Automated Random Embedding for Practical Bayesian Optimization with Unknown Effective Dimension

Hong Qian[1*] **Xiang Shu**[2], **Xiang Xia**[1], **Xuhui Liu**[3], **Yangde Fu**[1], **Bei Liang**[1], **Huibin Wang**[1] and **Liang Dou**[1]

[1]Shanghai Institute of AI for Education, and School of Computer Science and Technology, East China Normal University, Shanghai 200062, China

[2]Ant Group, Hangzhou 310013, China

[3]Nanjing University, Nanjing 210023, China

hqian@cs.ecnu.edu.cn, shuxiang.shu@antgroup.com, xxia@stu.ecnu.edu.cn, liuxh@lamda.nju.edu.cn, {ydfu, bliang, bingowang}@stu.ecnu.edu.cn, ldou@cs.ecnu.edu.cn

Abstract

Bayesian optimization is widely employed for optimizing complex black-box functions but struggles with the curse of dimensionality. Random embedding, as a dimension reduction strategy, simplifies tasks that possess the effective dimension by optimizing within a low-dimensional subspace. However, determining the effective dimension of a task in advance remains a significant challenge, which influences the selection of the subspace dimensionality and the optimization performance. Traditional methods use fixed subspace dimensions provided by experts or rely on trial and error to estimate subspace dimensions with resources consumed. To this end, this paper proposes an automated random embedding for high-dimensional Bayesian optimization with unknown effective dimension, called Dynamic Shared Embedding Bayesian Optimization (DSEBO). DSEBO starts with a low dimension and switches to a higher subspace if the solutions in the current subspace show preliminary convergence. DSEBO dynamically determines the dimension of the next subspace based on the quality of the solutions in different subspaces and shares the queried solutions with the new subspace for a better initialization. Theoretically, we derive a regret bound for DSEBO and demonstrate that DSEBO can better balance approximation and optimization errors. Extensive experiments on functions with dimensionality of varying magnitudes and real-world tasks with unknown effective dimensions reveal that, compared with state-of-the-art methods, alternating optimization across different subspaces results in significant improvements in high-dimensional optimization, both in terms of optimization regret and time.

1 Introduction

Optimization [Boyd and Vandenberghe, 2004] is widely used in various fields, such as economics [Mehta and Grosan, 2015], machine learning [Freund and Schapire, 1997; Elsken *et al.*, 2019], and reinforcement learning [Qian and Yu, 2021]. It aims to find the global optimal solution $\boldsymbol{x}^*$ of the objective function $f(\boldsymbol{x})$ in the D-dimensional search space $\mathcal{X} \subset \mathbb{R}^D$, formally expressed as $\boldsymbol{x}^* = \operatorname{argmin}_{\boldsymbol{x}\in\mathcal{X}\subset\mathbb{R}^D} f(\boldsymbol{x})$. However, in black-box optimization scenarios, the expression of the objective function is unknown [Shahriari *et al.*, 2016], where gradient-based optimization techniques are ineffective. In contrast, the derivative-free optimization methods [Zhou *et al.*, 2019; Shahriari *et al.*, 2016; Yu *et al.*, 2025] can handle these complex optimization problems.

Bayesian optimization (BO) [Srinivas *et al.*, 2010; Shahriari *et al.*, 2016; Garnett, 2023] is one of the most valuable derivative-free optimization methods for its excellent performance. However, the curse of dimensionality has persistently remained a critical issue in BO [Shahriari *et al.*, 2016; Wang *et al.*, 2016; Binois and Wycoff, 2022; Santoni *et al.*, 2024]. As the dimension increases, more evaluations of the objective function are required to adequately explore the solution space, making BO challenging to locate the optimal solution efficiently within finite computational resources.

Random embedding (RE) [Wang *et al.*, 2013; Wang *et al.*, 2016; Qian *et al.*, 2016] is an approach to addressing high-dimensional optimization tasks with effective dimension. By constructing a mapping between a lower-dimensional subspace and the original search space via an embedding matrix, optimization can be performed in the low-dimensional subspace, thus alleviating the scalability issue for BO methods. This approach exploits the inherent effective dimension of high-dimensional optimization tasks, where the function value is influenced by only a few relevant dimensions. By focusing on these dimensions, random embedding significantly enhances the efficiency and performance of high-dimensional BO within limited resources. However, a critical challenge is how to accurately determine the effective dimension. Selecting an appropriate low-dimensional subspace to perform optimization is crucial for the efficacy of embedding technique: a too-small subspace fails to adequately capture the effective dimension of the function, resulting in a loss of optimal solution, while a too-large subspace diminishes the advantages of the reduced dimensionality provided by random embedding.

*Corresponding Author.

However, previously in practice, the dimensionality of the low-dimensional subspace is typically determined via heuristic or trial-and-error approaches. These methods require performing multiple optimizations across various subspaces to achieve a more effective solution, which incurs significant computational costs. Currently, there is no direct method to predict the effective dimension required to capture the fundamental characteristics of the target function [Papenmeier *et al.*, 2022].

Problem. When using BO with RE for high-dimensional optimization tasks with unknown effective dimension, how to automatically determine the appropriate subspace dimension during optimization?

Contribution. To address the aforementioned problem, this paper proposes an effective approach to high-dimensional BO by developing a dynamic shared embedding Bayesian optimization (DSEBO) algorithm, which can automatically expand subspace dimension to handle tasks with unknown effective dimension. The shared embedding technique is introduced to leverage solutions from a low-dimensional subspace within a higher-dimensional subspace, thereby better guiding the initialization of the high-dimensional subspace and accelerating convergence. Based on the technique, DSEBO starts from a lower-dimensional subspace, dynamically determines the next subspace dimension according to the convergence of solutions in different subspaces, and facilitates transitions between various subspaces. The theoretical analysis derives a regret bound for DSEBO, highlighting its ability to balance approximation and optimization errors more effectively than GPUCB. Experiments on high-dimensional synthetic functions and real-world tasks show the effectiveness and superiority of DSEBO. The hyper-parameter experiments illustrate the robustness of DSEBO.

The subsequent sections present the related work and preliminaries, describe the proposed DSEBO method, show the theoretical and empirical results, and conclude the paper.

2 Related Work

This section reviews the related work: the high-dimensional optimization algorithms, and the multi-armed bandit (MAB) algorithms used for the subspace dimension selection.

2.1 High-Dimensional Optimization Algorithm

Embedding-based methods define a low-dimensional effective subspace where function values change dramatically, allowing optimization algorithms to operate within the subspace for lower computational costs. Linear embedding methods map low-dimensional points to high-dimensional space via an embedding matrix. Notable works include REMBO [Wang *et al.*, 2016], a foundational method, and SREBO [Qian *et al.*, 2016], which extends REMBO to sequential settings. HesBO [Nayebi *et al.*, 2019] uses hash matrices for linear mapping, SIRBO [Zhang *et al.*, 2019] employs sliced inverse regression to identify the effective dimension, and ALEBO [Letham *et al.*, 2020] provides a comprehensive analysis of the embedding process. Using nested random subspaces to dynamically increase the optimization space from a low dimensionality to a high dimensionality approaching D, BAxUS [Papenmeier *et al.*, 2022] can handle high-dimensional tasks. Nonlinear embedding methods use complex functions to map subspaces to the original space, like VAEBO [Gómez-Bombarelli *et al.*, 2018], which employs unsupervised learning to train variational encoders.

Apart from embedding-based methods, there are high-dimensional optimization methods that do not rely on subspace embedding. In the BO field, one method is TuRBO [Eriksson *et al.*, 2019], which handles high-dimensional tasks by dividing the optimization space into smaller regions for local optimization. SAASBO [Eriksson and Jankowiak, 2021] identifies and optimizes only on a few important dimensions. DuMBO [Bardou *et al.*, 2024] relaxes additive structure constraints by using decentralized message-passing and a refined acquisition function. MCTS-VS [Song *et al.*, 2022] employs Monte Carlo tree search to iteratively select a subset of variables and optimize them within a low-dimensional subspace. RDUCB [Ziomek and Bou-Ammar, 2023] uses random tree decompositions to build additive Gaussian process (GP) models, leveraging cycle-free pairwise-dimensional interactions for optimization. Recently, SBO-SE [Xu *et al.*, 2025] and VBO [Hvarfner *et al.*, 2024] introduce robust initialization strategies for the length-scale of GP kernel, addressing the vanishing gradient in high-dimensional BO. In the field of evolutionary algorithms, LMMAES [Loshchilov *et al.*, 2019] approximates the covariance with a small set of evolution paths. DCEM [Amos and Yarats, 2020] proposes a differentiable cross-entropy method using a smooth top-k operation.

All these methods rely on special assumptions, such as effective dimension. The failure to meet these underlying assumptions can lead to a significant decline in performance.

2.2 Multi-Armed Bandit

The problem of selecting subspace dimension can be modeled as an MAB problem. Specifically, different candidate dimensions are treated as different arms in the MAB problem, where the convergence value of the optimized function on these dimensions serves as the reward for each arm.

The MAB problem [Sutton and Barto, 2018; Jin *et al.*, 2022] centers on the trade-off between exploration and exploitation. For instance, Softmax [Sutton and Barto, 2018] employs the exponential probability rule for arm selection. Thompson Sampling (TS) [Jin *et al.*, 2022] is a probabilistic model-based approach that samples arms according to their posterior distributions. Additionally, other algorithms like Extreme, Random, ϵ-Greedy, Expectation, Upper Confidence Bound (UCB), and the UCB-E algorithm, which extends the traditional UCB algorithm, are also widely adopted methods in practice.

3 Preliminaries

This section briefly introduces Bayesian optimization (BO), the optimal ϵ-effective dimension, and random embedding to explain the necessary preliminary knowledge and notation.

Bayesian Optimization. BO [Shahriari *et al.*, 2016; Garnett, 2023] is a well-known derivative-free optimization method that strategically leverages prior knowledge to guide optimization. BO first constructs a surrogate model of the objective function, typically a Gaussian process, based on the observed dataset $\mathcal{D}$. With the model, BO estimates the

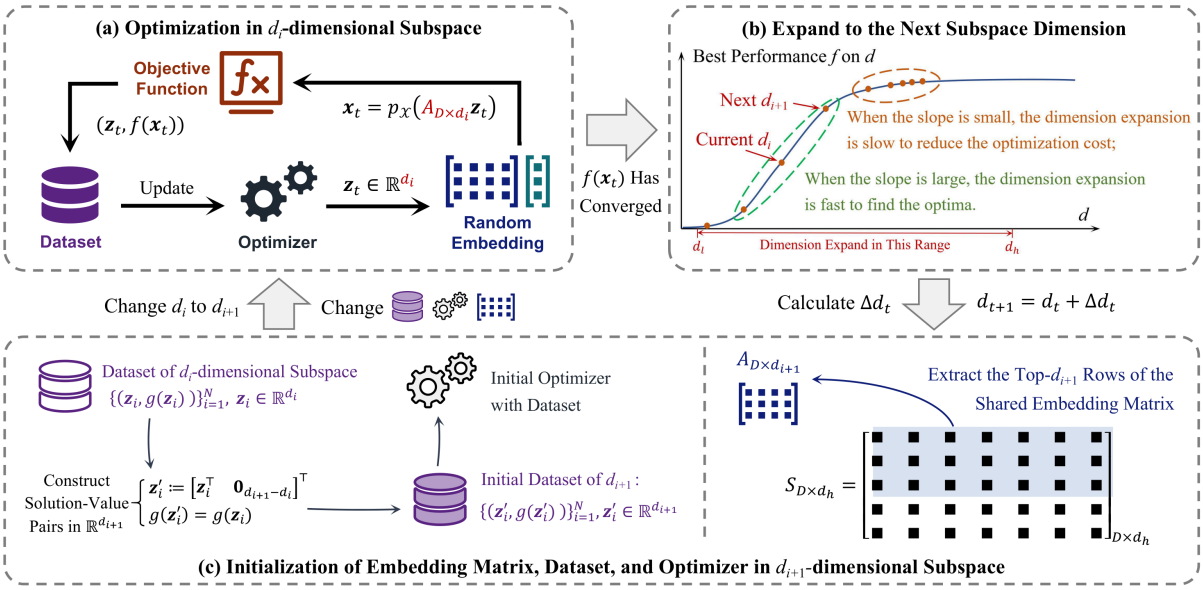

Figure 1: The framework of DSEBO. Subplot (a) shows BO with random embedding, where optimization occurs in a d_i-dimensional subspace, and solutions are mapped to the high-dimensional space through random embedding. Subplot (b) shows dynamic dimension expanding, which expands the subspace dimension to achieve an improved evaluation while keeping the dimension not too high. Subplot (c) shows the process of initializing a new subspace, extending the low-dimensional dataset, initializing the optimizer, and sharing the embedding matrix.

posterior distribution and calculates an acquisition function to balance "exploration" and "exploitation" in the search space $\mathcal{X} \subset \mathbb{R}^D$, thus determining the candidate $\boldsymbol{x} \in \mathcal{X}$. A common acquisition function is UCB [Srinivas *et al.*, 2010], defined as $\alpha_{\text{UCB}}(\boldsymbol{x}) = \mu_t(\boldsymbol{x}) + \sqrt{\kappa_{t+1}}\sigma_t(\boldsymbol{x})$, where $\mu_t(\boldsymbol{x})$ estimates the objective function and $\sigma_t(\boldsymbol{x})$ represents model uncertainty. The hyper-parameter κ controls the trade-off between "exploration" and "exploitation" for efficient optimization. Details are provided in the Appendix A and Appendix A-H can be found in https://github.com/X-Xia0828/DSEBO.

Random Embedding. Random embedding [Wang *et al.*, 2016; Wang *et al.*, 2013; Nayebi *et al.*, 2019] is a popular subspace embedding method, which can achieve dimensionality reduction when the high-dimensional optimization tasks have effective dimension (i.e., the function value is influenced by only a few relevant dimensions). However, it is challenging to meet the effective dimension assumption in real tasks. In this case, Qian *et al.* [2016] propose the concept of optimal ϵ-effective dimension to relax the assumption, defined as:

Definition 3.1 (Optimal ϵ-Effective Dimension [Qian *et al.*, 2016]). *For any $\epsilon > 0$, a function $f : \mathbb{R}^D \to \mathbb{R}$ is said to have an ϵ-effective subspace $\mathcal{V}_\epsilon$, if there exists a linear subspace $\mathcal{V}_\epsilon \subseteq \mathbb{R}^D$, s.t. for all $\boldsymbol{x} \in \mathbb{R}^D$, we have $|f(\boldsymbol{x}) - f(\boldsymbol{x}_\epsilon)| \leq \epsilon$, where $\boldsymbol{x}_\epsilon \in \mathcal{V}_\epsilon$ is the orthogonal projection of $\boldsymbol{x}$ onto $\mathcal{V}_\epsilon$. Let $\mathbb{V}_\epsilon$ denote the collection of all the ϵ-effective subspaces of f, and $dim(\mathcal{V})$ denote the dimension of $\mathcal{V}$. We define the optimal ϵ-effective dimension of f as $d_\epsilon = \min_{\mathcal{V}_\epsilon \in \mathbb{V}_\epsilon} dim(\mathcal{V}_\epsilon)$.*

For a high-dimensional optimization problem $\boldsymbol{x}^* = \text{argmin}_{\boldsymbol{x}\in\mathcal{X}\subset\mathbb{R}^D} f(\boldsymbol{x})$, a random matrix $A \in \mathbb{R}^{D\times d}$ with elements sampled from a Gaussian distribution $\mathcal{N}(0, \sigma^2)$ embeds the d-dimensional subspace $\mathcal{Z} \subset \mathbb{R}^d$ into the D-dimensional search space $\mathcal{X} \subset \mathbb{R}^D$. The optimizer only needs to optimize $\boldsymbol{z} \in \mathcal{Z}$ in the low-dimensional subspace, and then embed $\boldsymbol{z}$ into $\mathcal{X}$ through the matrix A to get the solution $\boldsymbol{x} = p_{\mathcal{X}}(A\boldsymbol{z})$. Here, $p_{\mathcal{X}}(A\boldsymbol{z}) = \text{argmin}_{\boldsymbol{x}\in\mathcal{X}} \|\boldsymbol{x} - A\boldsymbol{z}\|_2$ is to project illegal points (i.e. $A\boldsymbol{z} \notin \mathcal{X}$) back to the $\mathcal{X}$ region. Subsequently, the objective function f is evaluated in the solution $\boldsymbol{x}$, and the sample point $(\boldsymbol{z}, f(p_{\mathcal{X}}(A\boldsymbol{z})))$ is added to the subspace dataset. The optimizer uses the updated dataset to complete the next optimization iteration. See Appendix A for details.

4 The Proposed Method

This section introduces the proposed dynamic shared embedding Bayesian optimization (DSEBO), to handle optimization tasks with unknown effective dimension.

4.1 Overview

To automatically identify the appropriate subspace for high-dimensional BO tasks, the DSEBO algorithm is proposed, illustrated in Figure 1 with pseudo-code in Appendix B.

As shown in Figure 1, DSEBO operates in three stages: (a) optimizing in the d_i-dimensional subspace, (b) expanding to a new subspace, and (c) initializing the new subspace. DSEBO starts with a lower dimension d_l, iteratively optimizes within the subspace until convergence (the convergence criteria will be introduced later), then expands to a higher-dimensional subspace to continue optimization. By gradually increasing and optimizing each subspace, the objective function is optimized while dynamically expanding the subspace dimension. DSEBO considers the differences in the optimal solutions obtained from each subspace when determining dimensionality changes. It also adjusts the scale of changes during optimization to adapt to functions with different dimensionalities. Besides, DSEBO uses a shared embedding matrix to

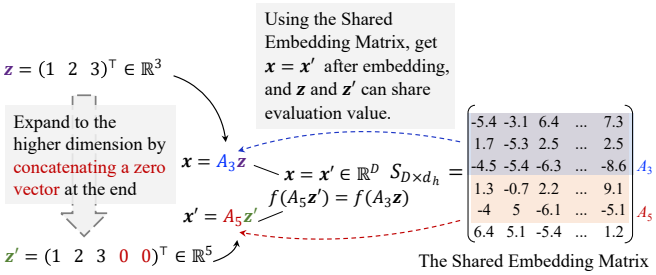

Figure 2: An example of the shared embedding technique.

share evaluation data among different subspaces, allowing data points from low-dimensional subspace to provide better initialization for new subspace, thus conserving the budget.

The following section will explain how data are shared between different subspaces and how dynamic strategy is designed to determine the next subspace dimension.

4.2 Dataset Initialization with Shared Embedding

Optimization is performed sequentially on multiple subspaces while dynamically expanding dimensions. However, independent random matrix mappings prevent sharing sampling points across different dimensional subspaces. Therefore, a proposed shared embedding technique enables the utilization of identical sampling points across different subspaces.

The shared embedding technique maintains a shared matrix $S \in \mathbb{R}^{D\times d_h}$, where D is the dimension of the search space and d_h is the dimension of the largest subspace. Each subspace uses a part of this matrix to embed solutions into the search space. Specifically, for a subspace of dimension d, the first d rows of the matrix S form the embedding matrix A_d. This ensures that for any two subspaces $\mathcal{V}_i$ and $\mathcal{V}_j$ with dimensions d_i and d_j (assuming $d_i < d_j$), A_{d_i} and the first d_i rows of A_{d_j} are identical. Thus, $\mathcal{V}_i$ is a subspace within $\mathcal{V}_j$, allowing $\mathcal{V}_j$ to share sampled data points from $\mathcal{V}_i$.

Figure 2 illustrates the shared embedding process across different subspaces. A 3-dimensional vector $\boldsymbol{z} \in \mathbb{R}^3$ is expanded to 5-dimensional $\boldsymbol{z}' \in \mathbb{R}^5$ by appending zeros. These vectors are then transformed into the search space $\mathbb{R}^D$ using matrices A_3 and A_5, respectively, from the shared embedding matrix S. After embedding, the solutions $\boldsymbol{x}$ and $\boldsymbol{x}'$ in the search space are identical, allowing them to share the same evaluation, i.e., $f(A_5\boldsymbol{z}') = f(A_3\boldsymbol{z})$.

Based on the method, each new subspace uses data from the existing lower-dimensional subspace to initialize its dataset. The initialization algorithm is shown in Appendix C. In the first iteration $t = 1$, the dataset is empty and initialized with a random point. For dimension expansion, the current dataset $\mathcal{D}^{(t)}$ is generated from the previous dataset $\mathcal{D}^{(t-1)}$ by appending zeros to each solution vector $\boldsymbol{z}$ to expand it to the new subspace while keeping the original evaluation value y.

Through the proposed shared embedding technique, higher-dimensional subspaces can reuse the evaluated data in lower-dimensional subspaces, since the function values of corresponding solutions remain invariant across subspaces due to the connection between subspace embedding matrices.

Algorithm 1 Dynamic Dimension Expanding Strategy

Input: Current dimension $d^{(t)}$, dimension range $[d_l, d_h]$
1: **if** $t \leq 2$ **then**
2: $\quad \Delta d^{(t)} = \lfloor \frac{d_h - d_l}{\beta} \rfloor$
3: **else**
4: $\quad$ Compute the $s_i = -\frac{b_{i+1} - b_i}{d_{i+1} - d_i}$, $i \in \{1, 2, \ldots, n-1\}$
5: $\quad s_{\min}, s_{\max} \leftarrow \min_{i=1}^{n-1} s_i, \max_{i=1}^{n-1} s_i$
6: $\quad$ **if** $s_{\min} = s_{\max}$ **then**
7: $\quad\quad \Delta d^{(t)} \leftarrow \Delta d^{(t-1)}$
8: $\quad$ **else**
9: $\quad\quad \Delta d^{(t)} \leftarrow \lfloor k \Delta d^{(t-1)} \rfloor$, where $k = \frac{s_{n-1} - s_{\min}}{s_{\max} - s_{\min}} + 0.5$
10: $\quad$ **end if**
11: **end if**
12: $d^{(t+1)} = \min(d^{(t)} + \Delta d^{(t)}, d_h)$
Output: The next dimension $d^{(t+1)}$.

4.3 Dynamic Dimension Expanding Strategy

While the shared embedding technique enables new subspaces to reuse data from lower-dimensional subspaces, identifying optimal switching timing and targets remains challenging.

Intuitively, the subspace dimension should be updated when the current subspace converges. Specifically, if the optimal value in the subspace remains unchanged for T times, the process can be considered converged. The initial value of T is set to $\lfloor budget/2\beta \rfloor$, controlled by a hyper-parameter β. To determine whether optimization has converged at the current dimension, T is updated as: $T = \lfloor (1 + (d^{(t)} - d_l)/(d_h - d_l)) \cdot budget/2\beta \rfloor$, where $d^{(t)}$ is the current subspace dimension, d_l and d_h are the lower and upper bounds of the dimension range. As $d^{(t)}$ increases, so does T. This aligns with the intuition that higher-dimensional subspaces demand more effort to converge. Based on the definition of T, when to update the subspace dimension is solved. Another problem is how much larger the dimension of the new subspace should be.

According to the relationship between subspace dimensions and convergence values, a dynamic dimension expanding strategy is designed to select the next subspace dimension within the specified range $[d_l, d_h]$. Starting from a small dimension d_l, the strategy dynamically determines the subsequent dimension, and ultimately identifies a subspace where it can converge to an improved solution without excessively larger dimension. To achieve this, the dimensions of the optimized subspace and the corresponding optimal values are recorded as d_i and b_i, where i indicates the i-th optimized subspace, with $i \in \{1, 2, \ldots, n\}$ representing that n subspaces have been optimized. Based on d_i and b_i, the dynamic dimension expanding strategy is designed as Algorithm 1. This strategy determines the next dimension at the t-th iteration by calculating the dimension change $\Delta d^{(t)}$. Initially, if fewer than two subspaces have been optimized, $\Delta d^{(t)}$ is set to $\lfloor (d_h - d_l)/\beta \rfloor$, as stated in lines 1–2. Otherwise, the slopes of b_i with respect to d_i are calculated according to: $s_i = -(b_{i+1} - b_i)/(d_{i+1} - d_i)$, where $i \in \{1, 2, \ldots, n-1\}$. This equation measures the rate at which the optimal value improves as the dimensionality of the optimized subspace increases, quantifying the improvement of expanding the subspace. s_i are calculated for minimum

optimization and should be inverted for maximum.

Next, a proportion value k is used to determine the change in the current dimension, defined by the formula: $k = (s_{n-1} - s_{\min})/(s_{\max} - s_{\min}) + 0.5$, where $s_{\min} = \min_{i=1}^{n-1} s_i$ and $s_{\max} = \max_{i=1}^{n-1} s_i$. According to the equation, the most recent slope is min–max normalized to the range $[0.5, 1.5]$ and then used to determine the change in the current dimension. The value of k reflects the impact of increasing the subspace dimension on the convergence value in the current context. If the value of k is within the range $[1.0, 1.5]$, it indicates a significant impact of the dimension on the convergence value; otherwise it suggests a lesser effect. DSEBO then dynamically scales the change in dimension $\Delta d^{(t-1)}$ from the previous iteration using the value of k, yielding the current dimension change $\Delta d^{(t)}$, as described in line 9. Subsequently, based on this dimension change, the new dimension is calculated as $d^{(t+1)} = \min(d^{(t)} + \Delta d^{(t)}, d_h)$, as shown in line 12.

We would like to emphasize that, unlike BAxUS [Papenmeier *et al.*, 2022], which increases dimensions by splitting existing dimensions and eventually optimizes in the full space, DSEBO adopts the proposed dynamic dimension expanding strategy. Specifically, BAxUS relies on an exponential expansion with a manually specified dimensionality growth rate, whereas DSEBO automatically and nearly linearly expands the subspace dimension within a controllable range $[d_l, d_h]$, leading to lower computational costs and better scalability.

5 Theoretical Analysis

In this section, we present the regret analysis of the naive GPUCB algorithm and DSEBO. To begin, we define $\epsilon(d)$ as the approximation error associated with the d-dimensional subspace, formally expressed as $\epsilon(d) = \min_{\boldsymbol{x}\in\mathbb{R}^d} f(\Phi(\boldsymbol{x})) - \min_{\boldsymbol{z}\in\mathbb{R}^D} f(\boldsymbol{z})$, where $\Phi : \mathbb{R}^d \to \mathbb{R}^D$ denotes the embedding that maps a low-dimensional point into the original D-dimensional search space. Using this notation, we derive the simple regret bound for the naive GPUCB algorithm.

Here, simple regret is defined as $r_f(T) = \min_{t=1}^{T} f(\Phi(\boldsymbol{x}_t)) - \min_{\boldsymbol{z}\in\mathbb{R}^D} f(\boldsymbol{z})$, which measures the difference between the best function value found by the algorithm (after embedding the low-dimensional solution back into the high-dimensional space) and the true global optimum.

Theorem 5.1. *Suppose that the search space $\mathcal{X}$ is compact and convex with dimension d, and every $\boldsymbol{x} \in \mathcal{X}$ satisfies $\|\boldsymbol{x}\|_\infty \le b$. For GP sample paths f with an RBF kernel, choose $\delta \in (0,1)$, and define $\beta^{(t)} = 2\log\left(t^2 2\pi^2/3\delta\right) + 2D\log\left(t^2 dbr\sqrt{\log\left(4Da/\delta\right)}\right)$, where a and b are constants satisfying $\Pr\left(\sup_{\boldsymbol{x}\in\mathcal{X}} |\partial f/\partial x_j| > L\right) \le ae^{-(L/b)^2}$. By running the GPUCB algorithm with β_t using an initialization with a zero mean function and a covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$, the simple regret is bounded by $O^*\left(2\epsilon(d) + \sqrt{(\log T)^{d+1}/T}\right)$, where O^* omits logarithmic factors.*

Due to space limitations, we defer the proof in this section to Appendix D. For DSEBO, the dimensions of the subspaces are updated periodically. Let the number of updates be H, and denote the subspaces in different phases as $\{\mathcal{Z}_h\}_{h=1}^H$, with dimensions $\{d_h\}_{h=1}^H$ and durations $\{T_h\}_{h=1}^H$. We then present the following theorem. Theorems 5.1 and 5.2 stem from Srinivas *et al.* [2010] and Qian *et al.* [2016].

Theorem 5.2. *Suppose each subspace $\mathcal{Z}_h$ is compact and convex, and every $\boldsymbol{x}$ in the subspaces satisfies $\|\boldsymbol{x}\|_\infty \le b$. For GP sample paths f with RBF kernel, pick $\delta \in (0,1)$ and define*

$$\beta^{(t)} = 2\log\left(t^2 2\pi^2/3\delta\right) + 2d_h \log\left(t^2 d_h br\sqrt{\log\left(4d_h a/\delta\right)}\right),$$

where a and b are constants such that $\Pr\left(\sup_{\boldsymbol{x}\in\mathcal{Z}_h} |\partial f/\partial x_j| > L\right) \le ae^{-(L/b)^2}$. Running DSEBO with β_t for the initialization of a GP with mean function zero and covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$, we obtain a simple regret bound of $O^*\left(2\sum_{h=1}^H \epsilon(d_h)T_h/T + \sqrt{\sum_{h=1}^H (\log T_h)^{d_h+1}/T}\right)$.

Remarks. We compare the theoretical results of DSEBO and GPUCB. It can be observed that the first term of DSEBO is larger than that of GPUCB, while the second term is smaller. This indicates that DSEBO effectively balances the trade-off between approximation error and optimization error. To illustrate this, assume that the approximation error satisfies $\epsilon(d') = O^*\left((D - d')/D\right)$. Consider a common setting where $T = 1000$, $H = 10$, $d = 20$, $D = 100$, and the dimension of subspaces increasing evenly up to d, substituting these values into the bound yields an approximation error ratio between DSEBO and GPUCB of $O^*(1.25)$. This implies a modest increase in approximation error for DSEBO compared with GPUCB. However, the optimization error ratio between GPUCB and DSEBO is $O^*(100)$, indicating a significant reduction in optimization error for DSEBO. ***This implies that DSEBO sacrifices a little approximation error in exchange for a much lower optimization error to realize a lower total error (i.e., a better trade-off)***.

6 Experiment

This section evaluates the performance of the proposed DSEBO and shows its effectiveness and superiority through a series of experiments on both synthetic functions and real-world tasks. The code is available in https://github.com/X-Xia0828/DSEBO.

The Setting of Synthetic Functions. We construct high-dimensional objective functions based on synthetic functions meeting the optimal ϵ-effective dimension (as Definition 3.1). Specifically, let $f : \mathbb{R}^{d_f} \to \mathbb{R}$ be a base testing function, with its domain adjusted to $[-1, 1]^{d_f}$. The high-dimensional synthetic function $F_c : \mathbb{R}^D \to \mathbb{R}$ is crafted to simulate the minimization of f, defined by: $F_c(\boldsymbol{x}) = f(\boldsymbol{x}_{1:d_f} - \mathbf{c}) - K^{-1}\sum_{i=d_f+1}^{D}(x_i - c)^2$, where $\boldsymbol{x} \in \mathbb{R}^D$ is the input to F_c, and $\boldsymbol{x}_{1:d_f}$ denotes the first d_f dimensions of $\boldsymbol{x}$. The constant vector $\mathbf{c} \in \mathbb{R}^d$, filled with the scalar c, is introduced to shift the optimal solution away from the origin. The constant K modulates the influence of dimensions beyond the initial d_f. Evidently, F_c possesses an optimal ϵ-effective dimensionality d_f, with $\epsilon \le K^{-1}$. In the experiments, we construct high-dimensional functions with $D = 1000, 10000$ respectively, $d_e = 30$ and $K = 10000$, based on 6 synthetic

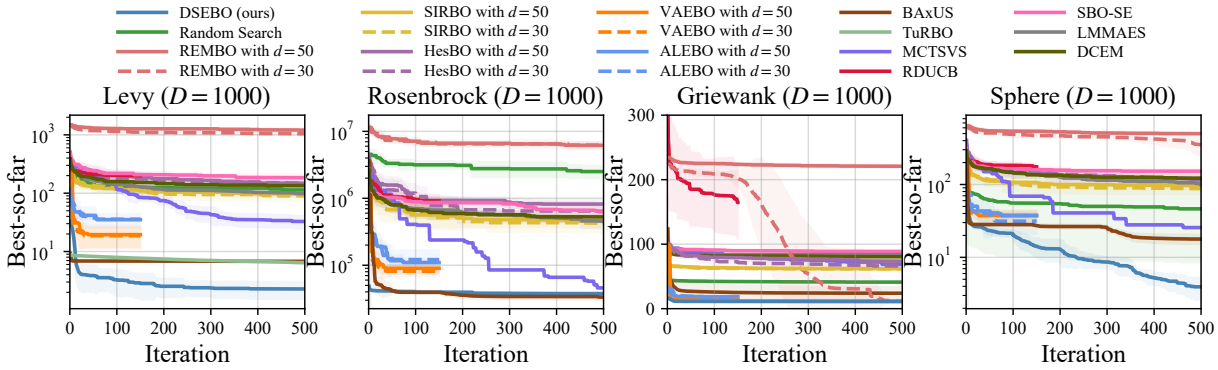

Figure 3: Results on partial synthetic functions compared with various high-dimensional optimization algorithms. All algorithms are independently repeated 10 times. The mean and standard deviation of the results are plotted.

functions from http://www.sfu.ca/~ssurjano/optimization.html, including Levy, Griewank, Sphere, and so on. All experiments on synthetic functions are minimum optimization problems.

The Setting of Real-World Tasks. We evaluate DSEBO on three real-world datasets. The first dataset is the Microsoft Learning to Rank (MSLR) [Qin *et al.*, 2010], specifically the MSLR-WEB-10K version, containing over 10000 queries, each with 136 features per website page. The second dataset is Lasso-Hard from LassoBench [Sehic *et al.*, 2022], a 1000-dimensional optimization task designed to identify sparse regression coefficients that minimize Lasso regression loss. The third dataset is LIMO [Eckmann *et al.*, 2022], a framework for molecular generation aimed at optimizing specific properties by operating in a 1024-dimensional latent space. Further details of these datasets are provided in the Appendix E. All experiments on real-world tasks are minimum optimization.

The Setting of DSEBO. DSEBO requires specifying the subspace dimension search range $[d_l, d_h]$, and a hyper-parameter β. For any high-dimensional problems, the recommended initial subspace dimension is $d_l = 5$, and the upper boundary of the search range is $d_h = \min(D, 100)$, which is the setting for all experiments. Given that the performance of the Bayesian optimizer declines sharply for dimensions above 100, the search range is capped at 100. The hyper-parameter β controls the scale of the dimension expansion, and we set $\beta = 12.0$. The shared embedding matrix $S \in \mathbb{R}_{D \times d_h}$ is initialized with elements independently sampled from the Gaussian distribution $\mathcal{N}(0, \sqrt{1/d_h})$.

6.1 Performance of High-dimensional Optimization

In this section, DSEBO is compared against high-dimensional optimization methods under limited resources. These methods include REMBO [Wang *et al.*, 2016], SIRBO [Zhang *et al.*, 2019], HesBO [Nayebi *et al.*, 2019], VAEBO [Gómez-Bombarelli *et al.*, 2018], and ALEBO [Letham *et al.*, 2020], all of which are based on subspace embedding techniques and REMBO serves as an ablated version of DSEBO without the dynamic dimension expanding strategy. We also compare with BAxUS [Papenmeier *et al.*, 2022], which dynamically selects the next subspace dimension. For methods that do not rely on subspace embedding, we conduct TuRBO [Eriksson *et al.*, 2019], SAASBO [Eriksson and Jankowiak, 2021], DuMBO [Bardou *et al.*, 2024], MCTS-VS [Song *et al.*, 2022], RDUCB [Ziomek and Bou-Ammar, 2023], SBO-SE [Xu *et al.*, 2025], LMMAES [Loshchilov *et al.*, 2019], DCEM [Amos and Yarats, 2020] and random search. For further details, refer to Appendix F. All methods are tested on high-dimensional optimization tasks with unknown effective dimensions, with hyper-parameters $d = 30, 50$ for synthetic functions and $d = 50, 80$ for real-world tasks. All methods are allocated 500 evaluation budget for optimization unless optimization either exceeds 8 hours of runtime (e.g., ALEBO, VAEBO, etc.) or encounters out-of-memory errors with 16 GB of RAM (e.g., SAASBO). All methods are independently repeated 10 times.

The best-so-far solutions found on partial synthetic functions and real-world datasets are shown in Figure 3 and 4. The horizontal axis represents the number of iterations, with each iteration consuming 1 budget unit. The vertical axis records the best-so-far function value. The remaining detailed results of synthetic functions with $D = 1000$, all results of synthetic functions with $D = 10000$ are shown in Appendix G. To further verify DSEBO's capability in handling high-dimensional optimization tasks, we also conduct experiments by adjusting the shifting scalar c and the effective dimension d_e of the synthetic functions, as shown in Appendix G.

The experimental results show (1) **Superiority.** Compared with other high-dimensional BO algorithms, DSEBO can find optimal solutions across almost all tasks, including real-world tasks and high-dimensional synthetic functions of different magnitudes. (2) **Efficiency.** In high-dimensional tasks with unknown effective dimensions, DSEBO dynamically expands the subspace dimension to find the optimal solution within limited resources. This allows DSEBO to achieve the optimal solution with minimal cost, whereas subspace-based methods require multiple complete optimization processes across different subspace dimensions [Wang *et al.*, 2016] and non-subspace-based methods require optimization in the original high-dimensional space, both consuming significant computational resources. (3) **Continuous Improvement.** Due to its ability to dynamically expand subspace dimensions, the performance of DSEBO continues to improve when other algorithms have converged. When the optimization process converges in the current subspace, DSEBO can expand to a larger subspace for further optimization, ensuring continuous progress. (4)

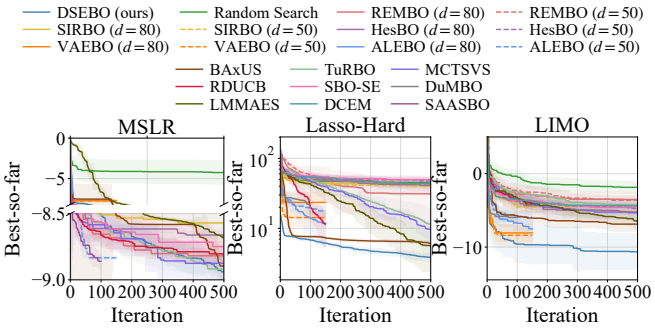

Figure 4: Results on real-world tasks compared with different high-dimensional optimization algorithms. All algorithms are run 10 times, and the mean and standard deviation are plotted.

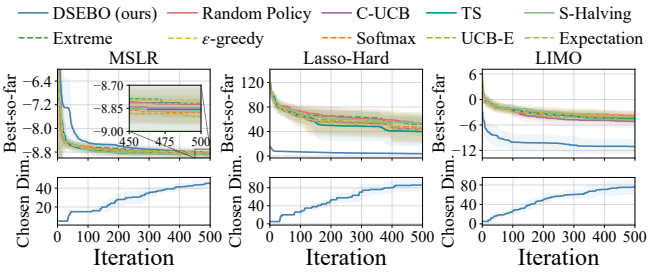

Figure 5: Results on real-world tasks compared with the MAB strategies, and the subspace dimensions selected by DSEBO during the optimization process. All experiments are repeated 10 times. The mean and standard deviation of the results are plotted.

Adaptability. DSEBO starts from a lower initial subspace, which makes its initial evaluation values different (such as having a good initial value on synthetic functions while getting poor initial performance on real-world datasets), but DSEBO adapts to the specific tasks and continues to converge towards the optimal solution. (5) **Necessity.** The experimental results show that some high-dimensional embedding BO algorithms (such as REMBO) have obvious performance differences on different subspaces. This illustrates the necessity of designing a dynamic subspace dimension expanding strategy.

6.2 Performance of Dynamic Dimension Expanding

To further evaluate the performance of the dynamic dimension expanding strategy, we compare a series of MAB strategies, including Extreme Bandits (Extreme), Classic UCB (C-UCB) [Auer *et al.*, 2002], ϵ-Greedy [Langford and Zhang, 2007], Softmax strategy [Sutton and Barto, 2018], Successive Halving (S-Halving) [Karnin *et al.*, 2013], UCB-E [Audibert *et al.*, 2010], Thompson Sampling (TS) [Jin *et al.*, 2022], Expectation strategy (Expectation), and Random strategy. Each MAB strategy selects different subspaces for optimization, with dimensions drawn from $\{10, 20, 30, 50, 70, 90, 100\}$. A detailed description of these algorithms is provided in the Appendix F. All strategies are allocated a budget of 500 evaluations, each subspace is initialized by one point, and the experiments are independently repeated 10 times. The experimental results on real-world tasks are presented in Figure 5, showcasing the best-so-far solution found (top) and the subspace dimension expansions (bottom). Results for synthetic functions are provided in the Appendix G.

Experimental results show that: (1) **Superiority and Tailor-Made.** DSEBO designs a dynamic dimension expanding strategy for high-dimensional BO problems with unknown effective dimensions. Compared with the general MAB strategies, it can converge to better performance on various tasks. (2) **Effectiveness of Shared Embedding.** Through experimental results on synthetic functions, it can be found that on tasks with particularly high dimensions, random strategy has the worst results, followed by strategies that explore larger subspace dimensions. Unlike DSEBO, other strategies cannot share data to explore different subspace dimensions. Therefore, exploration in different dimensions brings a huge overhead and reduces the convergence efficiency. (3) **Reasonable Expansions.** The results for four synthetic functions show that when the space dimension is below the effective dimension d_e, DSEBO rapidly increases it. However, once the effective dimension is exceeded, the growth rate slows down, reflecting the rationality of DSEBO in expanding subspace dimensions.

6.3 Hyper-Parameter Analysis

We conduct hyper-parameter analysis on β and the upper boundary of search range d_h to verify the robustness of DSEBO under different hyper-parameter settings. The detailed results and analysis are shown in Appendix H. The hyper-parameter analysis verifies that the chosen hyper-parameter values are reasonable, and shows the robustness of the DSEBO across different settings.

7 Conclusion and Discussion

Conclusion. This paper introduces the DSEBO method for automated random embedding in high-dimensional BO with unknown effective dimensions. We propose the dynamic shared embedding BO, which dynamically expands the subspace dimension during optimization. Utilizing a shared embedding matrix, one subspace can share the initial dataset from a lower-dimensional subspace. The dynamic dimension expanding strategy determines the dimension of the next subspace, thereby achieving better optimization performance within limited resources. The theoretical analysis establishes a regret bound for DSEBO, proofing its superior ability to balance approximation and optimization errors compared with GPUCB.

Discussion. Currently, due to the effective dimension being unknown during the optimization process, DSEBO is unable to halt dimension expansion near the effective dimension, which may lead to unnecessary increases in the subspace dimensionality. In the future, we will explore more adaptive strategies to identify the effective dimension and enable dynamic subspace adjustment (not just dimension expansion) and further integrate the dynamic dimension expansion strategy into other high-dimensional optimization methods, such as evolutionary algorithms, to improve optimization efficiency. Furthermore, we will explore the integration of DSEBO with multiple random embeddings [Cartis *et al.*, 2023] or learned embedding from data [Garnett *et al.*, 2014] methods to better address high-dimensional optimization tasks with effective dimension.

Ethical Statement

This work does not include any human subjects, personal data, or sensitive information. All testing datasets utilized are publicly accessible, and no proprietary or confidential information has been employed.

Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable and insightful comments. This work is supported by the National Key Research and Development Program of China under Grant 2024YFC3308503, the National Natural Science Foundation of China (No. 62476091) and Ant Group.

References

[Amos and Yarats, 2020] Brandon Amos and Denis Yarats. The differentiable cross-entropy method. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 291–302, Virtual Event, 2020.

[Audibert *et al.*, 2010] Jean-Yves Audibert, Sébastien Bubeck, and Rémi Munos. Best arm identification in multi-armed bandits. In *Proceedings of the 23rd Conference on Learning Theory*, pages 41–53, Haifa, Israel, 2010.

[Auer *et al.*, 2002] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 2002.

[Bardou *et al.*, 2024] Anthony Bardou, Patrick Thiran, and Thomas Begin. Relaxing the additivity constraints in decentralized no-regret high-dimensional Bayesian optimization. In *Proceedings of the 12th International Conference on Learning Representations*, 2024.

[Binois and Wycoff, 2022] Mickael Binois and Nathan Wycoff. A survey on high-dimensional Gaussian process modeling with application to Bayesian optimization. *ACM Transactions on Evolutionary Learning and Optimization*, 2(2):8:1–8:26, 2022.

[Boyd and Vandenberghe, 2004] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, 2004.

[Cartis *et al.*, 2023] Coralia Cartis, Estelle M. Massart, and Adilet Otemissov. Bound-constrained global optimization of functions with low effective dimensionality using multiple random embeddings. *Mathematical Programming*, 198(1):997–1058, 2023.

[Eckmann *et al.*, 2022] Peter Eckmann, Kunyang Sun, Bo Zhao, Mudong Feng, Michael K. Gilson, and Rose Yu. LIMO: Latent inceptionism for targeted molecule generation. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pages 5777–5792, Baltimore, MD, 2022.

[Elsken *et al.*, 2019] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20:55:1–55:21, 2019.

[Eriksson and Jankowiak, 2021] David Eriksson and Martin Jankowiak. High-dimensional Bayesian optimization with sparse axis-aligned subspaces. In *Proceedings of the 37th Conference on Uncertainty in Artificial Intelligence*, pages 493–503, Virtual Event, 2021.

[Eriksson *et al.*, 2019] David Eriksson, Michael Pearce, Jacob R. Gardner, Ryan Turner, and Matthias Poloczek. Scalable global optimization via local Bayesian optimization. In *Advances in Neural Information Processing Systems 32*, pages 5497–5508, Vancouver, Canada, 2019.

[Freund and Schapire, 1997] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[Garnett *et al.*, 2014] Roman Garnett, Michael A. Osborne, and Philipp Hennig. Active learning of linear embeddings for Gaussian processes. In *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence*, pages 230–239, Quebec, Canada, 2014.

[Garnett, 2023] Roman Garnett. *Bayesian Optimization*. Cambridge University Press, Cambridge, 2023.

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

[Hvarfner *et al.*, 2024] Carl Hvarfner, Erik Orm Hellsten, and Luigi Nardi. Vanilla Bayesian optimization performs great in high dimensions. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pages 20793–20817, Vienna, Austria, 2024.

[Jin *et al.*, 2022] Tianyuan Jin, Pan Xu, Xiaokui Xiao, and Anima Anandkumar. Finite-time regret of Thompson sampling algorithms for exponential family multi-armed bandits. In *Advances in Neural Information Processing Systems 35*, pages 38475–38487, New Orleans, LA, 2022.

[Karnin *et al.*, 2013] Zohar Shay Karnin, Tomer Koren, and Oren Somekh. Almost optimal exploration in multi-armed bandits. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1238–1246, Atlanta, GA, 2013.

[Langford and Zhang, 2007] John Langford and Tong Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in Neural Information Processing Systems 20*, pages 817–824, Vancouver, Canada, 2007.

[Letham *et al.*, 2020] Benjamin Letham, Roberto Calandra, Akshara Rai, and Eytan Bakshy. Re-examining linear embeddings for high-dimensional Bayesian optimization. In *Advances in Neural Information Processing Systems 33*, Virtual Event, 2020.

[Loshchilov *et al.*, 2019] Ilya Loshchilov, Tobias Glasmachers, and Hans-Georg Beyer. Large scale black-box optimization by limited-memory matrix adaptation. *IEEE*

Transactions on Evolutionary Computation, 23(2):353–358, 2019.

[Mehta and Grosan, 2015] Dhagash Mehta and Crina Grosan. A collection of challenging optimization problems in science, engineering and economics. In *Proceedings of the 17th IEEE Congress on Evolutionary Computation*, pages 2697–2704, Sendai, Japan, 2015.

[Nayebi *et al.*, 2019] Amin Nayebi, Alexander Munteanu, and Matthias Poloczek. A framework for Bayesian optimization in embedded subspaces. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 4752–4761, Long Beach, CA, 2019.

[Papenmeier *et al.*, 2022] Leonard Papenmeier, Luigi Nardi, and Matthias Poloczek. Increasing the scope as you learn: Adaptive Bayesian optimization in nested subspaces. In *Advances in Neural Information Processing Systems 35*, pages 11586–11601, New Orleans, LA, 2022.

[Qian and Yu, 2021] Hong Qian and Yang Yu. Derivative-free reinforcement learning: A review. *Frontiers of Computer Science*, 15(6):156336, 2021.

[Qian *et al.*, 2016] Hong Qian, Yi-Qi Hu, and Yang Yu. Derivative-free optimization of high-dimensional non-convex functions by sequential random embeddings. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 1946–1952, New York, NY, 2016.

[Qin *et al.*, 2010] Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. LETOR: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval*, 13(4):346–374, 2010.

[Santoni *et al.*, 2024] Maria Laura Santoni, Elena Raponi, Renato De Leone, and Carola Doerr. Comparison of high-dimensional Bayesian optimization algorithms on BBOB. *ACM Transactions on Evolutionary Learning and Optimization*, 4(3):17:1–17:33, 2024.

[Sehic *et al.*, 2022] Kenan Sehic, Alexandre Gramfort, Joseph Salmon, and Luigi Nardi. LassoBench: A high-dimensional hyperparameter optimization benchmark suite for Lasso. In *Proceedings of the 1st International Conference on Automated Machine Learning*, volume 188, pages 2/1–24, Baltimore, MD, 2022.

[Shahriari *et al.*, 2016] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[Song *et al.*, 2022] Lei Song, Ke Xue, Xiaobin Huang, and Chao Qian. Monte Carlo tree search based variable selection for high dimensional Bayesian optimization. In *Advances in Neural Information Processing Systems 35*, pages 28488–28501, New Orleans, LA, 2022.

[Srinivas *et al.*, 2010] Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias W. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on Machine Learning*, pages 1015–1022, Haifa, Israel, 2010.

[Sutton and Barto, 2018] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, 2018.

[Wang *et al.*, 2013] Ziyu Wang, Masrour Zoghi, Frank Hutter, David Matheson, and Nando de Freitas. Bayesian optimization in high dimensions via random embeddings. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pages 1778–1784, Beijing, China, 2013.

[Wang *et al.*, 2016] Ziyu Wang, Frank Hutter, Masrour Zoghi, David Matheson, and Nando de Freitas. Bayesian optimization in a billion dimensions via random embeddings. *Journal of Artificial Intelligence Research*, 55:361–387, 2016.

[Xu *et al.*, 2025] Zhitong Xu, Haitao Wang, Jeff M Phillips, and Shandian Zhe. Standard Gaussian process is all you need for high-dimensional Bayesian optimization. In *Proceedings of the 13th International Conference on Learning Representations*, 2025.

[Yu *et al.*, 2025] Yang Yu, Hong Qian, and Yi-Qi Hu. *Derivative-Free Optimization: Theoretical Foundations, Algorithms, and Applications*. Springer, 2025.

[Zhang *et al.*, 2019] Miao Zhang, Huiqi Li, and Steven W. Su. High dimensional Bayesian optimization via supervised dimension reduction. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 4292–4298, Macao, China, 2019.

[Zhou *et al.*, 2019] Zhi-Hua Zhou, Yang Yu, and Chao Qian. *Evolutionary Learning: Advances in Theories and Algorithms*. Springer Publishing Company, Incorporated, Berlin, 2019.

[Ziomek and Bou-Ammar, 2023] Juliusz Krysztof Ziomek and Haitham Bou-Ammar. Are random decompositions all we need in high dimensional Bayesian optimisation? In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pages 43347–43368, Honolulu, HI, 2023.