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 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.

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 [Erikkson 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

*The full code: https://github.com/Alibaba-MIIL/CobBO
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
4:         kernel $K_1$ on all of the observed data points from the
5:         full space $\Omega$.
6:     Stage 2:
7:         repeat
8:             BO on the same subspace $\Omega_r$, with a more flexible
9:             and possibly computationally demanding kernel
10:                $K_2$, using both the “virtual points” and truly
11:                   observed ones on $\Omega_r$.
12:         until Backoff stopping rule is met
13:     end for
14: return the best observed data point
```

To alleviate this problem, we introduce CobBO: a Bayesian
optimization algorithm with two-stage kernels and a coor-
dinate backoff stopping rule, as illustrated in Algorithm 1.
This method can be viewed as a variant of block coordi-
nate ascent tailored to Bayesian optimization. During each
round, a promising low dimensional subspace is restricted,
following a theoretically motivated (Section 4.1) and em-
pirically 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 cur-
rent 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 [Williams 2005]. It captures the global landscape by smoothing away local fluctuations.

Then, in the second stage of the same round, a more flexi-
ble and possibly computation heavier kernel is used within
the selected low dimensional subspace, as the computa-
tional 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 pro-
cur the local fluctuations in smaller selected sub-
spaces. 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 ob-
servations 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 meth-
ods, for most of the problems tested in Section 4.2.

2 RELATED WORK

Certain assumptions are often imposed on the latent struc-
ture in high dimensions. Typical assumptions include low
dimensional embedding and additive structures. Their ad-
vantages manifest on problems with a low effective dimen-
sion. 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], [Iyagi
and Cevher 2014], e.g., $f(x) = g(\Phi x)$ for a function $g(\cdot)$
and a matrix $\Phi$ 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], [Bi-
nos et al. 2020], [Letham et al. 2020], Hashing-enhanced
Subspace BO (HeSBO) [Munteanu et al. 2019], and Maha-
lanobis 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.

**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 dimen-
sionality 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]. How-
ever, 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.
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, 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 $x^* = \arg\max_{x \in \Omega} f(x)$ for a black-box function $f : \Omega \to \mathbb{R}$. The domain is normalized $\Omega = [0, 1]^D$ with the coordinates indexed by $I = \{1, 2, \ldots, D\}$. For a sequence of $t$ points $X_t = \{(x_1, y_1 = f(x_1)), \ldots, x_t\}$, 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 $\{f(x) | \mathcal{W}_t = \{(x_i, y_i)\}_{i=1}^t, x \in \Omega\}$ by conditioning on the observations $y_t = f(x_t)$ at queried points $\{x_i\}_{i=1}^t$ in the full space $\Omega \subseteq \mathbb{R}^D$, we change the conditional events, and consider $\{f(x) | R(P_{C_t} x_1, \ldots, x_t), \mathcal{W}_t \}, x \in \Omega_t, \Omega_t \subseteq \Omega$ for a projection function $P_{C_t}(\cdot)$ to a random subspace $\Omega_t$ and an estimation function $R(\cdot, \cdot)$. The projection $P_{C_t}(\cdot)$ maps the queried points to virtual points on a subspace $\Omega_t$ 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 $\Omega_t$, whose parameters would otherwise be expensive to learn in high dimensions.

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

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

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

$$x_{t+1} = \arg\max_{x \in \Omega_t, \mathcal{W}_t \cup \{P(f, \mathcal{W}_t)(x) | \mathcal{W}_t\}} Q_{f_{\Omega_t}}(x) \sim p(f|\mathcal{W}_t)(x|\mathcal{W}_t),$$

where the acquisition function $Q(x|\mathcal{W}_t)$ incorporates the posterior distribution of the Gaussian processes $p(f|\mathcal{W}_t)$. Typical acquisition functions include the expected improvement (EI) [Mockus 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], Henández-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 $p(f|\mathcal{W}_t)$, we replace the conditional events $\mathcal{W}_t$ by $\mathcal{W}_t = \{(x_i, y_i)\}_{i=1}^t$ with a projection function $P_{C_t}(\cdot)$.

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

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 $P_{C_t}(\cdot)$ on $\mathcal{X}_t$ and discarding duplicates generate a new set of distinct virtual points $\hat{X}_t = \{\hat{x}_1, \hat{x}_2, \hat{x}_3, \ldots, \hat{x}_t\}$, $\hat{x}_i \in \Omega_t \forall 1 \leq i \leq t$. In our implementation, the function values at $\hat{x}_i \in \hat{X}_t$ are interpolated as $y_i = R(\hat{x}_i, \mathcal{W}_t)$ using the standard radial basis function (RBF) kernel [Buhmann 2003] $k_l(u, v) = \exp(-||u - v||^2/l^2)$, 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...
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 $\Omega$ is the Automatic Relevance Determination (ARD) Matérn kernel [Rasmussen and Williams 2005]

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

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

3.1 STAGE 1: BLOCK COORDINATE ASCENT FOR SUBSPACE SELECTION

We induce a preference distribution $\pi_t$ 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 $\pi_t$ 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.

$$w_{t,j} = w_{t-1,j} \cdot \begin{cases} \alpha & \text{if } j \in C_t \text{ and } y_t > M_{t-1} \\ \frac{1}{\beta} & \text{if } j \in C_t \text{ and } y_t \leq M_{t-1} \\ 1 & \text{if } j \not\in C_t \end{cases}$$

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 $\alpha$ is chosen to be relatively large, e.g., $\alpha = 2.0$, and $\beta$ 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 3.2 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
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 $C$. 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 $I^*$ 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 steps. This block is induced by $\pi_t$ as specified next. Thus the expected cumulative loss of the forecaster is:

$$L_T = \sum_{t=1}^{T} \sum_{c \in C} \sum_{I_t \in S_c} \pi_t(I_t) \cdot \frac{1}{|I_t|} \sum_{i \in I_t} \ell_{t,i}$$

Assume that the best coordinate block is $I^*$, then the corresponding cumulative loss is:

$$L^*_T = \sum_{t=1}^{T} L_t(I^*) = \sum_{t=1}^{T} \frac{1}{|I^*|} \sum_{i \in I^*} \ell_{t,i}$$

We hence aim at bounding the regret $R_T = L_T - L^*_T$. Yet, $\alpha$ is chosen to be larger than $\beta$, 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 $I_t \subset \mathcal{I} = \{1, \cdots, D\}$ selected at time step $t$, the loss incurred by the forecaster is $L_{t,I_t} = \frac{1}{|I_t|} \sum_{i \in I_t} \ell_{t,i}$. 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

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

$$w_{t-1,i} \cdot \begin{cases} \alpha & \text{if } i \in C_t \text{ and } y_t > M_{t-1} \\ \beta & \text{if } i \in C_t \text{ and } y_t \leq M_{t-1} \\ 1 & \text{if } i \notin C_t, \end{cases}$$

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

The probability $\pi_{t,I_t}$ of selecting a certain coordinate block $I_t$ is induced by $\pi_t$ as specified next. Thus the expected cumulative loss of the forecaster is:

$$L_T = \sum_{t=1}^{T} \sum_{c \in C} \sum_{I_t \in S_c} \pi_{t,I_t} \cdot \frac{1}{|I_t|} \sum_{i \in I_t} \ell_{t,i}$$

The probability $\pi_{t,I_t}$ of selecting a certain coordinate block $I_t$ is induced by $\pi_t$ as specified next. Thus the expected cumulative loss of the forecaster is:

$$L_T = \sum_{t=1}^{T} \sum_{c \in C} \sum_{I_t \in S_c} \pi_t(I_t) \cdot \frac{1}{|I_t|} \sum_{i \in I_t} \ell_{t,i}$$

Assume that the best coordinate block is $I^*$, then the corresponding cumulative loss is:

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

We hence aim at bounding the regret $R_T = L_T - L^*_T$. Yet, $\alpha$ is chosen to be larger than $\beta$, since the frequency of improving the objective is expected to be smaller.
Assume \(C \supset \{\) where the number of experts is \(N\). The proof and detailed sampling policy are in appendix 4. 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 \(N = \sum_{c \in C} \tilde{p}_c \leq D^{|C|}\) in our combinatorial setup, as typically \(|C| \ll D\).

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 \(c \in C\) with probability \(p_c\) and \(c\) coordinates without replacement according to \(\pi_t\). Assume \(C \supset \{\) with probability \(p_c\) and \(c\) coordinates without replacement according to \(\pi_t\). Then the update rule in Eq. (2) with \(\alpha = \tilde{\alpha}^n, \beta = \tilde{\beta}^n\) and \(\eta = \sqrt{(\log(\tilde{\alpha} \tilde{\beta}))^2 - \log(p_1)}\) \(\geq 1\) yields

\[
R_T \leq O\left(\sqrt{(\log(\tilde{\alpha} \tilde{\beta})^2 - \log(p_1))} \cdot \sqrt{T \log(D)}\right),
\]

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

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 \(\eta\) 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 \(\hat{X}_t \cap \mathcal{I}_t\) directly observe the true function values. The function values on the rest ones in \(\hat{X}_t \setminus \mathcal{I}_t\) are estimated. For the trade-off between the inaccurate estimations and the exact observations in \(\Omega_t\), we design a stopping rule that determines the number of consistent queries in \(\Omega_t\). 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 \(t\), denoted by \(P_t\) the number of consecutive improvements (\(\Delta_t > 0\), \(s \leq t\)) and by \(N_t\) the total number of consecutive queries in the same subspace \(\Omega_t\). We set

\[
C_{t+1} = \begin{cases} 
\text{Sample a new block} & N_t \geq \tau \text{ and } \Delta_t \leq 0.1 \\
C_t & N_t < \tau \text{ or } \Delta_t > 0.1 \\
& \text{or } P_t > \xi 
\end{cases}
\]

where the values of the hyperparameters \(\xi\) and \(\tau\) 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 Figure 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 \(|C_t|\), shown in Appendix 6, Figure 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. Figure 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
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 $\Omega_t$. To validate its effectiveness, we compare it with schemes that use a fixed budget of queries for $\Omega_t$. 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 $\pi_{t,i}$ 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 [Eriksen et al. 2019], where it is compared with a comprehensive list of baselines, including BFGS, BOCK [Oh et al. 2018], BOHAMIAN, 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.
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 $f_{36}(x) = \text{Ackley}(x_1) + \text{Levy}(x_2) + \text{Rastrigin}(x_3) + \text{Hartmann}(x_4)$, where the first three terms express the exact functions and domains described in Section 4.2.2, with the Hartmann function defined over $[0, 1]^6$. The second has 56 dimensions, defined as $f_{56}(x) = \text{Ackley}(x_1) + \text{Levy}(x_2) + \text{Rastrigin}(x_3) + \text{Hartmann}(x_4) + \text{Rosenbrock}(x_5) + \text{Schwefel}(x_6)$, 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 $f_{56}$ 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 [Srinovnic 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


