CobBO: Coordinate Backoff Bayesian Optimization

Jian Tan j.tan
Niv Nayman niv.nayman
Mengchang Wang mengchang.wmc
Rong Jin jinrong.jr
Alibaba Group (@alibaba-inc.com)

Abstract

Bayesian optimization is a popular method for optimizing expensive black-box functions. The objective functions of hard real world problems are oftentimes characterized by a fluctuated landscape of many local optima. Bayesian optimization risks in over-exploiting such traps, remaining with insufficient query budget for exploring the global landscape. We introduce Coordinate Backoff Bayesian optimization (CobBO) to alleviate those challenges. CobBO captures a smooth approximation of the global landscape by interpolating the values of queried points projected to randomly selected promising coordinate subspaces. Thus also a smaller query budget is required for the Gaussian process regressions applied over the lower dimensional subspaces. This approach can be viewed as a variant of coordinate ascent, tailored for Bayesian optimization, using a stopping rule for backing off from a certain subspace and switching to another coordinate subset. Additionally, adaptive trust regions are dynamically formed to expedite the convergence, and stagnant local optima are escaped by switching trust regions. Further smoothness and acceleration are achieved by filtering out clustered queried points. Through comprehensive evaluations over a wide spectrum of benchmarks, CobBO is shown to consistently find comparable or better solutions, with a reduced trial complexity compared to the state-of-the-art methods in both low and high dimensions.

An implementation of CobBO is available at:
`https://github.com/Alibaba-MIIL/CobBO`

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 $D \leq 20$ [20]), Bayesian optimization based on Gaussian process regression has obstacles that impede its effectiveness.

Approximation accuracy: Gaussian process regression assumes a class of random smooth functions in a probability space as surrogates that iteratively yield posterior distributions by conditioning on the queried points. However, 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 could misinform the global function landscape for suggesting new query points. Could inaccurate conditional events help?

Curse of dimensionality: As a sample efficient method, Bayesian optimization often suffers from high dimensions. Computing the Gaussian process posterior and optimizing the acquisition function incur large computational costs and statistical insufficiency of exploration in high dimensions [15, 68]. How to mitigate the dimensionality problem?

Computational growth with query budget: The runtime of each iteration for Gaussian process regression scales cubically in the number of queried points. The computational complexity could grow prohibitively high and prevent the usage beyond a limited query budget. It is possible to bring the complexity down to be quadratic by carefully handling the Cholesky factorization [5, 53], or even linear by assuming additive structures [46]. Nevertheless, these methods are not generally applicable for our purpose. Instead, we resort to approximate Gaussian process regression [54, 10], using less points to describe the prior.

Stagnation at local optima: It is well known that Bayesian optimization could stagnate at local optima [50, 11, 57]. The imbalance between exploration

and exploitation could result in queries being trapped without fully searching other promising regions.

To alleviate these issues, we design coordinate backoff Bayesian optimization (CobBO), by asking "*whether it is always advantageous to use accurate information at the queried points?* ". We demonstrate a negative possibility, by showing that smoothing out local fluctuations can help in capturing the large-scale properties of the objective function $f(x)$ for Bayesian optimization. The design principle is to capture the landscape of $f(x)$ through interpolation of points projected to coordinate subspaces for inducing designated Gaussian processes over those subspaces. 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

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

for a projection function $P_{\Omega_t}(\cdot)$ to a random subspace $\Omega_t \subset \Omega$ and a radial basis interpolation 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 effective dimension [52]. The interpolation function $R(\cdot, \cdot)$ estimates the objective values at the virtual points using the queried points and their values as specified by $\mathcal{H}_t$. Together, the random projection, the smooth interpolation and the Gaussian process regression in the subspace Ω_k form a composite random kernel.

This method can be viewed as a variant of block coordinate ascent, tailored to Bayesian optimization by applying backoff stopping rules. There are three issues to be addressed:

1. Selecting a block of coordinates for ascending requires determining the block size as well as the coordinates therein. The selected coordinates shall sample more promising regions with higher probabilities.
2. A coordinate subspace requires a sufficient amount of query points acting as the conditional events for the Gaussian process regression. Without enough points and corresponding values, the function landscape within a subspace cannot be well-characterized [11].
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.

To this end, we design a holistic framework for addressing those challenges. The coordinate subspaces are selected by using both gradient estimation and multiplicative weights update to the preference probabilities associated with the dimensions. With the subspaces, the stopping rules for consistent queries and backoffs determine the trade-off between exploration and exploitation. In addition, aiming at providing a practical solution with good performance, we also introduce the following auxiliary features in conjunction with the above key features. We dynamically form trust regions, similarly to [1, 24, 42, 19], and apply Bayesian optimization in both the full space and localized regions alternately. To expedite the computation, we use a simple K-means clustering method [39] to filter the points.

Through comprehensive evaluations, CobBO demonstrates appealing performance for both low and high dimensional problems, with a reduced trial complexity. For a similar setting, CobBO obtains better solutions with fewer query points, in comparison with the state-of-the-art methods for almost all the problems tested in Section 3. An implementation of CobBO will be made available upon publication.

1.1 Related work

Given the large body of related work on Bayesian optimization [20, 8, 56], we summarize the most relevant ones in four topics: high dimensionality, trust regions, hybrid methods and batch sampling.

High dimensionality: To apply Bayesian optimization in high dimensions, certain assumptions are often imposed on the latent structure. 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 [35, 62], 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 [69, 15, 66, 37, 45, 75, 7], low-rank matrix recovery [15, 62], and learning subspaces by derivative information [15, 18]. In contrast to existing work on subspace selections, e.g., LineBO [34], Hashing-enhanced Subspace BO (HeSBO) [45] and DROPOUT [36], CobBO exploits subspace structure from a perspective of block coordinate ascent, independent of the dimensions. As a result, it shows great performance in both high and low dimensions, different from some algorithms that are only 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 [46], which is usually small. The Gaussian process is structured as an additive model [23, 32], e.g., projected-additive functions [37], ensemble Bayesian optimization (EBO) [64], latent additive structural kernel learning (HDBBO) [68] and group additive models [32, 37]. However, learning the unknown structure incurs a considerable computational cost [45], and is not applicable for non-separable functions, for which CobBO can still be applied.

Kernel methods: Various kernels have been used for resolving the difficulties in high dimensions, e.g., a hierarchical Gaussian process model [12], a cylindrical kernel [48] and a compositional kernel [16]. The implementation of CobBO presented in this work uses the standard Gaussian process library of Sklearn [22] with an Automatic Relevance Determination (ARD) Matérn 5/2 kernel [41]. It can be integrated with other sophisticated methods [57, 16, 48, 12, 43, 31, 58, 21, 55, 67], e.g., ATPE/TPE [6, 17] and SMAC [29].

Trust regions: Trust region Bayesian optimization 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, more efficient methods are applied, e.g., local Gaussian models (TurBO [19]), adaptive search on a mesh grid (BADS [1]) or quasi-Newton local optimization (BLOSSOM [42]). TurBO [19] uses Thompson sampling to allocate samples across multiple regions. CobBO dynamically forms a variable-size trust region around the optimum of the already queried points.

Hybrid methods: Combining Bayesian optimization and other techniques yields hybrid approaches. Bayesian adaptive direct search (BADS) [1] alternates between local BO and grid search, which only fits low dimensions, as commented in [1]. Gradients can be used with BO, e.g., derivative-enabled knowledge gradient (d-KG) [74]. EGO-CMA [44] is combined with CMA-ES [26]. In this regard, CobBO can be used as an effective component to form more sophisticated hybrid methods.

Batch sampling: The leverage of distributed and parallel computation systems for black-box optimization requires the generation of batch queries. Popular methods designed for batch sampling include, e.g., Batch Upper Confidence Bound (BUCB) [14], combining UCB and Pure Exploration by entropy reduction (UCB-PE) [13], local penalization [24], determinantal point processes [33], Monte-Carlo simulation [4], sampling according to a reward function [14] and using the reparameterization trick for acquisition functions [71]. These methods can be combined with CobBO. In addition, due to the sampling of coordinate subspaces, CobBO can be also paralleled in a batch mode by sampling multiple different subspaces simultaneously.

2 Algorithm

Without loss of generality, suppose that the goal is to solve the maximization problem

$$x^* = \text{argmax}_{x\in\Omega} f(x)$$

for a black-box function $f : \Omega \to \mathbb{R}$ which is expensive to evaluate. 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 $\mathcal{H}_t = \{(x_i, y_i = f(x_i))\}_{i=1}^{t}$. As a variant of coordinate ascent, a random subset $C_t \subseteq I$ of the coordinates is selected, forming a subspace $\Omega_t \subseteq \Omega$ at iteration t. Then, within Ω_t, Bayesian optimization is conducted to inexactly maximize $f(x)$ while fixing all the other coordinates $C_t^c = I \setminus C_t$, i.e., the complement subset of C_t. For this purpose, we require the subspace Ω_t to contain a pivot point, V_t, presumably to be the maximum point $\text{argmax}_{x\in\mathcal{X}_t} f(x)$, with $M_t = f(V_t)$. However, when the number q_t of consecutive queries at iteration t that fail to improve over M_{t-1} becomes larger than a threshold Θ (e.g., $\Theta = 70$), we set V_t as a selected sub-optimal random point in $\mathcal{X}_t$ in order to escape a trapped local maxima. Specifically, we randomly sample a few points in $\mathcal{X}_t$ with their values at the top half and pick the one with the largest distance to V_{t-1}.

For conducting Bayesian optimization 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) [43, 31], the upper confidence bound (UCB) [3, 58, 59], the entropy search [27, 28, 67], or the knowledge gradient [21, 55, 73]. Based on those candidates, CobBO uses ensemble learning for the applied acquisition function. Specifically, we use a bandit approach to select the acquisition functions by measuring the number of queried points that improves the observed function values. In addition, for UCB, the upper confidence bound typically is constructed as

Algorithm 1: CobBO(f, τ, T)

$\mathcal{H}_\tau \leftarrow$ use a Latin hypercube sampling to get τ 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 Backoff stopping rule (Section 2.1) **then**
    if $q_t > \Theta$ **then**
      $V_t, M_t \leftarrow$ Escape local maxima [Sample at random several tuples from $\mathcal{H}_\tau$ whose values are above the median. Choose the furthest away from V_{t-1} (Section 2)]
      $q_t \leftarrow 0$
    $K_t \leftarrow$ Update a virtual clock (Eq. 4)
    $\tilde{\Omega}_t \leftarrow$ FormTrustRegions(K_t, y_t, M_{t-1}, q_t) (Algorithm 2)
    $C_t \leftarrow$ Sample a promising coordinate block according to π_t (Section 2.1)
    $\Omega_t \leftarrow$ Take the subspace of $\tilde{\Omega}_t$ over the coordinate block C_t, such that $V_t \in \Omega_t$
  else
    $\Omega_t \leftarrow \Omega_{t-1}$
  $\hat{\mathcal{X}}_t \leftarrow P_{\Omega_t}(\mathcal{X}_t)$ [Project $\mathcal{X}_t$ onto Ω_t to obtain a set of virtual points (Eq. 1)]
  $\hat{\mathcal{H}}_t \leftarrow R\left(\hat{\mathcal{X}}_t, \mathcal{H}_t\right)$ [Smooth function values on $\hat{\mathcal{X}}_t$ by RBF interpolation using $\mathcal{H}_t$ (Eq. 3)]
  $p\left[\hat{f}_{\Omega_t}(x)|\hat{\mathcal{H}}_t\right] \leftarrow$ Compute the posterior distribution of the Gaussian process conditional on $\hat{\mathcal{H}}_t$ within the subspace Ω_t
  $x_{t+1} \leftarrow \text{argmax}_{x\in\Omega_t} Q_{\hat{f}\sim p(\hat{f}|\mathcal{H}_t)}(x|\mathcal{H}_t)$ [Suggest the next query point in Ω_t (Section 2)]
  $y_{t+1} \leftarrow$ 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 π_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

$\mu + \kappa\sigma$ where μ and σ represent the estimated mean and variance, respectively. We choose the parameter κ as a periodical function of q_t so that κ varies with q_t within an interval, e.g., $\kappa \in [2.0, 4.0]$.

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

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

with a radial basis function (RBF) interpolation $R(\cdot,\cdot)$ detailed in Eq. 3, and a projection function $P_{\Omega_t}(\cdot)$,

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

at coordinate j, which 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. 1.

Applying $P_{\Omega_t}(\cdot)$ on $\mathcal{X}_t$ and abandoning duplicates generate a new set of distinct virtual points $\hat{\mathcal{X}}_t = \{\hat{x}_1, \hat{x}_2, \hat{x}_3, \cdots, \hat{x}_{\hat{t}}\}$ such that $\hat{x}_i \in \Omega_t \forall\, 1 \le i \le \hat{t} \le t$. The function values at $\hat{x} \in \hat{\mathcal{X}}_t$ are interpolated using the standard radial basis function, such that $\hat{y} = R(\hat{x}, \mathcal{H}_t)$ using the observed points in $\mathcal{H}_t$, see [51, 9]. Thus, RBF serves as an effective method for capturing the large-scale landscape of $f(x)$ by smoothing out the local fluctuations.

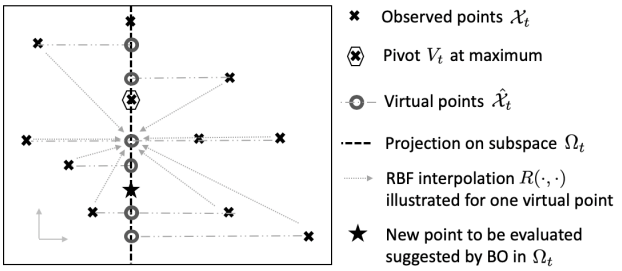

Figure 1: Illustration of subspace projection and function value interpolation

While CobBO involves several hyperparameters, extensive experiments demonstrate CobBO's robustness

to those as it achieves great performance across many tasks in Section 3 using the same default configuration.

2.1 Key features of the algorithm

The key features of CobBO are illustrated in Algorithm 1, with S and T denoting the number of initially sampled trials and the total number of trials, respectively. Note that the exact implementation varies, with the auxiliary features in Section 2.2 being combined with these key features.

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 [63] and the convergence rate can be characterized for smooth functions in the reproducing kernel Hilbert space [11]. 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 [70] 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 [25] that assumes $f(x)$ to be differentiable and strictly quasiconvex over a collection of blocks. In practice, we do not restrict ourselves to these assumptions.

In order to balance between exploitation and exploration, we alternate between two different approaches in selecting C_t. For the first approach that emphasizes exploitation, we estimate the top performing coordinate directions. A similar method is used in [40]. We select C_t to be the coordinates with the largest absolute gradient values of the RBF interpolated function at point V_t.

For the second approach that favors exploration, 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 increase in face of an improvement or decrease otherwise according to different multiplicative ratios α and β, respectively:

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

Then, π_t is normalized. This update characterizes how likely a coordinate block can generate a promising search subspace. The two multiplicative ratios are chosen to be relatively large, e.g., $\alpha = 1.2$ and $\beta = 1.8$, so that the most recent queries can impact the next chosen coordinate set C_{t+1} more influentially.

How to determine the size $|C_t|$? It is well-known that Bayesian optimization works well for low dimensions [20]. Thus, we specify an upper bound for the dimension size of the subspace (e.g. $|C_t| \leq 35$) except when the number of queried points contained in a localized trust region $\tilde{\Omega}_t$ (see section 2.2) is smaller than a threshold (e.g. 200), as for this small number of points the modest computational cost associated with the Gaussian process regression allows computations in higher dimensionality. In principle, $|C_t|$ can be any random number in this finite set. Empirically, we use a subset, e.g., $|C_t| \in \{2, 3, 5, 6, 9, 11, 13, 16, 19, 24, 27, 30, 35\}$. This is different from the common approach that partitions the coordinates into fixed blocks and selects one according to, e.g., cyclic order [72], random sampling or Gauss-Southwell [47].

The above method works well for low dimensions where $|C_t|/D$ is relatively large, as shown in Section 3.1. However, in high dimensions, $|C_t|/D$ could be small. In this case, instead of completely selecting at random according to π_t, also encourage cyclic order. With a certain probability p (e.g., $p = 0.3$), we select $|C_t|$ coordinates whose π_t values are the largest, and with probability $1 - p$, we randomly sample a coordinate subset according to the distribution π_t without replacement. Picking the coordinates with the largest values essentially implements a cyclic order, due to the selected weights update (Eq. 2) incurring probability oscillations. Since improvements tend to be less common than failures, the weights of the selected coordinates tend to decrease as the probability for choosing unselected coordinates increase in turn.

Backoff stopping rule for consistent queries: Applying Bayesian optimization in a subspace Ω_t requires a strategy to decide the number of consecutive queries to make 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. After a certain number of consecutive queries that fail to improve the objective, i.e.

$M_t = M_{t-1}$. This threshold is set higher with the query budget T and the problem dimension D. As switching to another subspace Ω_{t+1} ($\neq \Omega_t$) prematurely, without fully exploiting Ω_t, incurs an additional approximation error associated with the smooth interpolation of observations in Ω_t projected on to virtual points in Ω_{t+1}. In general, it is also possible to over-exploit a subspace, spending high query budget over some marginal improvements around a local optimum. In order to mitigate this, even when the a query leads to an improvement, other factors are considered for sampling a new subspace: its extent $M_t - M_{t-1}$, the point distance $||x_t - x_{t-1}||$, as well as the query budget T and the problem dimension D.

Smoothed interpolation: For Gaussian process regression conditional on $(\hat{x}, \hat{y}) \in \hat{\mathcal{H}}_t$, we need the corresponding function values $\{\hat{y}_i\}_{i=1}^t$, which however are unknown. These values are approximated by using RBF interpolation [51]. Specifically, for $i = 1, \ldots, t$ such that $\hat{x}_i \in \hat{\mathcal{X}}_t$ and $x_j \in \mathcal{X}_t$, we have,

$$\hat{y}_i = \sum\nolimits_{j=1}^{t} f(x_j)\, \phi(||\hat{x}_i - x_j||) \;;\; \hat{x}_j = P_{\Omega_t}(x_j) \quad (3)$$

Where $\phi(z) = \exp\left(-(z/\lambda)^2\right)$ is a Gaussian radial basis function, with λ approximating the average distance between points in $\mathcal{X}_t$. The distance $||\cdot||$ is chosen to be the Euclidean norm. When numerical stability issues are encountered, we use inverse distance weighting [30] to interpolate the function values by the corresponding nearest neighbors, weighted by the inverse of their mutual distances.

2.2 Auxiliary features of the algorithm

Both further smoothness and acceleration are achieved by filtering out clustered queried points, as alternating between adaptive trust regions promotes exploration in the interior of the domain and assists in escaping local optima.

Alternating between trust regions on two time scales: Trust regions have been shown to be effective in Bayesian optimization [19, 1, 24, 42]. Those are formed by shrinking the domain, e.g., by centering at V_t and halving the domain in each coordinate. We alternate between coarse and fine trust regions. Each is formed according to a slow and fast time scale, respectively. This brings yet another tradeoff between exploration and exploitation. Since sampled points tend to reside near the boundaries in high dimensions [48], inducing trust regions encourages sampling densely in the interior. However, aggressively shrinking those trust regions too fast around V_t can lead to an over-exploitation, getting trapped in a local optimum. Hence, to control this tradeoff, we alternate between two trust regions, following different time scales, as fast ones are formed inside slow ones. When the former allows fast exploitation of local optima, the latter avoids getting trapped in those.

The refinements of trust regions are triggered when a virtual clock K_t, characterizing the Bayesian optimization progress, reaches certain thresholds. Specifically,

$$K_{t+1} = \begin{cases} K_t + 1 & \text{if } \Delta_t \leq 0 \\ \gamma_t(\Delta_t, x_t, x_{t-1}) \cdot K_t & \text{if } 0 < \Delta_t \leq \delta \\ 0 & \text{if } \Delta_t > \delta \end{cases} \quad (4)$$

where $\Delta_t = \frac{M_t - M_{t-1}}{|M_{t-1}|}$ is the relative improvement and for example,

$$\gamma_t(\Delta, x_t, x_{t-1}) = \left(1 - \frac{\Delta}{\delta}\right) \cdot \left(1 - \frac{||x_t - x_{t-1}||}{\sqrt{|C_t|}}\right).$$

Starting from the full domain Ω, on a slow time scale, every time K_t reaches a threshold κ_S (e.g., $\kappa_S = 30$), a coarse trust region Ω_S is formed by further shrinking Ω as $K_t = 0$ Within the coarse trust region, on a fast time scale, when the number of consecutive fails exceeds a threshold κ_F ($< \kappa_S$, e.g, $\kappa_F = 6$), then a fine trust region is formed. In face of improvement, both the trust regions are enlarged back to the previous refinement of the coarse one. CobBO alternates between the two trust regions according to a duty cycle determined by τ_S and τ_F as specified by algorithm 2.

Algorithm 2: FormTrustRegions(K_t,y_t,M_{t-1},q_t)

Parameters:
    Slow/fast thresholds $\kappa_{S/F}$ respectively
    Slow/fast cycles $\tau_{S/F}$ respectively
Init: $\Omega_{S_0}, \Omega_{F_0} \leftarrow \Omega$
if $y_t > M_{t-1}$ **then**
    $\Omega_{S_t}, \Omega_{F_t} \leftarrow$ Double $\Omega_{S_{t-1}}$ around V_t
else if $mod(K_t, \kappa_S) = 0$ **then**
    $\Omega_{S_t} \leftarrow$ Halve $\Omega_{S_{t-1}}$ around V_t
else
    $\Omega_{S_t} \leftarrow \Omega_{S_{t-1}}$
if $mod(q_t, \kappa_F) = 0$ **then**
    $\Omega_{F_t} \leftarrow$ Halve $\Omega_{F_{t-1}}$ around V_t
if $mod(q_t, \tau_S + \tau_F) < \tau_S$ **then**
    $\tilde{\Omega}_t \leftarrow \Omega_{S_t}$
else
    $\tilde{\Omega}_t \leftarrow \Omega_{F_t}$
Output: Trust Region $\tilde{\Omega}_t$

Data filtering by K-means classification: Dealing with the cubic computation cost in queries [57], instead of using the sophisticated approximated Gaussian process regression [54, 10], above some quantity of aggregated observations, e.g. 1000, we leverage the K-means algorithm [39] for discarding clustered points.

Specifically, we only keep the point of maximal value within each cluster. Intuitively, if two nearby points have close function values, discarding the smaller one for a maximization problem seems innocuous. Sometimes, it could even be better, since Bayesian optimization assumes the function $f(x)$ to be smooth, from a reproducing kernel Hilbert space [11].

Batch queries: Due to sampling subspaces, CobBO can be easily paralleled in a batch mode. Specifically, we can sample multiple coordinate subspaces, each containing the latest observed pivot point V_t. Since the batch mode does not require synchronization, multiple concurrent subspaces may not necessarily use an identical V_t. In principle, we can integrate other batch methods [19, 14, 13, 24, 33, 4, 14, 71] with CobBO.

3 Numerical experiments

After tuning the hyper-parameters of CobBO over a number of commonly used benchmarks, we fix a default configuration for CobBO to be used across all of the experiments. The values are specified in the supplementary materials together with more experiments. Following the same default configuration, CobBO performs on par or outperforms a collection of state-of-the-art methods across the following experiments. This further demonstrates the robustness of CobBO. Most of the experiments are conducted using the same settings as in TurBO [19], where it is compared with a comprehensive list of baselines, including BFGS, BOCK [48], BOHAMIANN, CMA-ES [26], BOBYQA, EBO [64], GP-TS, HeSBO [45], 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 [66], Differetial Evolution (Diff-Evo) [60], Tree Parzen Estimator (TPE) [6] and Adaptive TPE (ATPE) [17]. The python code of the experiments and implementation of CobBO will be made publicly available.

3.1 Low dimensional tests

To evaluate the performance of CobBO on low dimensional problems, we use classic synthetic black-box functions [61], as well as two more challenging problems of lunar landing [38, 19] and robot pushing [65], by following the setup in [19] for most experiments. Confidence intervals (95%) over repeated 30 independent experiments for each problem are shown.

Classic synthetic black-box functions (minimization): Three popular synthetic 10 dimensional functions are chosen, including Ackley over $[-5, 10]^{10}$, Levy over $[-5, 10]^{10}$ and Rastrigin over $[-3, 4]^{10}$. TuRBO is configured identically the same as in [19], 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. 2. CobBO shows competitive or better performance for all of these problems. It finds the best optima on Ackley and Levy among all the algorithms and clearly outperforms those for the difficult Rastrigin function. Notably, BADS is more suitable for low dimensions, as commented in [1], and thus performs close to CobBO except for 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. REMBO appears unstable with large variations and is trapped in local optima.

Lunar landing (maximization): This controller learning problem (12 dimensions) is provided by the OpenAI gym [38] and evaluated in [19]. 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 [19]. Fig. 3 (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 [65] and extensively tested in [19]. We follow the setting in [19], 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, while TuRBO requires about 7000, as shown in Fig. 3 (lower left). TPE and ATPE converge to around 9, outperforming BADS and CEM-ES by large margins. The latter two exhibit large variations and get stuck in local optima.

3.2 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 the related work (section 1.1), additive latent structures have been explored for tackling challenges in high dimensions. We construct an additive function of 56 dimensions, defined as $f_{56}(x) = \text{Ackley}(\text{x}_1) + \text{Levy}(\text{x}_2) + \text{Rastrigin}(\text{x}_3) + \text{Hartmann}(\text{x}_4) + \text{Rosenbrock}(\text{x}_5) + \text{Schwefel}(\text{x}_6)$, where the first three terms express the exact functions and domains described in Section 3.1, the Hartmann function defiend over $[0, 1]^6$ and the Rosenbrock and Schwefel functions defined over $[-5, 10]^{10}$ 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

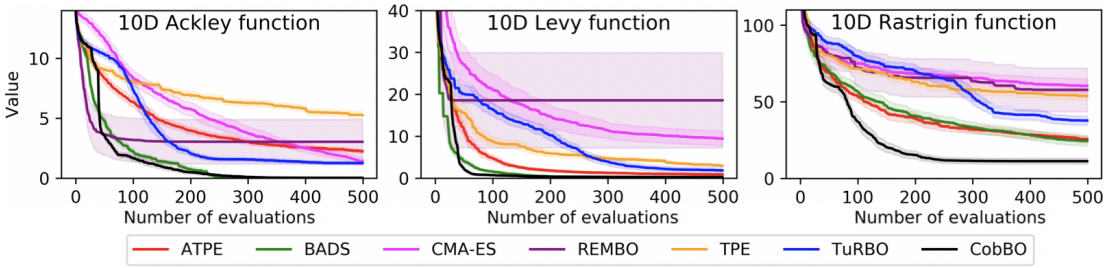

Figure 2: Performance over low dimensional problems: Ackley (left), Levy (middle) and Rastrigin (right)

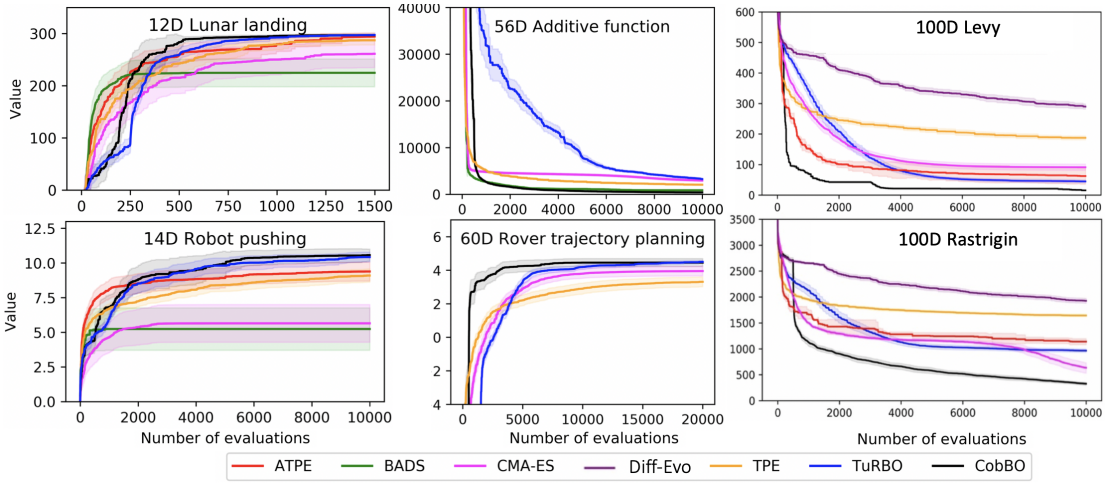

Figure 3: Performance over low (left) medium (middle) and high (right) dimensional problems

size 100. ATPE is excluded as it takes more than 24 hours per run to finish. The results are shown in Fig. 3 (upper middle), where CobBO quickly finds the best solution among the algorithms tested.

Rover trajectory planning (maximization): This problem (60 dimensions) is introduced in [65]. 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 [19]. ATPE, BADS and REMBO are excluded for this problem and the following ones, as they all last for more than 24 hours per run. The results are shown in Fig. 3 (lower middle). CobBO reaches the best solution among the tested algorithms faster than TuRBO, while TPE and CMA-ES reach inferior solutions.

The 100-dimensional Levy and Rastrigin functions (minimization): We minimize the Levy and Rastrigin functions over $[-5, 10]^{100}$ with 500 initial points. TuRBO is configured with 15 trust regions and a batch size of 100. As commented in [19], these two problems are challenging and have no redundant dimensions. Fig. 3 (right) shows that CobBO can dramatically reduce the trial complexity. For Levy, it quickly finds solutions close to the final one within $1,000$ trials, and eventually reach the best solution among all the algorithms tested. For Rastrigin, within $1,600$ trials CobBO surpasses the final solutions of all the other methods but CMA-ES, and within $3,000$ trials it surpasses also the final solution of CMA-ES, eventually with a large margin. TuRBO , TPE, ATPE and Diff-Evo [60] cannot find a competitive solution within $10,000$ trials. Furthermore, note that CobBO's sample variance for the Levy function across 10 independent experiments is extremely low, as can be seen in Fig. 3 (upper right).

4 Conclusion

CobBO is designed as a variant of coordinate ascent, tailored for Bayesian optimization. The number of consistent queries within each selected coordinate subspace is determined by a backoff stopping rule. Combining the key features of projection on random subspaces, function value interpolation and Gaussian process regression, we provide a practical Bayesian optimization method, with the performance further improved by auxiliary features, including trust regions alternation and data filtering. Empirically, using the same default configuration, 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] L. Acerbi and W. J. 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] S. Arora, E. Hazan, and S. Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(6):121–164, 2012.

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

[4] J. Azimi, A. Fern, and X. Z. Fern. Batch bayesian optimization via simulation matching. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 1*, NIPS'10, page 109–117, Red Hook, NY, USA, 2010. Curran Associates Inc.

[5] BayesOpt. `https://github.com/rmcantin/bayesopt`.

[6] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. 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.

[7] M. Binois, D. Ginsbourger, and O. 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.

[8] E. Brochu, V. M. Cora, and N. de Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *ArXiv*, abs/1012.2599, 2010.

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

[10] T. D. Bui, J. Yan, and R. E. Turner. A unifying framework for gaussian process pseudo-point approximations using power expectation propagation. *J. Mach. Learn. Res.*, 18(1):3649–3720, Jan. 2017.

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

[12] Z. Chen, S. Mak, and C. F. J. Wu. A hierarchical expected improvement method for bayesian optimization, 2019.

[13] E. Contal, D. Buffoni, A. Robicquet, and N. Vayatis. Parallel gaussian process optimization with upper confidence bound and pure exploration. In *Proceedings of the 2013th European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part I*, ECMLPKDD'13, page 225–240, Berlin, Heidelberg, 2013. Springer-Verlag.

[14] T. Desautels, A. Krause, and J. W. Burdick. Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization. *Journal of Machine Learning Research*, 15(119):4053–4103, 2014.

[15] J. Djolonga, A. Krause, and V. 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.

[16] D. Duvenaud, J. Lloyd, R. Grosse, J. Tenenbaum, and G. Zoubin. Structure discovery in nonparametric regression through compositional kernel search. In *International Conference on Machine Learning*, pages 1166–1174, 2013.

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

[18] D. Eriksson, K. Dong, E. Lee, D. Bindel, and A. 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.

[19] D. Eriksson, M. Pearce, J. Gardner, R. D. Turner, and M. Poloczek. Scalable global optimization via local bayesian optimization. In *Advances in Neural Information Processing Systems 32*, pages 5496–5507. Curran Associates, Inc., 2019.

[20] P. I. Frazier. A tutorial on bayesian optimization, 2018.

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

[22] Gaussian process regression. `https://scikit-learn.org/stable/modules/gaussian_process.html`.

[23] E. Gilboa, Y. Saatçi, and J. 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.

[24] J. Gonzalez, Z. Dai, P. Hennig, and N. Lawrence. Batch bayesian optimization via local penalization. In A. Gretton and C. 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.

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

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

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

[28] J. M. Henrández-Lobato, M. W. Hoffman, and Z. 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.

[29] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Proc. of LION-5*, page 507–523, 2011.

[30] Inverse distance weighting. https://en.wikipedia.org/wiki/Inverse_distance_weighting.

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

[32] K. Kandasamy, J. Schneider, and B. 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.

[33] T. Kathuria, A. Deshpande, and P. Kohli. Batched gaussian process bandit optimization via determinantal point processes. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 4213–4221, Red Hook, NY, USA, 2016. Curran Associates Inc.

[34] J. Kirschner, M. Mutny, N. Hiller, R. Ischebeck, and A. Krause. Adaptive and safe Bayesian optimization in high dimensions via one-dimensional subspaces. In K. Chaudhuri and R. 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.

[35] 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.

[36] C. Li, S. Gupta, S. Rana, V. Nguyen, S. Venkatesh, and A. 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.

[37] C.-L. Li, K. Kandasamy, B. Poczos, and J. Schneider. High dimensional bayesian optimization via restricted projection pursuit models. In A. Gretton and C. 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.

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

[39] J. MacQueen et al. Some methods for classification and analysis of multivariate observations, 1967.

[40] H. Mania, A. Guy, and B. 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.

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

[42] M. McLeod, M. A. Osborne, and S. J. Roberts. Optimization, fast and slow: Optimally switching between local and bayesian optimization. In *ICML*, 2018.

[43] 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.

[44] H. Mohammadi, R. Le Riche, and E. Touboul. Making ego and cma-es complementary for global optimization. In *International Conference on Learning and Intelligent Optimization*, pages 287–292. Springer, 2015.

[45] A. Munteanu, A. Nayebi, and M. Poloczek. A framework for Bayesian optimization in embedded subspaces. In K. Chaudhuri and R. 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.

[46] M. Mutny and A. 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.

[47] J. Nutini, M. Schmidt, I. H. Laradji, M. Friedlander, and H. 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.

[48] C. Oh, E. Gavves, and M. Welling. BOCK : Bayesian optimization with cylindrical kernels. In J. Dy and A. 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.

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

[50] C. Qin, D. Klabjan, and D. Russo. Improving the expected improvement algorithm. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5381–5391. Curran Associates, Inc., 2017.

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

[52] A. Rahimi and B. 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.

[53] R. Ram, S. Müller, F.-J. Pfreundt, N. R. Gauger, and J. Keuper. Scalable hyperparameter optimization with lazy gaussian processes, 2020.

[54] J. referencesQuiñonero Candela and C. E. Rasmussen. A unifying view of sparse approximate gaussian process regression. *J. Mach. Learn. Res.*, 6:1939–1959, Dec. 2005.

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

[56] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[57] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2951–2959. Curran Associates, Inc., 2012.

[58] N. Srinivas, A. Krause, S. Kakade, and M. 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.

[59] 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.

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

[61] S. Surjanovic and D. Bingham. `http://www.sfu.ca/~ssurjano/optimization.html`, 2013.

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

[63] E. Vazquez and J. 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.

[64] Z. Wang, C. Gehring, P. Kohli, and S. Jegelka. Batched large-scale bayesian optimization in high-dimensional spaces. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018.

[65] Z. Wang, C. Gehring, P. Kohli, and S. Jegelka. Batched large-scale bayesian optimization in high-dimensional spaces. In *International Conference on Artificial Intelligence and Statistics*, pages 745–754, 2018.

[66] Z. Wang, F. Hutter, M. Zoghi, D. Matheson, and N. De Freitas. Bayesian optimization in a billion dimensions via random embeddings. *J. Artif. Int. Res.*, 55(1):361–387, Jan. 2016.

[67] Z. Wang and S. Jegelka. Max-value entropy search for efficient Bayesian optimization. In D. Precup and Y. W. 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.

[68] Z. Wang, C. Li, S. Jegelka, and P. 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.

[69] Z. Wang, M. Zoghi, F. Hutter, D. Matheson, and N. 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.

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

[71] J. Wilson, R. Moriconi, F. Hutter, and M. P. Deisenroth. The reparameterization trick for acquisition functions. In *Proceedings of BayesOpt*, 2017.

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

[73] J. Wu and P. 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.

[74] J. Wu, M. Poloczek, A. G. Wilson, and P. Frazier. Bayesian optimization with gradients. In *Advances in Neural Information Processing Systems 30*, pages 5267–5278. Curran Associates, Inc., 2017.

[75] M. Zhang, H. Li, and S. 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.

Supplementary Materials

5 CobBO's default hyper-parameter configuration

Table 1 contains the default configuration of CobBO, which is used to test all the experiments in this paper.

Hyperparameter	Description	Default Value
Θ	The threshold for the number of consecutive fails q_t before changing V_t	70
α	Increase multiplicative ratio for the coordinate distribution update	1.2
β	Decay multiplicative ratio for the coordinate distribution update	1.8
p	Probability for selecting coordinates with the largest π_t values	0.3
κ_S	The slow threshold for the virtual clock value K_t before shrinking the coarse trust region Ω_S	30
κ_F	The fast threshold for the number of consecutive fails q_t before shrinking the fine trust region Ω_F	6
τ_S	The number of consecutive fails q_t in the course trust region Ω_S	24
τ_F	The number of consecutive fails q_t in the fine trust region Ω_F	6
δ	The relative improvement threshold governing the virtual clock update rule	0.1
	Gussian process kernel	Matern 5/2

Table 1: CobBO's hyperparameters configuration for all of the experiments

6 The Impact of the key/auxiliary features on the performance

6.1 Ablation of the backoff stopping rule and formation of trust regions

CobBO is configured with the default hyper-parameter configuration specified in section 5, including those governing a stopping rule for determining the number of consistent queries and the strategies to form coarse and fine trust regions on slow and fast time scales, respectively. In order to compare the impact of different configurations, we test the following combinations.

- Consistent query $\in$ {stopping rule, fixed constant $\mathrm{q}_{\max}$}
- $S \in \{\text{true}, \text{false}\}$, whether or not to employ coarse trust regions on a slow time scale
- $F \in \{\text{true}, \text{false}\}$, whether or not to employ refined trust regions on a fast time scale

The fixed constant $q_{\max}$ represents the maximum number of consistent queries that can be continuously imposed to the currently selected coordinate subspace. It induces a tradeoff between exploiting the potential of the current coordinate subspace and exploring other subspaces. When coarse trust regions are enabled on a slow time scale (i.e., S = true), the procedure exploits a neighborhood of V_t instead of the full domain. If fine trust regions are formed on a fast time scale (i.e., F = true), the Bayesian optimization better exploits the selected regions centered at V_t. The alternation between coarse and fine trust regions can help distributing new queries in both this centered area as well as near the boundary. We conduct extensive experiments to empirically demonstrate the contribution of these features to the performance of CobBO.

We apply CobBO on 30 dimensional synthetic functions (Ackley, Levy and Rastrigin) and the robot pushing problem using 5 different configurations, as shown in Table 2:

	CobBO^1	CobBO^2	CobBO^3	CobBO^4	CobBO^5
$q_{\max}$	stopping rule	stopping rule	stopping rule	1	15
S	false	true	false	true	true
F	false	false	true	true	true

Table 2: CobBO with different configurations

6.1.1 Ablation over 30 dimensional synthetic problems

We assign a budget of $2,500$ function evaluations to Ackley, Levy and Rastrigin, and $7,000$ function evaluations to the robot pushing problem. For each configuration, confidence intervals (95%) over repeated 30 independent experiments for each problem are shown. The tested value $q_{\max}$ is chosen to be 2 for $2,500$ function evaluations and 3 for $7,000$.

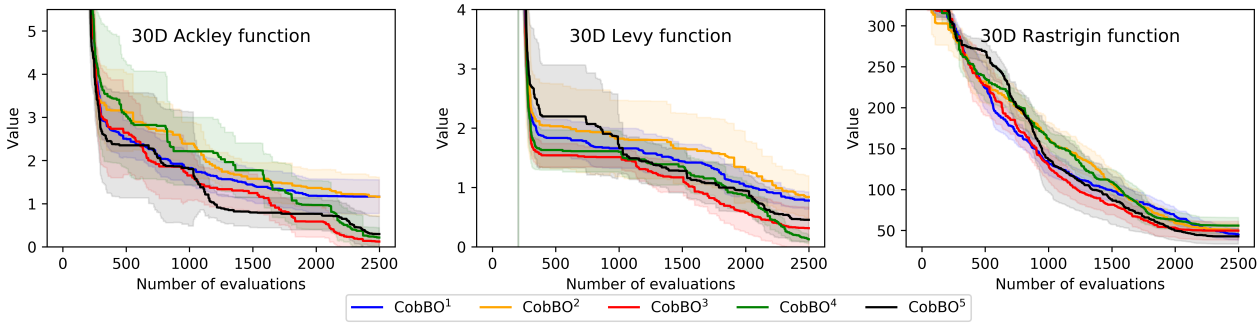

Figure 4: Performance of different configurations over synthetic problems of 30 dimensions: Ackley (left), Levy (middle) and Rastrigin (right)

CobBO^5, of a larger $q_{\max}$ value, performs slightly worse than CobBO^3 and CobBO^4, but better than CobBO^1 and CobBO^2. This implies that $q_{\max}$ and F have stronger impacts on the performance than S over the examined cases. When the fast trust region feature is enabled (F = true), CobBO^3 encourages more exploitation within smaller neighborhoods around the current best solutions, and consistently outperforms CobBO^1 and CobBO^2 on all three problems.

6.1.2 Ablation over the robot pushing problem

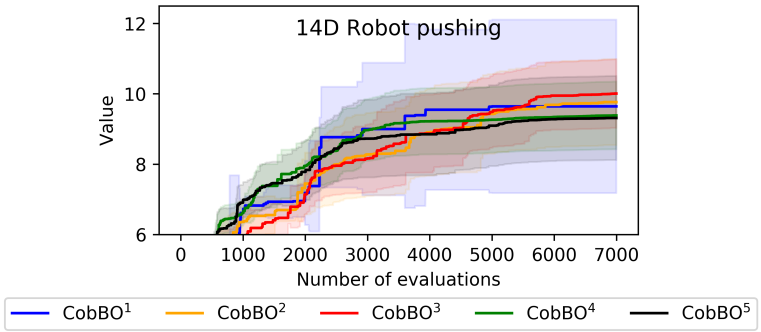

Figure 5: Performance of different configurations on the robot pushing problem

For the robot pushing problem, shown in Fig. 5, CobBO^3 slightly outperforms the rest on average, similar to the experiments shown in Fig 4. CobBO^5 performs badly, possibly due to its excessive exploitation of the selected coordinate subspaces. Different from the observations made in section 6.1.1, CobBO^1 and CobBO^2 find better solutions than CobBO^4 and CobBO^5 on average. This suggests that properly, and presumably adaptively,

balancing exploitation and exploration, e.g. through the formation of trust regions and the allocation of proper query budgets across selected subspaces, can impact the performance. The default configuration, detailed in table 1, includes fine trust regions. In this experiment, such configurations do not perform as well as CobBO^1 and CobBO^2. This indicates that better adaptive algorithms can be designed to further improve the performance of CobBO.

6.2 Ablation of escaping local maxima

CobBO is described in Algorithm 1, where Line 8 is about escaping local maxima by changing the pivot point V_t when the number of consecutive fails exceeds a threshold, i.e., $q_t > \Theta$. We use the experiments on Levy and Ackley functions of 100 dimensions, as described in section 3.2 to compute the fraction of queries that improve the already observed maximal points due to changing V_t according to Line 8.

Problem	Average # improved queries	Average # escaping queries
Ackley	228	15.3
Levy	155	3

Table 3: The number of improved queries due to escaping local maxima

We observe that optimizing the Levy function yields very few queries that improve the maximal points by changing the pivot, while optimizing the Ackley function can benefit more significantly from that.

7 Additional experiments

We provide more experiments for demonstrating the performance of CobBO. Confidence intervals (95%) are computed by repeating 30 independent experiments for each problem.

Medium-sized synthetic black-box functions (minimization): We test three synthetic functions (30 dimensions), including Ackley on $[-5, 10]^{30}$, Levy $[-5, 10]^{30}$, and Rastrigin on $[-3, 4]^{30}$. We also add experiments for an additive function of 36 dimensions, defined as $f_{36}(x) = \text{Ackley}(\text{x}_1) + \text{Levy}(\text{x}_2) + \text{Rastrigin}(\text{x}_3) + \text{Hartmann}(\text{x}_4)$, where the first three terms express the same functions over the same domains specified in Section 3.1 of this paper, with the Hartmann function over $[0, 1]^6$. TuRBO is configured identically the same as in Section 3.1, with a batch size of 10 and 5 trust regions with 10 initial points each. The other algorithms use 20 initial points. The results are shown in Fig. 6 and 7, where CobBO shows competitive or better performance compared to all of the methods tested across all of these problems.

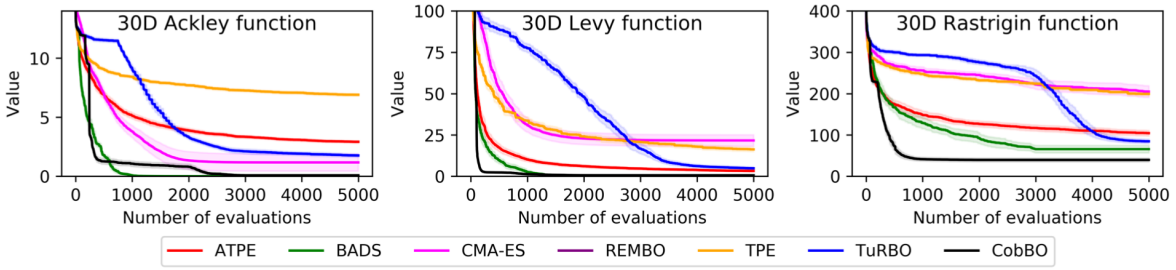

Figure 6: Performance over medium dimensional problems: Ackley (left), Levy (middle) and Rastrigin (right)

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 Ackley, Fig. 8 (left), CobBO reaches 4.0 within $1,800$ trials, while CMA-ES requires $7,000$ trials. TPE and TuRBO cannot find a comparable solution within $10,000$ trials. For Levy, Fig. 8 (right), CobBO quickly finds solutions close to its final one within $1,000$ trials, and eventually reaches the best solution among all of the algorithms tested. This shows that CobBO can dramatically reduce the trial complexity. The appealing trial complexity of CobBO suggests

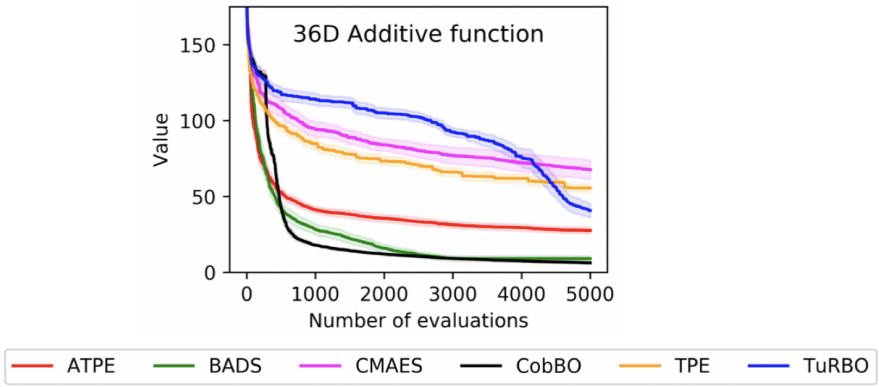

Figure 7: Performance over an additive function of 36 dimensions

that it can be applied in a hybrid method, e.g., used in the first stage of the query process when combined with gradient estimation methods or CMA-ES. Note that the variance for the Levy function across 10 independent experiments is very small, as shown in Fig. 9.

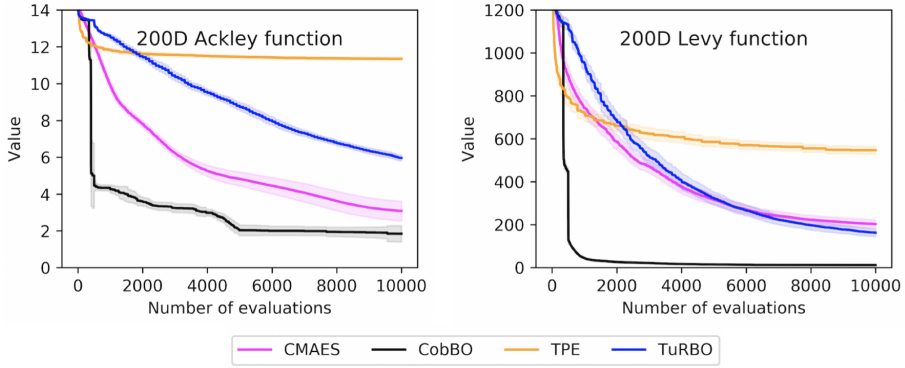

Figure 8: Performance over high dimensional synthetic problems: Ackley (left) and Levy (right)

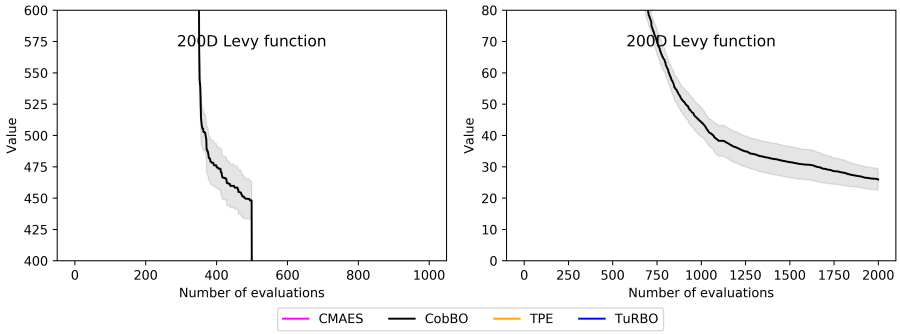

Figure 9: A closer look at the performance over the high dimensional synthetic Levy problem