Global Optimization of Gaussian Process Acquisition Functions Using a Piecewise-Linear Kernel Approximation

Yilin Xie Imperial College London
Shiqiang Zhang Imperial College London
Joel A. Paulson The Ohio State University
Calvin Tsay Imperial College London

Abstract

Bayesian optimization relies on iteratively constructing and optimizing an acquisition function. The latter turns out to be a challenging, non-convex optimization problem itself. Despite the relative importance of this step, most algorithms employ sampling- or gradient-based methods, which do not provably converge to global optima. This work investigates mixed-integer programming (MIP) as a paradigm for *global* acquisition function optimization. Specifically, our Piecewise-linear Kernel Mixed Integer Quadratic Programming (PK-MIQP) formulation introduces a piecewise-linear approximation for Gaussian process kernels and admits a corresponding MIQP representation for acquisition functions. The proposed method is applicable to uncertainty-based acquisition functions for any stationary or dot-product kernel. We analyze the theoretical regret bounds of the proposed approximation, and empirically demonstrate the framework on synthetic functions, constrained benchmarks, and a hyperparameter tuning task.

1 INTRODUCTION

Optimization of black-box functions is a compelling task in many scientific fields, such as hyperparameter tuning for machine learning (Ranjit et al., 2019), routing for robot control problems (Nambiar et al., 2022), designing energy systems (Thebelt et al., 2022b), and drug discovery (Colliandre and Muller, 2024). These tasks share challenges such as limited knowledge about the true underlying objective function and expensive evaluations on the target function, i.e., experiments. Given the latter, finding a high-quality sample point at each evaluation becomes extremely valuable.

Bayesian optimization (BO) is a popular class of algorithms designed for this setting. BO can be divided into two main components: (1) a Bayesian model of the objective function (usually a Gaussian process (GP)), and (2) an acquisition function to decide which point $\boldsymbol{x}^*$ to sample next. An acquisition function is a mathematical expression that quantifies the value of evaluating a particular point in the search space. The choice of acquisition function balances exploration and exploitation (Paulson and Tsay, 2024). Mathematical optimization of the acquisition function is thus a key step of BO. While many works formulate more indicative and sophisticated acquisition functions, comparatively less attention has been given to computing the optimal solutions of acquisition functions. While a well-formulated acquisition function is critical, its reliable optimization can be just as beneficial for BO performance (Wilson et al., 2018; Kim and Choi, 2021).

Gradient- and sampling-based methods remain the mainstream for acquisition function optimization. While the objective function of BO is a black box, closed-form expressions are available for many acquisition functions, e.g., Upper Confidence Bound (UCB) (Srinivas et al., 2010), Expected Improvement (EI) (Jones et al., 1998). Therefore, mathematical programming, especially (stochastic) gradient-based methods such as L-BFGS-B (Zhu et al., 1997) can perform well given first- and sometimes second-order derivatives. However, derivative information can be unreliable, e.g., due to numerical issues (Ament et al., 2024). Gradient-based methods can also be trapped at local optima and thus return sub-optimal solutions. On the other hand, *sampling* methods are invariably limited by the curse of dimensionality and are also likely to return sub-optimal solutions, given limited evaluation budgets. Algorithms that provide a *global* optimality guarantee for solving acquisition functions are thus lacking.

An alternative class of *deterministic* methods is mixed-

Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258.

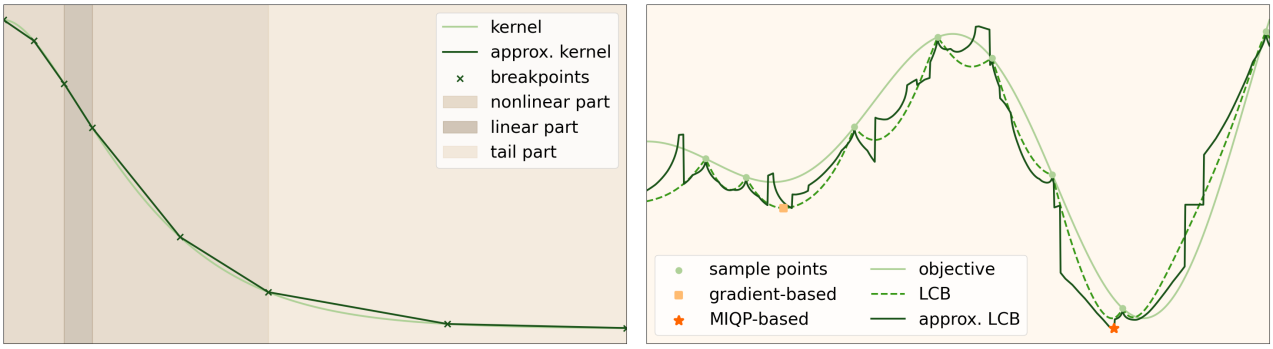


Figure 1: (**left**) Illustration of piecewise linear approximation of kernel function. (**right**) Visualization of the effect of kernel approximation on LCB acquisition function. The solution from gradient-based method (orange square) may end up at a local minimum, a sampling-based solution can miss the global minimum, and optimizing approximated LCB using global model (red star) will provide the global solution.

integer programming (MIP) (Belotti et al., 2013), where established algorithms can globally optimize an objective function subject to various constraints. Conditions of a given problem are expressed as linear, quadratic, nonlinear and/or integral constraints and are passed to modern solvers that utilize branch-and-bound (B&B) algorithms. By bounding the objective value, B&B algorithms provide some guarantees on the global optimality of the solution. To the best of our knowledge, the application of MIP in BO has only been studied by Schweidtmann et al. (2021). This work further explores MIP-based global optimization for BO, specifically with novel kernel approximation techniques tailored to the solution methods.

In this paper, we propose Piecewise-linear Kernel Mixed Integer Quadratic Programming (PK-MIQP), a global optimization framework for GP acquisition functions. Specifically, we introduce a piecewise linear approximation of the GP kernel function (Figure 1, left) that enables a mixed-integer quadratic programming (MIQP) formulation of the acquisition function. Note that the model becomes mixed-integer linear if only the GP mean is required. Lower confidence bound (LCB) is chosen as a representative example to demonstrate the proposed framework. We then use a B&B solver to globally optimize the approximated acquisition function. While gradient-based methods can return sub-optimal solutions without further indication (Figure 1, right), our method approximates the global optimum within a bounded neighborhood. The main contributions of this work are as follows:

1. We propose a MIP-motivated piecewise linear approximation for GP kernel functions in stationary or dot-product form.
2. We present an MIQP formulation to solve the resulting acquisition function optimization problems with global optimality guarantees.
3. We theoretically analyze the performance and worst-case error of PK-MIQP.
4. We embed PK-MIQP in a full BO procedure and evaluate its performance against state-of-the-art optimization methods on tasks including synthetic functions, constrained benchmarks, and a hyperparameter tuning task.

2 RELATED WORK

Recent research in the field of BO often focuses on solving long-standing problems, such as scaling BO to perform on problems with higher dimensionalities (Spagnol et al., 2019; Cartis et al., 2023; Eriksson and Jankowiak, 2021) and improving computational efficiency through more informative acquisition functions (Oh et al., 2018; De Ath et al., 2021; Ament et al., 2024). While these innovations can be important and effective, the acquisition functions in these works remain optimized using classic and traditional methods.

Gradient-based methods such as L-BFGS-B (Zhu et al., 1997) and stochastic gradient ascent (Kingma and Ba, 2015) are popular choices to optimize acquisition functions in BO. For example, Eriksson and Jankowiak (2021) optimize the EI acquisition function based on a sparse axis-aligned subspace GP using L-BFGS-B (Zhu et al., 1997). Oh et al. (2018) propose a cylindrical transformation of the search space to allow the chosen acquisition function to explore more near the center of search space using the Adam algorithm (Kingma and Ba, 2015). Though widely-used, solving acquisition functions using gradient-based methods has its limitations. Ament et al. (2024) highlight the issue of vanishing values and gradients when applying gradient-based methods to acquisition func-

tion optimization. Specifically, improvement-based acquisition functions (e.g., EI) can suffer from numerically zero acquisition values and gradients, making gradient-based methods return sub-optimal solutions due to lack of gradient information. Ament et al. (2024) thus propose a reformulation of the EI acquisition family, termed LogEI, where functions retain the same optima, but more stable in terms of gradient values; the authors demonstrated performance in BO using L-BFGS-B to solve the LogEI acquisition functions. Daulton et al. (2022) also address the limitations of gradient-based methods on maximizing acquisition functions in discrete and mixed search space. The authors propose a novel probabilistic representation of the acquisition function, where continuity of parameters in the acquisition function is restored, enabling the use of the gradient-based method Adam. Popular BO tools, e.g., BoTorch (Balandat et al., 2020), often consider multi-start gradient approaches to overcome local optima. However, these approaches may have difficulty in determining the non-flat regions of acquisition functions, especially in higher dimensional settings (Rana et al., 2017; Adebiyi et al., 2024).

As an alternative direction, sampling methods such as Nelder-Mead (Nelder and Mead, 1965) are also a popular tool for acquisition function optimization in BO methods. While gradient-based methods suffer from difficulties in getting reliable gradient information, sampling methods mainly suffer from the curse of dimensionality. Here, Kandasamy et al. (2015) propose an additive structure to model the objective function to exploit the efficiency of sampling methods in low-dimensional problems. Eriksson et al. (2019) employ Thompson sampling in local trust regions and search for global optima through multiple local search.

Global optimality in BO has primarily been discussed in terms of the black-box objective function, rather than the acquisition function evaluated at each iteration. Towards the latter, some researchers heuristically search for global optimality through multi-start local methods (Eriksson et al., 2019; Mathesen et al., 2021) or evolutionary algorithms such as Firefly (Song et al., 2024), CMA-ES (Hansen, 2006; Wilson et al., 2018) and TSEMO (Bradford et al., 2018). MIP comprises a well-known paradigm to solve optimization problems globally, but to-date has been rarely used in BO. Some works apply MIP to solve acquisition functions based on more compatible surrogate models, e.g., neural networks (Papalexopoulos et al., 2022) or decision trees (Thebelt et al., 2022b). MIP is also used to handle discrete decisions in hybrid/combinatorial BO (Baptista and Poloczek, 2018; Daxberger et al., 2020; Deshwal et al., 2021). Relatively few works apply MIP with GP-based surrogate models in BO. Thebelt et al. (2022a) formulate GPs using tree kernels and solve the resulting acquisition function as an MIQP. Schweidtmann et al. (2021) formulate acquisition functions for smooth-kernel GPs as a mixed-integer nonlinear program (MINLP), which is then solved using a B&B algorithm. However, MINLP can easily exceed computational budgets in real applications. This work seeks a more stable and practical algorithm for standard GPs that combines some global optimality guarantees, with more realistic computing times.

3 BACKGROUND

3.1 Gaussian Processes

A Gaussian process (GP) models a joint multivariate Gaussian distribution over some random variables (Schulz et al., 2018). A GP is fully specified by a prior mean function $\mu(\cdot)$ and a kernel $K(\cdot,\cdot)$:

$$f(\cdot) \sim \mathcal{GP}(\mu(\cdot), K(\cdot,\cdot))$$

Normally the prior mean is set to zero to simplify the subsequent posterior computation, leaving only the kernel function. Common choices here are the squared exponential (SE) kernel or *Matérn* kernel.

GP regression can be viewed as a Bayesian statistical approach for modelling and predicting functions. Given t observed data points $\{(\boldsymbol{x_i}, f(x_i))\}_{i=1}^{t}$, we denote $\boldsymbol{X} = [\boldsymbol{x_1}, \ldots, \boldsymbol{x_t}]$, $\boldsymbol{y} = [f(\boldsymbol{x_1}), \ldots, f(\boldsymbol{x_t})]$. For a new point $\boldsymbol{x} \in \mathbb{R}^n$, the posterior mean and variance of its function value is given by Frazier (2018):

$$\begin{aligned} \mu_t(\boldsymbol{x}) &= K_{xX} K_{XX}^{-1} \boldsymbol{y} \\ \sigma_t^2(\boldsymbol{x}) &= K_{xx} - K_{xX} K_{XX}^{-1} K_{Xx} \end{aligned} \tag{1}$$

where we omit the conditioning on $(\boldsymbol{X}, \boldsymbol{y})$ and use $K_{XY} = K(\boldsymbol{X}, \boldsymbol{Y})$ for simplicity. Note that K_{xx} is the kernel variance σ_f^2.

When the observed output is noisy, i.e., $y = f(\boldsymbol{x}) + \epsilon$, we assume $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$, giving a posterior mean and variance of:

$$\begin{aligned} \mu_t(\boldsymbol{x}) &= K_{xX}(K_{XX} + \sigma_\epsilon^2 \boldsymbol{I})^{-1} \boldsymbol{y} \\ \sigma_t^2(\boldsymbol{x}) &= K_{xx} - K_{xX}(K_{XX} + \sigma_\epsilon^2 \boldsymbol{I})^{-1} K_{Xx} \end{aligned} \tag{2}$$

Nevertheless, for simplicity, we omit the noise term and assume that $\|K_{XX}\|_2 \geq \sigma_\epsilon^2$ in our proofs.

3.2 Bayesian Optimization

A general BO procedure constructs an acquisition function (AF) based on the above GP and seeks to maximize its value. We consider the classic lower confidence bound (LCB) acquisition function:

$$\alpha_{LCB}(\boldsymbol{x}) = \mu_t(\boldsymbol{x}) - \beta_t^{1/2} \sigma_t(\boldsymbol{x})$$

Here μ_t and σ_t are the posterior mean and variance from surrogate model, and β_t is a hyperparameter that controls exploration and exploitation of the acquisition function. The next sample point proposed by acquisition function is then evaluated on the objective function, adding to the sample data set for next round of surrogate modelling and acquisition evaluation. Gaussian process regression and evaluation on acquisition function are performed repeatedly during the optimization process until a pre-defined budget on number of iterations is achieved.

3.3 Mixed-Integer Quadratic Programming

Mixed-integer quadratic programming (MIQP) considers problems with quadratic constraints/objectives:

$$\begin{aligned}
\min_{\boldsymbol{x}\in\mathbb{R}^D}\ & \boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x}+\boldsymbol{q}^T\boldsymbol{x} && \text{(objective)}\\
s.t.\ & \boldsymbol{A}\boldsymbol{x}\le\boldsymbol{b} && \text{(linear)}\\
& \boldsymbol{x}^T\boldsymbol{Q_j}\boldsymbol{x}+\boldsymbol{q_j}^T\boldsymbol{x}\le b_j,\ \forall j\in\mathcal{J} && \text{(quadratic)}\\
& x_i\in\mathbb{Z},\ \forall i\in\mathcal{I}\subset\{1,\dots,D\} && \text{(integrality)}\\
& \boldsymbol{x}_L\le\boldsymbol{x}\le\boldsymbol{x}_U && \text{(bounds)}
\end{aligned}$$

Section 4.2 presents our MIQP formulation for LCB with approximated posterior mean and variance. Several commercial solvers for MIQP are available; we use Gurobi v11.0.0 (Gurobi Optimization, LLC, 2024).

4 METHODOLOGY

4.1 Piecewise linearization of kernel

This section introduces our proposed piecewise-linear approximation for stationary or dot-product kernel functions. Consider a kernel function $k(r):\mathbb{R}_0^+\to\mathbb{R}^+$, where r is some distance measure or dot-product between data points $\boldsymbol{x},\boldsymbol{x'}\in\mathbb{R}^D$. Intuitively, fewer linear segments are needed for approximately-linear parts of the function. We therefore select piecewise-linear breakpoints based on the curvature of $k(\cdot)$, which is proportional to its second derivative. Specifically, we first set a threshold ϵ_k to define "near-linear" parts, i.e., segments of the kernel with $|k''(r)|\le\epsilon_k$.

The threshold value can be selected to balance the trade-off between the number of piecewise-linear segments and the approximation accuracy. We empirically choose ϵ_k to be half of the maximal value of $k''(r)$, which bounds the approximation error e_{approx} to relatively small values. Future work could investigate a more systematic derivation of this threshold value. Figure 2 visualizes our piecewise linearization strategy using the Matérn 3/2 kernel as an example.

For stationary kernels enjoying similar shape to the Matérn 3/2 kernel function, e.g., RBF, Matérn 5/2, the kernel is partitioned into three parts:

$$\underbrace{[r_0,r_1]\cup[r_2,r_3]}_{R_{\text{nonlinear}}}\cup\underbrace{[r_1,r_2]}_{R_{\text{linear}}}\cup\underbrace{[r_3,r_4]}_{R_{\text{tail}}}$$

where r_1,r_2,r_3 are derived by solving $|k''(r)|=\epsilon_k$, and r_0,r_4 denote the minimum and maximum values of r.

Based on the above, we use D segments in the domain of R_{linear} and $2D$ segments for both $R_{\text{nonlinear}}$ and R_{tail}. We use $2D$ segments in R_{tail} due to its relatively large range. The final set of breakpoints is:

$$\begin{aligned}
R &= S^{2D}_{r_0,r_1}\cup S^{D}_{r_1,r_2}\cup S^{2D}_{r_2,r_3}\cup S^{2D}_{r_3,r_4}\cup\{r_4\}\\
&=\{R_0,\cdots,R_M\mid R_i<R_j,\ \forall i<j\}
\end{aligned}$$

where $S^n_{l,r}$ denotes the set of n points evenly spaced over interval $[l,r)$, and M is the number of segments.

Using the approximated kernel, denoted by $\tilde{k}(\cdot)$, we can now define the approximated posterior mean $\tilde{\mu}$ and variance $\tilde{\sigma}^2$ analogously to (1):

$$\begin{aligned}
\tilde{\mu}(\boldsymbol{x}) &:= \tilde{K}_{xX}\tilde{K}_{XX}^{-1}\boldsymbol{y}\\
\tilde{\sigma}^2(\boldsymbol{x}) &:= \tilde{K}_{xx}-\tilde{K}_{xX}\tilde{K}_{XX}^{-1}\tilde{K}_{Xx}
\end{aligned}\tag{3}$$

which then give us the approximated LCB:

$$\tilde{\alpha}_{LCB}(\boldsymbol{x})=\tilde{\mu}(\boldsymbol{x})-\beta_t^{1/2}\tilde{\sigma}(\boldsymbol{x})\tag{4}$$

where by convention we take $\beta_t=0.2D\log 2t$ for its convergence properties (Kandasamy et al., 2015).

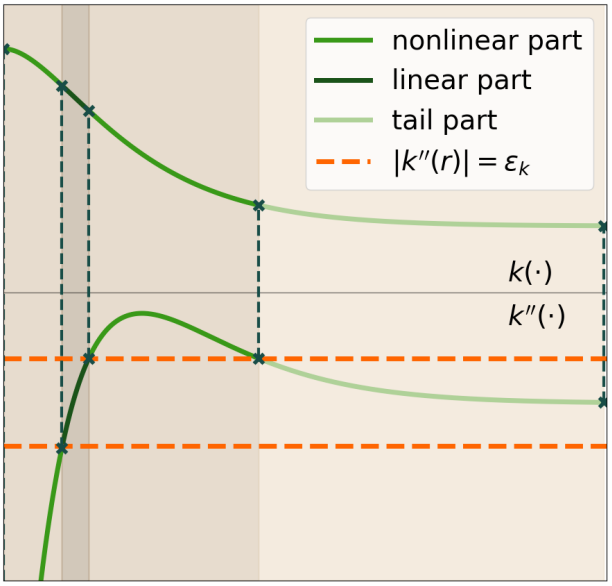


Figure 2: (**top**) Matérn 3/2 kernel function divided into 3 parts. (**bottom**) The second-order derivative of Matérn 3/2 kernel function. Parts within threshold are considered as "near-linear."

Table 1: Piecewise linearization parameter values for Matérn 3/2 and RBF kernels.

Variables	Matérn 3/2	RBF
ϵ_k	$\frac{3}{2}\exp(-2)\sigma_f^2$	$\exp(-\frac{3}{2})\sigma_f^2$
e_{approx}	$0.025\sigma_f^2$	$0.022\sigma_f^2$
r_1	0.4866	0.8280
r_2	0.7113	1.2099
r_3	2.1237	2.5213

We take two classic stationary kernels, Matérn 3/2 and RBF kernel, as examples. Their kernel functions are:

$$k_{Matérn\ 3/2}(r) = \sigma_f^2(1+\sqrt{3}r)\exp(-\sqrt{3}r)$$
$$k_{RBF}(r) = \sigma_f^2\exp(-\frac{1}{2}r^2)$$

where $r = \frac{\|\boldsymbol{x}-\boldsymbol{x'}\|_2}{l}$ and l is the lengthscale.

The values of parameters involved in their piecewise-linear approximations are presented in Table 1.

4.2 Mixed-integer Optimization Formulation

Given this piecewise linear kernel approximation, we now formulate the minimization of the approximated LCB (4) as an MIQP:

$$\min_{x\in\mathcal{X}} \tilde{\mu} - \beta_t^{1/2}\tilde{\sigma} \tag{5a}$$
$$s.t.\ \tilde{\mu} = \tilde{K}_{xX}\tilde{K}_{XX}^{-1}\boldsymbol{y} \tag{5b}$$
$$\tilde{\sigma}^2 \le \tilde{\sigma}_f^2 - \tilde{K}_{xX}\tilde{K}_{XX}^{-1}\tilde{K}_{Xx} \tag{5c}$$
$$r_i^2 = \frac{\|\boldsymbol{x}-\boldsymbol{x_i}\|_2^2}{l^2},\ \forall 1\le i\le N \tag{5d}$$
$$\tilde{K}_{xX_i} = \tilde{k}(r_i),\ \forall 1\le i\le N \tag{5e}$$

Note $\beta_t^{1/2}, \tilde{K}_{XX}^{-1}, \boldsymbol{y}, l$ are independent of $\boldsymbol{x}$ and their values are precomputed. Constraints (5b) and (5c) follow from (3), where the inequality comes from quadratic constraint relaxation. The scaled Euclidean distance is given by constraint (5d).

Constraint (5e) involves the piecewise linearization of $k(\cdot)$, whose encoding is well-studied in MIP literature (Beale and Tomlin, 1969; Forrest et al., 1974) and is provided in modern MIP solvers such as Gurobi (Gurobi Optimization, LLC, 2024). Here we present the classic encoding using our notations in Eq. (6). First, as shown in Eqs. (6a)–(6c), point $(r_i, \tilde{k}(r_i))$ is expressed as a convex combination of points $\{(R_j, \tilde{k}(R_j))\}_{0\le j\le M}$ with $\{w_i^j\}_{0\le j\le M}$ as the nonnegative and sum-to-one coefficients. Then, as given in Eqs. (6d)–(6g), an indicator variable λ_i^j is introduced to model that point $(r_i, \tilde{k}(r_i))$ lies in j-th linear segment between $(R_{j-1}, \tilde{k}(R_{j-1}))$ and $(R_j, \tilde{k}(R_j))$, i.e., only w_i^{j-1} and w_i^j can be nonzero if $\lambda_i^j = 1$.

$$r_i = \sum_{j=0}^{M} w_i^j R_j \tag{6a}$$
$$\tilde{K}_{xX_i} = \sum_{j=0}^{M} w_i^j \cdot \tilde{k}(R_j) \tag{6b}$$
$$\sum_{j=0}^{M} w_i^j = 1,\ w_i^j \ge 0,\ \forall 0\le j\le M \tag{6c}$$
$$\sum_{j=1}^{M} \lambda_i^j = 1,\ \lambda_i^j \in \{0,1\},\ \forall 1\le j\le M \tag{6d}$$
$$w_i^0 \le \lambda_i^1 \tag{6e}$$
$$w_i^j \le \lambda_i^j + \lambda_i^{j+1},\ \forall 1\le j< M \tag{6f}$$
$$w_i^M \le \lambda_i^M \tag{6g}$$

Initialization Heuristic. Note that finding a feasible initial point (upper bound) for (5) is simple: we need only select a point $\boldsymbol{x}$ and evaluate (3)–(4). We propose a heuristic to find initial points based on minimizing the posterior mean. Specifically, before solving (5), PK-MIQP first minimizes a sub-problem with $\tilde{\mu}$ as the objective and (5b), (5d)–(5e) as constraints. Without the quadratic constraint (5c), the sub-problem can be solved relatively quickly, producing an initial solution pool P_{sub} containing solutions with lowest mean. The full-problem (5) is then solved using the best solution among P_{sub} and several randomly sampled points P_{rand} as a good incumbent solution. While the approximation error is theoretically analyzed in the following section, in practice we employ two steps to polish the solution. First, PK-MIQP selects the best solution found while solving (5) with lowest $\alpha(\cdot)$, i.e., true LCB value. Then we apply a few steps of gradient descent to ensure we are exactly at a local optimum. The final solution $\boldsymbol{x_t}$ obtained is a minimum with global optimality guarantees (see Section 4.3). PK-MIQP can seamlessly handle *constrained* optimization problems by adding known constraints to the formulation (5). PK-MIQP also adapts to any acquisition function that can be linearly (or quadratically) represented, e.g., GLCB (Rodemann and Augustin, 2024), by replacing the objective function in (5a) accordingly. One iteration of PK-MIQP is outlined in Algorithm 1.

As PK-MIQP is a generic optimization framework, computational cost can be reduced using existing strategies, e.g., additive GP (add-GP) training (Duvenaud et al., 2011; Kandasamy et al., 2015). The basic idea of add-GP is to decompose black-box function f

Algorithm 1 PK-MIQP at t-th iteration

Input: sample points $\mathcal{D}_{t-1} = \{(\boldsymbol{x_i}, y_i)\}_{i=1}^{t-1}$, β_t.
Kernel approximation:
kernel parameters σ_f^2, $l \leftarrow$ GP fit to $\mathcal{D}_{t-1}$.
$\tilde{k}(\cdot) \leftarrow$ piecewise linearization.
Warm start (optional):
$P_{\text{sub}} \leftarrow$ solve sub-problem of (5).
$P_{\text{rand}} \leftarrow$ random feasible solutions of (5).
Acquisition optimization:
$P_{\text{full}} \leftarrow$ solve (5) (incumbent from $P_{\text{sub}} \cup P_{\text{rand}}$).
Solution polishing:
$\boldsymbol{x_t^0} = \arg\min_{\boldsymbol{x} \in P_{\text{full}}} \alpha(\boldsymbol{x}) \leftarrow$ solution with lowest LCB.
$\boldsymbol{x_t} \leftarrow$ correction starting at $\boldsymbol{x_t^0}$.
Output: next sample $\boldsymbol{x_t}$.

among N_g disjoint sets of dimensions:

$$f(\boldsymbol{x}) = \sum_{i=1}^{N_g} f^{(i)}(\boldsymbol{x}^{(i)})$$

where each set of dimensions has a kernel that only acts on the included dimensions. Since the sets of dimensions are independent, they can be optimized separately, e.g., applying PK-MIQP N_g times in parallel.

4.3 Theoretical Analysis

Given the approximation and methodology above, we now aim to establish a bound on regret $r_t := f(\boldsymbol{x_t}) - f(\boldsymbol{x^*})$ for using a GP with the proposed approximated kernel at the t^{th} iteration of a full BO loop. Here f is the black-box objective, $\boldsymbol{x^*}$ denotes the (oracle) optimum point that minimizes f, and $\boldsymbol{x_t}$ is the chosen point to evaluate at iteration t. We begin with two remarks on the proposed approximated kernel:

Remark 4.1. The difference between the true and approximated kernel functions is bounded by error ϵ_M, which is a function of the number of linear pieces M.

Remark 4.2. Error ϵ_M asymptotically converges to 0 as the number of linear pieces increases:

$$\lim_{M \to \infty} \epsilon_M = 0$$

With the above remarks, Theorem 4.3 shows that the approximated kernel mean (and variance) converges to the true mean (and variance) as $M \to \infty$.

Theorem 4.3. *Given N observed data points $\boldsymbol{X}$ with outputs $\boldsymbol{y}$, for any $\boldsymbol{x} \in \mathcal{D}$, we have:*

$$|\mu(\boldsymbol{x}) - \tilde{\mu}(\boldsymbol{x})| \leq C_\mu N^2 \epsilon_M, \ |\sigma(\boldsymbol{x}) - \tilde{\sigma}(\boldsymbol{x})| \leq C_\sigma N \epsilon_M^{1/2}$$

Proof (Sketch). Denote $\psi := K_{xX} - \tilde{K}_{xX}$ and $\Psi := K_{XX} - \tilde{K}_{XX}$. By the definition of $\mu(\boldsymbol{x})$ and $\tilde{\mu}(\boldsymbol{x})$,

$$\mu(\boldsymbol{x}) - \tilde{\mu}(\boldsymbol{x}) = -\tilde{K}_{xX} K_{XX}^{-1} \Psi \tilde{K}_{XX}^{-1} \boldsymbol{y} + \psi K_{XX}^{-1} \boldsymbol{y}$$

Since $\|\psi\|_2 \leq \sqrt{N}\epsilon_M$, $\|\Psi\|_2 \leq N\epsilon_M$, $\|\tilde{K}_{xX}\|_2 \leq \sqrt{N}\sigma_f^2$, and $\|\boldsymbol{y}\|_2 \leq \sqrt{N}$ (In our implementation, we scaled the objective $f(x) \in [0, 1]$), we have:

$$|\mu(\boldsymbol{x}) - \tilde{\mu}(\boldsymbol{x})| \leq C_\mu N^2 \epsilon_M$$

where the constant $C_\mu = \sigma_\epsilon^{-4} \sigma_{\max}^2$ and $\sigma_{\max}^2$ is the upper bound of kernel variance σ_f^2.

Similarly, the difference of variances is:

$$\begin{aligned}\sigma^2(\boldsymbol{x}) - \tilde{\sigma}^2(\boldsymbol{x}) =& \tilde{K}_{xX} K_{XX}^{-1} \Psi \tilde{K}_{XX}^{-1} \tilde{K}_{Xx} \\ & - 2\tilde{K}_{xX} K_{XX}^{-1} \psi^T - \psi K_{XX}^{-1} \psi^T\end{aligned}$$

Then we have:

$$|\sigma^2(\boldsymbol{x}) - \tilde{\sigma}^2(\boldsymbol{x})| \leq C_\sigma^2 N^2 \epsilon_M$$

where the constant $C_\sigma = \sigma_\epsilon^{-2} \sigma_{\max}^2$. Since variances are non-negative, we can derive:

$$|\sigma(\boldsymbol{x}) - \tilde{\sigma}(\boldsymbol{x})| \leq C_\sigma N \epsilon_M^{1/2}$$

which completes the proof. A more comprehensive proof is provided in Appendix. □

Theorem 4.3 holds for any continuous kernel with a proper piecewise linear approximation. With additional smoothness assumptions on the kernel, we can derive similar regret bounds as Srinivas et al. (2012). Following the same settings as Lemma 5.8 in Srinivas et al. (2012), Theorem 4.4 bounds the regret $r_t = f(\boldsymbol{x_t}) - f(\boldsymbol{x}^*)$, where $\boldsymbol{x_t}$ is the optimal solution of MIQP (5) at t-th iteration. Since we may need more pieces in our approximation, denote M_t as the number of linear pieces at t-th iteration.

Theorem 4.4. *Let $D \subset [0, r]^d$ be compact and convex, $d \in \mathbb{N}, r > 0$. Suppose kernel $K(\boldsymbol{x}, \boldsymbol{x'})$ satisfies the following high probability bound on the derivatives of GP sample paths f: for some constants $a, b > 0$:*

$$Pr\left\{\sup_{\boldsymbol{x} \in D} |\partial f / \partial x_j| > L\right\} \leq a e^{-(L/b)^2}, \ j = 1, 2, \ldots, d$$

Select $\delta \in (0, 1)$, and define $\beta_t = 2\log(2t^2\pi^2/(3\delta)) + 2d\log(t^2 dbr\sqrt{\log(4da/\delta)})$. Then the following regret bounds

$$r_t \leq 2\beta_t^{1/2} \sigma_{t-1}(\boldsymbol{x_t}) + 1/t^2 + 2C_\mu t^2 \epsilon_{M_t} + 4C_\sigma \beta_t^{1/2} t \epsilon_{M_t}^{1/2}$$

hold with probability $\geq 1 - \delta$.

Proof (Sketch). Lemma 5.5 and Lemma 5.7 in Srinivas et al. (2012) give us:

$$|f(\boldsymbol{x_t}) - \mu_{t-1}(\boldsymbol{x_t})| \leq \beta_t^{1/2} \sigma_{t-1}(\boldsymbol{x_t})$$
$$|f(\boldsymbol{x^*}) - \mu_{t-1}([\boldsymbol{x^*}]_t)| \leq \beta_{t-1}([\boldsymbol{x^*}]_t) + 1/t^2$$

where $[\boldsymbol{x}^*]_{\boldsymbol{t}}$ is the closest point in D_t to $\boldsymbol{x}^*$, and $D_t \subset D$ is a discretization at t-th iteration.

Recall the definition of $\boldsymbol{x_t}$, we have:

$$\tilde{\mu}_{t-1}(\boldsymbol{x_t}) - \beta_t^{1/2}\tilde{\sigma}_{t-1}(\boldsymbol{x_t}) \leq \tilde{\mu}_{t-1}([\boldsymbol{x}^*]_{\boldsymbol{t}}) - \beta_t^{1/2}\tilde{\sigma}_{t-1}([\boldsymbol{x}^*]_{\boldsymbol{t}})$$

Using Theorem 4.3 to replace μ, σ with $\tilde{\mu}$, $\tilde{\sigma}$, we have:

$$\begin{aligned} r_t &= f(\boldsymbol{x_t}) - f(\boldsymbol{x}^*) \\ &\leq \tilde{\mu}_{t-1}(\boldsymbol{x_t}) - \tilde{\mu}_{t-1}([\boldsymbol{x}^*]_{\boldsymbol{t}}) + \beta_t^{1/2}\tilde{\sigma}_{t-1}([\boldsymbol{x}^*]_{\boldsymbol{t}}) \\ &\quad + \beta_t^{1/2}\tilde{\sigma}_{t-1}(\boldsymbol{x_t}) + 1/t^2 + 2C_\mu t^2\epsilon_M + 2C_\sigma\beta_t^{1/2}t\epsilon_M^{1/2} \\ &\leq 2\beta_t^{1/2}\tilde{\sigma}_{t-1}(\boldsymbol{x_t}) + 1/t^2 + 2C_\mu t^2\epsilon_{M_t} + 2C_\sigma\beta_t^{1/2}t\epsilon_{M_t}^{1/2} \end{aligned}$$

Applying Theorem 4.3 again completes this proof. □

See Appendix A for full proofs of Theorems 4.3 and 4.4. Lemma 4.5 is a direct conclusion from Theorem 4.4, which implies that the same regret bounds as Srinivas et al. (2012) hold when using PK-MIQP to optimize approximated LCB.

Lemma 4.5. *For $\epsilon_{M_t} = 1/\mathcal{O}(t^{4+\epsilon})$, $\forall t \geq 1$ with some $\epsilon > 0$, running PK-MIQP for a sample f from a GP with zero mean and covariance $K(\boldsymbol{x}, \boldsymbol{x'})$, we can bound regret by $\mathcal{O}^*(\sqrt{dT\gamma_T})$ with high probability. Precisely, pick $\delta \in (0,1)$, with $C_1 = 8/\log(1+\sigma_f^{-2})$, $C_2 = \sum_{t=1}^{T}(1/t^2 + 2C_\mu t^2\epsilon_{M_t} + 4C_\sigma\beta_t^{1/2}t\epsilon_{M_t}^{1/2})$, we have:*

$$Pr\left\{R_T \leq \sqrt{C_1 T\beta_T\gamma_T} + C_2\right\} \geq 1-\delta$$

where $R_T = \sum_{t=1}^{T} r_t$ is the cumulative regret, γ_T is the maximal information gain after T rounds.

Proof. Replacing the term $1/t^2$ by $1/t^2 + 2C_\mu t^2\epsilon_{M_t} + 4C_\sigma\beta_t^{1/2}t\epsilon_{M_t}^{1/2}$ in the proof of Theorem 2 in Srinivas et al. (2012) finishes this proof. □

Remark 4.6. The smoothness assumption holds for any stationary kernel $K(\boldsymbol{x}, \boldsymbol{x'}) = k(\boldsymbol{x} - \boldsymbol{x'})$ that is four times differentiable, including squared exponential and Matérn kernels with $\nu > 2$. Whether the same conclusion for Matérn 3/2 holds is unclear, so we choose Matérn 3/2 in the main paper to empirically test its performance and find promising results.

5 RESULTS

We compare the performance of PK-MIQP against state-of-the-art minimizers over several benchmarks and a hyperparameter tuning problem. The Matérn 3/2 kernel and LCB are used for all methods; only the optimizer used for the acquisition function is changed. For gradient-based methods, we choose L-BFGS-B (Zhu et al., 1997), SLSQP (Kraft, 1988) and trust-constr (Byrd et al., 1999). For sampling-based methods, we consider COBYLA (Powell, 1994) and Nelder-Mead (Nelder and Mead, 1965). These methods are chosen since they are relatively general and can be directly deployed without altering the BO algorithm or acquisition function. We use the default implementations in `scipy` (Virtanen et al., 2020).

All experiments were performed on a 3.2 GHz Intel Core i7-8700 CPU with 16 GB memory. For each case, we report the mean with ±0.5 standard deviation of simple regret over 20 replications. For all benchmarks, we initially sample $\min\{10D, 30\}$ points using Latin hypercube sampling (LHS). For the SVM tuning problem, the size of the initial set is chosen as 10, since evaluations are time-consuming. At each iteration, the sample points are standardized to $[0, 1]$ before the GP training. We set size of solution pools P_{sub} and P_{rand} to 10 for PK-MIQP. We use GPflow (Matthews et al., 2017) to implement GP models and Gurobi v11.0.0 (Gurobi Optimization, LLC, 2024) to solve the resulting MIQPs (including solution pools). Full implementation details are provided in Appendix B. The code is available at GitHub.

5.1 Single acquisition function optimization

Before considering a full BO loop, we first study the performance of PK-MIQP on optimizing a given acquisition function. Specifically, we consider GP models given random samples from the prior in 1D–5D and employ different solvers to minimize the resulting LCB. The results for GP models with Matérn 3/2 kernel are given in Table 2. PK-MIQP consistently outperforms the other gradient- and sampling-based methods considered, suggesting the benefits of global optimization. Notably, increasing the problem dimension does not significantly impact the solution quality returned by PK-MIQP. The same conclusions can be made when using the RBF kernel (results given in Appendix C), which furthermore supports the generalizability of the proposed PK-MIQP framework.

5.2 Real-time results

To investigate the computational requirements to achieve global optimality using PK-MIQP, we report the computational time of each method used in Section 5.1. Add-GP training is applied to 1D–5D functions sampled from GP models, where each function is decomposed into $N_g = D$ groups. The GP model is then trained with the additive kernel, and optimization solvers are applied for each group independently.

Table 2: Comparison of solvers on optimizing random acquisition functions using Matérn 3/2 kernel. The mean of the optimal LCB values found over the 20 replications is reported with 0.5 standard deviation in parentheses. PK-MIQP consistently outperforms other gradient- and sampling-based methods.

method	1D	2D	3D	4D	5D
L-BFGS-B	-0.68(0.54)	-0.94(0.51)	-1.79(0.74)	-1.21(0.77)	-1.22(0.50)
Nelder-Mead	-0.65(0.51)	-0.96(0.51)	-1.32(0.42)	-1.98(1.07)	-1.07(0.41)
COBYLA	-1.09(0.44)	-1.26(0.52)	-1.48(0.75)	-1.58(0.75)	-0.68(0.49)
SLSQP	-0.59(0.53)	-0.99(0.70)	-1.58(0.75)	-1.49(0.86)	-1.43(0.51)
trust-constr	-0.54(0.53)	-1.19(0.55)	-1.95(0.53)	-1.71(0.88)	-1.61(0.49)
PK-MIQP	**-1.76(0.27)**	**-2.26(0.49)**	**-2.17(0.36)**	**-2.25(0.74)**	**-1.62(0.50)**

Table 3: Comparison of solvers on optimizing random acquisition functions using Matérn 3/2 kernel. The mean of the computational time in seconds over the 20 replications is reported with 0.5 standard deviation in parentheses.

method	1D	2D	3D	4D	5D
L-BFGS-B	0.12(0.06)	0.14(0.08)	0.46(0.25)	0.41(0.16)	0.80(0.37)
Nelder-Mead	0.27(0.11)	1.63(2.35)	1.96(1.28)	2.34(0.96)	3.75(1.51)
COBYLA	0.21(0.06)	0.48(0.29)	1.57(1.32)	2.21(1.52)	1.68(0.75)
SLSQP	0.08(0.04)	0.10(0.05)	0.40(0.17)	0.24(0.13)	0.61(0.23)
trust-constr	0.26(0.11)	0.78(0.21)	0.88(0.29)	2.67(1.76)	1.78(0.84)
add-GP PK-MIQP	2.08(0.35)	7.14(0.93)	15.41(2.04)	19.06(4.52)	22.45(3.34)
PK-MIQP	2.08(0.35)	11.37(3.95)	866.48(601.49)	1557.82(576.70)	3199.24(361.09)

As shown in Table 3, PK-MIQP has a higher time complexity compared to gradient- and sample-based methods, which is the expected cost of employing global optimization. Note that BO is particularly useful when evaluating the unknown objective function is expensive, meaning extended computational times may not be a deterrent in many BO settings. Introducing add-GP greatly reduces the time cost for PK-MIQP, and we hope that our work motivates future research in MIQP methods to accelerate solutions.

5.3 Bayesian optimization using PK-MIQP

This section tests the performance of PK-MIQP with the Matérn 3/2 kernel in a full BO loop over the following functions (see Appendix B for their formulations):

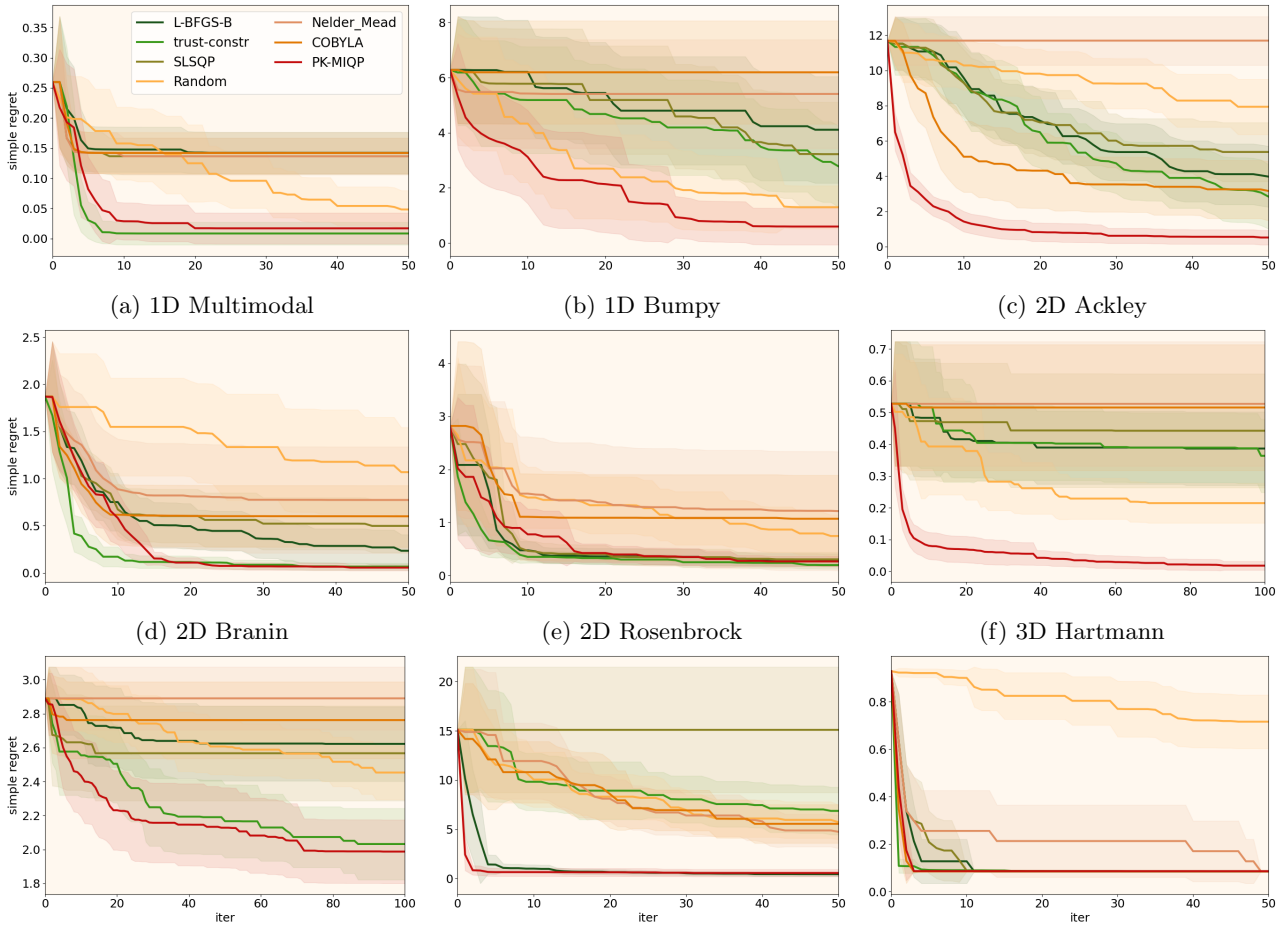
Unconstrained benchmarks: Bumpy (1D), Multimodal (1D), Ackley (2D), Branin (2D), Rosenbrock (2D), Hartmannn (3D), and Michalewicz (5D). Most of these functions are commonly used as synthetic benchmarks in BO literature. Additionally, to highlight cases where intuitively there may be significant difference between choosing a global minima and a local minima of the acquisition function, we also select Bumpy and Multimodal as benchmarks. Bumpy is periodic with multiple local and global minima, and Multimodal has multiple local minima and a unique global minimum.

Constrained benchmark: KS224 (2D) function with 4 linear constraints (Schittkowski, 2008). This function is used to demonstrate the ability of PK-MIQP to handle additional constraints.

SVM hyperparameter tuning: As a real-world example with long evaluation times, we consider the hyperparameter tuning task using support vector machine (SVM) as a text classifier. The regularization parameter and kernel coefficient in SVM are set as the hyperparameters to be tuned. The objective is to maximize the 5-fold cross-validation score of the SVM.

For the 3D and 5D benchmarks, we use add-GP training for all methods as in Section 5.2. Figures 3f and 3g show the results for the these cases.

For the constrained benchmark, PK-MIQP can easily handle linear and quadratic constraints by adding them into formulation (5). For methods that cannot directly incorporate constraints, such as L-BFGS-B and Nelder-Mead, we add a penalty term to LCB on the constraint violation with a scaling parameter λ. We run the experiment three times with λ set as $\{10, 100, 1000\}$ respectively. We pick the λ setting that achieves the lowest regret at the end of BO, and report the results in Figure 3h. For hyperparameter tuning, we use the SVM implementation in scikit-learn (Pedregosa et al., 2011), which is trained as a text classifier on the 20 news group text dataset (Lang, 1995).



(a) 1D Multimodal (b) 1D Bumpy (c) 2D Ackley

(d) 2D Branin (e) 2D Rosenbrock (f) 3D Hartmann

(g) 5D Michalewicz (h) 2D constrained KS224 (i) 2D hyperparameter tuning

Figure 3: Numerical results on Bayesian optimization using PK-MIQP with Matérn 3/2 kernel and the state-of-the-art minimizers. The mean with 0.5 standard deviation of simple regret is reported over 20 replications.

Optimization results on the benchmark functions and real-world application are presented in Figure 3. BO with PK-MIQP demonstrates comparable performance to the state-of-the-art comparison methods on smooth functions, e.g., 1D multimoddal, 2D Branin and 2D Rosenbrock. On more difficult functions with numerous local minima, e.g., 1D Bumpy and 2D Ackley, PK-MIQP outperforms other methods considerably. This follows the intuition that gradient-based methods are easily trapped at local minima, and sampling-based methods can miss the global minima. PK-MIQP manages an (approximately) global optimization step. With add-GP training, PK-MIQP remains competitive performance. Note that, as dimensionality increases, sampling-methods such as COBYLA and Nelder-Mead can fail to improve through iterations. Figure 3h illustrates the results of a constrained benchmark, where again PK-MIQP is found to be effective in handling the known constraints. Finally, for the hyperparameter tuning problem, PK-MIQP achieves a cross-validation accuracy score at 91.4% at the end of optimization, which outperforms most of other solvers.

6 CONCLUSION

This work proposes PK-MIQP, a mixed-integer programming-based paradigm for *global* optimization of GP-based acquisition functions. Our formulation introduces a piecewise-linear approximation for smooth GP kernels and a corresponding MIQP representation of acquisition functions. We analyze the theoretical regret bounds of PK-MIQP, and empirically demonstrate the framework on synthetic functions, constrained benchmarks, and a hyperparameter tuning task. We hope this work demonstrates the potential of mixed-integer programming in BO settings. Future work can improve and further scale PK-MIQP using tools from this community, e.g., cutting planes, tighter formulations, and/or branching rules.

Acknowledgments

The authors gratefully acknowledge support from a Department of Computing Scholarship (YX), BASF SE, Ludwigshafen am Rhein (SZ), a BASF/Royal Academy of Engineering Senior Research Fellowship (CT) and the US National Science Foundation grant 2237616 (JP). The authors also thank Alexander Thebelt, Akshay Kudva, Daniel Lengyel and Wei-Ting Tang for the helpful discussions throughout this work, as well as Jiongjian Cai for initial exploratory experiments.

References

T. A. Adebiyi, B. Do, and R. Zhang. Optimizing posterior samples for Bayesian optimization via rootfinding, 2024. URL https://arxiv.org/abs/2410.22322.

S. Ament, S. Daulton, D. Eriksson, M. Balandat, and E. Bakshy. Unexpected improvements to expected improvement for Bayesian optimization. In *NeurIPS*, 2024.

M. Balandat, B. Karrer, D. R. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy. Botorch: a framework for efficient Monte-Carlo Bayesian optimization. In *Advances in Neural Information Processing Systems 33*, 2020.

R. Baptista and M. Poloczek. Bayesian optimization of combinatorial structures. In *International conference on machine learning*, 2018.

E. Beale and J. Tomlin. Special facilities in a general mathematical programming system for nonconvex problems using ordered sets of variables. *Operational Research*, 69:447–454, 1969.

P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, and A. Mahajan. Mixed-integer nonlinear optimization. *Acta Numerica*, 22:1–131, 2013.

E. Bradford, A. M. Schweidtmann, and A. Lapkin. Efficient multiobjective optimization employing Gaussian processes, spectral sampling and a genetic algorithm. *Journal of Global Optimization*, 71(2):407–438, 2018.

R. H. Byrd, M. E. Hribar, and J. Nocedal. An interior point algorithm for large-scale nonlinear programming. *SIAM Journal on Optimization*, 9:877–900, 1999.

C. Cartis, E. Massart, and A. Otemissov. Global optimization using random embeddings. *Mathematical Programming*, 200(2):781–829, 2023.

L. Colliandre and C. Muller. Bayesian optimization in drug discovery. *Methods in Molecular Biology*, 2716: 101–136, 2024.

S. Daulton, X. Wan, D. Eriksson, M. Balandat, M. A. Osborne, and E. Bakshy. Bayesian optimization over discrete and mixed spaces via probabilistic reparameterization. In *NeurIPS*, 2022.

E. Daxberger, A. Makarova, M. Turchetta, and A. Krause. Mixed-variable Bayesian optimization. In *IJCAI*, 2020.

G. De Ath, R. M. Everson, A. A. M. Rahat, and J. E. Fieldsend. Greed is good: Exploration and exploitation trade-offs in Bayesian optimisation. *ACM Transactions on Evolutionary Learning and Optimization*, 1:1–22, 2021.

A. Deshwal, S. Belakaria, and J. R. Doppa. Mercer features for efficient combinatorial Bayesian optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

D. K. Duvenaud, H. Nickisch, and C. Rasmussen. Additive Gaussian processes. *Advances in neural information processing systems*, 24, 2011.

D. Eriksson and M. Jankowiak. High-dimensional Bayesian optimization with sparse axis-aligned subspaces. In *UAI*, 2021.

D. Eriksson, M. Pearce, J. Gardner, R. D. Turner, and M. Poloczek. Scalable global optimization via local Bayesian optimization. In *NeurIPS*, 2019.

J. J. H. Forrest, J. Hirst, and J. A. Tomlin. Practical solution of large mixed integer programming problems with UMPIRE. *Management Science*, 20(5): 736–773, 1974.

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

Gurobi Optimization, LLC. Gurobi optimizer reference manual, 2024. URL https://www.gurobi.com.

N. Hansen. The cma evolution strategy: a comparing review. *Towards a new evolutionary computation: Advances in the estimation of distribution algorithms*, 2006.

D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 1998.

K. Kandasamy, J. Schneider, and B. Póczos. High dimensional Bayesian optimisation and bandits via additive models. In *ICML*, 2015.

J. Kim and S. Choi. On local optimizers of acquisition functions in Bayesian optimization. In *Machine Learning and Knowledge Discovery in Databases*, 2021.

D. P. Kingma and J. Ba. Adam: a method for stochastic optimization. In *ICLR*, 2015.

D. Kraft. *A software package for sequential quadratic programming*. Wiss. Berichtswesen d. DFVLR, 1988.

K. Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339, 1995.

L. Mathesen, G. Pedrielli, S. H. Ng, and Z. B. Zabinsky. Stochastic optimization with adaptive restart: A framework for integrated local and global learning. *Journal of Global Optimization*, 79:87–110, 2021.

A. G. d. G. Matthews, M. van der Wilk, T. Nickson, K. Fujii, A. Boukouvalas, P. León-Villagrá, Z. Ghahramani, and J. Hensman. GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 18(40):1–6, 2017.

A. M. K. Nambiar, C. P. Breen, T. Hart, T. Kulesza, T. F. Jamison, and K. F. Jensen. Bayesian optimization of computer-proposed multistep synthetic routes on an automated robotic flow platform. *ACS Central Science*, 8(6):825–836, 2022.

J. A. Nelder and R. Mead. A simplex method for function minimization. *Comput. J.*, 7:308–313, 1965.

C. Oh, E. Gavves, and M. Welling. BOCK : Bayesian optimization with cylindrical kernels. In *ICML*, 2018.

T. P. Papalexopoulos, C. Tjandraatmadja, R. Anderson, J. P. Vielma, and D. Belanger. Constrained discrete black-box optimization using mixed-integer programming. In *ICML*, 2022.

J. A. Paulson and C. Tsay. Bayesian optimization as a flexible and efficient design framework for sustainable process systems. *arXiv preprint arXiv:2401.16373*, 2024.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: machine learning in Python. *Journal of Machine Learning Research*, 12, 2011.

M. J. Powell. *A direct search optimization method that models the objective and constraint functions by linear interpolation*. Springer, 1994.

S. Rana, C. Li, S. Gupta, V. Nguyen, and S. Venkatesh. High dimensional bayesian optimization with elastic gaussian process. In *International conference on machine learning*, 2017.

M. P. Ranjit, G. Ganapathy, K. Sridhar, and V. Arumugham. Efficient deep learning hyperparameter tuning using cloud infrastructure: intelligent distributed hyperparameter tuning with Bayesian optimization in the cloud. In *IEEE 12th International Conference on Cloud Computing (CLOUD)*, 2019.

J. Rodemann and T. Augustin. Imprecise Bayesian optimization. *Knowledge-Based Systems*, 2024.

K. Schittkowski. 306 test problems for nonlinear programming with optimal solutions - User's guide, 2008. URL `https://klaus-schittkowski.de/tpnp.htm`.

E. Schulz, M. Speekenbrink, and A. Krause. A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions. *Journal of Mathematical Psychology*, 85:1–16, 2018.

A. M. Schweidtmann, D. Bongartz, D. Grothe, T. Kerkenhoff, X. Lin, J. Najman, and A. Mitsos. Deterministic global optimization with Gaussian processes embedded. *Mathematical Programming Computation*, 13(3):553–581, 2021.

X. Song, Q. Zhang, C. Lee, E. Fertig, T.-K. Huang, L. Belenki, G. Kochanski, S. Ariafar, S. Vasudevan, S. Perel, and D. Golovin. The Vizier Gaussian process bandit algorithm, 2024. URL `https://arxiv.org/abs/2408.11527`.

A. Spagnol, R. L. Riche, and S. D. Veiga. Global sensitivity analysis for optimization with variable selection. *SIAM/ASA Journal on Uncertainty Quantification*, 7:417–443, 2019.

N. Srinivas, A. Krause, S. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: no regret and experimental design. In *ICML*, 2010.

N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger. Information-theoretic regret bounds for Gaussian process optimization in the bandit setting. *IEEE Transactions on Information Theory*, 58:3250–3265, 2012.

A. Thebelt, C. Tsay, R. Lee, N. Sudermann-Merx, D. Walz, B. Shafei, and R. Misener. Tree ensemble kernels for bayesian optimization with known constraints over mixed-feature spaces. *Advances in Neural Information Processing Systems*, 35:37401–37415, 2022a.

A. Thebelt, C. Tsay, R. M. Lee, N. Sudermann-Merx, D. Walz, T. Tranter, and R. Misener. Multi-objective constrained optimization for energy applications via tree ensembles. *Applied Energy*, 306: 118061, 2022b.

P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, et al. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17:261–272, 2020.

J. Wilson, F. Hutter, and M. Deisenroth. Maximizing acquisition functions for Bayesian optimization. In *NeurIPS*, 2018.

C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on mathematical software (TOMS)*, 23:550–560, 1997.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

A Extended Proofs

Proof of Theorem 4.3. Denote $\psi := K_{xX} - \tilde{K}_{xX}$ and $\Psi := K_{XX} - \tilde{K}_{XX}$. Since the difference between the true and approximated kernel functions is bounded by ϵ_M (see Remark 4.1), we can obtain:

$$\|\psi\|_2 \leq \sqrt{N}\epsilon_M, \ \|\Psi\|_2 \leq N\epsilon_M$$

where we use the fact that $\|\Psi\|_2 \leq N\|\Psi\|_{\max}$ and $\|\Psi\|_{\max} \leq \epsilon_M$.

By the definition of $\mu(\boldsymbol{x})$ and $\tilde{\mu}(\boldsymbol{x})$,

$$\begin{aligned}
\mu(\boldsymbol{x}) - \tilde{\mu}(\boldsymbol{x}) &= K_{xX}K_{XX}^{-1}\boldsymbol{y} - \tilde{K}_{xX}\tilde{K}_{XX}^{-1}\boldsymbol{y} \\
&= (\tilde{K}_{xX} + \psi)K_{XX}^{-1}\boldsymbol{y} - \tilde{K}_{xX}\tilde{K}_{XX}^{-1}\boldsymbol{y} \\
&= \tilde{K}_{xX}K_{XX}^{-1}\boldsymbol{y} + \psi K_{XX}^{-1}\boldsymbol{y} - \tilde{K}_{xX}\tilde{K}_{XX}^{-1}\boldsymbol{y} \\
&= \tilde{K}_{xX}(K_{XX}^{-1} - \tilde{K}_{XX}^{-1})\boldsymbol{y} + \psi K_{XX}^{-1}\boldsymbol{y} \\
&= \tilde{K}_{xX}K_{XX}^{-1}(\tilde{K}_{XX} - K_{XX})\tilde{K}_{XX}^{-1}\boldsymbol{y} + \psi K_{XX}^{-1}\boldsymbol{y} \\
&= -\tilde{K}_{xX}K_{XX}^{-1}\Psi\tilde{K}_{XX}^{-1}\boldsymbol{y} + \psi K_{XX}^{-1}\boldsymbol{y}
\end{aligned}$$

where we use the property $A^{-1} - B^{-1} = A^{-1}(B - A)B^{-1}$ for two invertible matrices A and B.

Note that $\|\tilde{K}_{xX}\|_2 \leq \sqrt{N}\sigma_f^2$ and $\|\boldsymbol{y}\|_2 \leq \sqrt{N}$ (recall our assumption that $f(x) \in [0, 1]$), then we have:

$$\begin{aligned}
|\mu(\boldsymbol{x}) - \tilde{\mu}(\boldsymbol{x})| &\leq \|\tilde{K}_{xX}\|_2\|K_{XX}^{-1}\|_2\|\Psi\|_2\|\tilde{K}_{XX}^{-1}\|_2\|\boldsymbol{y}\|_2 + \|\psi\|_2\|K_{XX}^{-1}\|_2\|\boldsymbol{y}\|_2 \\
&\leq N\|K_{XX}^{-1}\|_2(N\sigma_f^2\|\tilde{K}_{XX}^{-1}\|_2 + 1)\epsilon_M \\
&\leq C_\mu N^2\epsilon_M
\end{aligned}$$

where the constant $C_\mu = \sigma_\epsilon^{-4}\sigma_{\max}^2$ and $\sigma_{\max}^2$ is the upper bound of kernel variance σ_f^2. Note we ignore the small term since $N \gg 1$ and only consider the dominating term (the first term in this case).

Similarly, for the difference between variance $\sigma^2(\boldsymbol{x})$ and $\tilde{\sigma}^2(\boldsymbol{x})$, we have:

$$\begin{aligned}
\sigma^2(\boldsymbol{x}) - \tilde{\sigma}^2(\boldsymbol{x}) &= K_{xx} - K_{xX}K_{XX}^{-1}K_{Xx} - \tilde{K}_{xx} + \tilde{K}_{xX}\tilde{K}_{XX}^{-1}\tilde{K}_{Xx} \\
&= \tilde{K}_{xX}\tilde{K}_{XX}^{-1}\tilde{K}_{Xx} - K_{xX}K_{XX}^{-1}K_{Xx} \\
&= \tilde{K}_{xX}\tilde{K}_{XX}^{-1}\tilde{K}_{Xx} - (\tilde{K}_{xX} + \psi)K_{XX}^{-1}(\tilde{K}_{Xx} + \psi^T) \\
&= \tilde{K}_{xX}\tilde{K}_{XX}^{-1}\tilde{K}_{Xx} - \tilde{K}_{xX}K_{XX}^{-1}\tilde{K}_{Xx} - 2\tilde{K}_{xX}K_{XX}^{-1}\psi^T - \psi K_{XX}^{-1}\psi^T \\
&= \tilde{K}_{xX}(\tilde{K}_{XX}^{-1} - K_{XX}^{-1})\tilde{K}_{Xx} - 2\tilde{K}_{xX}K_{XX}^{-1}\psi^T - \psi K_{XX}^{-1}\psi^T \\
&= \tilde{K}_{xX}K_{XX}^{-1}\Psi\tilde{K}_{XX}^{-1}\tilde{K}_{Xx} - 2\tilde{K}_{xX}K_{XX}^{-1}\psi^T - \psi K_{XX}^{-1}\psi^T
\end{aligned}$$

where $K_{xx} = \tilde{K}_{xx} = \sigma_f^2$ in our approximation, and we reuse equation $\tilde{K}_{XX}^{-1} - K_{XX}^{-1} = K_{XX}^{-1}\Psi\tilde{K}_{XX}^{-1}$.

Then we have:

$$\begin{aligned}
|\sigma^2(\boldsymbol{x}) - \tilde{\sigma}^2(\boldsymbol{x})| &\leq \|\tilde{K}_{xX}\|_2\|K_{XX}^{-1}\|_2\|\Psi\|_2\|\tilde{K}_{XX}^{-1}\|_2\|\tilde{K}_{Xx}\|_2 + 2\|\tilde{K}_{xX}\|_2\|K_{XX}^{-1}\|_2\|\psi\|_2 + \|K_{XX}^{-1}\|_2\|\psi\|_2^2 \\
&\leq N\|K_{XX}^{-1}\|_2(N\sigma_f^4\|\tilde{K}_{XX}^{-1}\|_2 + 2\sigma_f^2 + \epsilon_M)\epsilon_M \\
&\leq C_\sigma^2 N^2\epsilon_M
\end{aligned}$$

where the constant $C_\sigma = \sigma_\epsilon^{-2}\sigma_{\max}^2$. Note that we again only consider the dominating term (the first term again).

Consider the following two cases:

Case 1: if $\max(\sigma(\boldsymbol{x}), \tilde{\sigma}(\boldsymbol{x})) \leq C_\sigma N\epsilon_M^{1/2}$, then:

$$|\sigma(\boldsymbol{x}) - \tilde{\sigma}(\boldsymbol{x})| = \max(\sigma(\boldsymbol{x}), \tilde{\sigma}(\boldsymbol{x})) - \min(\sigma(\boldsymbol{x}), \tilde{\sigma}(\boldsymbol{x})) \leq \max(\sigma(\boldsymbol{x}), \tilde{\sigma}(\boldsymbol{x})) \leq C_\sigma N\epsilon_M^{1/2}$$

since $\min(\sigma(\boldsymbol{x}), \tilde{\sigma}(\boldsymbol{x})) \geq 0$.

Case 2: if $\max(\sigma(\boldsymbol{x}), \tilde{\sigma}(\boldsymbol{x})) > C_\sigma N \epsilon_M^{1/2}$, then:

$$|\sigma(\boldsymbol{x}) - \tilde{\sigma}(\boldsymbol{x})| = \frac{|\sigma^2(\boldsymbol{x}) - \tilde{\sigma}^2(\boldsymbol{x})|}{\sigma(\boldsymbol{x}) + \tilde{\sigma}(\boldsymbol{x})} \leq \frac{|\sigma^2(\boldsymbol{x}) - \tilde{\sigma}^2(\boldsymbol{x})|}{\max(\sigma(\boldsymbol{x}), \tilde{\sigma}(\boldsymbol{x}))} \leq C_\sigma N \epsilon_M^{1/2}$$

Therefore, we conclude that:

$$|\sigma(\boldsymbol{x}) - \tilde{\sigma}(\boldsymbol{x})| \leq C_\sigma N \epsilon_M^{1/2}$$

□

Proof of Theorem 4.4. Except for the fact that $\boldsymbol{x_t}$ is chosen by minimizing the approximated LCB instead of the true LCB, all conditions of this theorem are the same as in Lemma 5.8 of Srinivas et al. (2012). Therefore, the conclusions of Lemma 5.5 and Lemma 5.7 in Srinivas et al. (2012) still hold here, and we first restate them without repeating proofs for simplicity:

$$|f(\boldsymbol{x_t}) - \mu_{t-1}(\boldsymbol{x_t})| \leq \beta_t^{1/2} \sigma_{t-1}(\boldsymbol{x_t})$$
$$|f(\boldsymbol{x^*}) - \mu_{t-1}([\boldsymbol{x^*}]_{\boldsymbol{t}})| \leq \beta_{t-1}([\boldsymbol{x^*}]_{\boldsymbol{t}}) + 1/t^2$$

where $[\boldsymbol{x^*}]_{\boldsymbol{t}}$ is the closest point in D_t to $\boldsymbol{x^*}$, and $D_t \subset D$ is a discretization at t-th iteration.

Recall the definition of $\boldsymbol{x_t}$, we have:

$$\tilde{\mu}_{t-1}(\boldsymbol{x_t}) - \beta_t^{1/2} \tilde{\sigma}_{t-1}(\boldsymbol{x_t}) \leq \tilde{\mu}_{t-1}([\boldsymbol{x^*}]_{\boldsymbol{t}}) - \beta_t^{1/2} \tilde{\sigma}_{t-1}([\boldsymbol{x^*}]_{\boldsymbol{t}})$$

Combining those three inequalities with our inequalities from Theorem 4.3 gives

$$\begin{aligned}
r_t &= f(\boldsymbol{x_t}) - f(\boldsymbol{x^*}) \\
&\leq \mu_{t-1}(\boldsymbol{x_t}) + \beta_t^{1/2} \sigma_{t-1}(\boldsymbol{x_t}) - \mu_{t-1}([\boldsymbol{x^*}]_{\boldsymbol{t}}) + \beta_t^{1/2} \sigma_{t-1}([\boldsymbol{x^*}]_{\boldsymbol{t}}) + 1/t^2 \\
&\leq \tilde{\mu}_{t-1}(\boldsymbol{x_t}) + \beta_t^{1/2} \tilde{\sigma}_{t-1}(\boldsymbol{x_t}) - \tilde{\mu}_{t-1}([\boldsymbol{x^*}]_{\boldsymbol{t}}) + \beta_t^{1/2} \tilde{\sigma}_{t-1}([\boldsymbol{x^*}]_{\boldsymbol{t}}) + 1/t^2 + 2C_\mu t^2 \epsilon_{M_t} + 2C_\sigma \beta_t^{1/2} t \epsilon_{M_t}^{1/2} \\
&\leq 2\beta_t^{1/2} \tilde{\sigma}_{t-1}(\boldsymbol{x_t}) + 1/t^2 + 2C_\mu t^2 \epsilon_{M_t} + 2C_\sigma \beta_t^{1/2} t \epsilon_{M_t}^{1/2} \\
&\leq 2\beta_t^{1/2} \sigma_{t-1}(\boldsymbol{x_t}) + 1/t^2 + 2C_\mu t^2 \epsilon_{M_t} + 4C_\sigma \beta_t^{1/2} t \epsilon_{M_t}^{1/2}
\end{aligned}$$

□

B Experimental Implementation Details

B.1 Problem setup

For our experiments, the GP models, including the additive GPs, are implemented with a Matérn 3/2 kernel using the GPflow package (Matthews et al., 2017). In our implementation, kernel variance σ_f^2 is bounded to be within $[0.05, 20]$ and kernel lengthscale l is bounded to be within $[0.005, 20]$. The GP parameters are optimized over 10 multiple starts, with random initial values for kernel parameters. The set of parameters with minimal negative log likelihood is then selected. We take the noise term $\sigma_\epsilon^2 = 10^{-6}$.

For PK-MIQP, all MIQPs are solved using Gurobi v11.0.0 (Gurobi Optimization, LLC, 2024). We list relevant solver hyperparameters in Table 4 and use default values for other hyperparameters.

As mentioned in the "Initialization Heuristic" paragraph of Section 4.2, PK-MIQP first solves a sub-problem to initialize the full-problem. The sub-problem drops all variance-related terms, resulting in

$$\begin{aligned}
&\min_{x \in \mathcal{X}} \ \tilde{\mu} \\
&s.t. \ \tilde{\mu} = \tilde{K}_{xX} \tilde{K}_{XX}^{-1} \boldsymbol{y} \\
&\quad r_i^2 = \frac{\|\boldsymbol{x} - \boldsymbol{x_i}\|_2^2}{l^2}, \ \forall 1 \leq i \leq N \\
&\quad \tilde{K}_{xX_i} = \tilde{k}(r_i), \ \forall 1 \leq i \leq N
\end{aligned}$$

Table 4: Gurobi hyperparameters.

Parameter	Value
NonConvex	2
ObjScale	0.5
ScaleFlag	1
MIPgap	0.5
PoolSolutions	10
TimeLimit	5400

For sub-problem, we set `TimeLimit` to 1800(s); most are solved relatively quickly.

We use scipy (Virtanen et al., 2020) to implement state-of-the-art optimizers for comparison, including:

L-BFGS-B: Limited-memory Broyden–Fletcher–Goldfarb–Shanno Bound (L-BFGS-B) (Zhu et al., 1997) algorithm is a gradient-based solver for minimizing differentiable function. L-BFGS-B uses the first derivative as well as an estimate of the inverse of Hessian matrix to steer the search. It doesn't support constrained minimization problems.

COBYLA: Constrained Optimization BY Linear Approximation (COBYLA) (Powell, 1994) algorithm is a derivative-free solver for minimizing scalar functions. COBYLA updates a linear approximation of both the objective and constraints and performs a simplex method within the trust region to solve the problem. It supports constrained minimization problems.

trust-constr: Trust-region algorithm for constrained optimization (trust-constr) (Byrd et al., 1999) is a gradient-based solver that applies the interior point algorithm to minimize a given function. Both gradients and Hessians are approximated during the solving. It supports constrained minimization problems.

Nelder-Mead: Nelder-Mead algorithm (Nelder and Mead, 1965) is a derivative-free method for solving multidimensional optimization problems. It uses a simplex algorithm based on function comparison and doesn't support constrained minimization problems.

SLSQP: Sequential Least SQuares Programming (SLSQP) (Kraft, 1988) algorithm is a gradient-based optimizer. It uses the Han-Powell quasi-Newton method combined with BFGS to solve a Lagrange function at each iteration. It supports constrained minimization problems.

For these methods, the tolerance of termination (`tol`) is set to 10^{-6}, and the maximum number of iterations (`maxiter`) is set to 1000. The starting point (`x0`) for applicable methods is set to $\mathbf{0}$.

B.2 Benchmarks

In this section, we provide the formulations for all benchmark functions used in our experiments.

Bumpy (1D):

$$\begin{aligned} \min \ & -\sum_{i=1}^{6} i \cdot \sin\left((i+1)x + i\right) \\ s.t. \ & x \in [-10, 10] \end{aligned} \tag{Bumpy}$$

Multimodal (1D):

$$\begin{aligned} \min \ & \sin x + \sin\left(\frac{10}{3}x\right) \\ s.t. \ & x \in [-2.7, 7.5] \end{aligned} \tag{Multimodal}$$

Ackley (2D)

$$\begin{aligned} \min \ & -20\exp\left(-0.2\sqrt{0.5(x_1^2 + x_2^2)}\right) - \exp\left(0.5(\cos(2\pi x_1)) + \cos(2\pi x_2)\right) + 20 + \exp(1) \\ s.t. \ & x_1, x_2 \in [-32, 16] \end{aligned} \tag{Ackley}$$

Branin (2D)

$$\begin{aligned} &\min\ a(x_2 - bx_1^2 + cx_1 - r)^2 + s(1-t)\cos(x_1) + s \\ &s.t.\ x_1 \in [-5, 10], x_2 \in [0, 15] \end{aligned} \tag{Branin}$$

where $a = 1, b = 5.1/(4\pi^2), c = 5/\pi, r = 6, s = 10, t = 1/(8\pi)$.

Rosenbrock (2D)

$$\begin{aligned} &\min\ (1 - x_1)^2 + 100(x_2 - x_1^2)^2 \\ &s.t.\ x_1 \in [-2, 2], x_2 \in [-1, 3] \end{aligned} \tag{Rosenbrock}$$

Hartmann (3D)

$$\begin{aligned} &\min\ -\sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{3} A_{ij}(x_j - P_{ij})^2\right) \\ &s.t.\ x_1, x_2, x_3 \in [0, 1] \end{aligned} \tag{Hartmann}$$

where $\alpha = (1, 1.2, 3, 3.2)^T$ and

$$A = \begin{pmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{pmatrix},\ P = 10^{-4}\begin{pmatrix} 3689 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8828 \end{pmatrix}$$

Michalewicz (5D)

$$\begin{aligned} &\min\ -\sum_{i=1}^{5} \sin(x_i)\sin^{20}\left(\frac{ix_i^2}{\pi}\right) \\ &s.t.\ x_i \in [0, \pi], i = 1, \cdots, 5 \end{aligned} \tag{Michalewicz}$$

Constrained KS224 (2D) For thise case, we solve the following constrained minimization problem with the 2D KS224 function (Schittkowski, 2008) as the objective.

$$\begin{aligned} &\min\ 2x_1^2 + x_2^2 - 48x_1 - 40x_2 \\ &s.t.\ -(x_1 + 3x_2) \le 0 \\ &\quad -(18 - x_1 - 3x_2) \le 0 \\ &\quad -(x_1 + x_2) \le 0 \\ &\quad -(8 - x_1 - x_2) \le 0 \\ &\quad x_1, x_2 \in [0, 6] \end{aligned} \tag{Constrained KS224}$$

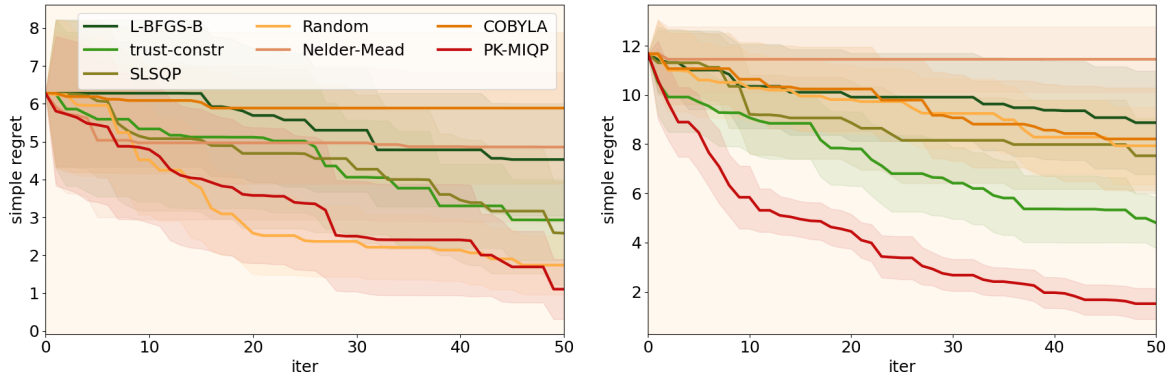
SVM hyperparameter tuning: The 20 newsgroup dataset (Lang, 1995) is a classic dataset consisting of posts on 20 topics for text classification. We implement a simple pipeline for this text classification in scikit-learn (Pedregosa et al., 2011) comprising first using a TF-IDF vectorizer to convert text to vectors and then applying C-Support Vector Classification (SVC). We use the default settings for hyperparameters in SVC except leaving the regularization hyperparameter $C \in [0.01, 1000]$ and kernel coefficient $\gamma \in [0.01, 1000]$ to be tuned. Negative 5-fold cross validation accuracy is set as the objective to be minimized.

C Numerical results using RBF kernel

We perform the same single acquisition function optimization experiment as in Section 5.1, but using the RBF kernel instead of the Matern 3/2 kernel. Table 5 reports the mean of the optimal LCB values found by different optimizers, along with half of the standard deviation. For Bayesian optimization, we empirically demonstrate the performance of PK-MIQP with RBF kernel in two benchmarks as shown in Figure 4.

Table 5: Comparison of solvers on optimizing random acquisition functions using RBF kernel. The mean of the optimal LCB values found over the 20 replications is reported with 0.5 standard deviation in parentheses. PK-MIQP consistently outperforms other gradient- and sampling-based methods.

method	1D	2D	3D	4D	5D
L-BFGS-B	-0.67(0.68)	-0.90(0.42)	-2.03(1.85)	-2.77(1.45)	-2.26(1.31)
Nelder-Mead	-0.98(0.63)	-0.87(0.67)	-1.53(0.74)	-2.70(1.43)	-2.12(1.38)
COBYLA	-1.25(0.82)	-1.44(0.60)	-2.18(1.83)	-1.47(0.99)	-2.49(1.37)
SLSQP	-1.04(1.00)	-0.72(0.41)	-2.34(1.83)	-2.53(1.49)	-2.28(1.29)
trust-constr	-0.57(0.68)	-1.27(0.38)	-1.90(0.70)	-3.06(1.56)	-2.89(1.32)
PK-MIQP	**-1.82(0.71)**	**-1.99(0.49)**	**-2.94(0.65)**	**-4.20(1.87)**	**-3.10(1.28)**



(a) 1D Bumpy

(b) 2D Ackley

Figure 4: Numerical results on Bayesian optimization using PK-MIQP with RBF kernel and the state-of-the-art minimizers. The mean with 0.5 standard deviation of simple regret is reported over 20 replications. PK-MIQP is similarly applicable to the RBF kernel and again outperforms other minimizers.