Bayesian Optimisation for Constrained Problems

JUAN UNGREDDA, Mathematics for Real-World Systems, University of Warwick, UK
JUERGEN BRANKE, Warwick Business School, University of Warwick, UK

Many real-world optimisation problems such as hyperparameter tuning in machine learning or simulation-based optimisation can be formulated as expensive-to-evaluate black-box functions. A popular approach to tackle such problems is Bayesian optimisation, which builds a response surface model based on the data collected so far, and uses the mean and uncertainty predicted by the model to decide what information to collect next. In this paper, we propose a generalisation of the well-known Knowledge Gradient acquisition function that allows it to handle constraints. We empirically compare the new algorithm with four other state-of-the-art constrained Bayesian optimisation algorithms and demonstrate its superior performance. We also prove theoretical convergence in the infinite budget limit.

Additional Key Words and Phrases: Simulation Optimisation, Gaussian Processes, Bayesian Optimisation, Constraints

ACM Reference Format:
Juan Ungredda and Juergen Branke. 2021. Bayesian Optimisation for Constrained Problems. *ACM Trans. Model. Comput. Simul.* 1, 1 (January 2021), 28 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Expensive black-box optimisation problems are common in many areas, including

- simulation optimisation, where a solution is evaluated by a discrete-event simulation [Amaran et al. 2016],
- hyperparameter tuning, where an evaluation involves training a machine learning model [Hernández-Lobato et al. 2016],
- the optimisation of the control policy of a robot under performance and safety constraints [Berkenkamp et al. 2016], or
- engineering design optimisation [Forrester et al. 2008].

For such applications, Bayesian optimisation (BO) has shown to be a powerful and efficient tool. After collecting some initial data, BO constructs a surrogate model, usually a Gaussian process (GP). Then it iteratively uses an acquisition function to decide what data would be most valuable to collect next, explicitly balancing exploration (collecting more information about yet unexplored areas) and exploitation (evaluating solutions that are predicted to be good). After sampling the next solution, the Gaussian process model is updated with the new information and the process is repeated until the available budget of evaluations has been consumed.

While many different BO algorithms have been proposed in the literature [Frazier 2018], handling constraints in BO is much less explored. The standard approach is to build separate surrogate models for the constraints, and then simply multiplying the value of an acquisition function for unconstrained problems with a solution's probability of being feasible (e.g., Chen et al. [2021]; Schonlau et al. [1998]). However this doesn't fully capture the value of the information gained about the constraints from sampling a solution.

Authors' addresses: Juan Ungredda, j.ungredda@warwick.ac.uk, Mathematics for Real-World Systems, University of Warwick, Coventry, UK, CV4 7AL; Juergen Branke, juergen.branke@wbs.ac.uk, Warwick Business School, University of Warwick, Coventry, UK, CV4 7AL.

XXXX-XXXX/2021/1-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

In this paper, we propose a generalisation of the well-known Knowledge Gradient (KG) acquisition function that fully takes into account the value of constraint information when deciding which solution to sample next. In particular, we make the following contributions.

(1) We develop a generalisation of the Knowledge Gradient acquisition function, called *constrained Knowledge Gradient* (cKG), capable of handling constraints. Different from most other acquisition functions proposed in the literature, it fully takes into account the value for constraint information when deciding where to sample next.
(2) We show how cKG can be efficiently computed.
(3) We prove for discrete spaces that in the limit cKG converges to the optimal solution.
(4) We apply our proposed approach to a variety of test problems with and without noise in the objective and the constraints, and show that cKG outperforms other available BO approaches for constrained problems.

We start with an overview of related work in Section 2, followed by a formal definition of the problem in Section 3. Section 4 explains the statistical models, presents the suggested sampling procedure, explains how it can be computed efficiently, and outlines some theoretical properties. Section 5 discusses the case of a risk averse decision maker when the algorithm needs to return the best sampled solution. We report on numerical experiments in Section 6. Finally, the paper concludes with a summary and some suggestions for future work.

2 LITERATURE REVIEW

Bayesian optimisation (BO) has gained wide popularity, especially for problems involving expensive black-box functions, for a comprehensive introduction see Frazier [2018] and Shahriari et al. [2016]. Although most work has focused on unconstrained problems, some extensions to constrained optimisation problems exist.

Many of the approaches are based on the famous Expected Improvement (EI) acquisition function [Jones et al. 1998]. Schonlau et al. [1998] and Gardner et al. [2014] extended EI to constrained EI (cEI) by computing the expected improvement of a point x over the best feasible point and multiplying it by its probability of being feasible. Bagheri et al. [2017] proposed a modified combination of probability of feasibility with EI that makes it easier to find solutions on the feasibility boundary. Other methods rely on relaxing the constraints instead of modifying the infill criteria, Gramacy et al. [2016] proposed an augmented Lagrangian approach that includes constraints as penalties in the objective function. Picheny et al. [2016] refined the previous approach by introducing slack variables and achieve better performance on equality constraints. Kleijnen et al. [2021] considered using the "Karush-Kuhn-Tucker" conditions to determine optimality and feasibility of a design vector.

Lam and Willcox [2017] proposed a lookahead approach for the value of feasibility information, selecting the next evaluation in order to maximise the long-term feasible increase of the objective function. This was formulated using dynamic programming where each simulated step gives a reward following cEI. Recently, Zhang et al. [2021] improved over Lam and Willcox [2017] by considering the likelihood ratio method to better estimate the gradients of the acquisition function. This allows for a faster computation and enables both sequential and batch settings. Letham et al. [2017] extended Expected Improvement to noisy observations (NEI) and noisy constraints by iterating the expectation over possible posterior distributions. For noise-free observations, their approach reduces to the original cEI.

The Knowledge Gradient (KG) policy [Scott et al. 2011] is an acquisition function that aims at maximising the new predicted optimal performance after one new sample, and it can be equally applied to deterministic as well as noisy objective functions. Compared to some other acquisition functions, Picheny et al. [2013] showed that KG is empirically superior, especially for larger levels of noise. Chen et al. [2021] recently proposed an extension of KG to constraints by multiplying the KG value of any new sampling location by its probability of feasibility. While these approaches allow to consider noise in the observations and constraints, they only use the current feasibility information and ignore the value of additional constraint information from sampling another solution.

Other acquisition functions have also been extended to tackle constraints. Hernández-Lobato et al. [2016] extended Predictive Entropy Search [Hernandez-Lobato et al. 2014] to constraints. This acquisition criterion involves computing the expected entropy reduction of the global solution to the constrained optimisation problem. Eriksson and Poloczek [2021] proposed SCBO, which extends Thompson sampling (TS) for constrained optimisation and also proposed a trust region to limit the search to locations close to the global optimum. Picheny [2014] proposed an optimisation strategy where the benefit of a new sample is measured by the reduction of the expected volume of the excursion set which provides a measure of uncertainty on the minimiser location where constraints can be incorporated in the formulation by a solution's probability of being feasible. However this can only be computed approximately using numerical integration. Candelieri [2019] proposed a two-stage approach where the feasible region is estimated during the first stage by a support-vector classifier, then the second stage uses the estimated boundaries and maximises the objective function value using the Upper Confidence Bound (UCB) as acquisition function.

The new cKG acquisition function is derived by reconsidering the assumption made in most constrained acquisition functions where the Bayesian optimisation algorithm returns only a previously evaluated design as a final solution. This can be considered as a sensible assumption if the decision maker is highly risk-averse and evaluations are noise-free, but if the decision-maker is willing to tolerate some risk then we may report a design that has uncertainty attached to it. Moreover, if evaluations have noise then the final recommended solution is necessarily uncertain. Therefore, we replace this assumption by allowing the algorithm to return any solution, even if it has not been previously evaluated.

Furthermore, most approaches build the constrained acquisition function by multiplying the value of an acquisition function for unconstrained problems with a solution's probability of being feasible. While this ensures that the acquisition function tends to sample designs with high probability of being feasible, it ignores the potential benefit of sampling infeasible solutions to recommend better future solutions. cKG takes this benefit into account by considering the possibility that as a result of the new sample, the predicted GP mean and constraints at other regions may change, therefore changing the predicted optimum location. This better captures the value of the information gained about the constraints from sampling a solution.

Table 1 summarises the characteristics of the approaches discussed in the paper, with one important criterion being the number of lookahead steps in the *constraints* the acquisition function performs to quantify the value of a sampling decision. "Zero-Step" refers to methods that only consider the estimated probability of feasibility of the current candidate design vector, e.g. cEI directly penalises the EI by multiplying it with the current estimate of the solution candidate's probability of feasibility, thus any design vector in infeasible regions is directly penalised. As a result, these methods avoid sampling design vectors outside feasible regions and may have difficulties to reach solutions close to the feasibility boundary. Moreover, they ignore the value of infeasible design vectors may have in reducing model uncertainty and allowing feasible design vectors to be found in future iterations. "One-Step" methods improve upon "Zero-Step" methods by considering the impact of the candidate design vector for the performance in the next iteration. PESC and cKG are the only two approaches that quantify the gain from a sampling decision in the "One-Step" category and that can also handle noisy settings. Lastly, "Multi-Step" methods assess the impact of a sampling decision more than one-step ahead. These methods are computationally more expensive, however they tend to outperform their myopic counterparts. Notice that Table 1 shows the current lack of research for "Multi-Step" algorithms that can effectively deal with noisy settings.

3 PROBLEM DEFINITION

We want to find the optimiser x^* of a black-box function $f : \mathbb{X} \rightarrow \mathbb{R}$ with constraints $c_k : \mathbb{X} \rightarrow \mathbb{R}$, i.e.,

Table 1. Summary of the approaches discussed in the paper. We consider three levels for the numbers of lookahead steps on the constraints. “Zero-Step" refers to methods that only estimate the feasibility of the candidate design vector, “One-Step" refers to methods that estimate the impact of a candidate design vector in the next iteration of the algorithm, and “Multi-Step" is used for methods that assess the impact in further iterations. Additionally, we report for each method the underlying acquisition function approach, e.g. cEI is based on the EI acquisition function. We consider Expected Improvement (EI), Upper Confidence Bound (UCB), Knowledge Gradient (KG), Excursion Volume (EV), and Entropy Search (ES) based methods. The top part of the table contains approaches that can only deal with deterministic objective functions and constraints, while the methods in the lower part are able to handle also noisy objective functions and constraints.

	Underlying	Number of Lookahead Steps on the **Constraints**		
		Zero-Step	One-Step	Multi-Step
Deterministic Objective and Constraints	EI	Gardner et al. [2014] (cEI) Kleijnen et al. [2021] Gramacy et al. [2016] Picheny et al. [2016]		Lam and Willcox [2017] Zhang et al. [2021]
	UCB	Candelieri [2019]		
	EV		Picheny [2014]	
Noisy Objective or Constraints	EI	Letham et al. [2017] (NEI)		
	ES		Hernández-Lobato et al. [2016] (PESC)	
	KG	Chen et al. [2021] (pKG)	cKG (this paper)	
	TS	Eriksson and Poloczek [2021] (SCBO)		

$$x^* = \arg\max_{x \in \mathbb{X}} f(x) \tag{1}$$

$$\text{s.t. } c_k(x) \leq 0\,, k = 1, \ldots, K. \tag{2}$$

The objective function f takes as arguments a design vector $x \in \mathbb{X} \subset \mathbb{R}^d$ and returns a *single* observation possibly corrupted by noise $y = f(x) + \epsilon$, where $\epsilon \sim N(0, \sigma_\epsilon^2)$, and a vector of constraint values $\mathbf{c} = [c_1(x), \ldots, c_K(x)]$, also possibly corrupted by noise, i.e., $\mathbf{c} = [c_1(x) + \epsilon_1, \ldots, c_k(x) + \epsilon_k, \ldots, c_K(x) + \epsilon_K]$ where $\epsilon_k \sim N(0, \sigma_{\epsilon_k}^2)$. We denote by x^* the optimiser of the non-noisy function under the non-noisy constraints $\{c_k\}_1^K$ and x^* is the true best solution that is possible to attain. Furthermore, we assume that f and $\mathbf{c}$ may be approximated by a Gaussian process process model (see Section 4.1).

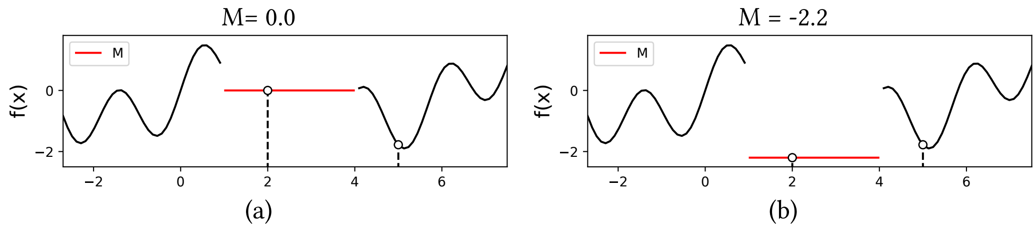


Fig. 1. objective function with a penalty value $M = 0$ (a) and $M = -2.2$ (b). Solutions $1 \leq x \leq 4$ are infeasible.

There is a total budget of B samples that can be spent. After consuming the budget, a recommended design, x_r, is returned to the user and its quality is determined by the difference in objective function to the best solution x^* given that x_r is feasible, i.e. $x \in F = \{x|c_k(x) \leq 0 \quad \forall k \in [1 \dots K]\}$. If x_r is not feasible then there is a penalty M for not having returned a feasible solution. Therefore, the performance may be measured as an Opportunity Cost (OC) to be minimised,

$$OC(x_r) = \begin{cases} f(x^*) - f(x_r) & \text{if } x_r \in F \\ f(x^*) - M & \text{otherwise.} \end{cases} \quad (3)$$

The value of the penalty for infeasibility, M, is problem and user dependent, and should in practice be set by an expert. The importance of this parameter is illustrated by Figure 1 shows the objective function with the penalty value M for unfeasible design vectors . Figure 1 (a) shows the objective function with a penalty $M = 0$. In this case, the penalty value set by the DM is higher than the objective function value in some feasible regions, and therefore, we may find that an unfeasible design vector may be preferable to a feasible design vector . In the case when the DM selects a design vector with an adequate penalty $M = -2.2$ (Figure 1(b)), any feasible design vector would present a favorable value compared to an unfeasible design vector .

As we show in Appendix 6.4, in the absence of such domain knowledge, M may be set to the minimum GP estimate of the objective function in the design space. Without loss of generality and to simplify later equations, we assume without loss of generality a penalty $M = 0$ in the remainder of this paper. Any other case may be transformed into the case $M = 0$ by subtracting M from the objective function. A derivation of cKG with general penalty M may be found in Appendix E. To test the proposed approach in Section 6 with $M = 0$, we compare on strictly positive test functions.

4 THE CKG ALGORITHM

We propose the Constrained Knowledge Gradient (cKG) algorithm which collects additional data taking into account constraints and potential noise in the objective function or constraints. In Section 4.1 we describe the statistical models for inferring the objective function and constraints. Section 4.2 explains the Knowledge Gradient for unconstrained problems proposed by Scott et al. [2011]. Then, Section 4.3 and 4.4 derive the cKG acquisition function and discuss its efficient implementation. Lastly, we summarise the overall cKG algorithm and discuss some of its properties in Sections 4.5 and 4.6, respectively.

4.1 Statistical Model

Let us denote all n design vectors sampled so far as $X = \{x_i\}_{i=1}^n$, the training data from the collection of objective function observations, $\mathscr{D}_f = \{(x, y)\}_{i=1}^n$, and constraints, $\mathscr{D}_c = \{(x, \mathbf{c})\}_{i=1}^n$. We model the objective function observations as a Gaussian process (GP) which is fully specified by a mean function $\mu_y^n(x) = \mathbb{E}[y(x)|\mathscr{D}_f]$ and its covariance $\mathrm{Cov}\left[y(x), y(x')|\mathscr{D}_f\right]$,

$$\mathbb{E}[y(x)|\mathcal{D}_f] = \mu_y^n(x)$$
$$= \mu_y^0(x) - k_y^0(x,X)(k_y^0(X,X) + I\sigma_\epsilon^2)^{-1}(Y - \mu_y^0(X)) \tag{4}$$
$$\text{Cov}\,[y(x), y(x')|\mathcal{D}_f] = k_y^n(x,x')$$
$$= k_y^0(x,x') - k_y^0(x,X)(k_y^0(X,X) + I\sigma_\epsilon^2)^{-1}k_y^0(X,x'). \tag{5}$$

Although we assume homoscedastic noise, it is also possible to perform inference assuming that the variance changes with the domain, $\sigma_\epsilon^2 = r(x)$, by modeling the log of the variance with a second Gaussian process [Kersting et al. 2007]. Similarly, each constraint is modelled as an independent GP over the training data $\mathcal{D}_c$ defined by a constraint mean function $\mu_k^n(x) = \mathbb{E}[c_k(x)|\mathcal{D}_c]$ and covariance $\text{Cov}\,[c_k(x), c_k(x')|\mathcal{D}_c]$. The prior mean is typically set to zero and the kernel allows the user to encode known properties such as smoothness and periodicity. We use the popular squared exponential kernel that assumes f and $\mathbf{c}$ are smooth functions, i.e., nearby x have similar outputs while widely separated points have unrelated outputs. Further details and alternative kernel functions can be found in Rasmussen and Williams [2006].

Although assuming independence works well for many constrained optimisation problems and has been widely used in the literature [Bagheri et al. 2017; Gardner et al. 2014; Letham et al. 2017], multi-output GP models may also be considered to model the correlation between the constraint functions and the objective [Álvarez et al. 2012]. Overall, the multi-output GP may provide a lower probability of feasibility estimation error than the independent model and may improve the sample efficiency of the Bayesian optimisation algorithm [Berkenkamp et al. 2021; Pelamatti et al. 2022].

4.2 Knowledge Gradient for Unconstrained problems (KG)

Scott et al. [2011] proposed the knowledge gradient with correlated beliefs (KG) acquisition function. This acquisition function compares the old highest value of the model posterior mean with the new highest value given the decision to take the next sample at design vector x^{n+1}. The design vector sampled is then the one that maximises

$$\text{KG}(x) = \mathbb{E}[\max_{x''\in\mathbb{X}} \left\{\mu_y^{n+1}(x'')\right\} - \max_{x'\in\mathbb{X}} \left\{\mu_y^n(x')\right\} | x^{n+1} = x]. \tag{6}$$

Different approaches have been developed to solve Eqn. 6. Scott et al. [2011] propose discretising the design space and solving a series of linear problems. However, increasing the number of dimensions requires more discretisation points and thus renders this approach computationally expensive. A more recent approach involves Monte-Carlo sampling of the observed value at design vector x^{n+1}, and solving an inner optimisation problem for each sample to identify the best posterior mean [Wu and Frazier 2017]. Using Monte-Carlo samples improves the scalability of the algorithm in terms of number of dimensions, but the inner optimisation still leads to high computational complexity. Pearce et al. [2020] propose a hybrid between between both approaches that consists of obtaining high value points from the predictive posterior GP mean that would serve as a discretisation. Combining both approaches allows to leverage the scalability of the Monte-Carlo based acquisition function and the computational performance of discretising the design space.

4.3 Knowledge Gradient for Constrained Problems (cKG)

At the end of the algorithm we must recommend a final design vector, x_r. Assuming a risk-neutral user, the utility of a design vector is the expected objective performance, $\mu_y^B(x)$, if feasible, and zero (M) if infeasible. Therefore, if the constraints and objective are independent, a recommended solution x_r may be obtained by

$$x_r = \arg\max_{x \in \mathbb{X}} \mu_y^B(x)\mathbb{P}[\mathbf{c}^B(x) \leq 0], \tag{7}$$

where $\mathbb{P}[\mathbf{c}^B(x) \leq 0]$ is the probability of feasibility of a design x. For each constraint k, $\mathbb{P}[c_k^B(x) \leq 0|\mathcal{D}_c]$ can be evaluated by a univariate Gaussian cumulative distribution (Φ) as

$$\mathbb{P}[c_k^B(x) \leq 0|\mathcal{D}_c] = \Phi\left(\frac{-\mu_k^B(x)}{\sqrt{k_k^B(x,x)}}\right).$$

Therefore, when design vectors have a low value of $\mu_k^B(x)$, the probability of feasibility tends closer to one. On the other hand, if $\mu_k^B(x)$ is high, the probability of feasibility tends closer to zero. Notably, the boundary of feasibility may be found when $\mu_k^B(x)$ is equal to zero. Following Gardner et al. [2014], we assume independent constraints, such that the overall probability of feasibility can be computed as

$$\text{PF}(x) = \mathbb{P}[\mathbf{c}(x) \leq 0|\mathcal{D}_c] = \prod_{k=1}^{K} \mathbb{P}[c_k(x) \leq 0|\mathcal{D}_c].$$

We aim for an acquisition function that quantifies the value of the objective function and constraint information we would gain from a given sampling decision. Note that obtaining feasibility information does not immediately translate to better expected objective performance but rather more accurate feasibility information where more updated information may change our current beliefs about where x_r is located. Therefore, to quantify the benefit of a design vector, we first find the design that would be recommended after having sampled data $\mathcal{D}_c$ and $\mathcal{D}_f$ as,

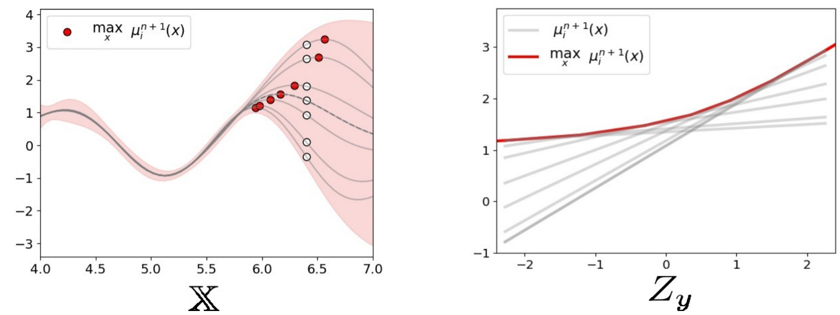
$$x_r^n = \arg\max_{x \in \mathbb{X}} \mu_y^n(x)\text{PF}^n(x). \tag{8}$$

A sensible compromise between the current step n and the one-step lookahead estimated performance is offered by augmenting the training data by the sampling decision x^{n+1} with its respective constraint and objective observations as $\mathcal{D}_c \cup \{x^{n+1}, \mathbf{c}^{n+1}\}$ and $\mathcal{D}_f \cup \{x^{n+1}, y^{n+1}\}$. The difference in performance between the current recommended design and the new best performance can be used as an acquisition function for a design x, where Eqn. 9 is positive for all the design space. Futhermore, given that $\mu_y^n(x_r^n) = \mathbb{E}_{y^{n+1}}[\mu_y^{n+1}(x_r^n)]$ and the independence between objective function and constraints, we obtain

$$\text{cKG}(x) = \mathbb{E}[\max_{x' \in \mathbb{X}} \left\{\mu_y^{n+1}(x')\text{PF}^{n+1}(x')\right\} - \mu_y^n(x_r^n)\text{PF}^{n+1}(x_r^n)|x^{n+1} = x]. \tag{9}$$

This acquisition function quantifies the benefit of a design vector and takes into account the change in the current performance value when more feasibility information is available. Also, when constraints are not considered, the formulation reduces to standard KG [Scott et al. 2011].

Note that penalising the posterior mean by the probability of feasibility as in Eqn. 8 acknowledges that it is risky to eventually recommend solutions that are exactly at the constraint boundary, where the probability of feasibility is 0.5 for a single constraint. This is sensible, as such a solution would not be preferable to a decision maker unless it promises a much higher quality μ_y. On the contrary, sampling an infeasible solution (or a solution on the boundary) during the optimization may be very beneficial if this provides valuable information about the objective function or the exact location of the constraint boundary, which is taken into account in the acquisition function Eqn. 9.



(a) (b)

Fig. 2. (a) Given a sample $x^{n+1} = x$ (white dots), the current GP mean (dotted grey) changes according to Z_y. This produces a different realisation and a new maximum (red dots) for $\mu_y^{n+1}(x_i^*)$. (b) shows the surface of the maximum posterior over the discrete set for Z_y.

4.4 Efficient Acquisition Function Computation

Obtaining a closed-form expression for cKG is not possible but as we show below, it can still be computed efficiently. We first convert $\mu_y^{n+1}(x)$ to quantities that can be computed in the current step n through the reparametrisation trick [Scott et al. 2011] as $\mu_y^{n+1}(x) = \mu_y^n(x) + \tilde{\sigma}_y(x, x^{n+1})Z_y$ where $Z_y \sim N(0, 1)$. The deterministic function $\tilde{\sigma}^n(x, x^{n+1})$ represents the standard deviation of $\mu_y^{n+1}(x)$ parametrised by x^{n+1} and given by $\tilde{\sigma}_y^n(x, x^{n+1}) = \frac{k_y^n(x, x^{n+1})}{\sqrt{k_y^n(x^{n+1}, x^{n+1}) + \sigma_\epsilon^2}}$.

For unconstrained problems, Pearce et al. [2020] proposed an efficient computation for the KG acquisition function (Eqn. 6). It first obtains a suitable small set of points $X_d = \{x_1^*, \dots, x_{n_z}^*\}$ identifying the maxima of the posterior GP mean, $\mu_y^{n+1}(x)$, for different Z_y quantiles (Figure 2 a). Those discrete design vectors are used as a discretisation where $\mu_y^{n+1}(x_i^*)$ is linear on Z_y (Figure 2 b). Then, KG can be computed in closed-form and solved analytically. This approach is both computationally efficient and scalable with the number of design vector dimensions, thus we adapt this method to our constrained problem.

Similar to the unconstrained KG, we may apply the reparametrisation trick to the posterior means and variances of the constraints, i.e, $\mu_k^{n+1}(x) = \mu_k^n(x) + \tilde{\sigma}_k(x, x^{n+1})Z_k$ and $k_k^{n+1}(x, x) = k_k^n(x, x) - \tilde{\sigma}_k{}^2(x, x^{n+1})$, where $Z_k \sim N(0, 1)$ for $k = 1, \dots, K$. Now, the probability of feasibility is also parametrised by x^{n+1} and all the stochasticity is determined by $\mathbf{Z}_c = [Z_1, \dots, Z_K]$. By plugging these parametrisations into Eqn. 9, we change our initial problem to variables that can be estimated in the current step where the stochasticity is given by standard normally distributed random variables for both constraints and the objective,

$$\mathrm{cKG}(x) = \mathbb{E}\Bigg[\overbrace{\max_{x' \in \mathbb{X}} \left\{\left[\mu_y^n(x') + \tilde{\sigma}_y(x', x^{n+1})Z_y\right]\mathrm{PF}^{n+1}(x'; x^{n+1}, \mathbf{Z}_c)\right\}}^{\text{Inner Optimisation } n+1} \\ - \mu_y^n(x_r)\mathrm{PF}^{n+1}(x_r^n; x^{n+1}, \mathbf{Z}_c) | x^{n+1} = x\Bigg], \tag{10}$$

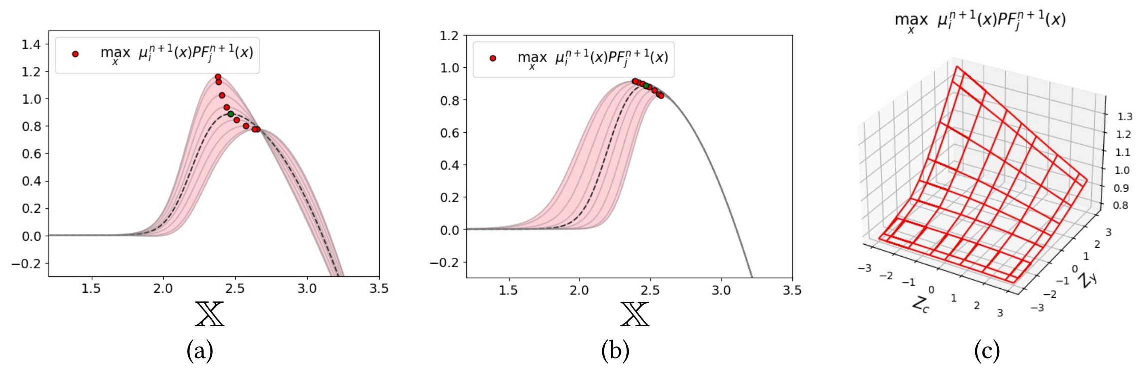


Fig. 3. (a) Given $\mathbf{Z_c} = 0$, current penalised GP mean (dotted grey) and maximum (green dot) where changing Z_y produces a different realisation and a new maximum (red dots). (b) Given $Z_y = 0$, different values of $\mathbf{Z_c}$ produces a new maximum according to the probability of feasibility. (c) shows the surface of the maximum posterior over the discrete set for all combinations of $\mathbf{Z}_c$ and Z_y.

where $\mathrm{PF}^{n+1}(x'; x^{n+1}, \mathbf{Z}_c)$ denotes the probability of solution x' being feasible given that solution x^{n+1} is evaluated next and we observe $\mathbf{Z}_c$.

To solve the above expectation we first find x_r^n according to Eqn. 8 using a continuous numerical optimiser. Then, we generate a discretisation X_d given a design x^{n+1}. This is done using n_y values from Z_y and n_c values from $\mathbf{Z_c}$ where the inner optimisation problems in Eqn. 10 are solved by a continuous numerical optimiser for all $n_z = n_c * n_y$ values. Each solution found by the optimiser, x_j^*, represents a peak location, and together they determine a discretisation X_d. Figure 3 visualises this process where each x_j^* location is generated by numerically identifying the "peaks" (red dots) by using different quantiles for Z_y and Z_c.

Furthermore, conditioned on $\mathbf{Z}_c$, the expectation in Eqn. 10 can be seen as marginalising the standard KG over the constraint uncertainty,

$$\mathrm{cKG}(x) = \mathbb{E}_{Z_c}\Bigg[\mathbb{E}_{Z_y}\Bigg[\overbrace{\max_{x' \in \mathbb{X}} \left\{ \left[\mu_y^n(x') + \tilde{\sigma}_y(x', x^{n+1}) Z_y\right] \mathrm{PF}^{n+1}(x'; x^{n+1}, \mathbf{Z}_c) \right\}}^{\text{Inner Optimisation } n+1}$$
$$- \mu_y^n(x_r)\mathrm{PF}^{n+1}(x_r^n; x^{n+1}, \mathbf{Z}_c) | x^{n+1} = x, \mathbf{Z}_c \Bigg]\Bigg], \tag{11}$$

where $\mu_y^n(x)$ and $\tilde{\sigma}_y(x, x^{n+1})$ are penalised by the (deterministic) function $\mathrm{PF}^{n+1}(x; x^{n+1}, \mathbf{Z}_c)$. The inner expectation can be solved in closed-form over the discrete set X_d using the discrete KG algorithm (KG_d) proposed by Scott et al. [2011] and described in Appendix F. The outer expectation may be computed by a Monte-Carlo approximation. This approximation depends on solving and taking the average over n_c different KG_d computations,

$$\mathrm{cKG}(x) = \frac{1}{n_c} \sum_{m=1}^{n_c} \mathrm{KG}_d(x^{n+1} = x; \mathbf{Z}_c^m). \tag{12}$$

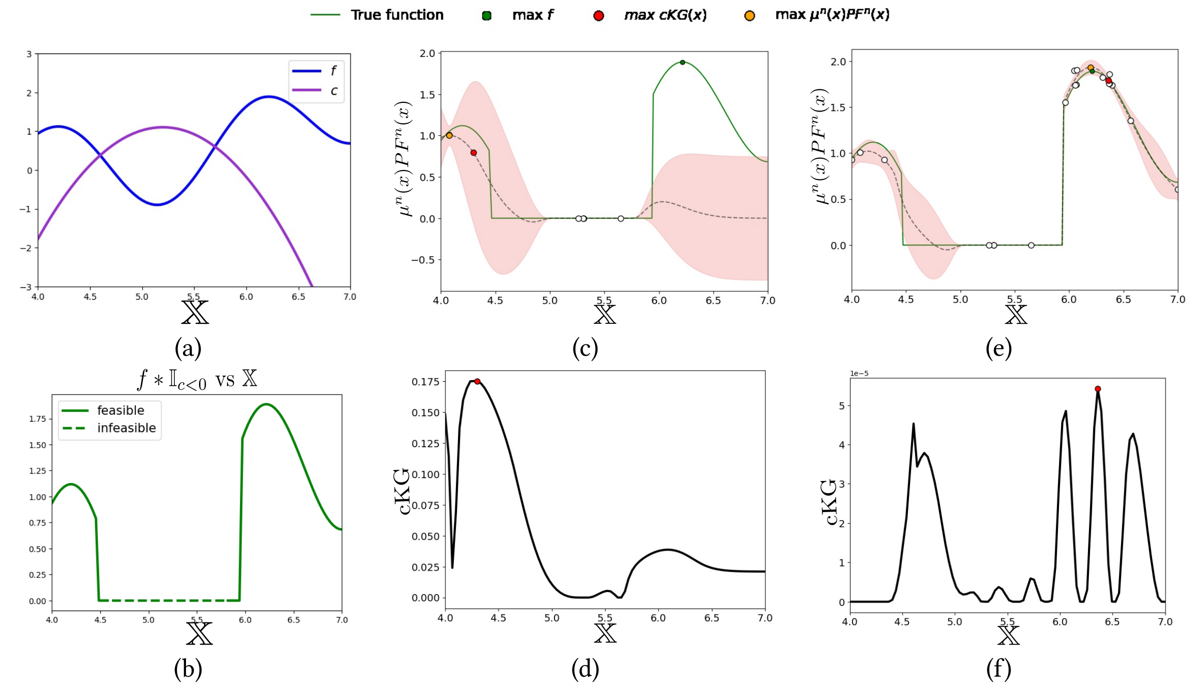


Fig. 4. (a) objective function and constraint where constraint values less than zero are feasible. (b) Feasible and infeasible regions with its corresponding values. (c) Initial design allocation where a model is built using a GP for the objective function penalised by the probability of feasibility using a GP for the constraints. (d) and (f) show the next sample decision (red dot) according to cKG using the fitted models. (e) shows the samples taken during the entire optimisation run (white dots) with the recommended design (orange dot) coinciding with the true best design vector (green dot).

Fig. 3 shows the influence of a sample x^{n+1} (white dots) on computing the expectation at $n+1$ in Eqn. 11. More specifically, if we fix $\mathbf{Z}_c$, Fig. 3 (a) shows how the current GP mean (dotted grey) and maximum (green dot) could change according to Z_y where each different realisation presents a new maximum (red dots). However, if we fix Z_y, Fig. 3 (b) shows how the maximum of the GP mean may change according to the probability of feasibility. Fig. 3 (c) shows the surface of the maximum locations for all combinations of $\mathbf{Z}_c$ and Z_y where, for each generated Z_c, there is an epigraph as in Fig. 2.

4.5 Overall Algorithm

Fig. 4 visualises some of the steps of cKG. Fig. 4 (a) shows an objective function (blue) and a constraint (purple) with negative constraint values representing feasible solutions. The aim is to find the best feasible solution at $x^* = 6.25$, see also Fig. 4 (b). Then, GPs are built based on initial samples, and (c) shows the posterior penalised GP mean (dotted line, posterior y times probability of feasibility). Then, the next design vector is obtained by maximising cKG (Fig. 4 (d)). Finally, after the budget of B samples has been allocated sequentially, a final recommendation x_r is selected according to Eqn. 8 where x_r (orange dot) ends up being very close to the true best x^* (green dot). Notice that cKG aims at improving the maximal posterior mean, not the quality at the sampled solution, and thus often tends to sample the neighborhood of x^* instead of the actual best design vector location.

cKG is outlined in Algorithm 2. On Line 0, the algorithm begins by fitting a Gaussian process model to the initial training data $\mathscr{D}_f$ and $\mathscr{D}_c$ obtained using a Latin hypercube (LHS) 'space-filling' experimental design. After initialisation, the algorithm continues in an optimisation loop until the budget B has been consumed. In each iteration, we sample a new design vector x^{n+1} according to cKG, as defined in Algorithm 1 (Line 2). The design vector x that maximises cKG determines the samples $(x, y)^{n+1}$ and $(x, \mathbf{c})^{n+1}$. The point is added to the training data $\mathscr{D}_f$ and $\mathscr{D}_c$ and each Gaussian process model is updated (Line 5). Finally, cKG recommends a design vector according to Eqn. 8 (Line 7). More implementation details may be found in Appendix F.

Algorithm 1: cKG computation.

Input: Sample x^{n+1}, size of Monte-Carlo discretisations n_c and n_y

0. Initialise discretisation $X_d^0 = \{\}$ and set $n_z = n_c n_y$
1. Compute $x_r^n = \arg\max_{x \in \mathbb{X}} \mu_y^n(x)\mathrm{PF}^n(x)$
2. **for** j **in** $[1, \dots, n_z]$ **:**
3. Generate $Z_y^j, Z_1^j, \dots, Z_K^j \sim N(0, 1)$
4. Compute $x_j^* = \max_{x \in X_d} \left\{ \left[\mu_y^n(x) + \tilde{\sigma}_y(x, x^{n+1}) Z_y^j \right] \mathrm{PF}^{n+1}(x; x^{n+1}, \mathbf{Z}_c^j) \right\}$
5. Update discretisation $X_d^j = X_d^{j-1} \cup \{x_j^*\}$
6. **for** m **in** $[1, \dots, n_c]$ **:**
7. Compute $\mathrm{KG}_d(x^{n+1} = x; \mathbf{Z}_c^m)$ using X_d
8. Compute Monte-Carlo estimation $\frac{1}{n_c} \sum_{m=1}^{n_c} \mathrm{KG}_d(x^{n+1}; \mathbf{Z}_c^m)$
9. **Return:** $\mathrm{cKG}(x^{n+1})$

Algorithm 2: cKG Overall Algorithm. The algorithm starts with an initialization phase to collect preliminary data, then, proceeds to a sequential phase.

Input: black-box function $f : X \longrightarrow \mathbb{R}$, constraints $c_k : X \longrightarrow \mathbb{R}$, size of Monte-Carlo n_c and n_y

0. Collect initial simulation data, $\mathscr{D}_f$,$\mathscr{D}_c$, and fit an independent Gaussian process for each constraint and the black-box function.
1. **While** $b < \mathrm{B}$ **do:**
2. Compute $x^{n+1} = \arg\max_{x \in X} \mathrm{cKG}(x, n_z, M)$.
3. Update $\mathscr{D}_f$, with sample $\{(x, y)^{n+1}\}$
4. Update $\mathscr{D}_c$, with sample $\{(x, \mathbf{c})^{n+1}\}$
5. Fit a Gaussian process to $\mathscr{D}_f$ and $\mathscr{D}_c$
6. Update budget consumed, $b \longleftarrow b + 1$
7. **Return:** Recommend solution, $x_r = \arg\max_{x \in \mathbb{X}} \{\mu_y^B(x)\mathrm{PF}^B(x)\}$

4.6 Properties of cKG

In the Appendix we prove consistency of cKG for an (arbitrary) discrete design space and noise in the objective function or constraints. However, we outline the main findings here. The proof builds on previous work on consistency for unconstrained problems (see Pearce et al. [2019]; Scott et al. [2011], and Poloczek et al. [2017]).

Theorem 1 shows that cKG infinitely samples all design vectors. This ensures that the algorithm learns the true value for all design vectors.

Theorem 1. Let X be a finite set and B the budget to be sequentially allocated by cKG. Let $N(x, B)$ be the number of samples allocated to point x within budget B. Then for all $x \in X$ we have that $\lim_{B\to\infty} N(x, B) = \infty$.
Proof: see Appendix G.

Finally, if the cKG value for all design vectors reaches zero and there exists a feasible region, then we know the location of the global optimiser. This is derived from Theorem 1 which guarantees that all design vectors are infinitely sampled, and therefore we obtain the true best underlying design vector .

Corollary 1 . Let's consider that the set of feasible design vectors $F = \{x|c_k(x) \leq 0 \text{ for } 1 \leq k \leq K\}$ is not empty. If $\text{cKG}(x) = 0$ for all $x \in X$ then $\arg\max_{x\in X} \mu_y^\infty(x)\text{PF}^\infty(x) = \arg\max_{x\in X} f(x)\mathbb{I}_{x\in F}$.

Proof: see Appendix G.

5 RISK-AVERSE DECISION MAKER

If the decision maker is risk averse, they would not accept a solution that has not been evaluated and identified as feasible. In such cases, the solution returned by the algorithm should be the best sampled solution,

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

However, KG is aiming to identify the maximum of the posterior GP mean and has not necessarily sampled this solution. Thus, for such problems, analogous to Pearce and Branke [2018], we suggest still using the cKG acquisition function for the first $B - 1$ iterations, but use an acquisition function that aims to sample at the best location in the final step, namely NEI.

NEI is an extension cEI proposed by Letham et al. [2017] that is capable of accommodating a noisy objective function and noisy constraints. In particular, without observation noise, NEI is identical to cEI. This method consists of generating different realisations of the GP at the observed points to provide different estimations of the best sampled performance. Then cEI is computed for each different realisation for a design vector. If we denote the objective and constraint values at observed design vector locations as $\tilde{\mathbf{f}}^n = [f^n(x_1), \dots, f^n(x_n)]$ and $\tilde{\mathbf{c}}^n = [\mathbf{c}^n(x_1), \dots, \mathbf{c}^n(x_n)]$ then this criteria can be expressed as

$$\text{NEI}(x) = \int_{\mathbf{f}^n,\mathbf{c}^n} \text{cEI}(x|\tilde{\mathbf{f}}^n, \tilde{\mathbf{c}}^n)p(\tilde{\mathbf{f}}^n|\mathcal{D}_f)p(\tilde{\mathbf{c}}^n|\mathcal{D}_c)\text{d}\tilde{\mathbf{f}}^n\text{d}\tilde{\mathbf{c}}^n,$$

where $\text{cEI}(x|\tilde{\mathbf{f}}^n, \tilde{\mathbf{c}}^n)$ is cEI such that the sampled best performance is recomputed for each realisation according to $\tilde{\mathbf{f}}^n, \tilde{\mathbf{c}}^n$. Letham et al. [2017] propose to compute the expectation using quasi Monte-Carlo integration.

6 EXPERIMENTS

We compare cKG against a variety of well-known acquisition functions that can deal with constraints, namely: constrained Expected Improvement (cEI) by Gardner et al. [2014], expected improvement to noisy observations (NEI) by Letham et al. [2017], Predictive Entropy Search with constraints (PESC) by Hernández-Lobato et al. [2016], Thompson sampling for constrained optimisation (TS) by Eriksson and Poloczek [2021] and a recently proposed constrained KG algorithm [Chen et al. 2021] which we call penalised KG (pKG) to distinguish it from our proposed formulation (further details on the benchmark algorithms can be found in Appendix C).

We used implementations of cEI and NEI available in BoTorch [Balandat et al. 2020]. For PESC, only the Spearmint optimisation package provided an available implementation of the algorithm that included constraints. The remaining algorithms have been re-implemented from scratch and can be accessed through github[1].

For all test problems, we fit an independent Gaussian process for each constraint c_k and the black-box objective function y with an initial design of size 10 for the synthetic test functions and 20 for the MNIST experiment, both chosen by Latin Hypercube Sampling. All Gaussian processes use an RBF kernel with hyperparameters tuned by maximum likelihood, including the noise σ_ϵ^2 in case of noisy problems.

6.1 Synthetic Tests

We test the algorithms on three different constrained synthetic problems: Mistery function, Test function 2, and New Branin from Sasena [2002]. Each function was tested with and without a noise level for the objective value and constraints. All test results were averaged over 30 replications. Further details of each function can be found in Appendix A.

Fig. 5 shows the results of these experiments. Fig. 5 (**first row**) depicts a contour plot of each objective function over its feasible area. The location of the optimum is highlighted by a green cross. Mistery and New Branin both have a single non-linear constraint whereas the infeasible area in Test function 2 is the result of a combination of 3 different constraints. Fig. 5 (**second row**) shows the convergence of the opportunity cost over the number of iterations, for the case without noise. As can be seen, cKG outperforms all benchmark approaches on the Branin and Mistery function, with cEI second best. On Test Function 2, pKG converges to the same quality as cKG, with the other methods performing much worse. A closer investigation revealed that cEI tends to concentrate design vectors in highly feasible regions, whereas cKG and pKG seem to benefit from sampling also in regions with lower feasibility. Overall, cKG is the only method that consistently yields superior performance across all three test problems. Fig. 5 (**third row**) shows the performance when the objective value observations are corrupted by noise. Since cEI was designed for deterministic problems, it was replaced by the more general NEI for the noisy problems. Not surprisingly, in all cases the performance of the different considered approaches deteriorated compared to the deterministic setting. The difference between cKG and the other methods is even more apparent, with no method coming close to cKG's performance on any of the benchmarks. Fig. 5 (**fourth row**) represents results with noise in the objective function values and in the constraint values. This is a more challenging task, since design vectors close to the feasibility boundary are more difficult to estimate. This issue is even magnified when several constraints are considered (Test Function 2). Similar to the other experiments, cKG converges consistently better than other methods considered. This shows that cKG is superior to all other tested methods, and particularly capable of handling noisy constrained optimisation problems.

Unlike NEI and cEI, which only consider the posterior at the point sampled, cKG considers the posterior objective and constraints over the full domain. Similarly, although pKG considers the full objective domain by using KG, pKG does not consider the possibility that the new sample may change the constraints, therefore changing the predicted optimum location. This tends to discourage exploration outside the feasible regions, and may lead to slower convergence. On the other hand, cKG takes advantage of looking further ahead than the compared acquisition functions and fully quantifies the impact of a new sampling decision to both, the objective and constraint landscape. Therefore, cKG would place a positive value to design vectors that cause the maximum of the penalised posterior mean to change, even if that design vector is infeasible. This can be considered as an advantage since infeasible design vectors may be useful for reducing model uncertainty which allows a better estimation of future feasible design vectors and therefore, better recommendations.

[1]The code for this paper is available at https://github.com/xxx/xxx (will be published after acceptance)

Although PESC is also a formulation that considers noise in the objective function and constraints by design, it showed comparatively low performance in all the experiments. Hernández-Lobato et al. [2016] showed state-of-the-art results in their comparison against cEI. However, Letham et al. [2017] and Eriksson and Poloczek [2021] observed that PESC mostly performs poorly under different noise scenarios.

Fig. 6 further highlights the robustness of cKG to noise. For each acquisition function, we compute the gap between the OC of the final recommended solution in case of a noisy objective function, or noisy objective function and noisy constraints, to the deterministic case. As expected, noise impacts the algorithm's ability to find the optimum, i.e., the computed gaps are positive, except for pKG on the Mistery function (but this has very large error bars). The detriment is larger if noise is applied to objective function as well as constraints rather than only the objective function. Generally, cKG has the smallest gaps, meaning it suffers the least from the addition of noise.

6.2 Tuning a Fast Fully Connected Neural Network

For this experiment we aim to tune the hyperparameters of a fully connected neural network subject to a limit on the prediction time of 1 ms (the constraint). The design space consists of 9 dimensions comprising the optimiser parameters and the number of neurons on each level, details of the neural network architecture may be found in Appendix B. Note that this is a mixed integer search space, as the number of nodes on a layer is discrete, and can range from 8 to 4096. Following others for similar problems [Hernández-Lobato et al. 2016; Letham et al. 2017; Pearce et al. 2020], we treated the discrete variables as continuous variables and rounded to the nearest integer before evaluation, for all acquisition functions tested. The prediction time is computed as the average time of 3000 predictions for samples of 250 images. The network is trained on the MNIST digit classification task using tensorflow and the objective to be minimised is the classification error rate on a test set. This value is stochastic, as it depends e.g. on the random split of the data into test and training set. At the end, the recommended design is evaluated 20 times to compute a "ground-truth" validation error. All results were averaged over 20 replications (BO runs) and generated using a 20-core Intel(R) Xeon(R) Gold 6230 processor.

Fig. 7 shows that cKG yields the highest validation accuracy compared to the other considered benchmark methods. TS and pKG also perform well, which is consistent with the synthetic experiments.

6.3 Experiments with a Risk-Averse Decision Maker

As explained in Section 5, for a risk-averse decision maker, the algorithm should return the best *sampled* solution. Figure 8 compares the OC performance of the solution with the best penalised posterior GP mean (Eqn. 8) and the best sampled solution, for different acquisition functions on Test Function 2 with deterministic objective function and constraints. In this example, for all acquisition functions the OC resulting from the best GP recommended solution is smaller (better), but the difference is particularly large for those acquisition functions based on KG (cKG and pKG). This is not surprising since as explained above, KG specifically aims to optimise the best GP recommended location.

Thompson sampling also presents a considerable gap. This is largely because the optimisation of the acquisition function, as it was conceived in Eriksson and Poloczek [2021], is performed by discretising the design space and returning the best design vector instead of fine-optimising. Therefore, the sampled solutions tend to be worse than the best fine-tuned GP recommended design.

Figure 9 demonstrates the benefit of executing a final NEI/cEI step before recommending a solution. Compared to Figure 8, making a last step using NEI/cEI significantly decreases the gap in cKG between the sampled and the GP recommended performance (black). Therefore, even if cKG samples design vectors in the neighborhood of x^*, a final NEI/cEI sample ensures good sampled performance.

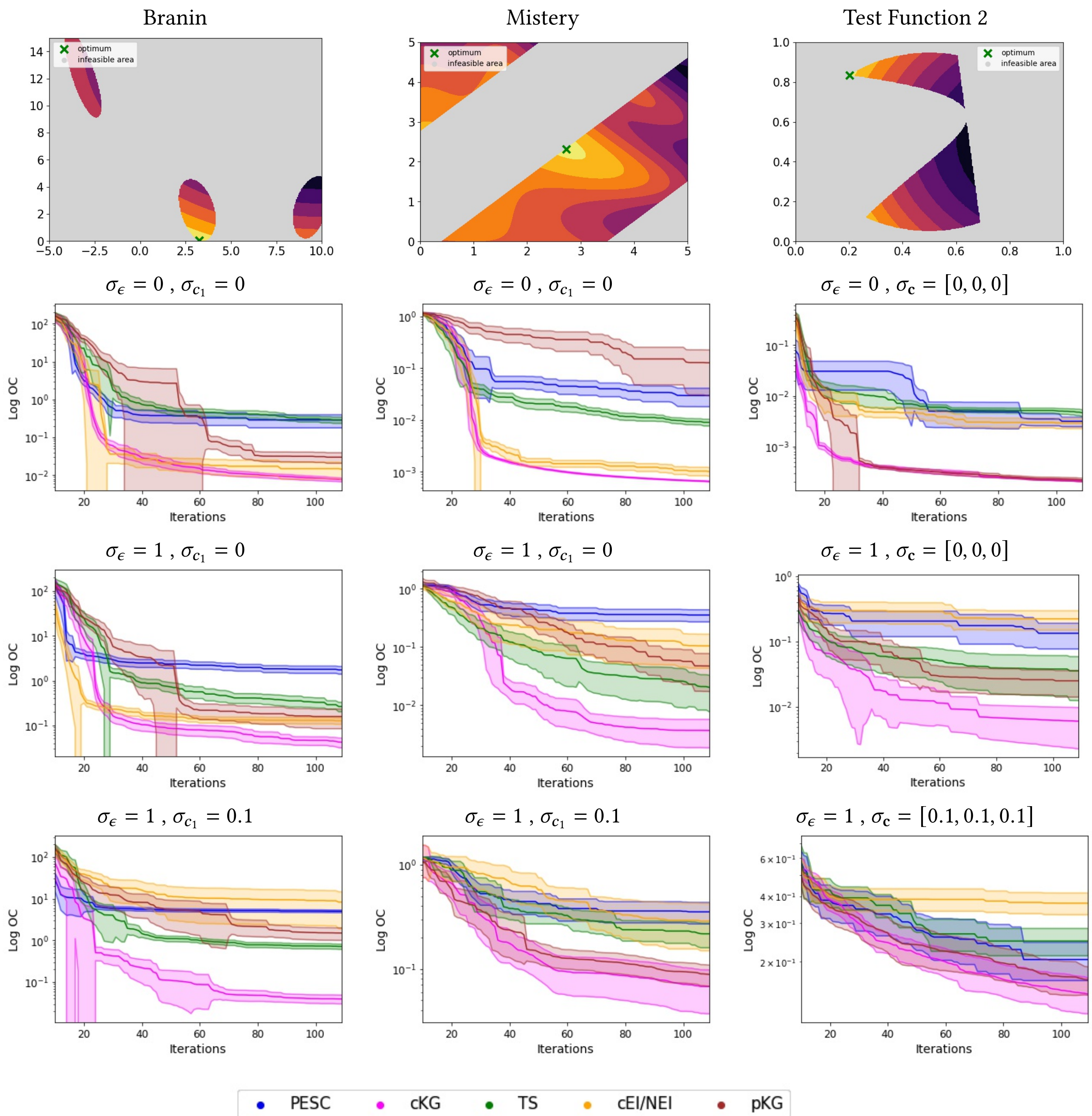


Fig. 5. (**first row**) shows contour plots of the synthetic test functions and their feasible and infeasible regions. For the remaining rows, we show the mean and 95% CI for the OC over iterations for deterministic experiments (**second row**), noise only in the objective function (**third row**), and noise in both the objective function and constraints (**fourth row**).

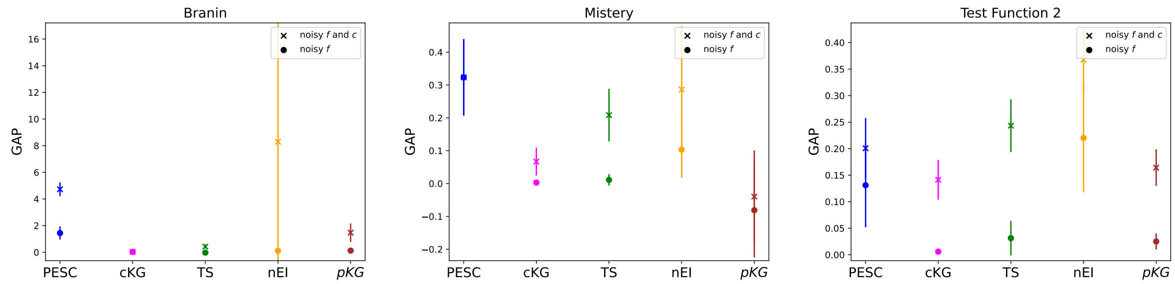
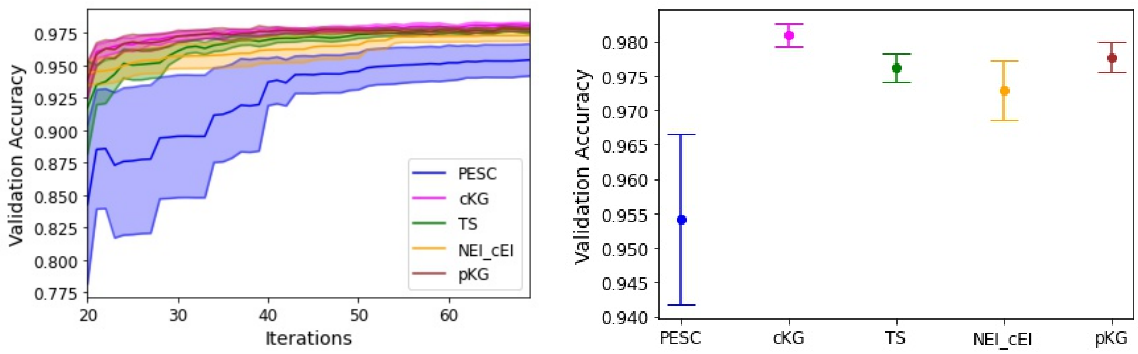


Fig. 6. Difference of OC, or gap, between the noisy experiments and the OC without noise for each algorithm. In all cases, cKG presents a robust performance when noise is added.



(a) (b)

Fig. 7. (a) Mean and 95% CI for the "ground-truth" validation accuracy over iterations. (b) Mean and 95% CI for the "ground-truth score" after 50 iterations.

6.4 Penalty Parameter M

In Section 3, we introduced a penalty M that represents the value that the decision maker assigns to an infeasible recommended design. In principle, its value should be low enough so that we prefer finding a feasible design over infeasible designs. If no domain knowledge is available and the value of M may be difficult to asses, it can simply be set to the worst predicted posterior GP mean. More technical details may be found in Appendix E.

Figure 10 shows the impact of the penalty M on the final solution recommended by cKG in the example of Test Function 2. Each subfigure shows the final recommendations for 30 replications, for different settings of M, from an extremely low value (-1.000.000), to different values of the underlying function (minimum, maximum, and mean), and when automatically setting M to the lowest GP mean prediction (adaptive).

When M is set to -1.000.000, any unfeasible design is heavily penalised, reflecting the situation in which the decision maker is averse to any potentially unfeasible recommendation. As a result, cKG tends to avoid risky solutions that are close to the boundary of feasibility and prefers regions with a high probability of being feasible (not at the extreme end of the feasible space, or even at the broader end of the lower feasible region). A very low M may also prevent the algorithm from crossing unfeasible areas to find new feasible regions. On the other hand, Figure 10 ($M = max f$) shows that when M is at least as high as the true optimal value, any unfeasible design is considered as good as the optimal solution, and cKG tends to recommend only unfeasible designs. Lastly,

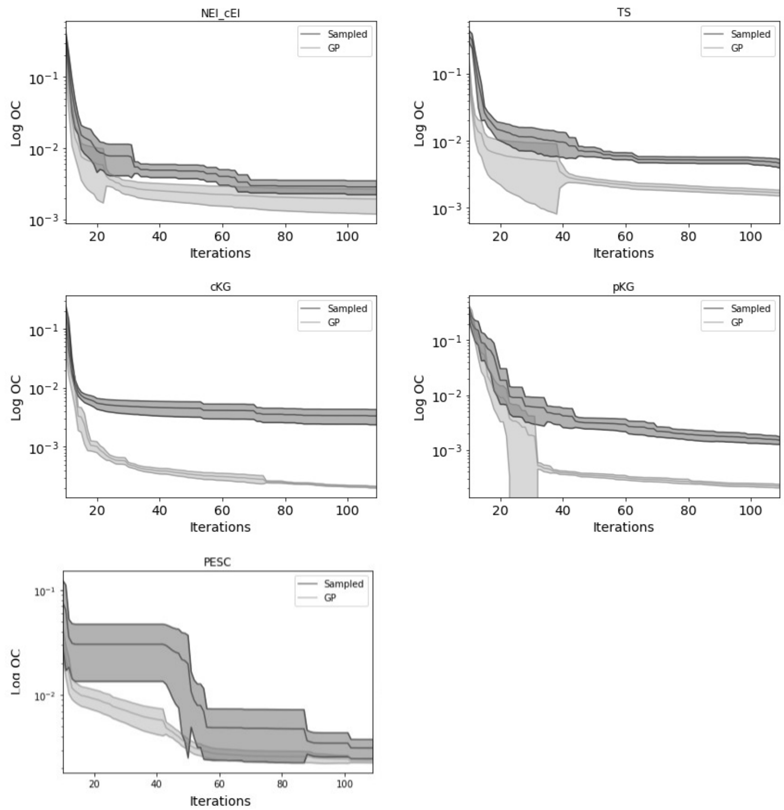


Fig. 8. Sampled and GP recommended solution performance mean and 95% CI for the OC over iterations on Test Function 2 without observation noise for the objective function and the constraints

Figure 10 (adaptive) shows that automatically setting M to the lowest model prediction ensures that cKG will always prefer feasible solutions over unfeasible ones and is reliably able to identify the true optimum.

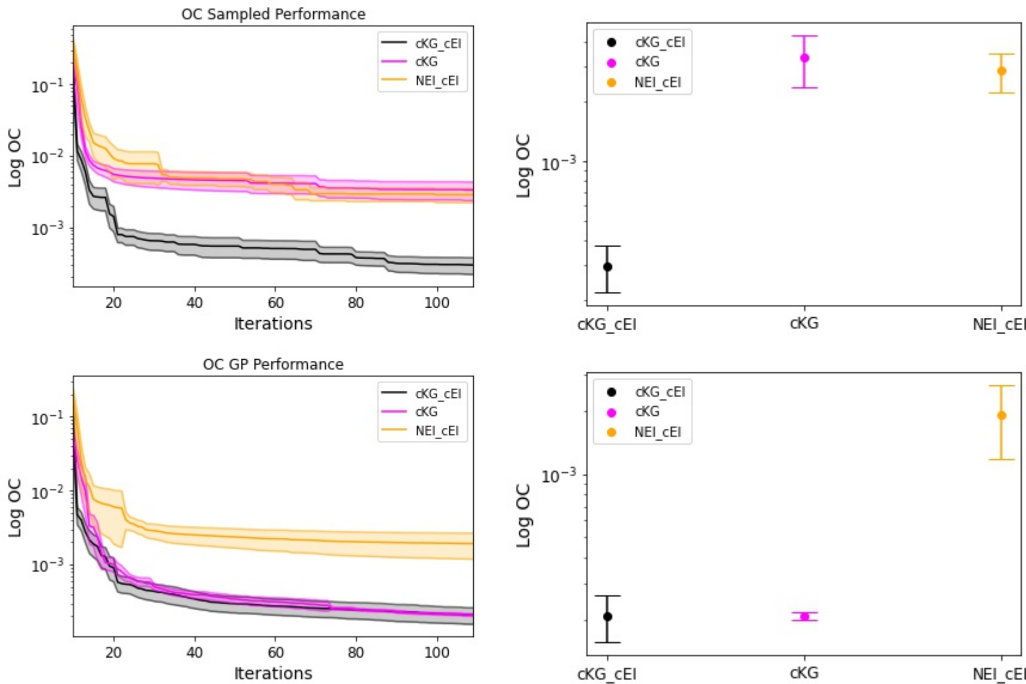


Fig. 9. Comparison between cKG and cKG taking a cEI last step before recommending a solution. **(First Column)** Best sampled and GP recommended solution performance mean and 95% CI for the OC. **(Second Column)**: Sampled and GP recommended solution performance mean and 95% CI for the OC at the end of the budget. Objective noise level: $\sigma_\epsilon^2 = 0$.

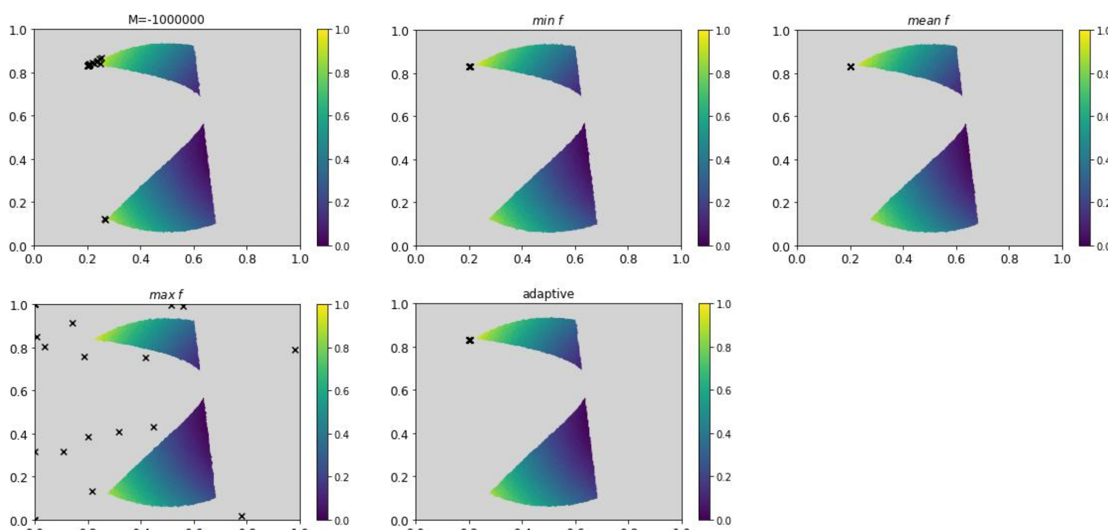


Fig. 10. Final recommended design of 30 replications using the deterministic Test Function 2 with different values of M: $M = -1.000.000$ (top left), minimum f value (top middle), mean f value (top right), maximum f (bottom left), and using the lowest model prediction (adaptive, bottom middle)

7 CONCLUSION

For the problem of constrained Bayesian optimisation, we proposed a generalisation of the well-known Knowledge Gradient acquisition function, constrained Knowledge Gradient (cKG), that is capable of handling constraints and noise. We show that cKG can be efficiently computed by adapting an approach proposed in [Pearce et al. 2020] which is a hybrid between discretisation and Monte-Carlo approximation that allows to leverage the benefits of fast computations of the discrete design space and the scalability of continuous Monte-Carlo sampling. We prove that the algorithm will find the true optimum in the limit. Finally, we empirically demonstrate the effectiveness of the proposed approach on several test problems. cKG consistently and significantly outperformed all benchmark algorithms on all test problems, with a particularly large improvement under noisy problem settings.

Despite the excellent results, the study opens some interesting avenues for future work. We assume the noise in the quality measure to be homoscedastic. Perhaps ideas from Stochastic Kriging can be used to relax this. Furthermore, in real problems, objective and constraints may be correlated where multi-output GP models may be considered to model correlation. This may lead to a better feasibility representation and performance of the algorithm. As the standard KG, we only look ahead one step with sequential evaluations. cKG may be further extended to multi-step look ahead or batched evaluations by using a "one-shot" formulation. Finally, we assume that an evaluation of a solution returns simultaneously its quality as well as its constraint value. In practice, it may be possible to evaluate quality and feasibility independently, or infeasible solutions may not return an objective value at all.

ACKNOWLEDGEMENTS

Removed for double blind review

REFERENCES

Mauricio A. Álvarez, Lorenzo Rosasco, and Neil D. Lawrence. 2012. Kernels for Vector-Valued Functions: A Review. *Found. Trends Mach. Learn.* 4, 3 (mar 2012), 195–266. https://doi.org/10.1561/2200000036

Satyajith Amaran, Nikolaos V. Sahinidis, Bikram Sharda, and Scott J. Bury. 2016. Simulation optimization: a review of algorithms and applications. *Annals of Operations Research* 240, 1 (2016), 351–380.

Samineh Bagheri, Wolfgang Konen, Richard Allmendinger, Juergen Branke, Kalyanmoy Deb, Jonathan Fieldsend, Domenico Quagliarella, and Karthik Sindhya. 2017. Constraint Handing in Efficient Global Optimization. In *Genetic and Evolutionary Computation Conference*. ACM, 673–680.

Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. 2020. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*. http://arxiv.org/abs/1910.06403

Felix Berkenkamp, Andreas Krause, and Angela P. Schoellig. 2016. Bayesian Optimization with Safety Constraints: Safe and Automatic Parameter Tuning in Robotics. *ArXiv* abs/1602.04450 (2016).

Felix Berkenkamp, Andreas Krause, and Angela P. Schoellig. 2021. Bayesian optimization with safety constraints: Safe and automatic parameter tuning in robotics - machine learning. https://link.springer.com/article/10.1007/s10994-021-06019-1

Antonio Candelieri. 2019. Sequential model based optimization of partially defined functions under unknown constraints. *Journal of Global Optimization* (2019), 1–23.

Wenjie Chen, Shengcai Liu, and Ke Tang. 2021. A New Knowledge Gradient-based Method for Constrained Bayesian Optimization. arXiv:2101.08743 [cs.LG]

Erhan Cinlar. 2011. *Probability and Stochastics*. Vol. Graduate Texts in Mathematics 261. Springer.

David Eriksson and Matthias Poloczek. 2021. Scalable Constrained Bayesian Optimization. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 130)*, Arindam Banerjee and Kenji Fukumizu (Eds.). PMLR, 730–738. https://proceedings.mlr.press/v130/eriksson21a.html

Alexander I. J. Forrester, Andras Sobester, and Andy J. Keane. 2008. *Engineering Design via Surrogate Modelling*.

Peter I. Frazier. 2018. A Tutorial on Bayesian Optimization. arXiv:1807.02811 [stat.ML]

Jacob Gardner, Matt Kusner, Eddie Xu, Kilian Weinberger, and John Cunningham. 2014. Bayesian Optimization with Inequality Constraints. *31st International Conference on Machine Learning, ICML 2014* 3.

Robert B. Gramacy, Genetha A. Gray, Sébastien Le Digabel, Herbert K. H. Lee, Pritam Ranjan, Garth Wells, and Stefan M. Wild. 2016. Modeling an Augmented Lagrangian for Blackbox Constrained Optimization. *Technometrics* 58, 1 (2016), 1–11. https://doi.org/10.1080/00401706.2015.1014065 arXiv:https://doi.org/10.1080/00401706.2015.1014065

José Miguel Hernández-Lobato, Michael A. Gelbart, Ryan P. Adams, Matthew W. Hoffman, and Zoubin Ghahramani. 2016. A General Framework for Constrained Bayesian Optimization Using Information-Based Search. *J. Mach. Learn. Res.* 17, 1 (Jan. 2016), 5549–5601.

Jose Miguel Hernandez-Lobato, Matthew W. Hoffman, and Zoubin Ghahramani. 2014. Predictive Entropy Search for Efficient Global Optimization of Black-Box Functions. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1* (Montreal, Canada) *(NIPS'14)*. MIT Press, Cambridge, MA, USA, 918–926.

Donald R. Jones, Matthias Schonlau, and William J. Welch. 1998. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization* 13 (Jan 1998), 455–492. https://doi.org/10.1023/A:1008306431147

Kristian Kersting, Christian Plagemann, Patrick Pfaff, and Wolfram Burgard. 2007. Most Likely Heteroscedastic Gaussian Process Regression. In *Proceedings of the 24th International Conference on Machine Learning* (Corvalis, Oregon, USA) *(ICML '07)*. Association for Computing Machinery, New York, NY, USA, 393–400. https://doi.org/10.1145/1273496.1273546

Jack P. C. Kleijnen, Inneke Van Nieuwenhuyse, and Wim van Beers. 2021. Constrained optimization in simulation: efficient global optimization and Karush-Kuhn-Tucker conditions. *CentER Discussion Paper Series No. 2021-031* (08 2021). https://doi.org/10.2139/ssrn.3958881

Remi R. Lam and Karen E. Willcox. 2017. Lookahead Bayesian Optimization with Inequality Constraints. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 1888–1898.

Benjamin Letham, Brian Karrer, Guilherme Ottoni, and Eytan Bakshy. 2017. Constrained Bayesian Optimization with Noisy Experiments. *Bayesian Analysis* 14 (06 2017). https://doi.org/10.1214/18-BA1110

Michael Pearce and Juergen Branke. 2018. Continuous multi-task bayesian optimisation with correlation. *European Journal of Operational Research* 270, 3 (2018), 1074–1085.

Michael Pearce, Janis Klaise, and Matthew Groves. 2020. Practical Bayesian Optimization of Objectives with Conditioning Variables. arXiv:2002.09996 [stat.ML]

Michael Pearce, Matthias Poloczek, and Juergen Branke. 2019. Bayesian Simulation Optimization with Common Random Numbers. In *Proceedings of the Winter Simulation Conference* (National Harbor, Maryland) *(WSC '19)*. IEEE Press, 3492–3503.

Julien Pelamatti, Rodolphe Le Riche, Céline Helbert, and Christophette Blanchet-Scalliet. 2022. Coupling and selecting constraints in Bayesian optimization under uncertainties. https://doi.org/10.48550/ARXIV.2204.00527

Victor Picheny. 2014. A stepwise uncertainty reduction approach to constrained global optimization. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 33)*, Samuel Kaski and Jukka Corander (Eds.). PMLR, Reykjavik, Iceland, 787–795. http://proceedings.mlr.press/v33/picheny14.html

Victor Picheny, Robert B Gramacy, Stefan Wild, and Sebastien Le Digabel. 2016. Bayesian optimization under mixed constraints with a slack-variable augmented Lagrangian. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Curran Associates, Inc., 1435–1443. http://papers.nips.cc/paper/6439-bayesian-optimization-under-mixed-constraints-with-a-slack-variable-augmented-lagrangian.pdf

Victor Picheny, Tobias Wagner, and David Ginsbourger. 2013. A benchmark of kriging-based infill criteria for noisy optimization. *Structural and Multidisciplinary Optimization* 48 (09 2013). https://doi.org/10.1007/s00158-013-0919-4

Matthias Poloczek, Jialei Wang, and Peter Frazier. 2017. Multi-Information Source Optimization. In *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/df1f1d20ee86704251795841e6a9405a-Paper.pdf

Carl E. Rasmussen and Christopher K. I. Williams. 2006. *Gaussian Processes for Machine Learning*. MIT Press.

Michael 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*. 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 arXiv:https://doi.org/10.1137/100801275

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. 2016. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proc. IEEE* 104, 1 (2016), 148–175.

Jian Wu and Peter I. Frazier. 2017. Discretization-free Knowledge Gradient Methods for Bayesian Optimization. arXiv:1707.06541 [stat.ML]

Yunxiang Zhang, Xiangyu Zhang, and Peter Frazier. 2021. Constrained Two-step Look-Ahead Bayesian Optimization. In *Advances in Neural Information Processing Systems 34 pre-proceedings*.

A SYNTHETIC TEST FUNCTIONS

The following subsections describe the synthetic test functions used for the empirical comparison [Sasena 2002].

A.1 Mystery Function

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

subject to

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

$$x_i \in [0, 5], \forall i = 1, 2$$

A.2 New Branin Function

$$\min f(x) = -(x_1 - 10)^2 - (x_2 - 15)^2$$

subject to

$$\left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 5 \leq 0$$

$$x_1 \in [-5, 10]$$

$$x_2 \in [0, 15]$$

A.3 Test Function 2

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

subject to

$$\left[(x_1 - 3)^2 + (x_2 + 2)^2\right]e^{x_2^7} - 12 \leq 0$$

$$10x_1 + x_2 - 7 \leq 0$$

$$(x_1 - 0.5)^2 + (x_2 - 0.5)^2 - 0.2 \leq 0$$

$$x_i \in [0, 1] \forall i = 1, 2$$

B MNIST HYPERPARAMETER EXPERIMENT

Design Space:

- `learning_rate` $\in [0.0001, 0.01]$, log scaled.
- `beta_1` $\in [0.7, 0.99]$, log scale.
- `beta_2` $\in [0.9, 0.99]$, log scale.
- `dropout_rate_1` $\in [0, 0.8]$, linear scale.
- `dropout_rate_2` $\in [0, 0.8]$, linear scale.
- `dropout_rate_3` $\in [0, 0.8]$, linear scale.
- `n_neurons_1` $\in [3, 12]$, no scaling.
- `n_neurons_2` $\in [3, 12]$, no scaling.
- `n_neurons_3` $\in [3, 12]$, no scaling.

Neural Network Architecture:

```
model = Sequential()

model = Dense(units = int(power(2,n_neurons_1)), input_shape=(784,))
model = Dropout(dropout_rate_1)
model = activation('relu')

model = Dense(units = int(power(2, n_neurons_2)))
model = Dropout(dropout_rate_2)
model = activation('relu')

model = Dense(units = int(power(2, n_neurons_3)))
model = Dropout(dropout_rate_3)
model = activation('relu')

model = Dense(units = 10)
model = activation('softmax')
```

Optimiser and Compilation:

```
adam = Adam(learning_rate=learning_rate,
            beta_1=beta_1,
            beta_2=beta_2)


model.compile(loss='categorical_crossentropy',
              optimizer=adam,
              metrics=['accuracy'])
```

C RELATED ALGORITHMS

C.1 Constrained Expected Improvement (cEI)

Schonlau et al. [1998] extends EI to deterministic constrained problems by multiplying it with the probability of feasibility in the acquisition function:

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

where $\mathrm{PF}^n(x)$ is the probability of feasibility of x and $\mathrm{EI}(x|f^*)$ is the expected improvement over the best feasible sampled observation, f^*, i.e.,

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

The posterior Gaussian distribution with mean $\mu^n_y(x)$ and variance $k^n_y(x,x)$ offers a closed form solution to EI where the terms only depend on Gaussian densities and cumulative distributions,

$$\mathrm{EI}(x|f^*) = (\mu^n_y(x) - f^*)\Phi(z) + k^n_y(x,x)\phi(z), \text{ where } z = \frac{\mu^n_y(x) - f^*}{k^n_y(x,x)}$$

C.2 Thompson Sampling with constraints (TS)

Eriksson and Poloczek [2021] extend Thompson sampling to constraints. Let $x_1, \ldots, x_r$ be candidate points. Then a realization is taken at the candidate points location $(\hat{f}(x_i), \hat{c}_1(x_i), \ldots, \hat{c}_m(x_i))$ for all x_i with $1 \leq i \leq r$ from the respective posterior distributions. Therefore, if $\hat{F} = \{x_i | \hat{c}_l(x_i) \leq 0 \text{ for } 1 \leq l \leq m\}$ is not empty, then the next design vector is selected by $\arg\max_{x \in \hat{F}} \hat{f}(x)$. Otherwise a point is selected according to the minimum total violation $\sum_{l=1}^{m} \max\{\hat{c}_l(x), 0\}$.

Eriksson and Poloczek [2021] 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. However, for comparison purposes, we only implement the selection criteria.

C.3 Constrained Predicted Entropy search (PESC)

Hernández-Lobato et al. [2016] seek to maximise the information about the feasible optimal location x^* given the collected data, as,

$$\text{PESC}(x) = \text{H}(y|\mathcal{D}_f, \mathcal{D}_c) - \mathbb{E}_{x^*}[\text{H}[y|\mathcal{D}_f, \mathcal{D}_c, x, x^*]]$$

The first term on the right-hand side of 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 $\hat{x}^* \sim \text{p}(x^*|\mathcal{D}_f, \mathcal{D}_c)$. 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 $\hat{x}^*$.

C.4 Penalised Knowledge Gradient (pKG)

Chen et al. [2021] extend KG to constrained problems by penalising any new sample by the probability of feasibility, i.e.,

$$\text{pKG}(x) = \mathbb{E}\left[\max_{x \in \mathbb{X}} \left\{\mu_y^{n+1}(x)\right\} - \max_{x \in \mathbb{X}} \left\{\mu_y^{n}(x)\right\} | x^{n+1} = x\right] \text{PF}^n(x^{n+1} = x). \tag{13}$$

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].

D COMPUTATIONAL TIME OF ACQUISITION FUNCTION EVALUATION

Although cKG shows superior performance compared to other methods, it comparatively requires more computational time to approximate cKG and decide where to sample next. However, for the real-world problems intended, the time to determine the next sample location is negligible in comparison to the objective function/constraints evaluation time. Therefore, increased query efficiency is justified over the relatively small optimisation overhead of the acquisition function. This is the setting where BO is almost always applied.

We show results using the New Branin Function with deterministic evaluations, i.e., without adding noise, and using a 20-core Intel(R) Xeon(R) Gold 6230 processor. In terms of purely the time to identify the next sample location by optimising the acquisition function, cKG is the slowest (8 seconds) and TS is the fastest (0.03 seconds). The other methods require a similar optimisation time (NEI: 3.2 seconds, pKG: 3.24 seconds, and PESC: 3.39 seconds). However, for simulation optimisation problems where a single simulation run takes several minutes to complete, the time taken by the acquisition function becomes negligible and all that counts is the performance relative to the number of required evaluations.

E FORMULATION WITH PENALTY PARAMETER M

Section. 6.4 shows the effect of the parameter M in cKG. In this section we give additional details on the cKG computation. If we consider a non-zero penalty (M) for infeasible solutions, we may recommend a solution x_r at a given n iteration according to

$$x_r^n = \arg\max_{x \in \mathbb{X}} \left\{ \mu_y^n(x) \mathrm{PF}^n(x) + M(1 - \mathrm{PF}^n(x)) \right\}, \tag{14}$$

The second term in the right-hand side represents the penalty incurred by infeasible solutions. When a clearly infeasible solution is selected, $(1 - \mathrm{PF}^n(x))$ is closer to 1 and the whole penalty term is closer to M. To modify cKG, we must simply take into account the penalty term as,

$$\begin{aligned} \mathrm{cKG}(x) = \mathbb{E}[\max_{x' \in \mathbb{X}} & \left\{ \mu_y^{n+1}(x') \mathrm{PF}^{n+1}(x') + M(1 - \mathrm{PF}^{n+1}(x')) \right\} \\ & - \mu_y^{n+1}(x_r^n) \mathrm{PF}^{n+1}(x_r^n) - M(1 - \mathrm{PF}^{n+1}(x_r^n)) | x^{n+1} = x]. \end{aligned} \tag{15}$$

cKG may be then computed as described in Section. 4.4, where the penalty term is included when each inner optimisation problem is solved.

F IMPLEMENTATION DETAILS OF CKG

Implementing cKG first requires to generate Z_y and $\mathbf{Z_c}$ for a candidate sample x. This may be done by randomly generating values from a standard normal distribution, or taking Quasi-Monte samples which provides more sparse samples and faster convergence properties [Letham et al. 2017]. However, we choose to adopt the method proposed by Pearce et al. [2020] where they use different Gaussian quantiles for the objective $Z_y = \{\Phi^{-1}(0.1), \dots, \Phi^{-1}(0.9)\}$. We further extend this method by also generating Gaussian quantiles for each constraint $k = 1, \dots, K$ and produce the n_z samples using the Cartesian product between the z-samples for y and $k = 1, \dots, K$. Once a set of n_z samples has been produced, we may find each sample in X_d by a L-BFGS optimiser, or any continuous deterministic optimisation algorithm. Finally, KG_d in Alg. 1 may be computed using the algorithm described in Alg. 3 by Scott et al. [2011].

To optimise cKG we first select an initial set of candidates according to a Latin-hypercube design and compute their values. We then select the best subset according to their cKG value and proceed to fine optimise each selected candidate design vector. We have noticed that discretisations, X_d, achieved by this subset of candidates do not change considerably during the fine optimisation, therefore we fix the discretisation found for each candidate and then fine optimise. A fixed discretisation allows to use a deterministic and continuous optimiser where approximate gradients may also be computed.

Algorithm 3: Knowledge Gradient by discretisation. This algorithm takes as input a set of linear functions parameterised by a vector of intercepts μ and a vector $\tilde{\sigma}$

Input: μ, $\tilde{\sigma}$, and best current performance μ^*

0. $O \leftarrow \text{order}(\tilde{\sigma})$ (get sorting indices of increasing $\tilde{\sigma}$)
1. $\mu \leftarrow \mu[O]$, $\tilde{\sigma} \leftarrow \tilde{\sigma}[O]$ (arrange elements)
2. $I \leftarrow [0, 1]$ (indices of elements in the epigraph)
3. $\tilde{Z} \leftarrow [-\infty, \frac{\mu_0 - \mu_1}{\tilde{\sigma}_1 - \tilde{\sigma}_0}]$ (z-scores of intersections on the epigraph)
4. **for** j **in** $[2, \ldots, n_z - 1]$ **:**
5. $\quad j \leftarrow \text{last}(I)$
6. $\quad z \leftarrow [-\infty, \frac{\mu_i - \mu_j}{\tilde{\sigma}_j - \tilde{\sigma}_i}]$
7. $\quad$ **if** $z < \text{last}(\tilde{Z})$:
8. $\quad\quad$ Delete last element of I and $\tilde{Z}$.
9. $\quad\quad$ Return to Line 5.
10. $\quad$ Add i to the end of I and z to $\tilde{Z}$
11. $\tilde{Z} \leftarrow [\tilde{Z}, \infty]$
12. $A \leftarrow \phi(\tilde{Z}[1:]) - \phi(\tilde{Z}[:-1])$
13. $B \leftarrow \Phi(\tilde{Z}[1:]) - \Phi(\tilde{Z}[:-1])$
14. $\text{KG} \leftarrow B^T \mu[I] - A^T \tilde{\sigma}[I] - \mu^*$
15. **Return:** KG

G THEORETICAL RESULTS FOR FINITE DOMAINS

In this section we further develop the statements in the main paper. In Theorem 1 we show that cKG infinitely samples all design vectors. This ensures that the algorithm learns the true value for all design vectors. Furthermore, if the cKG value for all design vectors reaches zero, then we know the location of the global optimiser (Theorem 1). For the following statements, we assume a discrete design space. Note that this space can be arbitrarily dense, thus it is valid for example if the algorithm is run on a computer with finite precision. Figure 11 compares the convergence curves of cKG in the continuous domain (cKG) and when the design space is discretised using 500, 500.000, and 5.000.000 design vectors generated by a Latin hypercube (LHS) experimental design. As can be seen, for all test functions, the convergence of cKG using a discretised design space tends to converge to cKG over a continuous domain as the discretisation is refined. The results of the following proofs, derived for a discrete domain, are thus still expected to hold also in the continuous domain.

To prove Theorem 1, we rely on the fact that cKG is a measure of improvement and thus must be non-negative.

Lemma 1. *Let $x \in \mathbb{X}$, then $cKG(x) \geq 0$*

Proof:

If we take the recommended design according to $x_r^n = \arg\max_{x \in \mathbb{X}} \mu_y^n(x)\text{PF}^n(x)$ to compute the proposed formulation,

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

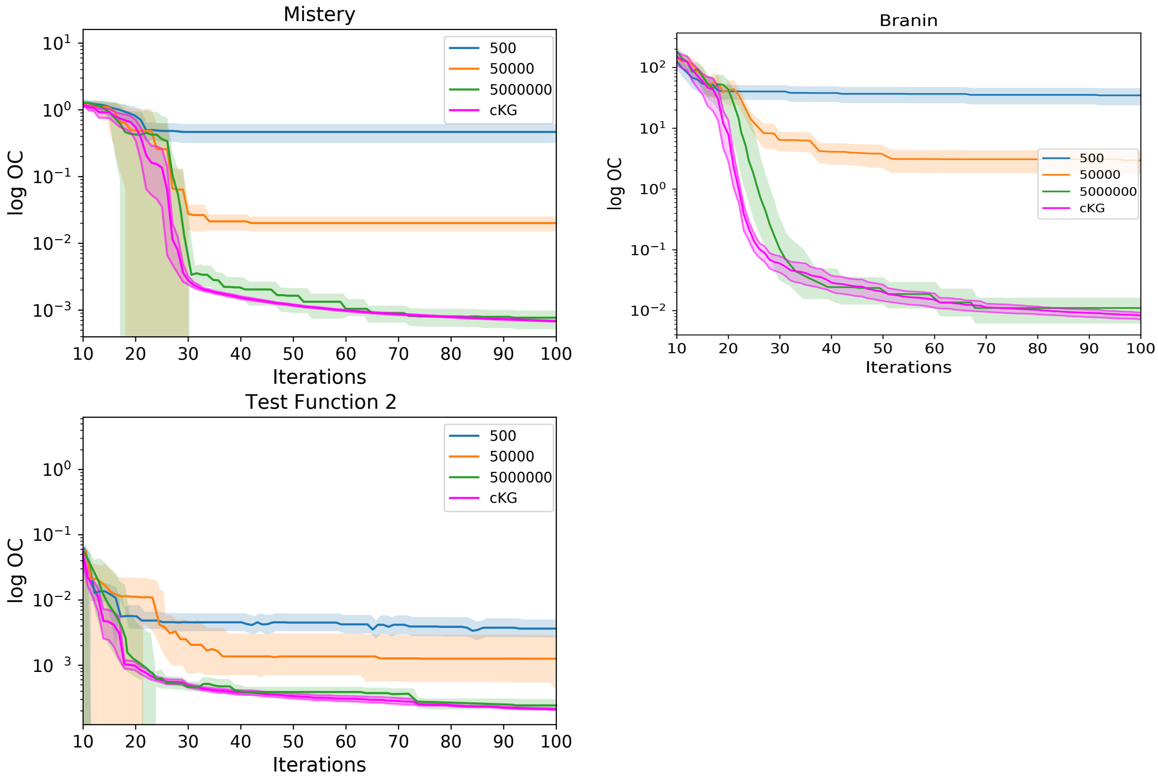


Fig. 11. mean and 95% CI for the OC over the iterations for the deterministic experiments when the design space is continuous (cKG) and when the design space is discretised using 500, 500000, and 5000000 design vectors generated using a Latin hypercube (LHS) experimental design.

it is straightforward to observe that the first term in the left-hand-side has a value greater or equal to the second term given by the inner optimisation operation. □

Then, Lemma 2 shows that for noisy observations, if we infinitely sample a design vector x then cKG(x) reaches a lower bound. In the case of deterministic observations, we only require to sample x once so cKG(x) reaches zero.

Lemma 2. *Let $x \in \mathbb{X}$ and the number of samples taken in x is denoted as $N(x)$, then $N(x) = \infty$ implies that $cKG(x) = 0$*

Proof:

If the observation is deterministic ($\sigma_\epsilon^2 = 0$) then sampling x^{n+1} at any design vector x produces $\sigma_y(x, x') = 0$ for the following iterations (see Lemma 10 in Pearce et al. [2019]). Therefore, cKG becomes zero for those sampled locations.

When there is noise in the observation ($\sigma_\epsilon^2 > 0$) or constraints ($\sigma_k^2 > 0$), and given infinitely many observations at x, we have that $k_y^\infty(x, x) = 0$ and $k_y^\infty(x, x') = 0$ for all $x \in X$ by the positive definiteness of the kernel (see Lemma 11 in Pearce et al. [2019]). Then it easily follows that $\tilde{\sigma_y}(x, x') = 0$ and $\tilde{\sigma_k}(x, x') = 0$ for all $x' \in X$ and $k = 1, \ldots K$. Therefore, $\mathrm{PF}^{n+1}(x; x^{n+1}, \mathbf{Z}_c) = \mathrm{PF}^n(x)$, and,

$$\begin{aligned}\text{cKG}(x) &= \mathbb{E}_{\mathbf{Z}_c}[\mathbb{E}_{Z_y}[\max_{x'\in X}\{[\mu_y^n(x') + \overbrace{\tilde{\sigma_y}(x,x')}^{=0} \cdot Z_y]\text{PF}^n(x')\} \\ &\quad - \mu_y^n(x_r)\text{PF}^n(x_r)|x^{n+1} = x, \mathbf{Z}_c]] \\ &= 0\end{aligned}$$

where the bottom line comes from obtaining the recommended design as $x_r = \arg\max\{\mu_y^n(x)\text{PF}^n(x)\}$. □

Lemma 3. *Let $x^{n+1} \in \mathbb{X}$ be a design vector for which $cKG(x^{n+1}) > 0$ then $N(x^{n+1}) < \infty$*

Proof:
cKG$(x^{n+1}) > 0$ implies that $\tilde{\sigma_y}(x, x^{n+1}) > 0$ and $\text{PF}^{n+1}(x; x^{n+1}, \mathbf{Z}_c) > 0$ for some x. By Lemma 3 in Poloczek et al. [2017], if $\tilde{\sigma_y}(x, x^{n+1}) > 0$ then $k^n(x, x^{n+1})$ is not a constant function of x. Therefore, only if x^{n+1} is infinitely sampled, $k^n(x, x^{n+1})$ becomes a constant function and the maximiser value x^* is perfectly known. Thus x^{n+1} is not infinitely sampled. □

Theorem 1. *Let X be a finite set and B the budget to be sequentially allocated by cKG. Let $N(x, B)$ be the number of samples allocated to point x within budget B. Then for all $x \in X$ we have that $\lim_{B\to\infty} N(x, B) = \infty$.*

Proof:
Lemma 1 and Lemma 3 imply that any point x that is infinitely sampled will reach a lower bound. Since cKG recommends samples according to argmax, any design vector x that has been infinitely sampled will not be visited until all other design vectors $x' \in X$ have cKG$(x') = 0$. Therefore, $N(x, B) = \infty$ for all points when $B \to \infty$. □

To prove Corollary 1 we rely on Lemma 4. Complete derivation may be found in Cinlar [2011], in Proposition 2.8, however, the proposition states that any sequence of conditional expectations of an integrable random variable under an increasing convex function is a uniformly integrable martingale.

Lemma 4. *Let $x, x' \in X$ and $n \in \mathbb{N}$. The limits of the series $(\mu^n(x)$ and $(V^n(x, x')$ (shown below) exist.*

$$\mu^n(x) = \mathbb{E}_n[f(x)] \tag{16}$$

$$\begin{aligned}V^n(x, x') &= \mathbb{E}_n[f(x) \cdot f(x')] &(17)\\ &= k^n(x, x') + \mu^n(x) \cdot \mu^n(x') &(18)\end{aligned}$$

Denote their limits by $\mu^\infty(x)$ and $V^\infty = (x, x')$ respectively.

$$\lim_{n\to\infty} \mu^n(x) = \mu^\infty(x) \tag{19}$$

$$\lim_{n\to\infty} V^n(x, x') = V^\infty(x, x') \tag{20}$$

If x' is sampled infinitely often, then $\lim_{n\to\infty} V^n(x, x') = \mu^\infty(x) \cdot \mu^\infty(x')$ holds almost surely.

Corollary 1. *Let's consider that the set of feasible design vectors* $F = \{x|c_k(x) \leq 0 \text{ for } 1 \leq k \leq K\}$ *is not empty. If* $cKG(x) = 0$ *for all* $x \in X$ *then* $\arg\max_{x \in X} \mu_y^{\infty}(x) PF^{\infty}(x) = \arg\max_{x \in X} f(x)\mathbb{I}_{x \in F}$.

Proof:

By Lemma 4, $\lim_{n\to\infty} \tilde{k}_y^n(x, x') = \tilde{k}_y^{\infty}(x, x')$ a.s for all $x, x' \in X$. Also, Theorem 1 implies that all x locations will be visited so each constraint c_k becomes known and $\text{PF}^{\infty}(x)$ converges to one if $c_k(x) \leq 0$ for all $k = 1, \ldots, K$. Therefore, if the posterior variance $\tilde{k}_y^{\infty}(x, x) = 0$ for all $x \in X$ then we know the global optimiser. Let's consider the case of design vectors such that $\hat{x} \in \hat{X} = \{x \in X|\tilde{k}^{\infty}(x, x) > 0 \text{ and } x \in F\}$, then,

$$\tilde{\sigma}_y^{\infty}(x, \hat{x}) = \frac{k_y^{\infty}(x, \hat{x})}{\sqrt{k_y^{\infty}(\hat{x}, \hat{x}) + \sigma_\epsilon^2}} > 0$$

If we assume $\tilde{\sigma}_y^{\infty}(x_1, \hat{x}) \neq \tilde{\sigma}_y^{\infty}(x_2, \hat{x})$ for $x_1, x_2 \in X$, then $cKG(x)$ must be strictly positive since for a value of $Z_0 \in Z$, $\mu_y^{\infty}(x_1) + \tilde{\sigma}_y^{\infty}(x_1, \hat{x}) > \mu_y^{\infty}(x_2) + \tilde{\sigma}_y^{\infty}(x_2, \hat{x})$ for $\{z' \in Z : z' > Z_0\}$, and vice versa. Therefore, $\tilde{\sigma}_y^{\infty}(x''', \hat{x}) = \tilde{\sigma}_y^{\infty}(x'', \hat{x})$ must hold for any $x''', x'' \in X$ in order for $cKG(x) = 0$, which results in,

$$\frac{k^{\infty}(x''', \hat{x})}{\sqrt{k^{\infty}(\hat{x}, \hat{x}) + \sigma_\epsilon^2}} = \frac{k^{\infty}(x'', \hat{x})}{\sqrt{k^{\infty}(\hat{x}, \hat{x}) + \sigma_\epsilon^2}}$$

Since $\sigma_\epsilon^2 > 0$, $k_y^{\infty}(x''', \hat{x}) - k_y^{\infty}(x'', \hat{x}) = 0$ and $\tilde{\sigma}_y^{\infty}(x, \hat{x})$ does not change for all $x \in X$. It must follow that $\tilde{\sigma}_y^{\infty}(x, \hat{x}) = 0$. Therefore, the optimiser is known $\arg\max_{x \in X} \mu_y^{\infty}(x)\text{PF}^{\infty}(x) = \arg\max_{x \in X} f(x)\mathbb{I}_{x \in F}$. □