Two-stage Kernel Bayesian Optimization in High Dimensions

Jian Tan [*,1] **Niv Nayman** [*,2]

[*]Equal contribution
[1]Alibaba Group , Sunnyvale, California, USA
[2]Technion - Israel Institute of Technology , Haifa, Israel

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 heavy. While a complex kernel with many length scales is prone to overfitting and expensive to train, a simple coarse kernel with too few length scales cannot effectively capture the variations of the high dimensional function in different directions. To alleviate this problem, we introduce CobBO: a Bayesian optimization algorithm with two-stage kernels and a coordinate backoff stopping rule. It adaptively selects a promising low dimensional subspace and projects past measurements into it using a computational efficient coarse kernel. Within the subspace, the computational cost of conducting Bayesian optimization with a more flexible and accurate kernel becomes affordable and thus a sequence of consecutive observations in the same subspace are collected until a stopping rule is met. Extensive evaluations [a]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 both the trial complexity and computational costs.

[a]The full code: `https://github.com/Alibaba-MIIL/CobBO`

1 INTRODUCTION

Bayesian optimization (BO) is an effective zero-order paradigm for optimizing expensive black-box functions. It has been widely used in various real applications, e.g., parameter tuning for recommendation systems, automatic database configuration tuning, and simulation-based optimization.

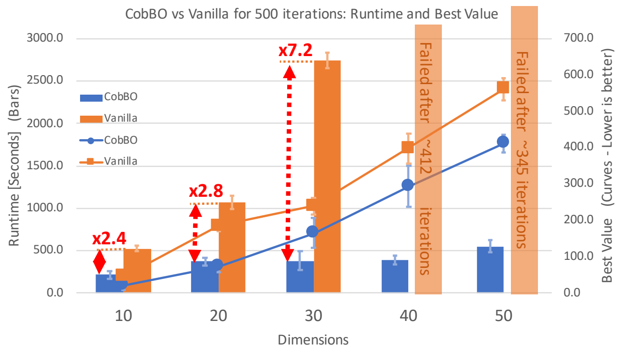

Figure 1: The measured runtime and best value (with their standard deviations averaged over 5 trials) of the Rastrigin function on $[-5, 10]^D$ observed by CobBO and vanilla BO with a budget of 500 iterations for $D = 10, 20, 30, 40, 50$. CobBO is much faster while obtaining better function values. In higher dimensions, vanilla BO cannot even complete the iteration budget (transparent bars for illustration only) while CobBO scales properly.

Though highly competitive in low dimensions (e.g., the dimension $D \leq 20$ Frazier [2018]), Bayesian optimization based on Gaussian Process (GP) regression has obstacles in high dimensions.
Curse of dimensionality: As a sample efficient method, Bayesian optimization often suffers from high dimensionality. Fitting the GP model (estimating the parameters, e.g., length scales Eriksson et al. [2019]) and optimizing the acquisition function all incur large computational costs in high dimensions. It also results in statistical insufficiency of exploration Djolonga et al. [2013], Wang et al. [2017]. As the GP regression's error grows with dimensions Bull [2011], more samples are required to balance that in high dimensions, which could cubically increase the computational costs in the worst case Mutny and Krause [2018].
Multiple length scales: The smoothness of the regression is determined by the specified kernel and the corresponding length scales, where the latter can be viewed as the measuring units along different axes in space. The landscapes

of the objective function over the global full space and on different local coordinate subspaces can vary significantly, while BO tries to approximate all of them in each iteration using a family of Gaussian functions. Thus, a single kernel with a fixed set of length scales cannot effectively fit all.

Algorithm 1 High level description of CobBO

```
 1: for each round $r$ do
 2:     Stage 1:
 3:     GP regression using a computation-efficient coarse
        kernel $K_1$ on all of the observed data points from the
        full space $\Omega$.
 4:     Select a subspace $\Omega_r$, project those data points into it
        and estimate their function values using $K_1$ to form
        “virtual points”.
 5:     Stage 2:
 6:     repeat
 7:         BO on the same subspace $\Omega_r$ with a more flexible
            and possibly computationally demanding kernel
            $K_2$, using both the “virtual points” and truly ob-
            served ones on $\Omega_r$.
 8:     until Backoff stopping rule is met
 9: end for
10: return the best observed data point
```

To alleviate this problem, we introduce CobBO: a Bayesian optimization algorithm with two-stage kernels and a coordinate backoff stopping rule, as illustrated in Algorithm 1. This method can be viewed as a variant of block coordinate ascent tailored to Bayesian optimization. During each round, a promising low dimensional subspace is restricted, following a theoretically motivated (Section 3.1) and empirically supported (Section 4.1) coordinate selection policy. To leverage information observed in all other subspaces, past data points in the full space are projected into the current subspace to form virtual points. In the first stage, their values are approximated using a simple coarse kernel that sacrifices the approximation accuracy for computational efficiency, e.g., RBF Buhmann [2003], for which efficient algorithms in $O(N \log N)$ for N observations have been studied Gumerov and Duraiswami [2007]. It captures the global landscape by smoothing away local fluctuations.

Then, in the second stage of the same round, a more flexible and possibly computation heavier kernel is used within the selected low dimensional subspace, as the computational cost of conducting Bayesian optimization therein becomes affordable. A possible choice is the Automatic Relevance Determination (ARD) Matérn Rasmussen and Williams [2005], which learns varying length scales to properly capture the local fluctuations in smaller selected subspaces. Then, a sequence of consecutive observations in the same subspace are collected. This refinement lasts until a stopping rule is met, determining when to back off from a certain subspace and switch to another.

This decoupling significantly reduces the computational burden in high dimensions, while fully leveraging the observations in the whole space rather than only relying on the few observations in each subspace. It can dramatically reduce both the model fitting time in the full space and the acquisition function optimization time in the subspace compared to performing ‘vanilla’ BO over the full space, as shown in Fig. 1.

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 embedding and additive structures. Their advantages manifest on problems with a low effective dimension. However, these assumptions do not necessarily always hold in practice, e.g., for non-separable functions without redundant dimensions.

Low dimensional embedding: The function f is assumed to have a low effective dimension Kushner [1964], Tyagi and Cevher [2014], e.g., $f(x) = g(\Phi x)$ for a function $g(\cdot)$ and a matrix Φ of $d \times D, d << D$. It essentially assumes that $f(x)$ does not change along certain directions. More generally, a non-linear auto-encoder can also be utilized to find the embedding. A variety of methods have been developed, including random embedding Djolonga et al. [2013], Wang et al. [2016], Munteanu et al. [2019], Binois et al. [2020], Letham et al. [2020], Hashing-enhanced Subspace BO (HeSBO) Munteanu et al. [2019], and Mahalanobis kernel ALEBO Letham et al. [2020]. Since not all the real-world problems fit the low dimensional embedding structure, CobBO is designed to optimize functions without redundant dimensions. It exploits the subspace structure, independent of the dimensions. Though the embedding-based algorithms and CobBO are based on different assumptions, REMBO Wang et al. [2016] and ALEBO Letham et al. [2020] are compared with CobBO in Appendix 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 Mutny and Krause [2018], which is usually small. The Gaussian process is structured as an additive model Gilboa et al. [2013], Kandasamy et al. [2015]. However, learning the unknown structure incurs a considerable computational cost Munteanu et al. [2019], and is not always applicable for non-separable functions, for which CobBO can still be applied.

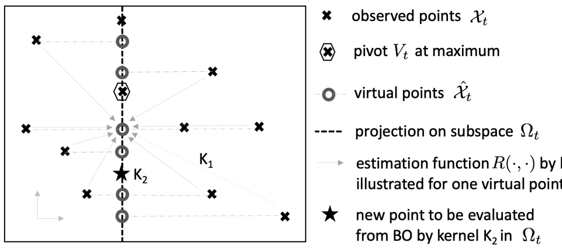
K1
K2
observed points $\mathcal{X}_t$
pivot V_t at maximum
virtual points $\hat{\mathcal{X}}_t$
projection on subspace Ω_t
estimation function $R(\cdot,\cdot)$ by kernel K1 illustrated for one virtual point
new point to be evaluated from BO by kernel K2 in Ω_t

Figure 2: An illustration of the two-stage kernels. Stage 1: subspace projection and function value estimation for virtual points using kernel K_1. Stage 2: BO in Ω_t using kernel K_2.

Trust regions and subspaces: Trust region BO has been proven effective for high-dimensional problems. Within the local trust regions, many efficient methods have been applied, e.g., local Gaussian models (TurBO Eriksson et al. [2019]), adaptive search on a mesh grid (BADS Acerbi and Ma [2017]) or quasi-Newton local optimization (BLOSSOM McLeod et al. [2018]). TurBO Eriksson et al. [2019] uses Thompson sampling to allocate samples across multiple regions. A related method is to use space partitions, e.g., LA-MCTS Wang et al. [2020] on a Monte Carlo tree search algorithm to learn efficient partitions. CobBO differs by selecting low dimensional subspaces and using two-stage kernels. Apart from the afore-mentioned works on axis-aligned subspaces Li et al. [2017], Oliveira et al. [2018], Moriconi et al. [2020], Eriksson and Jankowiak [2021], another closely related work is LineBO Kirschner et al. [2019]. It significantly reduces the acquisition function optimization time by restricting on one-dimensional subspaces. However, as it uses a single kernel, it does not address the computational issues of the GP regression in the full space. Furthermore, CobBO selects the block size as well as the coordinates therein by a multiplicative weights update method Arora et al. [2012] applied to the preference probability associated with each coordinate. Thus, it samples more promising subspaces with higher probabilities. See Appendix 2 for the comparison.

3 METHOD

Formally, suppose that the goal is to solve $x^* = \text{argmax}_{x\in\Omega} f(x)$ 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 t points $\mathcal{X}_t = \{x_1, x_2, \cdots, x_t\}$, we observe $\mathcal{W}_t = \{(x_i, y_i = f(x_i))\}_{i=1}^t$. A subset $C_t \subseteq I$ of the coordinates is selected, forming a subspace $\Omega_t \subseteq \Omega$.

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. For iteration t, instead of computing the Gaussian process posterior distribution $\{\hat{f}(x)|\mathcal{W}_t = \{(x_i, y_i)\}_{i=1}^t, x \in \Omega\}$ by conditioning on the observations $y_i = f(x_i)$ at queried points $\{x_i\}_{i=1}^t$ in the full space $\Omega \subset \mathbb{R}^D$, we change the conditional events, and consider $\{\hat{f}(x)|R(P_{\Omega_t}(x_1, \ldots, x_t), \mathcal{W}_t), x \in \Omega_t, \Omega_t \subset \Omega\}$ for a projection function $P_{\Omega_t}(\cdot)$ to a random subspace Ω_t and an estimation function $R(\cdot,\cdot)$. The projection $P_{\Omega_t}(\cdot)$ maps the queried points to virtual points on a subspace Ω_t of a lower dimension. The function $R(\cdot,\cdot)$ estimates means and variances of the objective values at the virtual points based on $\mathcal{W}_t$. The second stage uses a more flexible kernel within the subspace Ω_t, whose parameters would otherwise be expensive to learn in high dimensions.

As a variant of coordinate ascent, CobBO restricts the subspace Ω_t to contain a pivot point V_t, which is presumably the maximum point $x_t^M = \text{argmax}_{x\in\mathcal{X}_t} f(x)$ (or some perturbation over it to escape local optima) , whose function value is $M_t = f\left(x_t^M\right)$.

Then, BO is conducted within Ω_t, fixing all the other coordinates $\bar{C}_t = I \setminus C_t$, i.e., the complement of C_t.

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

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

where the acquisition function $Q(x|\mathcal{W}_t)$ incorporates the posterior distribution of the Gaussian processes $p(\hat{f}|\mathcal{W}_t)$. Typical acquisition functions include the expected improvement (EI) Močkus [1975], Jones et al. [1998], the upper confidence bound (UCB) Auer [2003], Srinivas et al. [2010], Srinivas et al. [2012], the entropy search Hennig and Schuler [2012], Henrández-Lobato et al. [2014], Wang and Jegelka [2017], and the knowledge gradient Frazier et al. [2008], Scott et al. [2011], Wu and Frazier [2016].

Instead of directly computing the posterior distribution $p(\hat{f}|\mathcal{W}_t)$, we replace the conditional events $\mathcal{W}_t$ by $\hat{\mathcal{W}}_t = R\left(P_{\Omega_t}(\mathcal{X}_t), \mathcal{W}_t\right) = \{(\hat{x}_i, \hat{y}_i)\}_{i=1}^t$ with a projection function $P_{\Omega_t}(\cdot)$,

$$[P_{\Omega_t}(x_i)]_j = \begin{cases} x_{i,j} & \text{if } j \in C_t \\ V_{t,j} & \text{if } j \notin C_t \end{cases} \quad ; \quad i \in \{1, \ldots t\} \quad (1)$$

at coordinate j. It simply keeps the values of x_t 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 $\hat{\mathcal{X}}_t = \{\hat{x}_1, \hat{x}_2, \hat{x}_3, \cdots, \hat{x}_{\hat{t}}\}$, $\hat{x}_i \in \Omega_t \,\forall\, 1 \leq i \leq \hat{t} \leq t$. In our implementation, the function values at $\hat{x}_i \in \hat{\mathcal{X}}_t$ are interpolated as $\hat{y}_i = R(\hat{x}_i, \mathcal{W}_t)$ using the standard radial basis function (RBF) kernel Buhmann [2003] $k_1(u, v) = \exp(-||u - v||^2/l^2)$, with a single length scale l, which is isotropic but easy to train. Multiple length scales in high dimensions can significantly increase the fitting time even though the time complexity is of the

Algorithm 2 Detailed description of CobBO(f, τ, T)

```
1: $\mathcal{W}_\tau \leftarrow$ Sample $t = \tau$ initial points and evaluate their values
2: $V_\tau, M_\tau \leftarrow$ The maximal point in $\mathcal{W}_\tau$
3: $q_\tau \leftarrow 0$ // Number of consecutive failed queries
4: $\pi_\tau \leftarrow$ Uniform preference for coordinates
5: **while** $t < T$ **do**
6:     **Stage 1**: Use a computation-efficient coarse kernel $K_1$ for the estimation function $R(\cdot,\cdot)$
7:     $\Omega_t \leftarrow$ Induce a new subspace over the coordinate block $C_t$, such that $V_t \in \Omega_t$
8:     $\hat{\mathcal{X}}_t \leftarrow P_{\Omega_t}(\mathcal{X}_t)$ // Form virtual points
9:     $\hat{\mathcal{W}}_t \leftarrow R\left(\hat{\mathcal{X}}_t, \mathcal{W}_t\right)$ // Compute the means (and optional variances) of the virtual points by $K_1$
10:    **Stage 2**: Use a flexible kernel $K_2$ for BO within $\Omega_t$ for consistent queries
11:    **repeat**
12:        $p\left[\hat{f}_{\Omega_t}(x)|\hat{\mathcal{W}}_t\right] \leftarrow$ Compute the posterior distribution of the Gaussian process in $\Omega_t$ conditional on $\hat{\mathcal{W}}_t$ by $K_2$; Note that the points in $\hat{\mathcal{W}}_t$ could have non-zero variances
13:        $x_{t+1} \leftarrow \text{argmax}_{x\in\Omega_t} Q_{\hat{f}\sim p(\hat{f}|\hat{\mathcal{W}}_t)}(x|\hat{\mathcal{W}}_t)$ // Keep using $K_2$
14:        $y_{t+1} = f(x_{t+1})$
15:        **if** $y_{t+1} > M_t$ **then**
16:            $V_{t+1} \leftarrow x_{t+1}$, $M_{t+1} \leftarrow y_{t+1}$, $q_{t+1} \leftarrow 0$
17:        **else**
18:            $V_{t+1} \leftarrow V_t$, $M_{t+1} \leftarrow M_t$, $q_{t+1} \leftarrow q_t + 1$
19:        **end if**
20:        Update $\pi_t$ according to Eq. (2)
21:        $\Omega_{t+1} \leftarrow \Omega_t$ // Remain in the same subspace for the next query
22:        $\mathcal{W}_{t+1} \leftarrow \mathcal{W}_t \bigcup \{(x_{t+1}, y_{t+1})\}$,
           $\mathcal{W}_{t+1} \leftarrow \mathcal{X}_t \bigcup \{x_{t+1}\}$,    $t \leftarrow t + 1$
23:    **until** The backoff stopping rule is met (Section 3.2) $\rightarrow$ Switch to a different subspace
24: **end while**
```

same order. Specifically, using a 'multiquadric' kernel with length scales approximating the average distance between points, CobBO can efficiently fit the model in the full space. Note that efficient algorithms for RBF in $O(N \log N)$ for N observations have been proposed Gumerov and Duraiswami [2007]. A possible choice for the second stage's kernel in subspace Ω_t is the Automatic Relevance Determination (ARD) Matérn kernel Rasmussen and Williams [2005]

$$k_2(u,v) = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\sqrt{2\nu}||d(u,v)||\right)^{\nu} K_\nu\left(\sqrt{2\nu}||d(u,v)||\right)$$

where $\Gamma(\cdot)$ is the gamma function, $K_\nu(\cdot)$ is a modified Bessel function ($\nu = 2.5$ twice differentiable), and $d(u,v) = ((u_1-v_1)/l_1, (u_2-v_2)/l_2, \cdots, (u_D-v_D)/l_D)$ with anisotropic length scales $l_1, \cdots, l_D$, that are more expensive to learn in high dimensions.

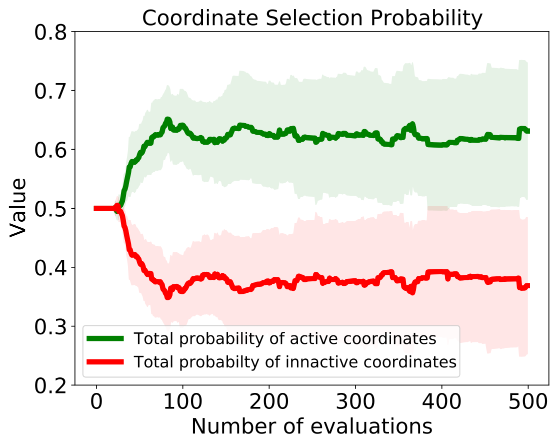

Figure 3: The preference probability concentrates on the 25 active coordinates of the Rastrigin function on $[-5, 10]^{25}$ (its summation in green), compared to the total probability assigned to 25 artificially added inactive coordinates (in red), that are ignored by the function. Mean values (solid lines) and 95% confidence intervals (shaded areas) over 10 independent experiments are presented.

3.1 STAGE 1: BLOCK COORDINATE ASCENT FOR SUBSPACE SELECTION

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 Arora et al. [2012]. 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,

$$w_{t,j} = w_{t-1,j} \cdot \begin{cases} \alpha & \text{if } j \in C_t \text{ and } y_t > M_{t-1} \\ 1/\beta & \text{if } j \in C_t \text{ and } y_t \leq M_{t-1} \\ 1 & \text{if } j \notin C_t \end{cases} \quad (2)$$

with $w_{0,j} = {}^1/_D$ and $\pi_{t,j} = {}^{w_{t,j}}/_{\sum_{j=1}^D w_{t,j}}$. 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 $y_t \leq M_{t-1}$.

While Fig. 3 and Section 4.1 provide an empirical support for the proposed block coordinate selection scheme, in Section 3.1.1, we provide a theoretical motivation for it.

While most existing methods partition the coordinates into fixed blocks and select one according to, e.g., cyclic

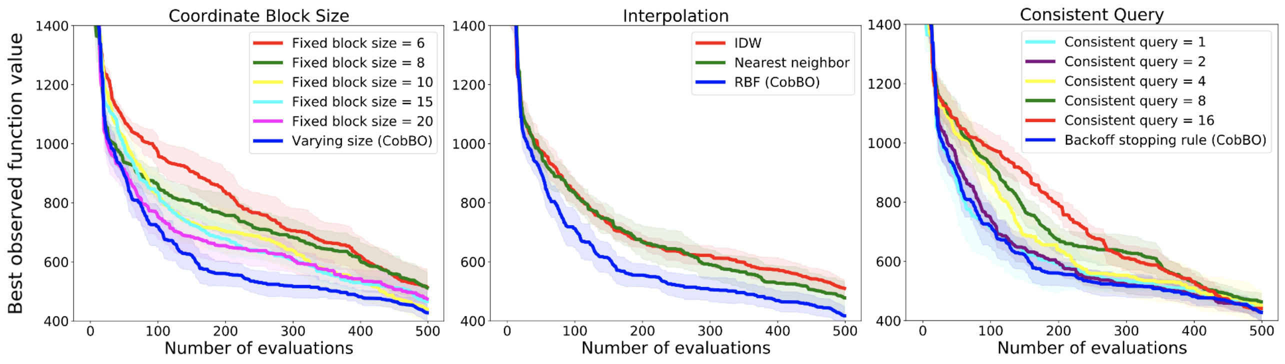

Figure 4: Ablation study over 5 trials using Rastrigin on $[-5, 10]^{50}$ with 20 initial random samples (lower is better)

order Wright [June 2015], random sampling or Gauss-Southwell Nutini et al. [July 2015], or selecting the size $|C_t|$, we specify an upper bound, e.g. $|C_t| \leq 30$, where $|C_t|$ can be any random number in a finite set $\mathcal{C}$. A sensitivity study for this upper bound appears in Appendix 6.

3.1.1 Theoretical motivation for the subspace selection

The selection of a block of coordinates can be viewed as a combinatorial mixture of experts problem, where each coordinate is a single expert and the forecaster aims at choosing the best combination of experts in each step Cesa-Bianchi and Lugosi [2006]. Under this view, we bound the regret of our selection method on an intuitive surrogate loss function with respect to the policy of selecting the best block of coordinates at each step. This is complementary to the regret analysis of the optimization performed at each subspace. Here we focus on justifying the coordinate selection alone.

Following the standard framework, we compare with 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. For any coordinate subset $\mathcal{A}$, we define the following loss function at time t, for coordinate i,

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

with $\tilde{\alpha}, \tilde{\beta} > 1$, where both y_t and M_{t-1} are fully determined by the previously selected coordinate subset $C_1, C_2, \cdots, C_{t-1}, C_t$. 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.

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 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

$$w_{t,i} = w_{t-1,i} \cdot e^{-\eta \ell_{t,i}(C_t; y_t, M_{t-1})} \tag{4}$$

$$= w_{t-1,i} \cdot \begin{cases} \tilde{\alpha}^\eta & \text{if } i \in C_t \text{ and } y_t > M_{t-1} \\ 1/\tilde{\beta}^\eta & \text{if } i \in C_t \text{ and } y_t \leq M_{t-1} \\ 1 & \text{if } i \notin C_t, \end{cases} \tag{5}$$

which, by setting $\alpha = \tilde{\alpha}^\eta$ and $\beta = \tilde{\beta}^\eta$, yields the update rule in Eq. (2).

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 forecaster 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 that 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 $\mathcal{R}_T = L_T - L_T^*$.

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 $\tilde{\pi}^c_{t,\mathcal{I}_t} = \prod_{i\in\mathcal{I}_t} \tilde{w}_{t,\mathcal{I}_t} / \sum_{c\in\mathcal{C}} \sum_{\hat{\mathcal{I}}\in\mathcal{S}_c} \prod_{j\in\hat{\mathcal{I}}} \tilde{w}_{t,\hat{\mathcal{I}}}$. 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*

$$\mathcal{R}_T \leq \mathcal{O}\left(\log(\tilde{\alpha}\tilde{\beta}) \cdot \sqrt{T|\mathcal{C}|D\log(D)}\right), \qquad (6)$$

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 the weights for block $\mathcal{I}_t$.

The upper bound in Eq. (6) is tight, as the lower bound can be shown to be of $\Omega(\sqrt{T\log(N)})$ Haussler et al. [1995] where the number of experts is $N = \sum_{c\in\mathcal{C}} \mathcal{S}_c \leq D^{|\mathcal{C}|D}$ 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 $\eta = \sqrt{\frac{\log(D)}{T(\log(\tilde{\alpha}\tilde{\beta})^2 - \log(p_1))}} \geq 1$ yields*

$$\mathcal{R}_T \leq \mathcal{O}\left(\sqrt{(\log(\tilde{\alpha}\tilde{\beta})^2 - \log(p_1))} \cdot \sqrt{T\log(D)}\right), \qquad (7)$$

where $p_c > 0$ for all $c \in \mathcal{C}$ and $\sum_{c\in\mathcal{C}} p_c = 1$.

The proof and detailed sampling policy are in appendix 4. The regret upper bound in Eq. 7 is tight, as the lower bound for an easier setup can be shown to be of $\Omega(\sqrt{T\log(D)})$ Haussler et al. [1995]. The implication on η is valid only for settings of a high dimension and low query budget. In particular, CobBO is designed for this kind of problems. 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 4.3.

3.2 STAGE 2: BACKOFF STOPPING RULE FOR CONSISTENT QUERIES

Note that only a fraction of the points in $\hat{\mathcal{X}}_t \cap \mathcal{X}_t$ directly observe the true function values. The function values on the rest ones in $\hat{\mathcal{X}}_t \backslash \mathcal{X}_t$ are estimated. For the trade-off between the inaccurate estimations and the exact observations in Ω_t, we design a stopping rule that determines the number of consistent queries in Ω_t. The more queries conducted in a given subspace, the more accurate the model therein, albeit at the expense of a smaller budget for exploring others.

For each iteration t, denote the relative improvement at iteration t by $\Delta_t = (y_t - M_{t-1})/|M_{t-1}|$. When looking backward in time from iteration t, we denote by P_t the number of consecutive improvements ($\Delta_s > 0, s \leq t$) and by N_t the total number of consecutive queries in the same subspace Ω_t. We set

$$C_{t+1} = \begin{cases} \text{Sample a new block} & \begin{matrix} N_t \geq \tau \text{ and } \Delta_t \leq 0.1 \\ \text{and } P_t \leq \xi \end{matrix} \\ \\ C_t & \begin{matrix} N_t < \tau \text{ or } \Delta_t > 0.1 \\ \text{or } P_t > \xi \end{matrix} \end{cases} \qquad (8)$$

where the values of the hyperparameters ξ and τ depend on the query budget T and the problem dimension D, as specified in Appendix 5. This heuristic stopping rule is robust to all the problems presented in this work and to many other that we have tested.

4 NUMERICAL EXPERIMENTS

This section presents detailed ablation studies of the key components and comparisons with other algorithms. The specifications of the testbed are as follows: Intel(R) Xeon(R) CPU E5-2682 v4 2.50GHz, Memory 32GB, GPU NVIDIA Tesla P100 PCIe 16GB.

4.1 ABLATION STUDY AND EMPIRICAL ANALYSIS

Ablation studies are designed to study the contributions of the key components in Algorithm 2 by experimenting with the Rastrigin function on $[-5, 10]^{50}$ with 20 initial points. Confidence intervals (95%) over 10 independent experiments for each configuration are presented in Fig. 4.

Coordinate blocks of a varying size: CobBO selects a block of coordinates C_t of a varying size, as described in Section 3.1. While CobBO is robust to the upper bound of the block size $|C_t|$, as shown in Appendix 6, Fig. 4 (left) shows that a varying size is better than a fixed one. Furthermore, although the average block size of CobBO is 15 in this setting, it enjoys both the fast exploration of larger block sizes (e.g. 22) and efficient exploitation of smaller block sizes (e.g. 6).

RBF interpolation in the first stage: RBF calculation is time efficient, which is beneficial in high dimensions. Fig. 1 (left) shows the computation time of plain Bayesian optimization compared to CobBO's. While the former applies the Matérn kernel in the high dimensional space directly, the later applies RBF interpolation in the high dimensional

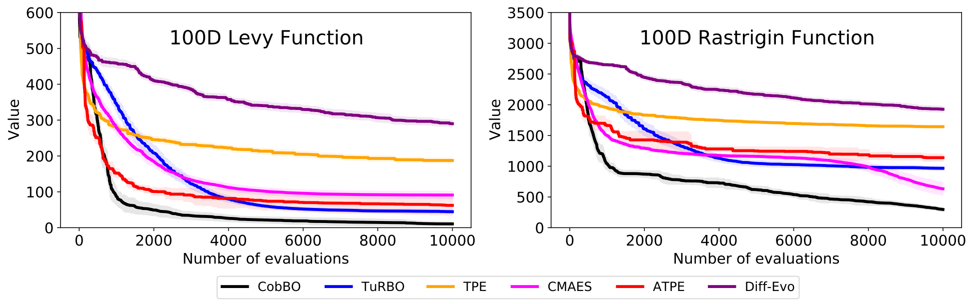

Figure 5: Performance (lower is better) over high dimensional synthetic problems: Levy (left) and Rastrigin (right)

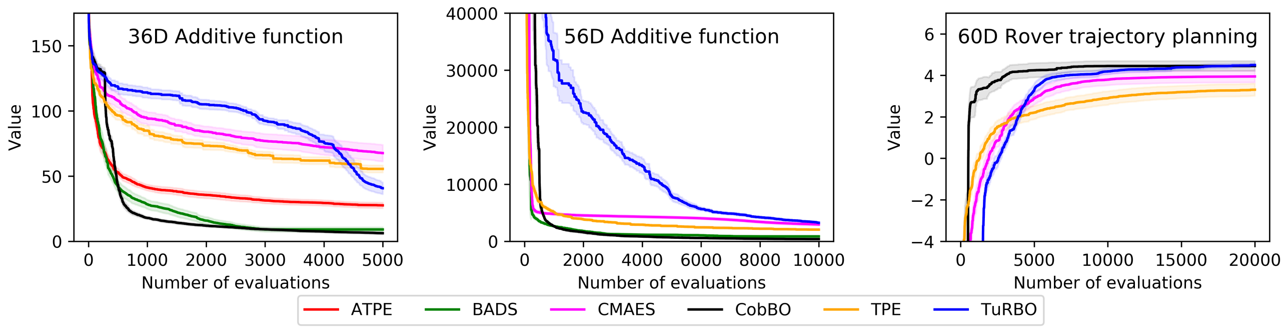

Figure 6: Performance over medium-size dimensional problems: 36D (left) and 56D (middle) additive functions (lower is better) and the 60D rover trajectory planning (right - higher is better)

space and the Matérn kernel in the low dimensional subspace. This two-stage kernel method leads to a significant speed-up. Other efficient alternatives are, e.g., the inverse distance weighting Shepard [1968] and the simple approach of assigning the value of the observed nearest neighbour. Fig. 4 (middle) shows that RBF is more 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. Fig. 4 (right) shows that the stopping rule yields superior results. Specifically, it enjoys both fast exploration of small query budget in each subspace (e.g. 1,2) and efficient exploitation of large ones (e.g. 16). Note that for different problems the best fixed number of consistent queries vary but the backoff stopping rule can adaptively achieve a good performance.

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. Fig. 3 shows the overall preference probability π_t for picking active ($\sum_{i=1}^{25} \pi_{t,i}$) and inactive coordinates ($\sum_{i=26}^{50} \pi_{t,i}$) at each iteration t. We see that the preference distribution concentrates on the active coordinates.

4.2 COMPARISONS WITH OTHER METHODS

A default configuration for CobBO is used for all of the experiments. CobBO performs on par or outperforms a collection of state-of-the-art methods. Most of the experiments are conducted using the same settings as in TurBO Eriksson et al. [2019], where it is compared with a comprehensive list of baselines, including BFGS, BOCK Oh et al. [2018], BOHAMIANN, CMA-ES Hansen and Ostermeier [2001], BOBYQA, EBO Wang et al. [2018], GP-TS, HeSBO Munteanu et al. [2019], 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 with BADS Acerbi and Ma [2017], Tree Parzen Estimator (TPE) Bergstra et al. [2011] and Adaptive TPE (ATPE) ElectricBrain [2018]. As mentioned in Section 2, the embedding algorithms (e.g., REMBO Wang et al. [2016] and ALEBO Letham et al. [2020]) and CobBO are based on different assumptions, which are compared in Appendix 1. Appendix 2 presents the comparison with LineBO Kirschner et al. [2019].

4.2.1 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 shown.

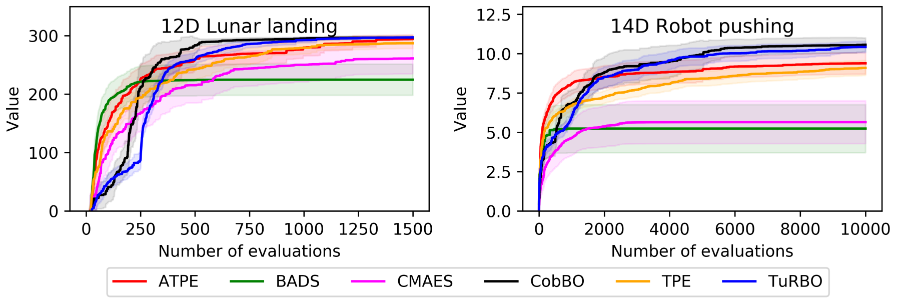

Figure 7: Performance (higher is better) over the lunar landing (left) and robot pushing (right) problems

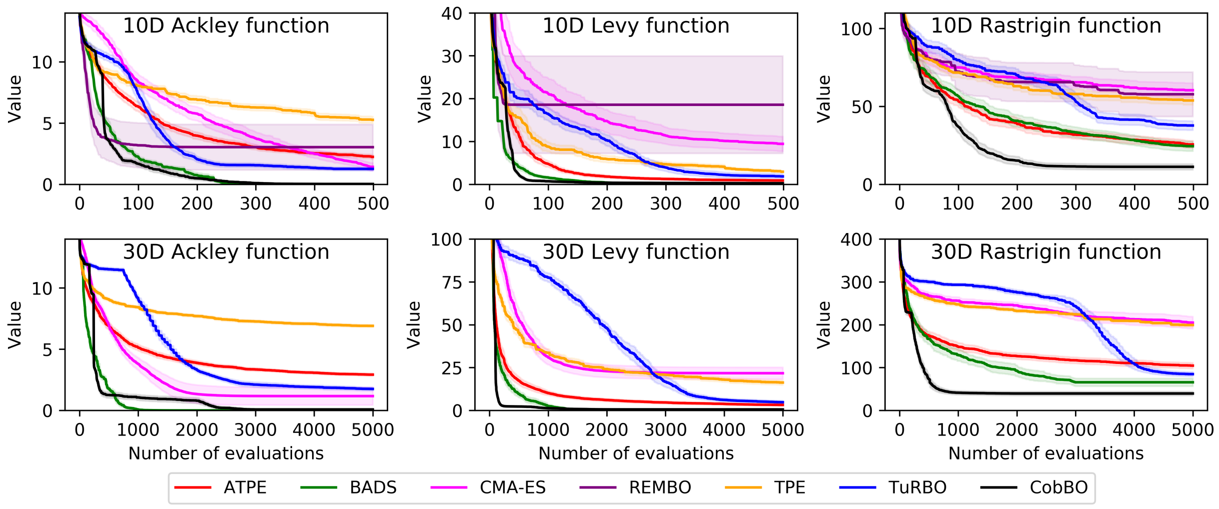

Figure 8: Performance on 10D (top) and 30D (bottom) synthetic functions: Ackley (left), Levy (middle) and Rastrigin (right)

The 100 dimensional synthetic black-box functions (minimization): We minimize the Levy and Rastrigin functions on $[-5, 10]^{100}$ with 300 initial points. These two problems are challenging since they have no redundant dimensions. TuRBO is configured with 1 trust regions and a batch size of 100. Fig. 5 (left) shows that CobBO can greatly reduce the trial complexity. For Levy and Rastrigin, CobBO surpasses the final solutions of all the other methods within $2,000$ and $5,000$ trials for a total budget of $10,000$ trials, respectively. REMBO is especially compared in Appendix 1.

In order to highlight the difference of the running time, we test Ackley 200D with $10,000$ trials. For a fair comparison, we change the configure so that both TurBO and CobBO have the same batch size of 1. CobBO runs for 12.8 CPU hours and TuRBO-1 runs for more than 80 CPU hours or 9.6 *GPU* hours. Other methods either take too long to make progress or find far worse solutions.

Additive latent structure (minimization): As mentioned in Section 2, additive latent structures have been explored for tackling challenges in high dimensions. We construct two additive functions. The first one has 36 dimensions, defined as $f_{36}(x) = \mathrm{Ackley}(\mathrm{x}_1) + \mathrm{Levy}(\mathrm{x}_2) + \mathrm{Rastrigin}(\mathrm{x}_3) + \mathrm{Hartmann}(\mathrm{x}_4)$, where the first three terms express the exact functions and domains described in Section 4.2.2, with the Hartmann function defiend over $[0, 1]^6$. The second has 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 four terms are the same as those of f_{36}, with the Rosenbrock and Schwefel functions defined over $[-5, 10]^{10}$ and $[-500, 500]^{10}$, respectively.

We compare CobBO with TPE, ATPE, BADS, CMA-ES and TuRBO, each with 100 initial points. Specifically, TuRBO is configured with 15 trust regions and a batch size 50 for f_{36} and 100 for f_{56}. ATPE is excluded for f_{56} as it takes more than 24 hours per run to finish. The results are shown in Fig. 6, where CobBO quickly finds the best solutions for both f_{36} and f_{56}.

As shown in Fig. 6, CobBO finds the best solutions for both f_{36} and f_{56}. BADS performs closely to CobBO. ATPE outperforms TPE, TuRBO and CMA-ES on f_{36}. TuRBO surpasses TPE and CMA-ES on f_{36} eventually, while TPE and CMA-ES converge faster than TuRBO on f_{56}.

Rover trajectory planning (maximization): This problem (60 dimensions) is introduced in Wang et al. [2018]. 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 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 Eriksson et al. [2019]. ATPE, BADS and REMBO are excluded for this problem, as they all last for more than 24 hours per run. The result is shown in Fig. 6. CobBO reaches the best solution with fewer evaluations than TuRBO, while TPE and CMA-ES reach inferior solutions.

4.2.2 Low dimensional tests

To evaluate the performance of CobBO on low dimensional problems, we use two challenging problems of lunar landing Eriksson et al. [2019] and robot pushing Wang et al. [2018], as well as classic synthetic black-box functions Surjanovic and Bingham [2013], by following the setup in Eriksson et al. [2019] for most of the experiments. Confidence intervals (95%) over repeated 30 independent experiments for each problem are shown.

Lunar landing (maximization): This controller learning problem (12 dimensions) is provided by the OpenAI gym and evaluated in Eriksson et al. [2019]. 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 Eriksson et al. [2019]. Fig. 7 shows that, among the 30 independent tests, CobBO quickly exceeds 300 along some good sample paths, outperforming other algorithms.

Robot pushing (maximization): This control problem (14 dimensions) is introduced in Wang et al. [2018] and extensively tested in Eriksson et al. [2019]. We follow the setting in Eriksson et al. [2019], where TuRBO is configured with a batch size of 50 and 15 trust regions with 30 initial points each. We exclude REMBO that takes too long per run (more than 24 hours). Each experiment has a budget of $10,000$ evaluations. On average CobBO exceeds 10.0 within 5,500 trials, while TuRBO requires about 7,000, as shown in Fig. 7. TPE and ATPE converge to around 9.0, outperforming BADS and CEM-ES with large margins. The latter two exhibit large variations and get stuck at local optima.

Classic synthetic black-box functions (minimization): Three popular synthetic functions (10 and 30 dimensions) are chosen, including Ackley over $[-5,10]^{10}$ and $[-5,10]^{30}$, Levy over both $[-5,10]^{10}$ and $[-5,10]^{30}$, and Rastrigin over both $[-3,4]^{10}$ and $[-3,4]^{30}$. TuRBO is configured identically the same as in Eriksson et al. [2019], with a batch size of 10 and 5 concurrent trust regions where each has 10 initial points. The other algorithms use 20 initial points. The results are shown in Fig. 8. CobBO shows competitive or better performance for all of these problems. It finds the global optima on Ackley and Levy, and clearly outperforms the other algorithms for the difficult Rastrigin function. Notably, BADS is more suitable for low dimensions, as commented in Acerbi and Ma [2017], which performs close to CobBO except on Rastrigin. TuRBO performs better than TPE and worse than BADS. ATPE outperforms TPE. CMA-ES eventually catches up with TPE, ATPE and REMBO on Ackley. For 10 dimensions, REMBO appears unstable with large variations and is trapped at local optima. For 30 dimensions, REMBO is excluded as it takes too long to finish; see Appendix 1.

5 LIMITATIONS AND FUTURE RESEARCH DIRECTIONS

While the crafted backoff stopping rule, introduced in Section 3.2, works well in practice and is robust to the many problems experimented with, it is build on pure heuristics. Deriving a more theoretically motivated role for switching subspaces might be a beneficial research direction. In addition, while the regret is analysed for the subspace selection scheme alone in Section 3.1.1, a unified regret analysis, that also includes the Bayesian optimization performed at each subspace, might provide a more complete picture of the method. Finally, considering that sparse Gaussian processes have been successfully used for Bayesian Optimization, e.g. in McIntire et al. [2016], in principle those can be used when performing Bayesian optimization in every subspace selected by CobBO, while filtering the projected virtual points properly. While this is out of the scope of this work, this is an interesting direction for future research.

6 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 the 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.

Acknowledgements

The authors would like to thank Mengchang Wang for conducting the experiments, and Rong Jin for helpful discussions. Niv Nayman would especially like to thank Lihi Zelnik-Manor for her tremendous support and guidance.

References

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. ISBN 9781510860964.

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.

Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *J. Mach. Learn. Res.*, 3(null): 397–422, March 2003. ISSN 1532-4435.

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.

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.

Martin D Buhmann. *Radial basis functions: theory and implementations*, volume 12. Cambridge university press, 2003.

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

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

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.

ElectricBrain. Blog: Learning to optimize, 2018. URL `https://www.electricbrain.io/post/learning-to-optimize`.

David Eriksson and Martin Jankowiak. High-dimensional bayesian optimization with sparse axis-aligned subspaces. In *the 37th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2021.

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.

Peter I. Frazier. A tutorial on bayesian optimization, 2018.

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, September 2008. ISSN 0363-0129.

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.

Nail A. Gumerov and Ramani Duraiswami. Fast radial basis function interpolation via preconditioned krylov iteration. *SIAM Journal on Scientific Computing*, 29(5):1876–1899, 2007.

Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, June 2001. ISSN 1063-6560.

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.

Philipp Hennig and Christian J. Schuler. Entropy search for information-efficient global optimization. *J. Mach. Learn. Res.*, 13(1):1809–1837, June 2012. ISSN 1532-4435.

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.

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.

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.

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.

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.

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.

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.

Mitchell McIntire, Daniel Ratner, and Stefano Ermon. Sparse gaussian processes for bayesian optimization. In *UAI*, 2016.

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

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. ISBN 978-3-540-37497-8.

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.

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.

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.

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.

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.

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.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN 026218253X.

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.

Donald Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM National Conference*, ACM '68, page 517–524, New York, NY, USA, 1968. Association for Computing Machinery. ISBN 9781450374866.

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.

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.

Sonja Surjanovic and Derek Bingham. Optimization test problems, 2013. URL `http://www.sfu.ca/~ssurjano/optimization.html`.

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. ISSN 1063-5203.

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.

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.

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.

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.

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, January 2016. ISSN 1076-9757.

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

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. ISBN 9781510838819.