing Systems* (2014), 918–926.

José Miguel Hernández-Lobato, Michael A Gelbart, Ryan P Adams, Matthew W Hoffman, Zoubin Ghahramani, J M Hernández-Lobato, M A Gelbart, R P Adams, M W Hoffman, and Z Ghahramani Hernández-Lobato. 2016. A General Framework for Constrained Bayesian Optimization using Information-based Search. *Journal of Machine Learning Research* 17 (2016), 1–53.

José Miguel Hernández-Lobato, Michael A Gelbart, Matthew W Hoffman, Ryan P Adams, and Zoubin Ghahramani. 2015. Predictive Entropy Search for Bayesian Optimization with Unknown Constraints. In *International Conference on Machine Learning*.

Quoc Phong Nguyen, Wan Theng, Ruth Chew, Le Song, Bryan Kian, Hsiang Low, and Patrick Jaillet. 2024. Optimistic Bayesian Optimization with Unknown Constraints. In *The Twelfth International Conference on Learning Representations*.

Matteo Papini, Alberto Maria Metelli, Lorenzo Lupo, and Marcello Restelli. 2019. Optimistic Policy Optimization via Multiple Importance Sampling. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 4989–4999. https://proceedings.mlr.press/v97/papini19a.html

Victor Picheny, Robert B Gramacy, Stefan M Wild, and Sébastien Le Digabel. 2016. Bayesian optimization under mixed constraints with a slack-variable augmented Lagrangian. In *Neural Information Processing Systems*.

Carl Edward Rasmussen and Christopher K I Williams. 2005. *Gaussian Processes for Machine Learning*. The MIT Press. https://doi.org/10.7551/mitpress/3206.001.0001

Michael James Sasena. 2002. *Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations*. Ph. D. Dissertation.

Matthias Schonlau, William Welch, and Donald Jones. 1998. Global versus local search in constrained optimization of computer models. In *Lect Notes Monogr Ser*. Vol. 34. 11–25. https://doi.org/10.1214/lnms/1215456182

Warren Scott, Peter Frazier, and Warren Powell. 2011. The Correlated Knowledge Gradient for Simulation Optimization of Continuous Parameters using Gaussian Process Regression. *SIAM Journal on Optimization* 21, 3 (2011), 996–1026. https://doi.org/10.1137/100801275

Juan Ungredda and Juergen Branke. 2024. Bayesian Optimisation for Constrained Problems. *ACM Transactions on Modeling and Computer Simulation* 34, 2 (1 2024), 1–26. https://doi.org/10.1145/3641544

Juan Ignacio Ungredda Suárez. 2022. *Efficient Information Collection for Bayesian Optimisation with Constraints, User Preferences and Input Uncertainty*. Ph. D. Dissertation. http://wrap.warwick.ac.uk/180092

Samee ur Rehman, Matthijs Langelaar, and Fred van Keulen. 2014. Efficient Kriging-based robust optimization of unconstrained problems. *Journal of Computational Science* 5, 6 (2014), 872–881. https://doi.org/10.1016/j.jocs.2014.04.005

S. Liu W. Chen and K. Tang. 2021. A new knowledge gradient-based method for constrained bayesian optimization. *Arxiv.org* (2021). https://arxiv.org/abs/2101.08743

Jian Wu and Peter Frazier. 2016. The Parallel Knowledge Gradient Method for Batch Bayesian Optimization. In *Advances in Neural Information Processing Systems*, D Lee, M Sugiyama, U Luxburg, I Guyon, and R Garnett (Eds.), Vol. 29. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2016/file/18d10dc6e666eab6de9215ae5b3d54df-Paper.pdf

J. Wu and P. I. Frazier. 2017. Discretization-free knowledge gradient methods for Bayesian optimization. *Discretization-free knowledge gradient methods for Bayesian optimization* (2017).

7 APPENDIX

A TEST FUNCTIONS

A.1 Mystery Function

A two-dimensional problem with one constraint,

$$\min f(x) = 2 + 0.01(x_2 - x_1^2)^2 + (1 - x_1)^2 + 2(2 - x_2)^2 + 7\sin(0.5x_1)\sin(0.7x_1x_2)$$

subject to

$$-\sin\left(x_1 - x_2 - \frac{\pi}{8}\right) \leq 0$$

on the domain

$$x_1 \in [0, 5]; x_2 \in [0, 5];$$

A.2 Test Function 2

The two-dimensional problem with three constraints,

$$\min f(x) = -(x_1 - 1)^2 - (x_2 - 0.5)^2$$

subject to

$$(x_1 - 3)^2 + (x_2 + 1)^2 - 12 \leq 0$$
$$10x_1 + x_2 - 7 \leq 0$$
$$(x_1 - 0.5)^2 + (x_2 - 0.5)^2 - 0.2 \leq 0$$

on the domain

$$x_1 \in [0, 1]; x_2 \in [0, 1];$$

B FITTING A GAUSSIAN PROCESS

We discuss how to fit an $n-$dimensional Gaussian Process when given the training dataset denoted by $D_f = \{(x_i, y_i)\}_{i=1}^n$. Note that "fitting" a GP means finding its mean and variance. Note that kernel (correlation structure) is prior, and we essentially impose a correlation between the objective function values at different points, so we need to use some prior understanding of the problem setting to give reasonable hyperparameters for the kernel function. A typical choice is squared exponential kernel, which for two design vectors $\bar{x}, \bar{x}'$ takes the form:

$$k(\bar{x}, \bar{x}') = \sigma_f^2 \exp(-\sum_{i=1}^{n} \frac{1}{2} \frac{x_i - x_i'}{l_i^2}) + \sigma_{noise}^2 \delta(\bar{x}, \bar{x}'),$$

where we here use the notation $m(x)$ for the mean and $k(x, x')$ for the covariance.

Here $\sigma_f, \sigma_{noise}, l_i$ are hyperparameters that determine the smoothness of the functions. In practice, we estimate them via maximum likelihood. Note that even when dealing with noiseless data, it is often useful to suppose in the model that there is tiny noise, e.g. 10^{-4}. This often improves numerical stability during inference.

Suppose f is Gaussian process, then vector

$$(f(x_1), \ldots, f(x_n), f(x)) \sim N_{n+1}(0, \Sigma)$$

,

where

$$\Sigma = \begin{pmatrix} k(x_1, x_1) & \ldots & k(x_1, x_n) & k(x_1, x) \\ \ldots & & & \\ k(x_n, x_1) & \ldots & k(x_n, x_n) & k(x_n, x) \\ k(x, x_1) & \ldots & k(x, x_n) & k(x, x) \end{pmatrix} = \begin{pmatrix} k_{XX} & k_X(x, \bar{x}) \\ k_X(x, \bar{x})^T & k(x, x) \end{pmatrix}$$

Here we use notation

$$X = (x_1, \dots, x_n),$$

$$K_{XX} = \begin{pmatrix} k(x_1, x_1) & \dots & k(x_1, x_n) \\ \dots & & \\ k(x_n, x_1) & \dots & k(x_n, x_n) \end{pmatrix}$$

$$k_X(x, \bar{x}) = (k(x_1, x), \dots, k(x_n, x))^T,$$

$$F = (f(x_1), \dots, f(x_n))^T.$$

Then we can derive through certain properties of Gaussians that the exact expression for the conditional distribution is:

$$f(x)|(f(x_1), \dots, f(x_n)) \sim N(\bar{m}(x, \bar{x}), \bar{k}(x, \bar{x})),$$

where

$$\bar{m}(x, \bar{x}) = k_X(x, \bar{x})^T K_{XX}^{-1} F,$$

$$\bar{k}(x, \bar{x}) = k(x, x) - k_X(x, \bar{x})^T K_{XX}^{-1} k_X(x, \bar{x})$$

In the case of noisy observations, that is $y_i = f(x_i) + N(0, \sigma^2)$, the mutual distribution is the following:

$$(f(x_1), \dots, f(x_n), f(x)) \sim N_{n+1}(0, \Sigma),$$

where

$$\Sigma = \begin{pmatrix} k_{XX} + \sigma^2 I & k_X(x, \bar{x}) \\ k_X(x, \bar{x})^T & k(x, x) \end{pmatrix}$$

Hence the conditional distribution is given by

$$f(x)|(f(x_1), \dots, f(x_n)) \sim N(\bar{m}(x, \bar{x}), \bar{k}(x, \bar{x})),$$

where

$$\bar{m}(x, \bar{x}) = k_X(x, \bar{x})^T (K_{XX} + \sigma^2 I)^{-1} F,$$

$$\bar{k}(x, \bar{x}) = k(x, x) - k_X(x, \bar{x})^T (K_{XX} + \sigma^2 I)^{-1} k_X(x, \bar{x})$$

Received June 2024; revised June 2024; accepted June 2024