SCIREA Journal of Mathematics
ISSN: 2995-5823
http://www.scirea.org/journal/Mathematics
July 24, 2026
Volume 11, Issue 1, February 2026
https://doi.org/10.54647/mathematics110575

A Robust Sampling-Based Method for Batch Bayesian Optimization

Yiming Ye [1], Xin Qin [2,*] and Lening Ma [3]

[1]College of Finance, Guangdong University of Finance, Guangzhou, Guangdong, China.

[2]International Partnership of Education Research and Communication, Beijing, China.

[3]Ready Global Academy, Beijing, China.

*Corresponding author. E-mail: xin.qin@iperc.org (Xin Qin)

Abstract

This paper presents a sampling-based method called Sampling-Calculation-Optimization (SCO) for the experimental design of batch Bayesian optimization. SCO does not construct new multi-point acquisition functions but samples from the existing one-point acquisition function to obtain candidate designs. The rejection sampling is modified to address different shape of acquisition function. Besides, general discrepancy is computed to compare different designs. To reduce the uncertainty of designs, the genetic algorithm is applied to optimize the designs. Several strategies are proposed to reduce the burden of calculation in the SCO. After calculation and optimization, the SCO designs are less uncertain than other sampling-based methods. Numerical results shows that SCO is not only comparable with other BBO methods in many analytic problem, but also robust when the smoothness of objective function and kernel function are inconsistent.

Keywords: Batch Bayesian optimization, Design of experiment, Acquisition function, General discrepancy

1. Introduction

Global optimization, which seeks to identify the global minimum of an objective function over a predefined feasible domain, remains a fundamental and challenging problem across scientific and engineering disciplines. Formally, it can be stated as:

$$f^* = \min_{\boldsymbol{x}\in\boldsymbol{\mathcal{X}}} f(\boldsymbol{x}),$$

where $\mathcal{X} \subset \mathbb{R}^d$ denotes the d-dimensional feasible search space. This problem becomes particularly intractable when the objective function f exhibits non-convexity, multi-modality, and non-differentiable, and each function evaluation is computationally prohibitive or experimentally costly.

Such expensive black-box objective functions are ubiquitous in a wide range of real-world applications, including hyper-parameter tuning of deep learning models and large language model fine-tuning in artificial intelligence (Victoria & Maragatham, 2021; Wang et al., 2024), optimal design of engineering systems and manufacturing processes (Yu, Picard, & Ahmed, 2025), and stock price forecasting (Dai & Chen, 2026). Bayesian optimization (BO) has emerged as the gold-standard sequential optimization framework for addressing these challenging problems, offering unparalleled sample efficiency compared to traditional grid search, random search, and evolutionary algorithms.

In this paper, we assume $\mathcal{X} = [a, b] \subset \mathbb{R}^d$, and the general idea of Bayesian optimization is as follows:

(i) Build or update the surrogate model $\hat{f}$ based on the database $Y_{\text{data}} = f(X_{\text{data}})$, where $\hat{f}$ usually provides the prediction or description of the experiment response and is easy to solve;

(ii) Construct the acquisition function $\alpha(\boldsymbol{x}) = \alpha\left(\boldsymbol{x}\middle|X_{\text{data}}, Y_{\text{data}}, \hat{f}\right)$ to evaluate the value of further experiments on according to the model and experiment purposes;

(iii) Choose the most valuable points $\boldsymbol{x}^* = \arg\max \alpha(\boldsymbol{x})$ (or $\boldsymbol{x}^* = \arg\min \alpha(\boldsymbol{x})$) by some auxiliary optimization to arrange the sequential experiments;

(iv) Conduct the sequential experiments, and update the database.

The acquisition function $\alpha(x): \mathcal{X} \to \mathbb{R}$ is usually the function of one point $\boldsymbol{x}$. The classic Bayesian optimization method, efficient global optimization (EGO) (Jones, Schonlau, & Welch, 1998), use the expected improvement (EI) as acquisition function, which is efficient and widely used (Zhan & Xing, 2020). Other acquisition functions including the probability of improvement (PI) (de Villiers, Couckuyt, & Dhaene, 2017; Kushner, 1963) and the lower confidence bounds (LCB) (Srinivas, Krause, Kakade, & Seeger, 2010; Sun, Li, Fang, & Li, 2021) are also popular.

According to (iii), the sequential experiments only include one point in each experiment period. However, this is very time-consuming, especially when there are many parallel experimental resources. In order to make better use of these resources, the experimenter needs to identify multiple experimental points for each period. As a result, batch Bayesian optimization (BBO) comes into being.

In BBO, the experimenter should decide n points $X = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$ in one experiment period, where n is the batch size depending on the experimental resources, X is the batch sequential design. One way to obtain X is to construct a multi-point acquisition function $\alpha(x_1, ..., x_n): \mathcal{X}^n \to \mathbb{R}$ such as q-EI (Briffoteaux et al., 2020; Ginsbourger, Riche, & Carraro, 2010). However, $\alpha(\boldsymbol{x_1}, ..., \boldsymbol{x_n})$ is usually highly dimensional, which make it difficult to compute and optimize. To reduce the computations, many researches take asynchronous approaches to design X one after another (Desautels, Krause, & Burdick, 2012; Gonz´alez, Dai, Hennig, & Lawrence, 2016; Zhan, Qian, & Cheng, 2017). At each step, only one point $\boldsymbol{x}_i$ are obtained by maximizing $\alpha_i(\boldsymbol{x})$, then $\alpha_i(\boldsymbol{x})$ is modified into $\alpha_{i+1}(\boldsymbol{x})$ without knowing y_i. The asynchronous methods above reduce the computations by changing a $n \times d$-dimension problem into n separate d-dimension problems. The other way of BBO does not need new acquisition functions, but constructs X based on the original $\alpha(\boldsymbol{x})$. Nguyen, Rana, Gupta, Li, and Venkatesh (2017) uses the infinite Gaussian mixture model to identify the peaks of $\alpha(\boldsymbol{x})$, then the means of Gaussian mixture model are selected as X. Liu, Jiang, and Zheng (2021)

applies the multi-start local optimization and the automatic cluster approach to identify the extremes of $\alpha(\boldsymbol{x})$, and X are selected from the extremes by some criteria.

The approaches above are deterministic in theory. When $\hat{f}$ and α are accurate, the extreme points of α mark the possible optimal point of f, the performance of the Bayesian optimization is greatly improved by auxiliary optimization of α. In practice, however, there is always deviations between $\hat{f}$ and f. In this case, α can only describe a trend of the optimal value of f. As a result, some sampling-based methods are proposed. Instead of finding the extrema of α, the sampling-based methods transform α into a probability density function g and then sample form g to get X. Some sample techniques are applied to obtain better samples. Cai, Qiu, Gao, Yang, and Shao (2017) applies the mode-pursuing sampling method to gather X toward the peak. A three phases sampling method is used in Tao, Zhao, and Ren (2019) to deal with the constrained BBO problems. Ning, Xiao, and Xiong (2020) uses the sampling-importance-resampling from a number theoretic net to accelerate EGO. Li, Yi, and Yang (2021) obtains multiple sampling points by solving a multi-objective problem. Tang, Zhang, Zhang, Lv, and Wang (2022) applies the adaptive weighted importance sampling to reliability assessment. All the sampling-based methods do not require constructing new acquisition function and are easy to conduct. However, no matter what kind of sampling technique they use, the samples are arbitrary and uncertain to some extent, which is not appropriate in expensive experiments.

This article presents a new sampling-based method with less uncertainty to design the sequential experiments of BBO. After we obtain the samples, some calculations and optimizations then proceed to reduce the uncertainty. We call the method as Sampling-Calculation-Optimization (SCO). The rest of this paper is organized as follows. Section 2 introduces some basic theories that are needed. The details of the SCO method are described in Section 3. Some numerical analyses are presented in Section 4. The conclusions are summarized in Section 5.

2. Basic theories

In this section, we first introduce the commonly used acquisition functions, which we will sample from. Then the rejection sampling method is introduced as the major sampling technique. To compare the uncertainty of samples, we finally introduce the general discrepancy.

2.1. Acquisition functions based on Gaussian process

Gaussian process (GP) model is a widely used surrogate model (Rasmussen & Williams, 2009). It provides both the prediction of f as well as the variance. The prior assumption of GP model can be expressed as:

$f \sim \mathrm{N}(\mu, \kappa)$,

where μ is the mean function and κ is a positive-definite kernel function. Matérn kernels are a very flexible class of stationary kernels with the form of

$$\kappa_{\mathrm{M}}(\boldsymbol{x}, \boldsymbol{x}'; \nu) = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\sqrt{2\nu}r\right)^{\nu} J_{\nu}\left(\sqrt{2\nu}r\right),$$

where ν is the smoothness parameter, $r^2 = \left(\boldsymbol{x} - \boldsymbol{x}'\right)^{\mathrm{T}} \Lambda \left(\boldsymbol{x} - \boldsymbol{x}'\right)$ and Λ is a diagonal matrix of d squared length scale parameters θ_i^2. Note that the exponential kernel

$$\kappa_{\mathrm{E}}(\boldsymbol{x}, \boldsymbol{x}') = \exp(-r),$$

is a special case of the Matérn kernel with $\nu = 1/2$, and the squared exponential (Gaussian) kernel

$$\kappa_{\mathrm{S}}(\boldsymbol{x}, \boldsymbol{x}') = \exp(-r^2/2),$$

is the limiting kernel when $\nu \to \infty$.

The posterior of f conditioned on data X_{data}, Y_{data} is also a GP, with the posterior mean

$$\mu(\boldsymbol{x}) = \mu(\boldsymbol{x}|X_{\mathrm{data}}, Y_{\mathrm{data}}) = k(\boldsymbol{x})^T \left[\boldsymbol{K} + \sigma_0^2 \boldsymbol{I}\right]^{-1} Y_{\mathrm{data}},$$

and the posterior variance

$$\sigma^2(\boldsymbol{x}) = \sigma^2(\boldsymbol{x}|X_{\mathrm{data}}, Y_{\mathrm{data}}) = \kappa(\boldsymbol{x}, \boldsymbol{x}) - k(\boldsymbol{x})^T \left[\boldsymbol{K} + \sigma_0^2 \boldsymbol{I}\right]^{-1} k(\boldsymbol{x}),$$

where $k(\boldsymbol{x}) = [\kappa(\boldsymbol{x_1}, \boldsymbol{x}), ..., \kappa(\boldsymbol{x_n}, \boldsymbol{x})]^{\mathrm{T}}$, $\boldsymbol{K}$ is the matrix such that $(\boldsymbol{K})_{i,j} = \kappa(\boldsymbol{x_i}, \boldsymbol{x_j})$.

This posterior is used to form the acquisition function $\alpha(\boldsymbol{x})$, and $\alpha(\boldsymbol{x})$ needs to balance between exploitation and exploration so that global optimization can be achieved by maximizing (or minimizing) $\alpha(\boldsymbol{x})$. We introduce two widely used acquisition functions here. One is the EI function:

$$\alpha_{\mathrm{EI}}(\boldsymbol{x}) \triangleq \mathrm{E}[\max\{0, y^* - \mu(\boldsymbol{x})\}] = (y^* - \mu(\boldsymbol{x}))\Phi\left(\frac{y^* - \mu(\boldsymbol{x})}{\sigma(\boldsymbol{x})}\right) + \sigma(\boldsymbol{x})\phi\left(\frac{y^* - \mu(\boldsymbol{x})}{\sigma(\boldsymbol{x})}\right),$$

where y^* is the minimum of Y_{data}, ϕ and Φ are the probability density function (PDF) and the cumulative distribution function (CDF) of standard Gaussian distribution respectively. The other is the LCB function:

$$\alpha_{\mathrm{LCB}}(\boldsymbol{x}) = \mu(\boldsymbol{x}) - \beta\sigma(\boldsymbol{x}),$$

where β is a tunable parameter to balance exploitation against exploration. Note that α_{EI} is always no less than 0, while α_{LCB} is not. For sampling, we define $\alpha^*_{\mathrm{LCB}}(\boldsymbol{x}) = \alpha^* - \alpha_{\mathrm{LCB}}(\boldsymbol{x})$ in this paper, where α^* is the maximum of α_{LCB}.

2.2. Rejection sampling

Rejection sampling (RS) is a widely used sampling method, we introduce it briefly, see Bishop (2006) for more information.

Suppose we would like to sample from a target PDF $g(\cdot) \propto \alpha(\cdot)$ on $\mathcal{X}$, which is hard to sample directly but easy to compute. The main steps of RS are as follows:

Step 0: Choose a proposal distribution with density $h(\cdot)$, which is easy to sample.

Step 1: Find a constant L which is large enough to ensure $\alpha(\boldsymbol{x}) \leq Lg(\boldsymbol{x}), \forall \boldsymbol{x} \in X$.

Step 2: Generate $\boldsymbol{u}_i \sim h(\cdot)$ and $v_i \sim U(0,1)$.

Step 3: If $Lg(\boldsymbol{u}_i)v_i \leq \alpha(\boldsymbol{u}_i)$, then accept $\boldsymbol{u}_i$ as a sample $\boldsymbol{s}_i$. Otherwise, reject $\boldsymbol{u}_i$.

Note $U = \boldsymbol{u}_1, ..., \boldsymbol{u}_N$ as the pre-sample set, and $S = \boldsymbol{s}_1, ..., \boldsymbol{s}_m$ as the sample set. As indicated by Bishop (2006), $\boldsymbol{s}_j \sim g(\cdot)$ and the probability to accept $\boldsymbol{u}_i$ is inversely proportional to L. Generally, one needs to choose a smallest L to ensure $\alpha(\boldsymbol{x}) \leq Lg(\boldsymbol{x}), \forall \boldsymbol{x} \in X$.

2.3. General Discrepancy

There are many interpretations of the discrepancy (Li, Kang, & Hickernell, 2020). In this paper, we refer to the interpretation in Section 2.4 of Fang, Liu, Qin, and Zhou (2018).

Let $K(\cdot\,,\cdot)\colon \mathcal{X}\times\mathcal{X}\to\mathbb{R}$ be a symmetric and positive semi-definite kernel function, G is a CDF in

$$\mathcal{K}=\left\{G\left|\int_{\mathcal{X}\times\mathcal{X}}K(\boldsymbol{u},\boldsymbol{v})\,dG(\boldsymbol{u})dG(\boldsymbol{v})<\infty\right.\right\}.$$

Define the inner product of two arbitrary $G,H\in\mathcal{K}$ as

$$<G,H>_K=\int_{\mathcal{X}\times\mathcal{X}}K(u,v)\,dG(u)dH(v).$$

The general discrepancy of a design $X=\{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n\}$ with respect to the target distribution G using the kernel K is defined as

$$\begin{aligned}D^2(X,G,K)&\triangleq\left|\left|G_X-G\right|\right|_K^2=\langle G_X-G,G_X-G\rangle_K\\&=\int_{\mathcal{X}\times\mathcal{X}}K(\boldsymbol{u},\boldsymbol{v})\,\mathrm{d}(G-G_X)(\boldsymbol{u})\mathrm{d}(G-G_X)(\boldsymbol{v})\\&=\int_{\mathcal{X}\times\mathcal{X}}K(\boldsymbol{u},\boldsymbol{v})\,dG(\boldsymbol{u})dG(\boldsymbol{v})-\frac{2}{n}\sum_{i=1}^{n}\int_{\mathcal{X}}K(\boldsymbol{u},\boldsymbol{x_i})\,dG(\boldsymbol{u})\\&\quad+\frac{1}{n^2}\sum_{i,j=1}^{n}K(\boldsymbol{x_j},\boldsymbol{x_i})\end{aligned}\qquad\text{eq 2.1}$$

where G_X is the empirical distribution of X. $D^2(X,G,K)$ can be taken as the distance between G and G_X, so we prefer a design with low discrepancy to better represent the target distribution.

The kernel function K corresponds to an inner product of the Hilbert space. By taking different kernels, we can define different kinds of discrepancy, such as the widely used centered discrepancy, wrapped discrepancy, or mixed discrepancy. In this article, we use the kernel function of the wrapped discrepancy (Fang et al., 2018):

$$K(\boldsymbol{u},\boldsymbol{v})=\prod_{i=1}^{d}\left[\frac{3}{2}-|u_i-v_i|+(u_i-v_i)^2\right].\qquad\text{eq 2.2}$$

Because G is arbitrary, the high-dimension integral in eq 2.1 is difficult to compute; we use

the Monte Carlo method to estimate $D^2(X, G, K)$. Rewrite eq 2.2 as

$$D^2(X, G, K) = \int_{\mathcal{X}\times\mathcal{X}} K(\boldsymbol{u}, \boldsymbol{v}) \cdot g(\boldsymbol{u}) \cdot g(\boldsymbol{v}) \mathrm{d}\boldsymbol{u}\mathrm{d}\boldsymbol{v} - \frac{2}{n}\sum_{i=1}^{n}\int_{\mathcal{X}} K(\boldsymbol{u}, x_i) \cdot g(u)\mathrm{d}\boldsymbol{u} + \frac{1}{n^2}\sum_{i,j=1}^{n} K(\boldsymbol{x_j}, \boldsymbol{x_i}), \quad \text{eq 2.3}$$

where g is the PDF of G. We then get an estimation of $D^2(X, G, K)$ as

$$\widehat{D}^2(X, G, K) = \frac{1}{N^2}\sum_{i,j=1}^{N} K(\boldsymbol{u_j}, \boldsymbol{u_i}) \cdot g(\boldsymbol{u_j}) \cdot g(\boldsymbol{u_i}) - \frac{2}{nN}\sum_{i=1}^{n}\sum_{j=1}^{N} K(\boldsymbol{u_j}, \boldsymbol{x_i}) \cdot g(\boldsymbol{u_j}) + \frac{1}{n^2}\sum_{i,j=1}^{n} K(x_j, x_i), \quad \text{eq 2.4}$$

where $u_i \sim \mathrm{U}(\mathcal{X}), i = 1,2, \ldots, N$.

3. Method for batch Bayesian optimization

In this section, we propose a method to implement BBO. The new method is based on original one-point acquisition function, and its main idea is to exchange (iii) in Section 1 with a revised (iii') as follows:

(iii') Take $g \propto \alpha$ as the PDF of G and construct a sequential design X^* to fit G best;

For convenience, note $D^2(X, \alpha, K) \triangleq D^2(X, G, K)$, where G is induced by the density $g \propto \alpha$. We use $D^2(X, \alpha, K)$ to measure the fitness described above in (iii'). If $G \sim U(\mathcal{X})$, this kind of design is also called a uniform design. The methods for constructing uniform design are various and efficient (Chen, Hsu, Hung, & Wang, 2014; Fang, Lin, Winker, & Zhang, 2000; Zhou & Fang, 2013). However, it becomes complicated when G is arbitrary. $D^2(X, G, K)$ has no analytic form when G is arbitrary, calculating $D^2(X, G, K)$ is already difficult, let alone optimizing it. As a result, the construction of X^* has to make a trade-off between optimality and feasibility, aiming to construct a design with relatively low discrepancy in an acceptable time.

In this section, we first used the modified RS to generate sample points S. Subsequently, the candidate design X will be chosen from S. To avoid the arbitrariness of sampling method,

general discrepancy is calculated to compare the candidate designs. Ultimately, X^* is determined through optimization to reduce the uncertainty. We call this method Sampling-Calculation-Optimization (SCO). Several strategies are proposed to reduce the burden of calculation.

3.1. Sampling

We first use RS to obtain sample set S from $g \propto \alpha$, the proposal distribution $h(\cdot)$ is chosen as $\mathrm{U}(\mathcal{X})$. In this way, $h(\boldsymbol{x})$ is constant and easy to compute, and the pre-sample set U can be used in estimating $D^2(X, \alpha, K)$. Subsequently, as a requirement of RS, L should be large enough to ensure $\alpha(\boldsymbol{x}) \leq Lh(\boldsymbol{x})$=$L^*$. We set $L^* = \alpha(\boldsymbol{x}^*) = \max_{\boldsymbol{x}\in\mathcal{X}} \alpha(\boldsymbol{x})$, and take $\boldsymbol{x}^*$ as the first point of X^*, then only $n-1$ points leaved to be designed. In this way, our BBO method can be compatible with the original one-point Bayesian optimization method. Consequently, we propose the following two strategies.

Strategy 1. *Save the pre-sample set U and* $\alpha_i = \alpha(\boldsymbol{u}_i)$ *for calculation.*

Strategy 2. *Maintain* $\boldsymbol{x}^*$ *in the final design* X^*.

In order to better implement the strategies above, we modified the RS. Since the pre-sample set U will be used in estimating $D^2(X, \alpha, K)$, and sample set S will be used in optimizing X^*. The size of U and S, which is N and m, should balance between accuracy and efficiency. In this paper, we set $N \geq N_{\min}$ to ensure the accuracy of $\widehat{D}^2(X, \alpha, K)$, and $N \leq N_{\max}$ to accommodate the computational burden. Similarly, we set $m_{\min} \leq m \leq m_{\max}$ so that X^* has proper points to optimize. Due to the property of RS, the amount of N and m cannot be controlled at the same time, they are depending on L and the shape of α. To control N and m, we modify RS as **Algorithm 1**.

First, we generate $N_{\min}$ points $\boldsymbol{u}_i$, v_i at once, and calculate $\alpha_i = \alpha(\boldsymbol{u}_i)$. $L^* = \max \alpha(\boldsymbol{x}) = \alpha(\boldsymbol{x}^*)$ can be found by a local optimizer starting at the $\boldsymbol{u}_i$ with best α_i. All $\alpha_i = \alpha(\boldsymbol{u}_i)$ including $\alpha(\boldsymbol{x}^*)$ are saved as $\mathbf{A}$ for further computation. We fix $\boldsymbol{x}^* \in S$ and the rest of S can be chosen by $L^* v_i \leq \alpha_i$. Considering two shapes of α, if α is steep, resulting in there is less than $m_{\min}$ points in S, we continue RS until $m = m_{\min}$. For more extreme cases, if U becomes more than N_{max} points as RS proceeding, we stop adding $\boldsymbol{u}_i$ then turn to regenerate v_i, $i =$

$1, \ldots, N$ to screen U repeatedly until there are enough points in S. If α is relatively flat so that S contains more than $m_{\max}$ points, we only choose the first m_{max} points in it. In practice, we set $N_{\min} = 10^4,\ N_{\max} = 10^5,\ m_{\min} = 10n,\ m_{\max} = 100n$.

Algorithm 1 Sampling of SCO

Input: $\alpha(\boldsymbol{x})$, $N_{\min}$, $N_{\max}$, $m_{\min}$, $m_{\max}$
Output: S, U, **A**
$N = N_{\min}$, generate $U = \{\boldsymbol{u}_i, i = 1, \ldots, N\}$ uniformly distributed on $\mathcal{X}$;
calculate $\alpha_i = \alpha(\boldsymbol{u}_i), i = 1, \ldots, N$ and save them as **A**;
find $L^* = max\ \alpha(\boldsymbol{x}) = \alpha(\boldsymbol{x}^*)$ starting at the $\boldsymbol{u}_i$ with best α_i;
generate $v_i, i = 1, \ldots, N$ uniformly distributed on $(0, 1)$;
$S = \{\boldsymbol{x} *\} \cup \{\boldsymbol{u}_i | L^* v_i \le \alpha_i\}$; ▷ $\boldsymbol{x}^*$ is the first point of S
while $|S| < m_{\min}$ **do**
    if $N < N_{\max}$ **then** ▷ continue RS
        $N = N + 1$;
        generate $\boldsymbol{u}_N, v_N$, calculate α_N;
        $U = U \cup \{\boldsymbol{u}_N\}, \mathbf{A} = \mathbf{A} \cup \{\alpha_N\}$;
        if $\alpha_N \le L^* v_N$ **then**
        $S = S \cup \{\boldsymbol{u}_N\}$
        end if
    else ▷ re-screen U
        regenerate $v_i, i = 1, \ldots, N$;
    supplement S by $S = S \cup \{\boldsymbol{u}_i | \alpha_i \le M^* v_i\}$
    end if;
end while
if $|S| > m_{\max}$ **then**
    S = the first $m_{\max}$ points of S;
end if

3.2. Calculation

We choose several candidate design X_k from S, and calculate $D^2(X_k, \alpha, K)$ to compare them. For convenience, let $D_k^2 \triangleq D^2(X_k, \alpha, K)$, and I_k denote the indices of the points of X_k in S. Upon reviewing eq 2.4, if we take the proposal distribution $h(\ )$ as $\mathrm{U}(\mathcal{X})$, the pre-sample set U in RS can be used to estimate D^2. The details are as follows. First of all, to transform an acquisition function as a PDF, we need to normalize it by $g(\boldsymbol{x}) = \alpha(\boldsymbol{x}) / \int_{\mathcal{X}} \alpha(\boldsymbol{u}) \mathrm{d}\boldsymbol{u}$, where $\int_{\mathcal{X}} \alpha(\boldsymbol{u}) \mathrm{d}\boldsymbol{u}$ can be estimated by

$$\int_{\mathcal{X}} \alpha(\boldsymbol{u}) d\boldsymbol{u} \hat{=} \frac{1}{N} \sum_{i=1}^{N} \alpha_i \triangleq \frac{S_\alpha}{N}, \qquad \text{eq 3.1}$$

where α_i have been computed in sampling. Referring then to equation eq 2.4, $D^2(X, \alpha, K)$ can be written as

$$\hat{D}^2(X, \alpha, K) = \frac{1}{S_\alpha^2} \sum_{i,j=1}^{N} K\left(\boldsymbol{u}_j, \boldsymbol{u}_i\right) \cdot \alpha_j \cdot \alpha_i \qquad \text{eq 3.2}$$

$$-\frac{2}{nS_\alpha}\sum_{i=1}^{n}\sum_{j=1}^{N}K\left(\boldsymbol{u}_j,\boldsymbol{x}_i\right)\cdot\alpha_j+\frac{1}{n^2}\sum_{i,j=1}^{n}K\left(\boldsymbol{x}_j,\boldsymbol{x}_i\right)$$
$$\triangleq M_0(\alpha,K)-\frac{2}{n}\sum_{i=1}^{n}M_1\left(\boldsymbol{x}_i,\alpha,K\right)+M_2(X,K),$$

where $M_0(\alpha,K)=\sum_{i,j=1}^{N}K\left(\boldsymbol{u}_j,\boldsymbol{u}_i\right)\alpha_j\alpha_i/S_\alpha^2$ is irrelative to X, $M_2(X,K)=\sum_{i,j=1}^{n}K\left(\boldsymbol{x}_j,\boldsymbol{x}_i\right)/n^2$ is irrelative to α, and $M_1(\boldsymbol{x}_i,\alpha,K)=\sum_{j=1}^{N}K\left(\boldsymbol{u}_j,\boldsymbol{x}_i\right)\alpha_j/S_\alpha$ is irrelative to other elements of X expect $\boldsymbol{x}_i$. To reduce the burden of calculation, we propose the following two strategies.

Strategy 3. *Since M_0 is independent of X and it is most computationally intensive, we define*

$$\hat{D}_{-}^{2}(X,\alpha,K)=-\frac{2}{n}\sum_{i=1}^{n}M_1\left(\boldsymbol{x}_i,\alpha,K\right)+M_2(X,K). \qquad \text{eq 3.3}$$

We only compute eq 3.3 *to compare the fitness of different candidate designs. If exact $D^2(X,\alpha,K)$ is necessary for analysis, we calculate $M_0(\alpha,K)$ once at the end, and add it to $\widehat{D}^2(X,\alpha,K)$.*

Strategy 4. *Since M_1 is only related to one point, we calculate $M_1(\boldsymbol{s}_i,\alpha,K),\forall\boldsymbol{s}_i\in S$ and save them as* **M** *for future calculation and optimization.*

In summary, we divided the calculation into two parts. In the first part, we build **M** for future calculation and optimization. $\alpha_i\in\mathbf{A}$ can be retrieved when calculating $K\left(\boldsymbol{u}_j,\boldsymbol{s}_i\right)\alpha_i$ in $M_1(\boldsymbol{s}_i,\alpha,K)$. In the second part, we calculate D^2 with the help of **M**. The calculation of SCO is shown in **Algorithm 2**.

Algorithm 2 Calculation of SCO

Input: S, U, **A**
Output: **M**, X_k, I_k, and D^2
for $i=1:m$ **do** ▷ build **M**
  calculate $M_1(\boldsymbol{s}_i,\alpha,K)$ by $\boldsymbol{s}_i$, U and **A**;
  $\mathbf{M}=\mathbf{M}\cup M_1(\boldsymbol{s}_i,\alpha,K)$
end for
for $i=1:k$ **do** ▷ calculate D^2
  $I_k=\{1\}\cup\{\text{randperm}(2:m,n-1)\}$; ▷ $\boldsymbol{x}^*$ is the first point of X_k
  $X_k=\{\boldsymbol{s}_i|\forall i\in I_k\}$;
  $D_k^2=-2\sum_{i\in I_k}\mathbf{M}(i)/n+M_2(X_k,K)$
end for
calculate $M_0(\alpha,K)$ by U and **A** ▷ If exact D^2 is necessary
$D^2=D^2+M_0(\alpha,K),\forall i=1,\ldots,k$;

3.3. Optimization

To reduce the uncertainty of the final design X^*, we would like to optimize the candidate designs under the criterion of $D^2(X, \alpha, K)$. There are two difficulties in optimization. One is the burden of calculation. No matter what kind of algorithm we choose, every time we generate a new design, calculating eq 3.2, even eq 3.3, is time-consuming. The other difficulty is the large number of combinations of candidate sets, all possible combinations are C_n^m, which makes optimization impractical.

Strategy 5. *Assume a new design* X_{new} *with general discrepancy* D_{new} *is generated from an old design* X_{old} *with general discrepancy* D_{old}*, and all the points are restricted in* S*. We update* D_{new} *by* D_{old}*.*

Referring to eq 3.2, we have

$$\begin{aligned} D_{\text{new}}^2 &= D_{\text{old}}^2 + \frac{2}{n}\sum_{i\in I_{\text{old}}} \mathbf{M}(i) - \frac{2}{n}\sum_{j\in I_{\text{new}}} \mathbf{M}(j) - M_2(X_{\text{old}}, K) + M_2(X_{\text{new}}, K) \\ &\triangleq D_{\text{old}}^2 + \Delta(X_{\text{new}}, X_{\text{old}}), \end{aligned} \qquad \text{eq 3.4}$$

where D_{old} and $\mathbf{M}$ are already known. We only retrieve the changed terms of $\mathbf{M}$ in I_{old}, I_{new}, and calculate $M_2(X_{\text{old}}, K)$ and $M_2(X_{\text{new}}, K)$, which is easy to compute.

To deal with the large number of combinations, we apply the genetic algorithm (GA) to optimize X^*. GA is a classical intelligent optimization algorithm containing the imitation process of fitness, selection, crossover, and mutation in the genetic (Goldberg, 1988). All these process are well suited for optimizing experimental design. Since $D^2(X, \alpha, K)$ is positive and limits to zero, $D^{-2}(X, \alpha, K)$ is a good choice of fitness function. We take $\boldsymbol{x}_i$ as the gene of X, then the offspring can be generated by exchanging the corresponding points of parents with probability P_{c} (default 0.5). S can be taken as the gene pool, so that $\boldsymbol{x}_i$ can mutate into $\boldsymbol{s}_j$ with probability P_{m} (default 0.1). The main processes of the GA are described in **Algorithm 3**, and we keep the first point of all design, which is $\boldsymbol{x}^*$ unchanged during crossover and mutation.

Algorithm 3 Optimization of SCO

Input: X_k, I_k, D^2, S, and $\mathbf{M}$; Parameters: P_{c} and P_{m}
Output: X^* and D^*
1: initialize population: Parents $= X_1, \ldots, X_k$;
2: $k^* = \arg\min_k D_k^2$, $X^* = X_{k^*}$, $D^* = D_{k^*}^2$;
3: **while** Stopping criterion is not met **do**

Offspring $= \{X^*\}$ ▷ Retain the best individual
Fit$(X_i) = D^{-2}$, $i = 1, \ldots, k$;
while |Offspring| $< m$ **do**
    (Father, Mother) $=$ Selection(Parents; Fit);
    Child $=$ Crossover(Father, Mother; P_c); ▷ keep $\boldsymbol{x}^*$ unchanged
    Child $=$ Mutation(Child, S; P_m); ▷ keep $\boldsymbol{x}^*$ unchanged
    $D_{\text{Child}} = D_{\text{Father}} + \Delta(D_{\text{Child}}, D_{\text{Father}})$; ▷ update D_{Child}
    Offspring $=$ Offspring $\cup$ {Child};
end while
update X^* and D^*;
Parents $=$ Offspring;
end while

3.4. Summary

We use an example to display the uncertainty of SCO and another sampling-based method. We build a GP model according to $Y_{\text{data}} = f(X_{\text{data}})$, where f is the Branin function and X_{data} is the set of 3×3 mesh points on $X = [-5, 10] \times [0, 15]$, then draw the contours of α_{EI}, see the left of **Figure 1**. The pre-sample set U, the sample set S, and the final design X^* are plotted in different color. Compared with U, the sample set S greatly reduce the number of possible combinations without losing the features of α_{EI}. In addition, X^* is less uncertain than the sample set S after optimization, this point of view can also be illustrated by the right of **Figure 1**. We compare the uncertainty of two methods—the SCO method and the accelerated EGO (aEGO) method (Ning et al., 2020). The aEGO chooses the best $\boldsymbol{x}^*$ with maximum EI as the first point of X, then supplement the rest points of X by SIR samples from the quasi uniform random points in $\mathcal{X}$, without computation and optimization. We set $n = 5, 10, 20$ and generate 100 designs to compare their general discrepancies with the boxplots, which are grouped as aEGO-5, SCO-5 and so on. From the right of **Figure 1**, we can see that the general discrepancy of SCO designs are more concentrated, which illustrate that the SCO designs are much less uncertain. In addition, the median of SCO designs is even better, in terms of general discrepancy, than that of aEGO designs with the double sample size.

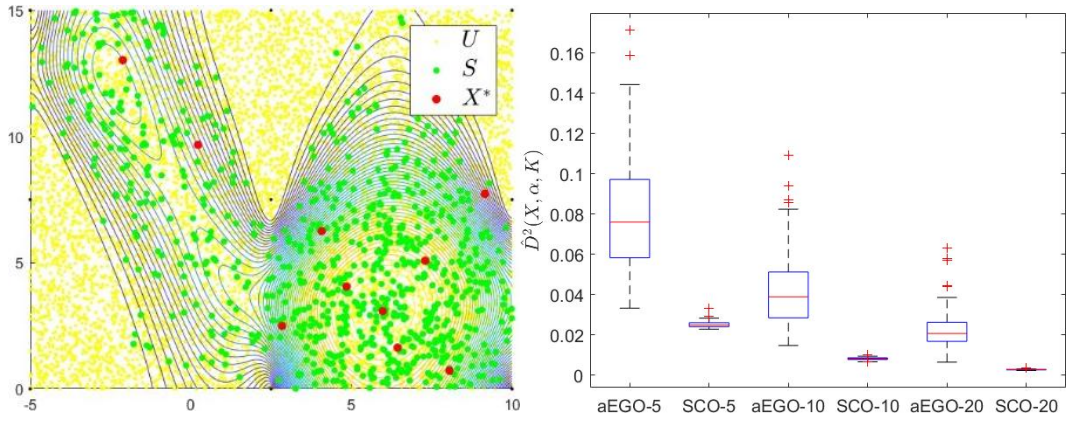

Figure 1 Comparison between SCO and aEGO. Left: Scatter plots of different sets in SCO, X^* is less uncertain than S after optimization. Right: General discrepancy of SCO and aEGO design, $\widehat{D}(X, \alpha, K)$ of SCO designs are more concentrated.

4. Numerical results

In this section, we first compare various BBO methods on some analytic problems. Further comparisons are then shown under different assumptions of GP model. Finally, a practical case is presented to show the application scenario of SCO.

4.1. Performance on solving analytic problems

We compare the performance of SCO on some analytic problems, including Branin, Goldstein-Price, Six-hump, Shubert, Hartmann-3, Hartmann-6, and Ackley. The details of these problems can be found in (http://www.sfu.ca/~ssurjano/optimization.html), we only list some characteristics in **Table 1**.

Table 1 Characteristics of the synthetic functions

Function	Dimension	Bounds	Global minimum
Branin	2	$[-5, 10] \times [0, 15]$	0.3979
Goldstein-Price	2	$[-2, 2]^2$	3
Six-hump	2	$[-3, 3] \times [-2, 2]$	-1.032
Shubert	2	$[-10, 10]^2$	-186.7
Hartmann-3	3	$[0, 1]^3$	-3.863
Hartmann-6	6	$[0, 1]^6$	-3.322
Ackley	10	$[-32.77, 32.77]^{10}$	0

For each problem, we begin with $5 \times d$ random points to get the initial data $X_{\text{data}}, Y_{\text{data}}$, then build the GP model by the Gaussian kernel, the parameters in GP model are optimized by

maximum likelihood estimation. 6 benchmark BBO methods are compared to our SCO method, they are q-EI (Briffoteaux et al., 2020; Ginsbourger et al., 2010), batch LCB (BLCB) (Cheng, Jiang, Zhou, Hu, & Shu, 2021; Desautels et al., 2012), adaptive local search (ALS) (Liu et al., 2021), mode-pursuing sampling (MPS) Cai et al. (2017), and accelerated EGO (aEGO) (Ning et al., 2020). q-EI and BLCB are the classic BBO methods based on α_{EI} and α_{LCB} respectively, their acquisition functions are modified step by step. We use the Kriging Believer strategy in q-EI, and set the parameter $\beta = 2$ in LCB. ALS is an adaptive method which can find the extremes of acquisition function. α_{EI} and α_{LCB} are both applied in ALS, noted as ALS-EI and ALS-LCB. The confidence bound parameter p of ALS is set to be 0.4. Since the batch size of ALS is adaptive, we supplement random points when its batch size is less then we set. MPS and aEGO are two sampling-based methods, and they both sample form α_{EI}. To analyze the performance of different acquisition functions, we apply α_{EI} and α^*_{LCB} respectively in SCO,

noted as SCO-EI and SCO-LCB. For $n = 2, 4, 8$, the optimal responses found in 5 iterations of each BBO method were recorded. We repeated each test 10 times, and listed the means and standard deviation in **Table 2**. The best results of each row are highlighted in bold.

Totally speaking, the proposed two SCO methods are comparable with the other 6 BBO methods. SCO-LCB outperforms all other methods in solving the Ackley problem, which is high dimension. SCO-EI performs better than the similar aEGO in almost every problem. While both two SCO methods have few advantage when $n = 2$. This shows that the SCO methods are not suitable for small batch sizes.

Table 2 Test results on analytic functions

Function	n	q-EI	BLCB	ALS-EI	ALS-LCB	MPS	aEGO	SCO-EI	SCO-LCB
Branin	2	**1.0707**	1.9772	1.2257	2.1132	1.3459	1.5416	2.4048	1.8467
		±0.962	±1.614	±1.599	±1.142	±0.942	±1.005	±1.526	±1.194
	4	0.4248	1.8705	0.5079	0.5282	0.6636	0.488	1.5546	**0.4084**
		±0.044	±1.649	±0.138	±0.150	±0.202	±0.102	±1.204	±0.014
	8	0.4565	0.4268	0.4031	0.402	0.4282	0.4002	0.4225	**0.3983**
		±0.071	±0.060	±0.008	±0.004	±0.020	±0.003	±0.058	±0.001
Goldstein	2	80.3457	56.2767	102.2068	73.6919	**49.5559**	73.7716	53.7987	53.8261
		±78.947	±39.590	±112.116	±61.144	±28.893	±65.905	±44.077	±42.482
	4	64.8274	45.3015	51.9064	48.3626	33.4362	49.1273	28.1143	**24.2261**

		±67.357	±25.031	±51.940	±35.814	±34.873	±29.828	±19.403	±18.720
	8	64.0252	24.0441	36.7649	43.211	30.5153	38.1262	**23.2315**	36.5226
		±73.994	±11.479	±34.906	±37.382	±12.990	±16.865	±13.179	±29.236
Six-hump	2	0.4727	0.2067	0.5544	**-0.1075**	0.6599	0.3833	0.306	0.9336
		±1.403	±1.045	±1.477	±0.887	±1.280	±1.674	±1.007	±1.633
	4	0.2369	**0.0918**	0.2314	0.7308	0.4002	0.1655	0.8547	0.6899
		±1.022	±0.629	±0.709	±1.015	±0.747	±0.555	±1.948	±1.101
	8	0.0835	-0.3505	-0.5242	-0.3214	-0.4763	0.3011	**-0.6075**	-0.0181
		±0.877	±0.860	±0.558	±0.507	±0.406	±1.097	±0.507	±0.555
Shubert	2	-54.4652	**-79.5178**	-67.4358	-57.4947	-67.0883	-60.4264	-71.7474	-59.7068
		±41.825	±64.620	±49.289	±52.419	±42.548	±44.293	±57.947	±49.795
	4	-62.8365	-55.2703	-64.9533	-57.9921	-51.9397	-62.2919	**-79.9902**	-50.7535
		±31.170	±34.939	±40.056	±42.892	±29.324	±45.796	±45.296	±45.272
	8	-73.1007	-97.1223	-86.6082	**-109.0976**	-82.1433	-74.1798	-64.4033	-77.2148
		±33.309	±38.924	±37.605	±52.759	±44.406	±39.271	±31.780	±54.799
Hartmann-3	2	-3.7957	-3.7623	-3.7135	-3.7233	-3.6984	**-3.8106**	-3.689	-3.628
		±0.084	±0.260	±0.250	±0.239	±0.092	±0.075	±0.230	±0.273
	4	-3.8499	**-3.8627**	-3.8554	-3.8028	-3.7345	-3.8561	-3.8276	-3.8214
		±0.008	±0.000	±0.005	±0.056	±0.056	±0.016	±0.017	±0.030
	8	-3.8475	**-3.8627**	-3.8588	-3.8561	-3.7364	-3.8592	-3.836	-3.8358
		±0.024	±0.000	±0.005	±0.012	±0.081	±0.003	±0.018	±0.026
Hartmann-6	2	-2.0832	**-2.3988**	-2.1921	-2.2242	-1.9848	-2.3537	-2.0138	-2.1778
		±0.605	±0.396	±0.524	±0.573	±0.543	±0.545	±0.622	±0.761
	4	-2.5115	-2.6237	-2.7523	-2.5891	-2.2458	-2.7622	**-2.9534**	-2.3598
		±0.635	±0.394	±0.673	±0.604	±0.421	±0.400	±0.426	±0.737
	8	-2.8514	-3.09	-3.0268	-2.4615	-2.4151	-2.4289	**-3.1656**	-2.5709
		±0.365	±0.306	±0.227	±0.529	±0.359	±0.465	±0.157	±0.400
Ackley	2	17.0047	16.9755	15.9259	16.4886	18.2989	15.6907	16.9137	**15.6365**
		±1.243	±1.532	±1.477	±1.167	±0.613	±1.921	±1.706	±2.258
	4	16.5909	15.899	14.711	15.8355	17.5574	14.3158	15.8946	**12.191**
		±1.350	±1.045	±0.908	±0.804	±0.998	±1.328	±1.780	±2.551
	8	16.2492	15.1158	12.6204	12.774	17.3537	11.9993	14.2091	**10.6465**
		±1.329	±1.786	±0.961	±0.735	±0.700	±1.998	±2.136	±2.946

4.2. Robustness of SCO

In last section, we displayed the performance of SCO on some analytic problem. The kernel function in GP model is selected as the Gaussian kernel. That is to say, both the objective function and the kernel function are infinitely smooth. In practical experiment, the objective function is always non-analytic and unsmooth. The surrogate model should more flexible to deal with it. In GP model, experimenters usually choose Matérn kernel as a flexible one. In

addition to the scale parameter, the Matérn kernel also contains the smoothness parameter.

In practical experiment, no matter the parameters are set in advance or optimized by methods such as maximum likelihood, they may be deviated from the true situation. In this section, we analyze the robustness of BBO methods when the smoothness of objective function and the kernel function are inconsistent. We generate the objective functions by the sample paths of Bayesian prior. Let $f \sim \mathrm{N}(0, \kappa_{\mathrm{M}})$, and set all the scale parameters $\theta_i = 1$. We generate several sample paths f_i from $\mathrm{N}(0, \kappa_{\mathrm{M}}(\cdot, \cdot\cdot; \nu_0))$. Examples of different f_i are shown in **Figure 2**. The left of **Figure 2** displays 5 sample paths with different generators when $\nu_0 = 2.5$, The right of **Figure 2** displays 5 sample paths with same generators but different ν_0. Usually, we do not know the smoothness of observation function, not to mention choosing suitable ν_1 to build GP. To analyze the robustness of BBO methods, GP is built with prior $\mathrm{N}(0, \kappa_{\mathrm{M}}(\cdot, \cdot\cdot; \nu_1))$. We set $d = 2$ and generate 10 f_i for each $\nu_0 = 0.1, 0.5, 2.5, \infty$, then set $n = 4$ and optimize f_i by BBO methods with $\nu_1 = 0.1, 0.5, 2.5, \infty$ respectively. **Figure 3** displays the average of the best results found in iterations for each combination of (ν_0, ν_1).

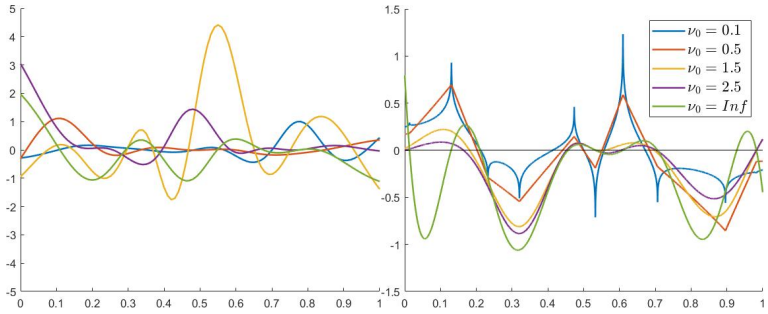

Figure 2 Sample paths of Bayesian prior. Left: Sample paths with different generators when $\nu_0 = 2.5$. Right: Sample paths with same generators but different ν_0.

The deterministic methods and the sampling-based methods show different performance when ν_0 and ν_1 are different. In those graph above the diagonal, $\nu_0 \leq \nu_1$, the deterministic methods perform well in most situation, and BLCB is the most competitive. While in those graph below the diagonal, $\nu_0 > \nu_1$, and the sampling-based methods—MPS, aEGO and SCO performs well. Especially when $\nu_1 = 0.1$, the deterministic methods are almost failed, and MPS shows particular advantage. In summary, the SCO methods show robustness when the smoothness of objective function and the kernel function are inconsistent. The sampling-

based nature makes SCO good at dealing with the situation when $\nu_0 > \nu_1$. At the same time, the calculation and optimization make it deterministic to some extent, which guarantees the efficiency when $\nu_0 \leq \nu_1$.

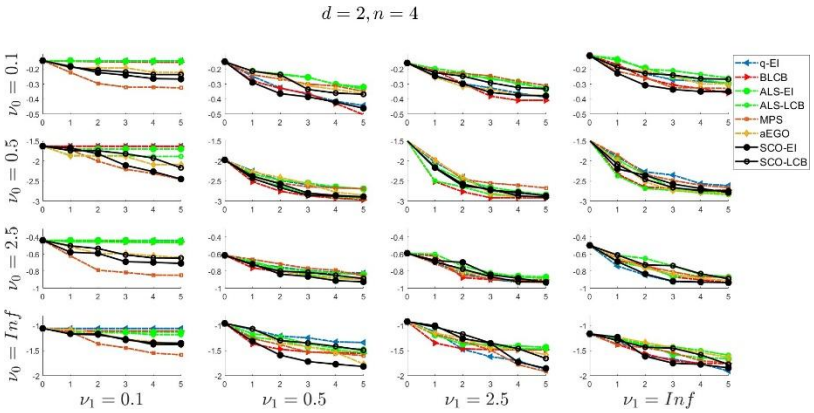

Figure 3 Comparison results of BBO methods on sample paths of Bayesian prior with different smoothness. Each row corresponds to the f_i generated with same ν_0. Each column corresponds to the models built with same ν_1.

5. Conclusion

This article proposes a sampling-based method for the experimental design of BBO. The main processes of this method include Sampling-Calculation-Optimization. To reduce the uncertainty, the samples are optimized under the criterion of general discrepancy. We propose several strategies to reduce the burden of calculation and make the optimization possible. After calculation and optimization, the uncertainty of the SCO was much less than the sampling-only method. Two widely used acquisition functions are applied in SCO. Numerical results shows that the two SCO methods are not only comparable with other batch Bayesian optimization methods in many analytic problems, but also robust when the smoothness of objective function and kernel function are inconsistent.

SCO provides a new idea to reduce the uncertainty of sampling-based experimental design. The keystone is to calculate and optimize a criterion of the candidate design. In this article, we measure the fitness of X^* and G in (iii') by general discrepancy with wrapped kernel. The performance of other kind of discrepancy or other measurement is leaved to be further studied.

Replication of results

The datasets generated during the current study are available from the corresponding author on reasonable request.

References

[1] Bishop, C.M. (2006). *Pattern recognition and machine learning*.

[2] Briffoteaux, G., Gobert, M., Ragonnet, R., Gmys, J., Mezmaz, M.-S., Melab, N., Tuyttens, D. (2020). Parallel surrogate-assisted optimization: Batched Bayesian neural network-assisted GA versus q-EGO. *Swarm Evol. Comput.*, *57* , 100717.

[3] Cai, X., Qiu, H., Gao, L., Yang, P., Shao, X. (2017). A multi-point sampling method based on kriging for global optimization. *Structural and Multidisciplinary Optimization*, *56* (1), 71–88.

[4] Chen, R.-B., Hsu, Y.-W., Hung, Y., Wang, W. (2014). Discrete particle swarm optimization for constructing uniform design on irregular regions. *Computational Statistics and Data Analysis*, *72* , 282–297.

[5] Cheng, J., Jiang, P., Zhou, Q., Hu, J., Shu, L. (2021). A parallel constrained lower confidence bounding approach for computationally expensive constrained optimization problems. *Appl. Soft Comput.*, *106* , 107276.

[6] Dai, R., & Chen, W. (2026). Research on stock price forecasting models based on lstm and Bayesian optimization algorithms. *Proceedings of the 2025 5th international conference on big data, artificial intelligence and risk management* (p. 853–860). New York, NY, USA: Association for Computing Machinery.

[7] Desautels, T., Krause, A., Burdick, J.W. (2012). Parallelizing exploration-exploitation tradeoffs with gaussian process bandit optimization. *J. Mach. Learn. Res.*, *15* , 3873-3923.

[8] de Villiers, D.I.L., Couckuyt, I., Dhaene, T. (2017). Multi-objective optimization of reflector antennas using kriging and probability of improvement. *2017 IEEE International Symposium on Antennas and Propagation & USNC/URSI National Radio Science*

Meeting , 985-986.

[9] Fang, K.T., Lin, D.K., Winker, P., Zhang, Y. (2000). Uniform design: Theory and application. *Technometrics*, *42* (3), 237–248.

[10] Fang, K.-T., Liu, M.-Q., Qin, H., Zhou, Y.-D. (2018). *Theory and application of uniform experimental designs*.

[11] Ginsbourger, D., Riche, R.L., Carraro, L. (2010). Kriging is well-suited to parallelize optimization. (pp. 131–162).

[12] Goldberg, D.E. (1988). *Genetic algorithms in search, optimization, and machine learning*.

[13] Gonz´alez, J.I., Dai, Z., Hennig, P., Lawrence, N.D. (2016). Batch Bayesian optimization via local penalization. *Aistats*.

[14] Jones, D.R., Schonlau, M., Welch, W.J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, *13* (4), 455–492.

[15] Kushner, H.J. (1963). A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering* , *86* , 97-106.

[16] Li, M., Yi, B., Yang, Y. (2021). An efficient global optimization method with multi-point infill sampling based on kriging. *Engineering Optimization*, *0* (0), 1-18.

[17] Li, Y., Kang, L., Hickernell, F.J. (2020). Is a transformed low discrepancy design also low discrepancy? In: Fan, J., Pan, J. (eds) *Contemporary Experimental Design, Multivariate Analysis and Data Mining*. Springer, Cham. https://doi.org/10.1007/978-3-030-46161-4_5

[18] Liu, J., Jiang, C., Zheng, J. (2021). Batch Bayesian optimization via adaptive local search. *Appl. Intell.*, *51* , 1280-1295.

[19] Nguyen, V., Rana, S., Gupta, S., Li, C., Venkatesh, S. (2017). Budgeted batch Bayesian optimization with unknown batch sizes. *ArXiv* , *abs/1703.04842* .

[20] Xiao, Y., Ning, J., Xiong, Z. et al. (2022). Batch sequential adaptive designs for global optimization. *J. Korean Stat. Soc.* 51, 780–802.

[21] Rasmussen, C.E., & Williams, C.K.I. (2009). Gaussian processes for machine learning.

Adaptive computation and machine learning.

[22] Srinivas, N., Krause, A., Kakade, S.M., Seeger, M.W. (2010). Gaussian process optimization in the bandit setting: No regret and experimental design. *Icml*.

[23] Sun, G., Li, L., Fang, J., Li, Q. (2021). On lower confidence bound improvement matrix-based approaches for multi-objective Bayesian optimization and its applications to thin-walled structures. *Thin-walled Structures*, *161* , 107248.

[24] Tang, C., Zhang, F., Zhang, J., Lv, Y., Wang, G. (2022). Novel reliability evaluation method combining active learning kriging and adaptive weighted importance sampling. *Structural and Multidisciplinary Optimization*.

[25] Tao, T., Zhao, G., Ren, S. (2019, 11). An Efficient Kriging-Based Constrained Optimization Algorithm by Global and Local Sampling in Feasible Region. *Journal of Mechanical Design*, *142* (5), 1-48.

[26] Victoria, A.H., & Maragatham, G. (2021). Automatic tuning of hyperparameters using Bayesian optimization. *Evolving Systems*, *12* , 217-223.

[27] Wang, Z., Dahl, G.E., Swersky, K., Lee, C., Nado, Z., Gilmer, J., . . . Ghahramani, Z. (2024). Pre-trained gaussian processes for Bayesian optimization. *Journal of Machine Learning Research*, *25* (1), 83.

[28] Yu, R., Picard, C., Ahmed, F. (2025). Fast and accurate Bayesian optimization with pre-trained transformers for constrained engineering problems: Fast and accurate Bayesian optimization with pre-trained transformers. R. yu et al. *Structural & Multidisciplinary Optimization*, *68* (3).

[29] Zhan, D., Qian, J., Cheng, Y. (2017). Pseudo expected improvement criterion for parallel EGO algorithm. *Journal of Global Optimization*, *68* , 641-662.

[30] Zhan, D., & Xing, H. (2020). Expected improvement for expensive optimization: a review. *Journal of Global Optimization*, 1-38.

[31] Zhou, Y.-D., & Fang, K.-T. (2013). An efficient method for constructing uniform designs with large size. *Computational Statistics*, *28* (3), 1319– 1331.