CobBO: Coordinate Backoff Bayesian Optimization with Two-Stage Kernels

Anonymous Author(s)
Affiliation
Address
email

Abstract

Bayesian optimization is a popular method for optimizing expensive black-box functions. Yet it oftentimes struggles in high dimensions where the computation could be prohibitively expensive and a sufficient estimation of the global landscape requires more observations. We introduce Coordinate backoff Bayesian optimization (CobBO) with two-stage kernels to alleviate this problem. In each iteration, a promising subset of coordinates is selected in the first stage, as past observed points in the full space are projected to the selected subspace adopting a simple kernel that sacrifices the approximation accuracy for computational efficiency. Then in the second stage of the same iteration a more sophisticated kernel is applied for estimating the landscape in the selected low dimensional subspace where the computational cost becomes affordable. Effectively, this second stage kernel refines the approximation of the global landscape estimated by the first stage kernel through a sequence of observations in the local subspace. This refinement lasts until a stopping rule is met determining when to back off from a certain subspace and switch to another coordinate subset. This decoupling significantly reduces the computational burden in high dimensions, while the two-stage kernels of the Gaussian process regressions fully leverage the observations in the whole space rather than only relying on observations in each coordinate subspace. Extensive evaluations show that CobBO finds solutions comparable to or better than other state-of-the-art methods for dimensions ranging from tens to hundreds, while reducing the trial complexity and computational costs.

1 Introduction

Bayesian optimization (BO) has emerged as an effective zero-order paradigm for optimizing expensive black-box functions. The entire sequence of iterations rely only on the function values of the already queried points without information on their derivatives. Though highly competitive in low dimensions (e.g., the dimension $D \leq 20$ [15]), Bayesian optimization based on Gaussian Process (GP) regression has obstacles that impede its effectiveness, especially in high dimensions.

Approximation accuracy: GP regression assumes a class of random functions in a probability space as surrogates that iteratively yield posterior distributions by conditioning on the queried points. When suggesting new query points, for complex functions with numerous local optima and saddle points due to local fluctuations, always exactly using the values on the queried points as the conditional events may mismatch the function's local landscape by overemphasizing the approximation accuracy of the global landscape.

Curse of dimensionality: As a sample efficient method, Bayesian optimization often suffers from high dimensions. Fitting the GP model (estimating the parameters, e.g., length_scales [14]), computing the Gaussian process posterior and optimizing the acquisition function in high dimensions all

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

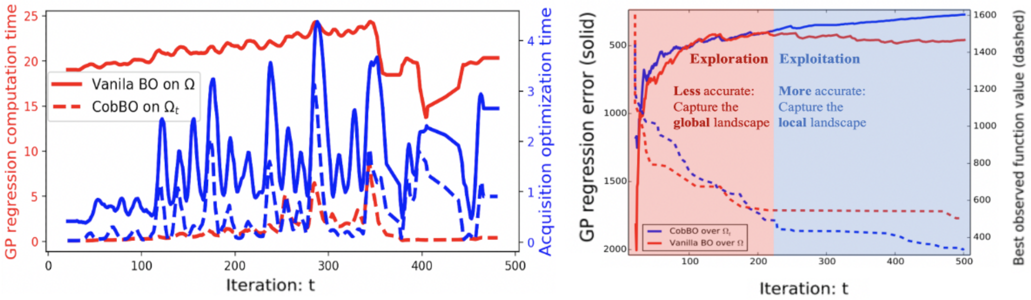

Figure 1: Minimize the fluctuated Rastrigin function on $[-5, 10]^{50}$ with 20 initial samples. [Left] Computation times for training the GP regression model and maximizing the acquisition function at each iteration. CobBO significantly reduces the execution time compared with a vanilla BO, e.g. ×13 faster in this case. [Right] The average error between the GP predictions before making queries and the true function values at the queried points (solid curves, the higher the better) and the best observed function value (dashed curves, the lower the better) at iteration t. CobBO captures the global landscape less accurately using the RBF kernel, and then explores selected subspaces Ω_t more accurately using the Matern kernel. This eventually better exploits the promising subspaces.

incur large computational costs. It also results in statistical insufficiency of exploration [11, 65]. As the GP regression's error grows with dimensions [8], more samples are required to balance that in high dimensions, which could cubically increase the computational costs in the worst case [45].

To alleviate these issues, we design coordinate backoff Bayesian optimization (CobBO) with two-stage kernels, by challenging a seemingly natural intuition stating that it is always better for Bayesian optimization to have a more accurate approximation of the objective function at all times. We demonstrate that this is not necessarily true, by showing that smoothing out local fluctuations and using the estimated function values instead of the true observations to serve as the conditional events in selected subspaces can not only significantly reduce the computation time due to the curse of dimensionality but also help in capturing the large-scale properties of the objective function $f(x)$.

Specifically, CobBO introduces the two-stage kernels with a stopping rule. The first stage of each iteration adopts a simple kernel that sacrifices the approximation accuracy of $f(x)$ for computational efficiency. For example, by using a universal radial basis function (RBF) approximation without learnable parameters [50], CobBO can eliminate the model fitting time in the full space. It captures a smooth approximation $\hat{f}(x)$ of the global landscape by interpolating the values of queried points projected to selected promising subspaces. These projected points serve as the conditional events for GP regression. In a selected coordinate subspace, the second stage of the same iteration applies a sophisticated kernel that can tolerate high computational cost in low dimensions. For example, CobBO uses the Automatic Relevance Determination (ARD) Matérn 5/2 kernel [40]. It refines the approximation of the local landscape by a sequence of observations determined by a stopping rule that backs off from a certain subspace and switches to another coordinate subset. In addition, computing the Gaussian process posterior and optimizing the acquisition function are both efficiently conducted in the low dimensional subspaces, bypassing the curse of dimensionality.

For iteration t, instead of directly computing the Gaussian process posterior distribution $\left\{\hat{f}(x)\middle|\mathcal{H}_t = \{(x_i, y_i)\}_{i=1}^{t}, x \in \Omega\right\}$ by conditioning on the observations $y_i = f(x_i)$ at queried points x_i in the full space $\Omega \subset \mathbb{R}^D$ for $i = 1, \ldots, t$, we change the conditional events, and consider

$$\left\{\hat{f}(x)\middle|R\left(P_{\Omega_t}(x_1, \ldots, x_t), \mathcal{H}_t\right), x \in \Omega_t, \Omega_t \subset \Omega\right\}$$

for a projection function $P_{\Omega_t}(\cdot)$ to a random subspace Ω_t and an interpolation function $R(\cdot, \cdot)$, e.g., using a RBF approximation without learnable parameters [50] as the simple kernel for the first stage. The projection $P_{\Omega_t}(\cdot)$ maps the queried points to virtual points on a subspace Ω_t of a lower dimension [51]. The interpolation function $R(\cdot, \cdot)$ estimates the objective values at the virtual points using the queried points and their values as specified by $\mathcal{H}_t$. The second stage within the subspace Ω_t uses the more sophisticated kernel, e.g., Matérn 5/2 kernel [40], which has a number of parameters that otherwise would be expensive to be learned in high dimensions.

This method can be viewed as a variant of block coordinate ascent tailored to Bayesian optimization by applying backoff stopping rules for switching coordinate blocks. While similar work exists [43, 48], CobBO differs by introducing the two-stage kernels and addressing the following three issues:

1. Selecting a block of coordinates for ascending requires determining the block size as well as the coordinates therein. CobBO selects the coordinate subsets by a multiplicative weights update method [2] to the preference probability associated with each coordinate. Thus, it samples more promising subspaces with higher probabilities.
2. A coordinate subspace requires a sufficient number of query points acting as the conditional events for the GP regression. CobBO leverages all observations in the whole space by interpolating the values of queried points projected to selected promising subspaces, rather than simply starting from scratch in each subspace.
3. Querying a certain subspace, under some trial budget, comes at the expense of exploring other coordinate blocks. Yet prematurely shifting to different subspaces does not fully exploit the full potential of a given subspace. Hence determining the number of consecutive function queries within a subspace makes a trade-off between exploration and exploitation. CobBO uses a stopping rule in each subspace to switch the selected coordinates. When consecutively querying data points in the same subspace, CobBO does not need to conduct the first-stage function approximation in the full space, which is far more efficient.

Through comprehensive evaluations, CobBO demonstrates appealing performance for dimensions ranging from tens to hundreds. It obtains comparable or better solutions with fewer queries, in comparison with the state-of-the-art methods, for most of the problems tested in Section 4.2.

2 Related work

Certain assumptions are often imposed on the latent structure in high dimensions. Typical assumptions include low dimensional structures and additive structures. Their advantages manifest on problems with a low dimension or a low effective dimension. However, these assumptions do not necessarily hold for non-separable functions with no redundant dimensions.

Low dimensional structure: The black-box function f is assumed to have a low effective dimension [30, 58], e.g., $f(x) = g(\Phi x)$ with some function $g(\cdot)$ and a matrix Φ of $d \times D, d << D$. A number of different methods have been developed, including random embedding [66, 11, 63, 36, 44, 70, 5, 32], low-rank matrix recovery [11, 58], and learning subspaces by derivative information [11, 13]. In contrast to existing work on subspace selections, e.g., Hashing-enhanced Subspace BO (HeSBO) [44], Mahalanobis kernel for linear embeddings [33], DROPOUT [35] and LineBO [29] (which receives a special treatment in Appendix F), CobBO efficiently leverages all the observations in the whole space using the two-stage kernels and the stopping rule in each subspace for consecutive observations, rather than only relying on limited observations in each coordinate subspace. It exploits subspace structure from a perspective of block coordinate ascent, independent of the dimensions, different from some algorithms that are more suitable for low dimensions, e.g., BADS [1].

Additive structure: A decomposition assumption is often made by $f(x) = \sum_{i=1}^{k} f^{(i)}(x_i)$, with x_i defined over low-dimensional components. In this case, the effective dimensionality of the model is the largest dimension among all additive groups [45], which is usually small. The Gaussian process is structured as an additive model [17, 28], e.g., projected-additive functions [36], ensemble Bayesian optimization (EBO) [61], latent additive structural kernel learning (HDBBO) [65] and group additive models [28, 36]. However, learning the unknown structure incurs a considerable computational cost [44], and is not applicable for non-separable functions, for which CobBO can still be applied.

Trust regions and space partitions: Trust region BO has been proven effective for high-dimensional problems. A typical pattern is to alternate between global and local search regions. In the local trust regions, many efficient methods have been applied, e.g., local Gaussian models (TurBO [14]), adaptive search on a mesh grid (BADS [1]) or quasi-Newton local optimization (BLOSSOM [41]). TurBO [14] uses Thompson sampling to allocate samples across multiple regions. A related method is to use space partitions, e.g., LA-MCTS[60] on a Monte Carlo tree search algorithm to learn efficient partitions. CobBO differs by selecting low dimensional subspaces. It can also incorporate trust regions in the first-stage global approximation, as shown in the Appendix.

3 Algorithm

Without loss of generality, suppose that the goal is to solve a maximization problem $x^* = \text{argmax}_{x\in\Omega} f(x)$ for a black-box function $f : \Omega \to \mathbb{R}$. The domain is normalized $\Omega = [0,1]^D$ with the coordinates indexed by $I = \{1, 2, \cdots, D\}$.

For a sequence of points $\mathcal{X}_t = \{x_1, x_2, \cdots, x_t\}$ with t indexing the most recent iteration, we observe $\mathcal{H}_t = \{(x_i, y_i = f(x_i))\}_{i=1}^t$. A random subset $C_t \subseteq I$ of the coordinates is selected, forming a subspace $\Omega_t \subseteq \Omega$ at iteration t. As a variant of coordinate ascent, the subspace Ω_t contains a pivot point V_t, which presumably is the maximum point $x_t^M = \text{argmax}_{x\in\mathcal{X}_t} f(x)$ with $M_t = f\left(x_t^M\right)$. CobBO may set V_t different from x_t^M to escape local optima. Then, BO is conducted within Ω_t while fixing all the other coordinates $C_t^c = I \setminus C_t$, i.e., the complement of C_t.

```
Algorithm 1: CobBO(f, τ, T)
$\mathcal{H}_\tau \leftarrow$ sample $\tau$ initial points and evaluate their values
$V_\tau, M_\tau \leftarrow$ Find the tuple with the maximal objective value in $\mathcal{H}_\tau$
$q_\tau \leftarrow 0$ Initialize the number of consecutive failed queries
$\pi_\tau \leftarrow$ Initialize a uniform preference distribution on the coordinates
for $t \leftarrow \tau$ to $T$ do
    if switch $\Omega_{t-1}$ by the backoff stopping rule (Section 3.2) then
        $C_t \leftarrow$ Sample a promising coordinate block according to $\pi_t$ (Section 3.1)
        $\Omega_t \leftarrow$ Take the subspace of $\Omega_t$ over the coordinate block $C_t$, such that $V_t \in \Omega_t$
    else
        $\Omega_t \leftarrow \Omega_{t-1}$
    $\hat{\mathcal{X}}_t \leftarrow P_{\Omega_t}(\mathcal{X}_t)$ $\left[\text{Project } \mathcal{X}_t \text{ onto } \Omega_t \text{ to obtain a set of virtual points (Eq. 1)}\right]$
    $\hat{\mathcal{H}}_t \leftarrow R\left(\hat{\mathcal{X}}_t, \mathcal{H}_t\right)$ $\left[\text{Smooth function values on } \hat{\mathcal{X}}_t \text{ by interpolation using } \mathcal{H}_t\right]$
    $p\left[\hat{f}_{\Omega_t}(x)|\hat{\mathcal{H}}_t\right] \leftarrow$ Compute the posterior distribution of the Gaussian process in $\Omega_t$
      conditional on $\hat{\mathcal{H}}_t$
    $x_{t+1} \leftarrow \text{argmax}_{x\in\Omega_t} Q_{\hat{f}\sim p(\hat{f}|\hat{\mathcal{H}}_t)}(x|\hat{\mathcal{H}}_t)$ $\left[\text{Suggest the next query in } \Omega_t \text{ (Section 3)}\right]$
    $y_{t+1} \leftarrow$ Evaluate the black-box function $y_{t+1} = f(x_{t+1})$
    if $y_{t+1} > M_t$ then
        $V_{t+1} \leftarrow x_{t+1}$, $M_{t+1} \leftarrow y_{t+1}$, $q_{t+1} \leftarrow 0$
    else
        $V_{t+1} \leftarrow V_t$, $M_{t+1} \leftarrow M_t$, $q_{t+1} \leftarrow q_t + 1$
    $\pi_{t+1} \leftarrow$ Update $\pi_t$ by a multiplicative weights update method (Eq. 2)
    $\mathcal{H}_{t+1} \leftarrow \mathcal{H}_t \bigcup \{(x_{t+1}, y_{t+1})\}$, $\mathcal{X}_{t+1} \leftarrow \mathcal{X}_t \bigcup \{x_{t+1}\}$
end
```

For BO in Ω_t, we use Gaussian processes as the random surrogates $\hat{f} = \hat{f}_{\Omega_t}(x)$ to describe the Bayesian statistics of $f(x)$ for $x \in \Omega_t$. At each iteration, the next query point is generated by solving

$$x_{t+1} = \text{argmax}_{x\in\Omega_t, V_t\in\Omega_t} Q_{\hat{f}_{\Omega_t}(x)\sim p(\hat{f}|\mathcal{H}_t)}(x|\mathcal{H}_t),$$

where the acquisition function $Q(x|\mathcal{H}_t)$ incorporates the posterior distribution of the Gaussian processes $p(\hat{f}|\mathcal{H}_t)$. Typical acquisition functions include the expected improvement (EI) [42, 27], the upper confidence bound (UCB) [3, 54, 55], the entropy search [24, 25, 64], and the knowledge gradient [16, 53, 69].

Instead of directly computing the posterior distribution $p(\hat{f}|\mathcal{H}_t)$, we replace the conditional events $\mathcal{H}_t$ by

$$\hat{\mathcal{H}}_t := R\left(P_{\Omega_t}\left(\mathcal{X}_t\right), \mathcal{H}_t\right) = \{(\hat{x}_i, \hat{y}_i)\}_{i=1}^t$$

with an interpolation function $R(\cdot,\cdot)$ and a projection function $P_{\Omega_t}(\cdot)$,

$$P_{\Omega_t}(x)^{(j)} = \begin{cases} x^{(j)} & \text{if } j \in C_t \\ V_t^{(j)} & \text{if } j \notin C_t \end{cases} \tag{1}$$

at coordinate j. It simply keeps the values of x whose corresponding coordinates are in C_t and replaces the rest by the corresponding values of V_t, as illustrated in Fig. 2.

Applying $P_{\Omega_t}(\cdot)$ on $\mathcal{X}_t$ and discarding duplicates generate a new set of distinct virtual points $\hat{\mathcal{X}}_t = \{\hat{x}_1, \hat{x}_2, \hat{x}_3, \cdots, \hat{x}_{\hat{t}}\}$, $\hat{x}_i \in \Omega_t \, \forall \, 1 \leq i \leq \hat{t} \leq t$. The function values at $\hat{x}_i \in \hat{\mathcal{X}}_t$ are interpolated as $\hat{y}_i = R(\hat{x}_i, \mathcal{H}_t)$ using the standard radial basis function [6, 7] and the observed points in $\mathcal{H}_t$. It not only significantly reduces the GP regression time due to the efficiency of RBF [6] and the acquisition function optimization in low dimensions [11], but also eventually improves the model accuracy using the more sophisticated kernel applied on Ω_t.

Note that only a fraction of the points in $\hat{\mathcal{X}}_t \cap \mathcal{X}_t$ directly observe the exact function values. The function values on the rest ones in $\hat{\mathcal{X}}_t \backslash \mathcal{X}_t$ are estimated by interpolation, which captures the landscape of $f(x)$ by smoothing out the local fluctuations. To control the trade-off between the inaccurate estimations and the exact observations in Ω_t, we design a stopping rule that optimizes the number of consistent queries in Ω_t. The more consistent queries conducted in a given subspace, the more accurate observations could be obtained, albeit at the expense of a smaller remaining budget for exploring other regions.

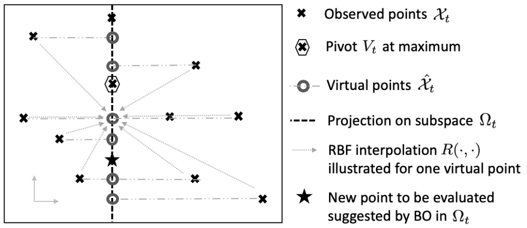
Observed points $\mathcal{X}_t$
Pivot V_t at maximum
Virtual points $\hat{\mathcal{X}}_t$
Projection on subspace Ω_t
RBF interpolation $R(\cdot,\cdot)$
illustrated for one virtual point
New point to be evaluated
suggested by BO in Ω_t

Figure 2: Two-stage kernels: subspace projection and function value interpolation

The key features of CobBO are listed in Algorithm 1, with more details in the following sections. Several auxiliary components are utilized and presented in Appendix C to deal with a larger variety of problems and corner cases.

3.1 Block coordinate ascent and subspace selection

For Bayesian optimization, consider an infeasible assumption that each iteration can exactly maximize the function $f(x)$ in Ω_t. This is not possible for one iteration but only if one can consistently query in Ω_t, since the points converge to the maximum, e.g., under the expected improvement acquisition function with fixed priors [59] and the convergence rate can be characterized for smooth functions in the reproducing kernel Hilbert space [8]. However, even with this infeasible assumption, it is known that coordinate ascent with fixed blocks can cause stagnation at a non-critical point, e.g., for non-differentiable [67] or non-convex functions [49]. This motivates us to select a subspace with a variable-size coordinate block C_t for each query. A good coordinate block can help the iterations to escape the trapped non-critical points. For example, one condition can be based on the result in [21] that assumes $f(x)$ to be differentiable and strictly quasi-convex over a collection of blocks. In practice, we do not restrict ourselves to these assumptions.

We induce a preference distribution π_t over the coordinate set I, and sample a variable-size coordinate block C_t accordingly. This distribution is updated at iteration t through a multiplicative weights update method [2]. Specifically, the values of π_t at coordinates in C_t starts off uniform and increase in face of an improvement or decrease otherwise according to different multiplicative ratios $\alpha > 1$ and $\beta > 1$, respectively,

$$w_{t,j} = w_{t-1,j} \cdot \begin{cases} \alpha & \text{if } j \in C_t \text{ and } y_t > M_{t-1} \\ 1/\beta & \text{if } j \in C_t \text{ and } y_t \leq M_{t-1} \\ 1 & \text{if } j \notin C_t \end{cases} \quad ; \quad w_{0,j} = \frac{1}{D} \quad ; \quad \pi_{t,j} = \frac{w_{t,j}}{\sum_{j=1}^{D} w_{t,j}} \tag{2}$$

This update characterizes how likely a coordinate block can generate a promising search subspace. The multiplicative ratio α is chosen to be relatively large, e.g., $\alpha = 2.0$, and β relatively small, e.g., $\beta = 1.1$, since the queries that improve the best observations $y_t > M_{t-1}$ happen more rarely than the opposite $y_t \leq M_{t-1}$.

How to dynamically select the size $|C_t|$? It is known that Bayesian optimization works well for low dimensions [15]. Thus, we specify an upper bound for the dimension of the subspace (e.g. $|C_t| \leq 30$). In principle, $|C_t|$ can be any random number in a finite set of possible block sizes $\mathcal{C}$. This is different from the method that partitions the coordinates into fixed blocks and selects one according to, e.g., cyclic order [68], random sampling or Gauss-Southwell [46].

3.2 Backoff stopping rule for consistent queries

Applying BO on Ω_t requires a strategy to determine the number of consecutive queries for making a sufficient progress. This strategy is based on previous observations, thus forming a stopping rule. In principle, there are two different scenarios, exemplifying exploration and exploitation, respectively. Persistently querying a given subspace refrains from opportunistically exploring other coordinate combinations. Abruptly shifting to different subspaces does not fully exploit the potential of a given subspace.

CobBO designs a heuristic stopping rule in compromise. It takes the above two scenarios into joint consideration, by considering not only the number of consecutive queries that fail to improve the objective function but also other factors including the improved difference $M_t - M_{t-1}$, the point distance $||x_t - x_{t-1}||$, the query budget T and the problem dimension D. On the one hand, switching to another subspace Ω_{t+1} ($\neq \Omega_t$) prematurely without fully exploiting Ω_t incurs an additional approximation error associated with the interpolation of observations in Ω_t projected to Ω_{t+1}. On the other hand, it is also possible to over-exploit a subspace, spending high query budget on marginal improvements around local optima. In order to mitigate this, even when a query leads to an improvement, other factors are considered for sampling a new subspace.

3.3 Theoretical Analysis

One can view our block coordinate selection approach in section 3.1 as a combinatorial mixture of experts problem [10], where each coordinate is a single expert and the forecaster aims at choosing the best combination of experts in each step. Under this view, we bound the regret of our selection method with respect to the policy of selecting the best (unknown) block of coordinates at each step.

Assume that there is a fixed optimal choice $\mathcal{I}^*$ for the block of coordinates to pick at all steps. This block is characterized by improving the objective function for the largest number of times among all the possible coordinate blocks when performing Bayesian optimization over the corresponding subspaces. The following particular design of losses expresses this cause:

$$\ell_{t,i} = \begin{cases} -\log(\tilde{\alpha}) & \text{if } i \in C_t \text{ and } y_t > M_{t-1} \\ \log(\tilde{\beta}) & \text{if } i \in C_t \text{ and } y_t \leq M_{t-1} \\ 0 & \text{if } i \notin C_t \end{cases} \quad ; \quad \tilde{\alpha}, \tilde{\beta} > 1 \tag{3}$$

as all the coordinates participating in the selected block incur the same loss that effectively rewards these coordinates for improving the objective and penalizes these for failing to improve the objective. All other coordinates that are not selected receive a zero loss and remain untouched.

Note that $\tilde{\alpha}$ and $\tilde{\beta}$ express the extent of reward and penalty, e.g. for $\tilde{\alpha} = \tilde{\beta} = e$ we have losses of $\ell_{t,i} \in \{-1, 1, 0\}$. Yet, $\tilde{\alpha}$ is better chosen to be larger than $\tilde{\beta}$, since the frequency of improving the objective is expected to be smaller.

The loss received by the forecaster is to reflect the same motivation. This is done by averaging the losses of the individual coordinates in the selected block, so that the size of the block does not matter explicitly, i.e. a bigger block should not incur more loss just due to its size but only due to its performance. Such that for each coordinate block $\mathcal{I}_t \subset \mathcal{I} = \{1, \cdots, D\}$ selected at time step t, the loss incurred by the forecaster is $L_{t,\mathcal{I}_t} = \frac{1}{|\mathcal{I}_t|} \sum_{i \in \mathcal{I}_t} \ell_{t,i}$. This is also the common loss incurred by all the coordinates participating in that block.

In each step we have the following multiplicative update rule of the weights associated with each coordinate (setting $\alpha = \tilde{\alpha}^\eta$ and $\beta = \tilde{\beta}^\eta$ yields the update rule in Eq. 2):

$$w_{t,i} = w_{t-1,i} \cdot e^{-\eta \ell_{t,i}} = w_{t-1,i} \cdot \begin{cases} \tilde{\alpha}^\eta & \text{if } i \in C_t \text{ and } y_t > M_{t-1} \\ 1/\tilde{\beta}^\eta & \text{if } i \in C_t \text{ and } y_t \leq M_{t-1} \\ 1 & \text{if } i \notin C_t \end{cases} \tag{4}$$

The probability $\tilde{\pi}_{t,\mathcal{I}_t}$ of selecting a certain coordinate block $\mathcal{I}_t$ is induced by π_t as specified next. Thus the expected cumulative loss of the foreceaster is:

$$L_T = \sum_{t=1}^{T} \sum_{c \in \mathcal{C}} \sum_{\mathcal{I}_t \in \mathcal{S}_c} \tilde{\pi}_{t,\mathcal{I}_t} \cdot \frac{1}{|\mathcal{I}_t|} \sum_{i \in \mathcal{I}_t} \ell_{t,i}$$

Assume the best coordinate block is $\mathcal{I}^*$, then the corresponding cumulative loss is:

$$L_T^* = \sum_{t=1}^{T} L_{t,\mathcal{I}^*} = \sum_{t=1}^{T} \frac{1}{|\mathcal{I}^*|} \sum_{i \in \mathcal{I}^*} \ell_{t,i}$$

We hence aim at bounding the regret $Regret_T = L_T - L_T^*$.

Theorem 1. *Sample from the combinatorial space of all possible coordinate blocks $\mathcal{I}_t \in \bigcup_{c\in\mathcal{C}} \mathcal{S}_c$ with probability $\tilde{\pi}_{t,\mathcal{I}_t} = \prod_{i\in\mathcal{I}_t} \tilde{w}_{t,\mathcal{I}_t} / \sum_{c\in\mathcal{C}} \sum_{\hat{\mathcal{I}}\in\mathcal{S}_c} \prod_{j\in\hat{\mathcal{I}}} \tilde{w}_{t,\hat{\mathcal{I}}}$. Then the update rule in Eq. 2 with $\alpha = \tilde{\alpha}^\eta$, $\beta = \tilde{\beta}^\eta$ and $\eta = \log(\tilde{\alpha}\tilde{\beta})^{-1}\sqrt{T^{-1}|\mathcal{C}|D\log(D)}$ yields:*

$$Regret_t \le \mathcal{O}\left((\log(\tilde{\alpha}\tilde{\beta}) \cdot \sqrt{T|\mathcal{C}|D\log(D)}) \right) \tag{5}$$

where $\tilde{w}_{t,\mathcal{I}_t} = \prod_{i\in\mathcal{I}_t} w_{t,i}^{1/|\mathcal{I}_t|}$ is the geometric mean of weights in block $\mathcal{I}_t$. The upper bound in Eq. 5 is tight, as the lower bound can be shown to be of $\Omega(\sqrt{T\log(N)})$ [23] where the number of experts is $N = \sum_{c\in\mathcal{C}} \mathcal{S}_c \le D^{|\mathcal{C}|D}$ in our combinatorial setup, as typically $|\mathcal{C}| \ll D$.

In practice, the direct sampling policy introduced in Theorem 1 involves high computational costs due to the exponential growth of combinations in D. Thus CobBO suggests an alternative computationally efficient sampling policy with a linear growth in D.

Theorem 2. *Sample a block size $c \in \mathcal{C}$ with probability p_c and c coordinates without replacement according to π_t. Assume $\mathcal{C} \supset \{1\}$, then the update rule in Eq. 2, with $\alpha = \tilde{\alpha}^\eta$, $\beta = \tilde{\beta}^\eta$ and $\eta = \sqrt{\frac{\log(D)}{T(\log(\tilde{\alpha}\tilde{\beta})^2 - \log(p_1))}} \ge 1$ yields:*

$$Regret_t \le \mathcal{O}\left(\sqrt{(\log(\tilde{\alpha}\tilde{\beta})^2 - \log(p_1))} \cdot \sqrt{T\log(D)}) \right) \tag{6}$$

where $p_c > 0$ for all $c \in \mathcal{C}$ and $\sum_{c\in\mathcal{C}} p_c = 1$, e.g., uniformly set $p_c \equiv |\mathcal{C}|^{-1}$.The proof and detailed sampling policy are in Appendix A. The regret upper bound in Eq. 6 is tight, as the lower bound for an easier setup can be shown to be of $\Omega(\sqrt{T\log(D)})$ [23]. The implication on η is valid only for settings of a very high dimensionality and low query budget. In particular, CobBO is designed for this kind of problems.

Remark: Similar analysis and results follow when incorporating consistent queries from Section 3.2 and sampling a new coordinate block once every several steps. This is done by effectively performing less steps of aggregated temporal losses, as shown in Appendix A.

4 Numerical Experiments

This section presents detailed ablation studies of the key components presented in Section 3 and comparisons with other algorithms.

4.1 Empirical analysis and ablation study

Ablation studies are designed to study the contributions of the key components in Algorithm 1 by experimenting with the Rastrigin function on $[-5, 10]^{50}$ with 20 initial points. The best performing run out of 5 experiments for each configuration is presented in Figure 3.

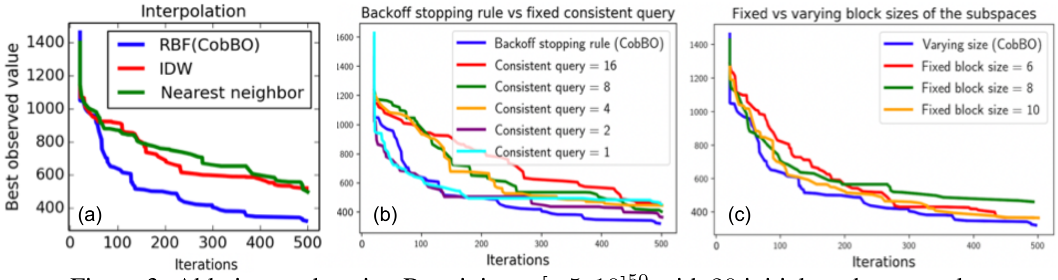

Figure 3: Ablation study using Rastrigin on $[-5, 10]^{50}$ with 20 initial random samples

RBF interpolation: RBF calculation is time efficient. Specifically, this is much beneficial in high dimensions. Figure 1 (left) shows the computation time of plain Bayesian optimization compared to CobBO's. While the former applies GP regression using the Matérn kernel in the high dimensional space directly, the later applies RBF interpolation in the high dimensional space and GP regression with the Matérn kernel in the low dimensional subspace. This two-step composite kernel leads to a significant speed-up. Other time efficient alternatives are, e.g., the inverse distance weighting [26] and the simple approach of assigning the value of the observed nearest neighbour. Figure 3 (a) shows that RBF is the most favorable.

Backoff stopping rule: CobBO applies a stopping rule to query a variable number of points in subspace Ω_t (Section 3.2). To validate its effectiveness, we compare it with schemes that use a fixed budget of queries for Ω_t. Figure 3 (b) shows that the stopping rule yields superior results.

Coordinate blocks of a varying size: CobBO selects a block of coordinates of a varying size C_t (Section 3.1). Figure 3 (c) shows that a varying size is better than fixed.

Preference probability over coordinates: For demonstrating the effectiveness of coordinate selection (Section 3.1), we artificially let the function value only depend on the first 25 coordinates of its input and ignore the rest. It forms two separate sets of active and inactive coordinates, respectively. We expect CobBO to refrain from selecting inactive coordinates. Figure 4 shows the entropy of this preference probability π_t over coordinates and the overall probability for picking active and inactive coordinate at each iteration. We see that the entropy decreases, as the preference distribution concentrates on the significant active coordinates.

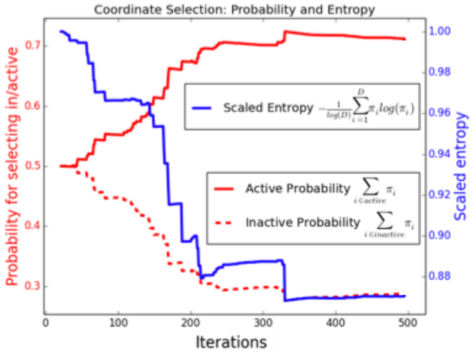

Figure 4: The preference probability focuses on active coordinates as the entropy decreases

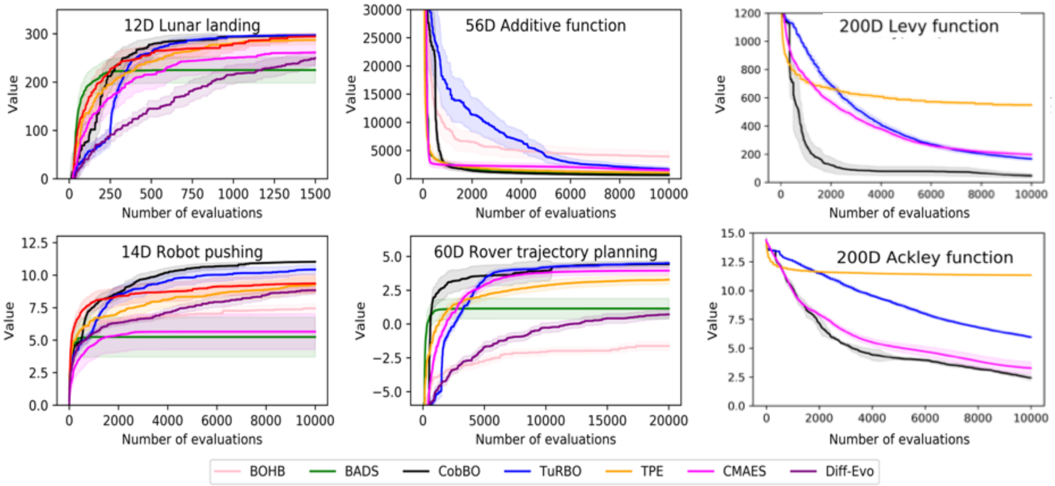

Figure 5: Performance over low (left) medium (middle) and high (right) dimensional problems

4.2 Comparisons with other methods

The default configuration for CobBO is specified in the supplementary materials. CobBO performs on par or outperforms a collection of state-of-the-art methods across the following experiments. Most of the experiments are conducted using the same settings as in TurBO [14], where it is compared with a comprehensive list of baselines, including BFGS, BOCK [47], BOHAMIANN, CMA-ES [22], BOBYQA, EBO [61], GP-TS, HeSBO [44], Nelder-Mead and random search. To avoid repetitions, we only show TuRBO and CMA-ES that achieve the best performance among this list, and additionally compare CobBO with BADS [1], REMBO [63], Differetial Evolution (Diff-Evo) [56], Tree Parzen Estimator (TPE) [4] and Adaptive TPE (ATPE) [12].

4.2.1 Low dimensional tests

To evaluate CobBO on low dimensional problems, we use the lunar landing [38, 14] and robot pushing [62], by following the setup in [14]. Confidence intervals (95%) over 30 independent experiments for each problem are shown in Fig. 5.

Lunar landing (maximization): This controller learning problem (12 dimensions) is provided by the OpenAI gym [38] and evaluated in [14]. Each algorithm has 50 initial points and a budget of $1,500$ trials. TuRBO is configured with 5 trust regions and a batch size of 50 as in [14]. Fig. 5 (upper left) shows that, among the 30 independent tests, CobBO quickly exceeds 300 along some good sample paths.

Robot pushing (maximization): This control problem (14 dimensions) is introduced in [62] and extensively tested in [14]. We follow the setting in [14], where TuRBO is configured with a batch size of 50 and 15 trust regions, each of which has 30 initial points. Each experiment has a budget of $10,000$ evaluations. On average CobBO exceeds 10 within 5500 trials, as shown in Fig. 5 (lower left).

4.2.2 High dimensional tests

Since the duration of each experiment in this section is long, confidence intervals (95%) over repeated 10 independent experiments for each problem are presented.

Additive latent structure (minimization): As mentioned in Section 2, additive latent structures have been exploited in high dimensions. We construct an additive function of 56 dimensions, defined as $f_{56}(x) = \mathrm{Ackley}(\mathrm{x}_1) + \mathrm{Levy}(\mathrm{x}_2) + \mathrm{Rastrigin}(\mathrm{x}_3) + \mathrm{Hartmann}(\mathrm{x}_4) + \mathrm{Rosenbrock}(\mathrm{x}_5) + \mathrm{Schwefel}(\mathrm{x}_6)$, where the first three terms express the exact functions and domains described in Section 4.2.1, the Hartmann function on $[0,1]^6$ and the Rosenbrock and Schwefel functions on $[-5,10]^{10}$ and $[-500,500]^{10}$, respectively.

We compare CobBO with TPE, BADS, CMA-ES and TuRBO, each with 100 initial points. Specifically, TuRBO is configured with 15 trust regions and a batch size 100. ATPE is excluded as it takes more than 24 hours per run to finish. The results are shown in Fig. 5 (upper middle), where CobBO quickly finds the best solution among the algorithms tested.

Rover trajectory planning (maximization): This problem (60 dimensions) is introduced in [62]. The objective is to find a collision-avoiding trajectory of a sequence consisting of 30 positions in a 2-D plane. We compare CobBO with TuRBO, TPE and CMA-ES, each with a budget of $20,000$ evaluations and 200 initial points. TuRBO is configured with 15 trust regions and a batch size of 100, as in [14]. ATPE, BADS and REMBO are excluded for this problem and the following ones, as they all take more than 24 hours per run. Fig. 5 (lower middle) shows that CobBO has a good performance.

The 200-dimensional Levy and Ackley functions (minimization): We minimize the Levy and Ackley functions over $[-5,10]^{200}$ with 500 initial points. TuRBO is configured with 15 trust regions and a batch size of 100. These two problems are challenging and have no redundant dimensions. For Levy, in Fig. 5 (upper right), CobBO reaches 100.0 within $2,000$ trials, while CMA-ES and TuRBO obtain 200.0 after $8,000$ trials. TPE cannot find a comparable solution within $10,000$ trials in this case. For Ackley, in Fig. 5 (lower right), CobBO reaches the best solution among all of the algorithms tested.

Regarding running times, for Ackley, CobBO runs for 12.8 CPU hours and TuRBO-1 run for more than 80 CPU hours or 9.6 *GPU* hours. Most other methods either cannot make any progress or find far worse solutions.

5 Conclusion

CobBO is a variant of coordinate ascent tailored for Bayesian optimization with a stopping rule to switch coordinate subspaces. The sampling policy of subspaces is proven to have tight regret bounds with respect to the best subspace in hindsight. Combining this projection on random subspaces with a two-stage kernels for function value interpolation and GP regression, we provide a practical Bayesian optimization method of affordable computational costs in high dimensions. Empirically, CobBO consistently finds comparable or better solutions with reduced trial complexity in comparison with the state-of-the-art methods across a variety of benchmarks.

References

[1] Luigi Acerbi and Wei Ji Ma. Practical bayesian optimization for model fitting with bayesian adaptive direct search. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1834–1844, Red Hook, NY, USA, 2017. Curran Associates Inc.

[2] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(6):121–164, 2012.

[3] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *J. Mach. Learn. Res.*, 3(null):397–422, Mar. 2003.

[4] James S. Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2546–2554. Curran Associates, Inc., 2011.

[5] Mickaël Binois, David Ginsbourger, and Olivier Roustant. On the choice of the low-dimensional domain for global optimization via random embeddings. *Journal of Global Optimization*, 76(1):69–90, January 2020.

[6] Martin D. Buhmann. *Radial Basis Functions: Theory and Implementations*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2003.

[7] Martin D. Buhmann and M. D. Buhmann. *Radial Basis Functions*. Cambridge University Press, USA, 2003.

[8] Adam D. Bull. Convergence rates of efficient global optimization algorithms. *Journal of Machine Learning Research*, 12(88):2879–2904, 2011.

[9] Roberto Calandra, André Seyfarth, Jan Peters, and Marc Peter Deisenroth. Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence*, 76(1):5–23, 2016.

[10] Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.

[11] Josip Djolonga, Andreas Krause, and Volkan Cevher. High-dimensional gaussian process bandits. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 1025–1033. Curran Associates, Inc., 2013.

[12] ElectricBrain. Blog: Learning to optimize, 2018.

[13] David Eriksson, Kun Dong, Eric Lee, David Bindel, and Andrew G Wilson. Scaling gaussian process regression with derivatives. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6867–6877. Curran Associates, Inc., 2018.

[14] David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable global optimization via local bayesian optimization. In *Advances in Neural Information Processing Systems 32*, pages 5496–5507. Curran Associates, Inc., 2019.

[15] Peter I. Frazier. A tutorial on bayesian optimization, 2018.

[16] Peter I. Frazier, Warren B. Powell, and Savas Dayanik. A knowledge-gradient policy for sequential information collection. *SIAM J. Control Optim.*, 47(5):2410–2439, Sept. 2008.

[17] Elad Gilboa, Yunus Saatçi, and John P. Cunningham. Scaling multidimensional Gaussian processes using projected additive approximations. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, page I–454–I–461. JMLR.org, 2013.

[18] Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D Sculley. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1487–1495, 2017.

[19] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

[20] Javier Gonzalez, Zhenwen Dai, Philipp Hennig, and Neil Lawrence. Batch bayesian optimization via local penalization. In Arthur Gretton and Christian C. Robert, editors, *Proceedings of the*

19th International Conference on Artificial Intelligence and Statistics, volume 51 of *Proceedings of Machine Learning Research*, pages 648–657, Cadiz, Spain, 09–11 May 2016. PMLR.

[21] Luigi Grippo and Marco Sciandrone. On the convergence of the block nonlinear gauss-seidel method under convex constraints. *Operations Research Letters*, 26(3):127–136, 2000.

[22] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, June 2001.

[23] David Haussler, Jyrki Kivinen, and Manfred K Warmuth. Tight worst-case loss bounds for predicting with expert advice. In *European Conference on Computational Learning Theory*, pages 69–83. Springer, 1995.

[24] Philipp Hennig and Christian J. Schuler. Entropy search for information-efficient global optimization. *J. Mach. Learn. Res.*, 13(1):1809–1837, June 2012.

[25] José Miguel Henrández-Lobato, Matthew W. Hoffman, and Zoubin Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'14, page 918–926, Cambridge, MA, USA, 2014. MIT Press.

[26] IDW. https://en.wikipedia.org/wiki/Inverse_distance_weighting.

[27] Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

[28] Kirthevasan Kandasamy, Jeff Schneider, and Barnabás Póczos. High dimensional bayesian optimization and bandits via additive models. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 295–304. JMLR.org, 2015.

[29] Johannes Kirschner, Mojmir Mutny, Nicole Hiller, Rasmus Ischebeck, and Andreas Krause. Adaptive and safe Bayesian optimization in high dimensions via one-dimensional subspaces. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3429–3438, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[30] H. J. Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86(1):97–106, mar 1964.

[31] Rémi Lam, Matthias Poloczek, Peter Frazier, and Karen E Willcox. Advances in bayesian optimization with applications in aerospace engineering. In *2018 AIAA Non-Deterministic Approaches Conference*, page 1656, 2018.

[32] Ben Letham, Roberto Calandra, Akshara Rai, and Eytan Bakshy. Re-examining linear embeddings for high-dimensional bayesian optimization. *Advances in Neural Information Processing Systems*, 33, 2020.

[33] Ben Letham, Roberto Calandra, Akshara Rai, and Eytan Bakshy. Re-examining linear embeddings for high-dimensional bayesian optimization. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1546–1558. Curran Associates, Inc., 2020.

[34] Benjamin Letham, Brian Karrer, Guilherme Ottoni, Eytan Bakshy, et al. Constrained bayesian optimization with noisy experiments. *Bayesian Analysis*, 14(2):495–519, 2019.

[35] Cheng Li, Sunil Gupta, Santu Rana, Vu Nguyen, Svetha Venkatesh, and Alistair Shilton. High dimensional bayesian optimization using dropout. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 2096–2102, 2017.

[36] Chun-Liang Li, Kirthevasan Kandasamy, Barnabas Poczos, and Jeff Schneider. High dimensional bayesian optimization via restricted projection pursuit models. In Arthur Gretton and Christian C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 884–892, Cadiz, Spain, 09–11 May 2016. PMLR.

[37] Daniel J Lizotte, Tao Wang, Michael H Bowling, and Dale Schuurmans. Automatic gait optimization with gaussian process regression. In *IJCAI*, volume 7, pages 944–949, 2007.

[38] LunarLander v2. https://gym.openai.com/envs/LunarLander-v2/.

[39] Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search of static linear policies is competitive for reinforcement learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 1800–1809. Curran Associates, Inc., 2018.

[40] Matern kernel. https://scikit-learn.org/stable/modules/generated/sklearn.gaussian_process.kernels.Matern.html.

[41] Mark McLeod, Michael A. Osborne, and Stephen J. Roberts. Optimization, fast and slow: Optimally switching between local and bayesian optimization. In *ICML*, 2018.

[42] J. Močkus. On bayesian methods for seeking the extremum. In G. I. Marchuk, editor, *Optimization Techniques IFIP Technical Conference Novosibirsk, July 1–7, 1974*, pages 400–404, Berlin, Heidelberg, 1975. Springer Berlin Heidelberg.

[43] Riccardo Moriconi, K. S. Sesh Kumar, and Marc Peter Deisenroth. High-dimensional bayesian optimization with projections using quantile gaussian processes. *Optimization Letters*, 14:51–64, 2020.

[44] Alexander Munteanu, Amin Nayebi, and Matthias Poloczek. A framework for Bayesian optimization in embedded subspaces. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4752–4761, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[45] Mojmir Mutny and Andreas Krause. Efficient high dimensional bayesian optimization with additivity and quadrature fourier features. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 9005–9016. Curran Associates, Inc., 2018.

[46] Julie Nutini, Mark Schmidt, Issam H. Laradji, Michael Friedlander, and Hoyt Koepke. Coordinate descent converges faster with the gauss-southwell rule than random selection. *ICML'15: Proceedings of the 32nd International Conference on International Conference on Machine Learning*, 37, July 2015.

[47] ChangYong Oh, Efstratios Gavves, and Max Welling. BOCK : Bayesian optimization with cylindrical kernels. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3868–3877, Stockholm, Sweden, 10–15 Jul 2018. PMLR.

[48] Rafael Oliveira, Fernando Rocha, Lionel Ott, Vitor Guizilini, Fabio Ramos, and Valdir Jr. Learning to race through coordinate descent bayesian optimisation. In *IEEE International Conference on Robotics and Automation (ICRA)*, February 2018.

[49] M.J.D. Powell. *On Search Directions for Minimization Algorithms*. AERE-TP. AERE, Theoretical Physics Division, 1972.

[50] Radial basis function. https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.Rbf.html.

[51] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1177–1184. Curran Associates, Inc., 2008.

[52] Akshara Rai, Rika Antonova, Seungmoon Song, William Martin, Hartmut Geyer, and Christopher Atkeson. Bayesian optimization using domain knowledge on the atrias biped. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1771–1778. IEEE, 2018.

[53] Warren Scott, Peter Frazier, and Warren Powell. The correlated knowledge gradient for simulation optimization of continuous parameters using gaussian process regression. *SIAM Journal on Optimization*, 21(3):996–1026, 2011.

[54] Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 1015–1022, Madison, WI, USA, 2010.

[55] 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(5):3250–3265, 2012.

[56] Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[57] Sonja Surjanovic and Derek Bingham. Optimization test problems, 2013.

[58] Hemant Tyagi and Volkan Cevher. Learning non-parametric basis independent models from point queries via low-rank methods. *Applied and Computational Harmonic Analysis*, 37(3):389 – 412, 2014.

[59] Emmanuel Vazquez and Julien Bect. Convergence properties of the expected improvement algorithm with fixed mean and covariance functions. *Journal of Statistical Planning and Inference*, 140(11):3088 – 3095, 2010.

[60] Linnan Wang, Rodrigo Fonseca, and Yuandong Tian. Learning search space partition for black-box optimization using monte carlo tree search. *ArXiv*, abs/2007.00708, 2020.

[61] Zi Wang, Clement Gehring, Pushmeet Kohli, and Stefanie Jegelka. Batched large-scale bayesian optimization in high-dimensional spaces. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018.

[62] Zi Wang, Clement Gehring, Pushmeet Kohli, and Stefanie Jegelka. Batched large-scale bayesian optimization in high-dimensional spaces. In *International Conference on Artificial Intelligence and Statistics*, pages 745–754, 2018.

[63] Ziyu Wang, Frank Hutter, Masrour Zoghi, David Matheson, and Nando De Freitas. Bayesian optimization in a billion dimensions via random embeddings. *J. Artif. Int. Res.*, 55(1):361–387, Jan. 2016.

[64] Zi Wang and Stefanie Jegelka. Max-value entropy search for efficient Bayesian optimization. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3627–3635, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[65] Zi Wang, Chengtao Li, Stefanie Jegelka, and Pushmeet Kohli. Batched high-dimensional bayesian optimization via structural kernel learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3656–3664. JMLR.org, 2017.

[66] Ziyu Wang, Masrour Zoghi, Frank Hutter, David Matheson, and Nando De Freitas. Bayesian optimization in high dimensions via random embeddings. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, page 1778–1784. AAAI Press, 2013.

[67] J. Warga. Minimizing certain convex functions. *Journal of the Society for Industrial and Applied Mathematics*, 11(3):588–593, 1963.

[68] Stephen J. Wright. Coordinate descent algorithms. *Mathematical Programming: Series A and B*, June 2015.

[69] Jian Wu and Peter I. Frazier. The parallel knowledge gradient method for batch bayesian optimization. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 3134–3142, Red Hook, NY, USA, 2016. Curran Associates Inc.

[70] Miao Zhang, Huiqi Li, and Steven Su. High dimensional bayesian optimization via supervised dimension reduction. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4292–4298. International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[71] Yichi Zhang, Daniel W Apley, and Wei Chen. Bayesian optimization for materials design with mixed quantitative and qualitative variables. *Scientific reports*, 10(1):1–13, 2020.

Broader Impact

As stated in [32], Bayesian optimization is a powerful optimization technique used in a wide range of industries and applications, such as robotics [37, 9, 52], internet tech companies [18, 34], designing novel molecules for pharmaceutics [19], material design for increasing efficiency of solar cells [71], and aerospace engineering [31]. All of these settings have high-dimensional optimization problems, and advances in BO will reflect on improved capabilities on these fields as well. We have fully open-sourced our code for CobBO using the MIT license to be available for researchers and practitioners in these fields, and many others. The ability to optimize a larger number of parameters than has previously been possible will bring further improvements to and further accelerate work in these areas.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or

[N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Section 3.
   (b) Did you describe the limitations of your work? [No]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Broader Impact.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 3.3.
   (b) Did you include complete proofs of all theoretical results? [Yes] See the Appendix A.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] In the supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Table 2 in the appendix, which contains the default hyperparameters.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Fig.5 and Fig.8-10.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See page 9, line 324.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [Yes] MIT license; see the appendix.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] See Section 4.2.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] It does not contain personal identifiable information or offensive content.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]