# Increasing the Scope as You Learn: Adaptive Bayesian Optimization in Nested Subspaces

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

Recent advances have extended the scope of Bayesian optimization (BO) to expensive-to-evaluate black-box functions with dozens of dimensions, aspiring to unlock impactful applications, for example, in the life sciences, neural architecture search, and robotics. However, a closer examination reveals that the state-of-the-art methods for high-dimensional Bayesian optimization (HDBO) suffer from degrading performance as the number of dimensions increases or even risk failure if certain unverifiable assumptions are not met. This paper proposes BAXUS that leverages a novel family of nested random subspaces to adapt the space it optimizes over to the problem. This ensures high performance while removing the risk of failure, which we assert via theoretical guarantees. A comprehensive evaluation demonstrates that BAXUS achieves better results than the state-of-the-art methods for a broad set of applications.

## 1 Introduction

The optimization of expensive-to-evaluate black-box functions where no derivative information is available has found many applications, for example, in chemical engineering [8, 24, 27, 52, 54], materials science [20, 25, 26, 28, 47, 57, 61], aerospace engineering [35, 39], hyperparameter optimization [3, 29, 34, 55], neural architecture search [32, 51], vehicle design [11, 30], hardware design [16, 44], drug discovery [46], robotics [9, 10, 37, 41, 49], and the life sciences [15, 53, 58]. Increasing the number of dimensions (or parameters) of the optimization problem usually allows for better solutions. For example, for a chemical reaction, we obtain a more granular control by exposing more process parameters; for a design problem, we may optimize a larger number of design decisions jointly; in robotics, we gain access to more sophisticated control policies. A series of breakthroughs facilitated the increasing adoption of Bayesian optimization in other domains that have pushed the envelope of high-dimensional Bayesian optimization. Methods that rely on local 'trust regions' usually achieve good performance if the input dimensionality of the problem does not exceed a couple of dozens. However, we observe that their performance degrades for higher-dimensional problems. This is not surprising, given that the surrogate model of the trust regions has the full dimensionality of the problem. Other state-of-the-art methods suppose the existence of a low-dimensional active subspace and enjoy great scalability if they find such a space. The caveat is that its existence is usually not known for practical applications. Moreover, the user needs to 'guess' a good upper bound on its dimensionality to enjoy a good sample efficiency.

In this work, we propose a theoretically-founded approach for high-dimensional Bayesian optimization, BAXUS (**B**ayesian optimization with **A**daptive a**x**is-aligned **U**niform **S**ubspaces), that reliably achieves a high performance on a comprehensive set of applications. BAXUS utilizes a family of nested embedded spaces which allows increasing the dimensionality of the domain that it currently optimizes over as it collects more data to refine its surrogate. As a byproduct, BAXUS can leverage

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

an active subspace, if it exists, without requiring the user to 'guess' its dimensionality. BAXUS is based on a novel random linear subspace embedding that enables a more efficient optimization and has strong theoretical guarantees. We make the following contributions:

1. We develop BAXUS, a reliable algorithm for high-dimensional BO that scales to higher input dimensions than the state-of-the-art.
2. A novel family of nested random embeddings that has the following properties: a) The BAXUS embedding has a strictly higher probability of containing an optimum than the HESBO embedding proposed by Nayebi et al. [45]. b) The BAXUS embedding the optimal embedding of the *count-sketch* type, as defined in Def. 1. c) its probability containing an optimum converges to the one of the HESBO embedding as the input dimensionality $D \to \infty$.
3. A comprehensive evaluation on a representative collection of benchmarks that demonstrates that BAXUS outperforms the state-of-the-art methods.

The remainder of this paper is structured as follows. Section 2 states the problem and discusses related work. Section 3 presents the BAXUS algorithm and the corresponding embedding. Section 4 evaluates BAXUS on a variety of benchmarks. Lastly, we discuss our work in Section 5.

## 2 Background

The task is to find a minimizer

$$\boldsymbol{x}^* \in \arg\min_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x}),$$

where $\mathcal{X} = [-1, +1]^D$. The objective function $f : \mathcal{X} \to \mathbb{R}$ is an expensive-to-evaluate black-box function; hence the number of function evaluations needed to find an optimizer is crucial. Evaluations may be subject to observational noise, i.e., $f(\boldsymbol{x}_i) + \varepsilon_i$, with $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$. The focus of this work is on *scalable high-dimensional Bayesian optimization*, where the *input dimensionality* $D$ is in the hundreds, and the sampling budget may comprise a thousand or more function evaluations.

**Linear embeddings.** A successful approach for high-dimensional Bayesian optimization is to assume the existence of an *active subspace* [14], i.e., there exist a space $\mathcal{Z} \in \mathbb{R}^{d_e}$, with $d_e < D$ and a function $g : \mathcal{Z} \to \mathbb{R}$, such that for all $\boldsymbol{x}$: $g(\boldsymbol{z}) \approx f(\boldsymbol{x})$ where $\boldsymbol{z}$ is a projection of $\boldsymbol{x}$ in $\mathcal{Z}$ and $d_e$ is the *effective dimensionality* of the problem. In practice, both $d_e$ and $\mathcal{Z}$ are unknown.

REMBO (Random embedding BO) [63] and HESBO (Hashing-enhanced subspace BO) [45] try to capture this active subspace by a randomly chosen subspace. Therefore, they generate a random projection matrix $S^\intercal$ that maps from a $d$-dimensional subspace $\mathcal{Y} \in \mathbb{R}^d$ with $d \ll D$ (the *target space*) to $\mathcal{X}$. We call $d$ the *target dimensionality*. For REMBO, each entry in $S^\intercal$ is normally distributed. REMBO uses a heuristic to determine a hyperrectangle in $\mathcal{Y}$ that it then optimizes over. Note that the bounded domain may not contain a point that maps to an optimizer of $f$, a risk aggravated by distortions introduced by the projection. [4, 5, 6] proposed ideas to mitigate the issue. HESBO's random projection assigns one target dimension and sign ($\pm 1$) to each input dimension. This embedding is inspired by the count-sketch algorithm [12] for estimating frequent items in data streams. The sparse projection matrix $S^\intercal$ is, except for the signs, binary, and each row has exactly one non-zero entry. Even though this embedding avoids REMBO's distortions, it has a lower probability of containing the optimum [36]. ALEBO [36] uses a Mahalanobis kernel and imposes linear constraints on the acquisition function to avoid projecting outside of $\mathcal{X}$.

**Non-linear embeddings.** Several works use autoencoders to learn non-linear spaces for optimization, trading in sample efficiency. Tripp et al. [60] change the training objective of a variational autoencoder (VAE) [33] to make the target space more suitable for optimization. They give higher weight to better-performing points when training the variational autoencoder (VAE) and show that this improves optimization. Moriconi et al. [42] incorporate the training of an autoencoder directly into the likelihood maximization of a Gaussian process (GP) surrogate. The computational cost is cubic in the number of samples and the input dimension. Lu et al. [38] and Maus et al. [40] use autoencoders to learn embeddings of highly structured input spaces such as kernels or molecules. Other approaches include partial least squares [7] or sliced inverse regression [13].

**High-dimensional BO in the input space.** One method to make HDBO in the input space feasible is trust regions (TRs) [19, 48, 50, 66]. The TuRBO algorithm [19] optimizes over bounded TRs instead of the global space, adapting their side lengths and the center points during the optimization process. By restricting function evaluations to trust regions, TuRBO addressed the problem of over-exploration; see [19] for details. Note that the TRs have full input dimensionality, which may impact TuRBO's ability to scale to very large dimensions. Nonetheless, TuRBO set a new state-of-the-art by scaling to dozens on input dimensions and thousands of function evaluations. Wan et al. [62] extended the idea of TRs to categorical and mixed spaces by using the Hamming distance to define the TR boundaries. SAASBO [17] uses sparse priors on the GP length scales which seems particularly valuable if the active subspace is axis-aligned. Indeed, SAASBO can outperform TuRBO on certain benchmarks [17]. The cost of inference scales cubically with the number of function evaluations; thus, SAASBO is not expected to scale beyond small sampling budgets, which is confirmed by our experiments. Another line of research relies on the assumption that the input space has an additive structure [21, 31, 43, 64]. Additive GPs rely on computationally expensive sampling methods to learn a decomposition of the input variables, which limits the scalability of such methods to problems of moderate dimensionalities and sampling budgets [19, 45].

In Sect. 4 we evaluate the performances of TuRBO, SAASBO, ALEBO, and HeSBO. Moreover, we study the popular CMA-ES [22] and random search [2].

# 3 The BAxUS algorithm

Wang et al. [63] showed that the REMBO embedding can represent the optimum in the target space if $d \geq d_e$ and there are no bounds on the target and input spaces, i.e., $\mathcal{Y} = \mathbb{R}^d$ and $\mathcal{X} = \mathbb{R}^D$. If there is a point in the target space that maps to an optimum in the input space for arbitrary functions $f$ attaining an optimum in $\mathcal{X}$, then $\mathcal{Y}$ contains that optimum. For $d < d_e$, it is generally impossible to represent an optimum in $\mathcal{X}$ for arbitrary $f$ as $S^\intercal$ projects to a $d$-dimensional subspace in $\mathcal{X}$. We call the probability of a target space to contain the optimum the *success probability*. For $d \geq d_e$, there is a positive success probability that increases with $d$ [36, 63]. The main problem is to set $d$ as small as possible to avoid the detrimental effects of the curse of dimensionality but as large as necessary to achieve a high probability of $\mathcal{Y}$ to contain an optimum.

In reality, the effective subspace and its dimensionality are usually unknown. The performance of methods, such as REMBO [63], HeSBO [45], and ALEBO [36] depends on choosing $d$ such that the success probability is high. Therefore, they implicitly rely on guessing the effective dimensionality $d_e$ appropriately. We argue that choosing the target dimensionality is problematic in many practical applications. If chosen too small, the subspace cannot represent $f$ sufficiently well. If it is chosen too large, the curse of dimensionality slows down the optimization.

The proposed algorithm, BAxUS, operates on target spaces of increasing dimensionality while preserving previous observations. Let $d_{\text{init}}$ be the initial target dimensionality and $m$ the total evaluation budget. BAxUS starts with a $d_{\text{init}}$-dimensional embedding that is increased over time until, after $m_D \leq m$ evaluations, it roughly reaches the input dimensionality $D$. With this strategy, we can leverage the efficiency of BO in low-dimensional spaces while being guaranteed to find an optimum in the limit. Increasing the target dimensionality is enabled by our embedding, which lets us convey observations from previous, lower-dimensional target spaces into more high-dimensional target spaces. We further use a TR-based approach based on Eriksson et al. [19] to carry out optimization for high target dimensions effectively. BAxUS uses a GP surrogate [65] to model the function in the target space. Algorithm 1 gives the pseudocode for BAxUS. In Appendix D, we prove global convergence for BAxUS. We will now discuss the different components.

## 3.1 The Axis-aligned Uniform Subspace Embedding

The BAxUS embedding is derived from the HeSBO embedding [45]. Given the input dimensionality $D$ and a target dimensionality $d$, the HeSBO embedding samples a target dimension $\in \{1, \ldots, d\}$ and a sign $\in \{\pm 1\}$ for each input dimension uniformly at random. Conversely, each target dimension has a set of signed contributing input dimensions. We call the set of contributing input dimensions to a target dimension a *bin*. These relations implicitly define the embedding matrix $S \in \{0, \pm 1\}^{d \times D}$, where each column has exactly one non-zero entry [12]. In the HeSBO embedding, the number of contributing input dimensions varies between 0 and $D$.

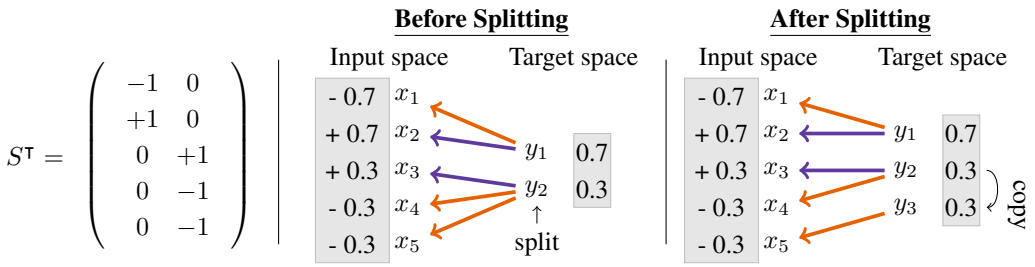


Figure 1: Observations are kept when increasing the target dimensionality: Example of the splitting method where $D = 5$ and $d = 2$. The first target dimension $y_1$ has two contributing input dimensions, $x_1$ and $x_2$; $y_2$ has three contributing input dimensions, $x_3$, $x_4$, and $x_5$. By $S^\intercal$, a point $(0.7, 0.3)^\intercal$ in the target space is mapped to $(-0.7, +0.7, +0.3, -0.3, -0.3)^\intercal$ in the input space. Assigning the fifth input dimension to a new target dimension and copying the function values from the second target dimension does not change the observation in the input space. The new $S^\intercal$ is not shown but has one additional column with $-1$ in the last row, and the last row of the second column is set to 0.

The interpretation of contributing input dimensions allows for an intuitive way to increase the embedding, which is shown in Figure 1. We update the embedding matrix such that contributing input dimensions of the target dimension are re-assigned to $b + 1$ bins. We then say that we *split* the corresponding target dimension. Importantly, this type of embedding allows for retaining observations (see Figure 1) without any distortions. Assume, for example, that $y_i$ is the dimension to be split. The contributing input dimensions are re-assigned to $y_i$ and two new target dimensions $y_j$ and $y_k$ (here, $b = 2$); the observations can be retained by copying the observation for $y_i$ to $y_j$ and $y_k$. The observations remain undistorted in the input space because an input dimension re-assigned to a new target dimension still has the same value. By doing so, the observations from the previous target dimension get copied over to the new target dimension. Algorithm 2 describes the procedure in detail.

In the BAxUS embedding, we force each bin of a target dimension to have roughly the same number of contributing input dimensions; the bin sizes differ by at most one. First, we create a random permutation of the input dimensions $1, \ldots, D$. The list of input dimensions is split into $\min(d, D)$ individual bins. If $d$ does not divide $D$, not all bins can have the same size. We split the permutation

**Algorithm 1** BAxUS

**Input:** $b$: new bins per dimension split, $D$: input dimension, $m_D$: # evaluations by which the input dimension is reached.
**Output:** minimizer $\boldsymbol{x}^* \in \arg\min_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x})$.

```
d ← initial target dimensionality of the subspace given by Eq. (4).
Compute initial projection matrix S^T : 𝒴 → 𝒳 by BAxUS embedding for D and d.
Sample initial data 𝒟 = {(y_1, f(S^T y_1)), ...} and fit the GP surrogate.
n ← 1
while evaluation budget not exhausted do
    L ← L_init                                          ▷ Initialize the trust region
    Calculate fail tolerance as described in Sec. 3.4: τ^s_fail ← max(1, min(τ̃^s_fail, d))
    while L > L_min and evaluation budget not exhausted do
        Find y by maximizing Thompson sample in TR, evaluate f(S^T y), and add to 𝒟.
        Re-fit the GP hyperparameters.
        Adjust trust region, see Sec. 3.2 for details.
    if d < D then
        d ← min(d · (b + 1), D).
        Increase S^T by Algorithm 2.                    ▷ See Appendix E
    else
        Re-sample initial data and discard previous observations.
    n ← n + 1
Return S^T(arg min_{y∈𝒟} f(S^T y)).                     ▷ Return best point in input space
```

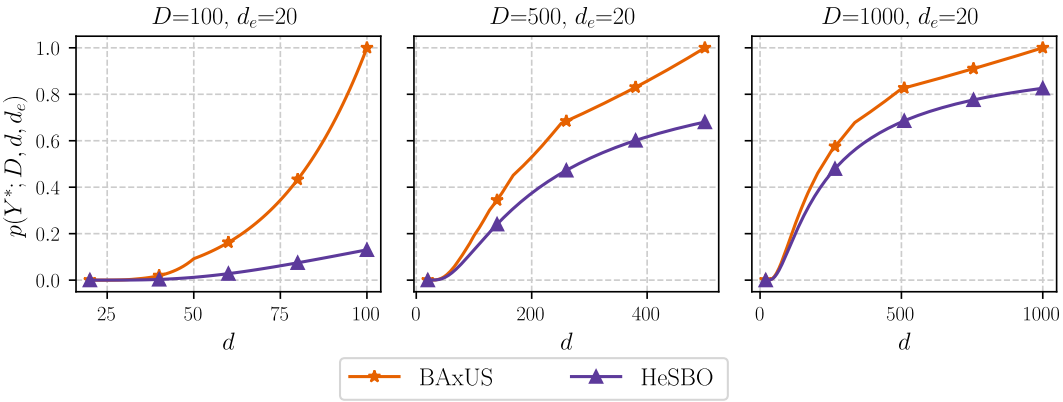


Figure 2: Success probabilities $p(Y^*; D, d, d_e)$ of the BAxUS and HeSBO embeddings for different input dimensionalities ($D = 100$ (l), $D = 500$ (c), $D = 1000$ (r)) as a function of the target dimensionality $d$. The effective dimensionality is $d_e = 20$. The BAxUS embedding has a strictly higher success probability than the HeSBO embedding. The improvement is enormous for input dimensionalities in the low hundreds and still substantial for 1000D tasks. Following the theoretical analysis, the difference vanishes as the input dimension $D$ grows large.

of input dimensions such that the $i$-th bin has size $\lceil D/d \rceil$, if $i + d\lfloor D/d \rfloor \leq D$, and $\lfloor D/d \rfloor$ otherwise. The first bins have one additional element with this construction if $d$ does not divide $D$. We further sample a sign for each input dimension. The sign of the input dimensions and their assignment to target dimensions then implicitly define $S^\intercal$ (see Figure 1). We now show that target spaces of the BAxUS embedding have a higher success probability than for the HeSBO embedding.

**Definition 1** (**Count-sketch type embedding**). We say that a matrix $S \in \{0, \pm 1\}^{D \times d}$ is of the *count-sketch type* if and only if each column in $S$ has precisely one non-zero entry.

**Definition 2** (**Success of a count-sketch type embedding**). A success of count-sketch type embedding is the event $Y^*$ = "For any sparse axis-aligned function $f : \mathcal{X} \to \mathbb{R}$ that attains minima in $\mathcal{X}$, there exists a $\boldsymbol{y}^*$ in $\mathcal{Y}$ such that $S^\intercal \boldsymbol{y}^* \in \arg\min_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x})$", where $S^\intercal$ maps from the target space $\mathcal{Y}$ to the input space $\mathcal{X}$.

We refer to Definition 4 in Appendix C for a formal definition of a sparse axis-aligned function. In Lemma 1 we give the success probability of the BAxUS embedding. All proofs have been deferred to Appendix C.

**Lemma 1** (**Success probability of the BAxUS embedding**). *Let $D$ be the input dimensionality and $d \geq d_e$ the dimensionality of the embedding. Let $\beta_{\text{small}} = \lfloor \frac{D}{d} \rfloor$ and $\beta_{\text{large}} = \lceil \frac{D}{d} \rceil$ be the small and large bin sizes. Then, the probability of $Y^*$ (see Definition 2) with the* BAxUS *embedding is*

$$p_A(Y^*; D, d, d_e) = \frac{\sum_{i=0}^{d_e} \binom{d(1+\beta_{\text{small}})-D}{i} \binom{D-d\beta_{\text{small}}}{d_e-i} \beta_{\text{small}}^i \beta_{\text{large}}^{d_e-i}}{\binom{D}{d_e}}. \tag{1}$$

Figure 2 shows three cases of the BAxUS and HeSBO embedding's success probabilities. The success probability of the HeSBO embedding is given by $p_H(Y^*; d, d_e) = d!/((d-d_e)!d^{d_e})$ [36]. It is independent of $D$ but is shown for varying $d$-ranges on the $x$-axis, therefore the visual discrepancy. Except for when $D \to \infty$, the HeSBO embedding always has a non-zero probability of failing. The BAxUS embedding ensures that the success probability is one whenever $d \geq D$. Discontinuities in the curve of the BAxUS embedding occur due to the unequal bin sizes in the BAxUS embedding's success probability. The difference between the two embeddings in Figure 2 is predominant when $d_e$ is high. For small $d_e$, both embeddings have a high success probability for moderate $d$. If $d_e$ is high, for example $d_e = 20$, HeSBO requires $d = 1000$ to reach a success probability of approximately 0.8, while the BAxUS embedding has a success probability of 1 as soon as $d = D$. For finite $D$, the success probability of the HeSBO embedding is lower.

**Corollary 1.** *In the limit, the success probability of the* BAxUS *embedding is*

$$\lim_{D \to \infty} p_A(Y^*; D, d, d_e) = \frac{d!}{(d-d_e)!d^{d_e}} \tag{2}$$

*and hence matches* HESBO*'s success probability* $p_H(Y^*; d, d_e)$.

We now show that the BAXUS embedding is optimal among count-sketch type embeddings.

**Corollary 2.** *With the same input, target, and effective dimensionalities ($D$, $d$, and $d_e$), no count-sketch type embedding has a higher success probability than the* BAXUS *embedding.*

### 3.2 Trust-region approach

Similar to TURBO [19], BAXUS operates in TRs. TRs are hyper-rectangles in the input space; their shape is defined by their *base side length* $L$ and the GP length scales. A TR always has volume $L$, but the side length for each dimension is proportional to the corresponding length scale of the GP kernel. The idea of this construction is that short GP length scales signify important dimensions and reduce the extent of the hyper-rectangle in that dimension. A TR is shrunk when TURBO fails $\tau_{\text{fail}}$ consecutive times to make progress. If the algorithm consecutively makes $\tau_{\text{success}} = 3$ times progress, it expands the TR. It restarts when the base side length $L$ of the current TR falls below a threshold $L_{\min} := 2^{-7}$. In that case, it discards all observations for the TR and initializes a new TR on new samples. TRs enable TURBO to focus on regions of the space that are close to the incumbent. To choose the next evaluation point, TURBO minimizes posterior samples of the GP on a set of candidates drawn from the TR (Thompson sampling [59]).

TRs are an essential component of BAXUS as the target dimensionality grows exponentially. We use the same hyperparameter settings as TURBO [19] with the following exceptions: First, we change the criterion for when to restart a TR. Instead of restarting a TR when it becomes too small, we increase the target dimensionality by splitting each target dimension into several new bins unless BAXUS had already reached the input dimensionality. Secondly, we change the fail tolerance $\tau_{\text{fail}}$, such that BAXUS can roughly reach the input dimensionality in a fixed number of evaluations. The method for this is described in Section 3.4. We use the same algorithmic setup as TURBO with the aforementioned exceptions.

### 3.3 Splitting strategy

Starting in a low-dimensional embedded space, BAXUS successively increases the target dimensionality to increase the probability of containing the optimum. By Corollary 2, it is optimal to keep the distribution of contributing input dimensions in the target bins as uniform as possible. At each splitting point, BAXUS re-assigns the contributing input dimension of each target dimension to $b + 1$ bins, where $b$ is the number of new bins we create per target dimension, i.e., the target dimensionality grows exponentially. The number of splits required to reach some input dimensionality $D$ is logarithmic in $D$. Compared to the constant growth of the target dimensionality, BAXUS has a larger evaluation budget in each split and is less dependent on the value of $D$. The algorithm starts in target dimensionality $d_{\text{init}}$. After $k$ splits, the target dimensionality is $d_k = d_{\text{init}}(b + 1)^k$.

### 3.4 Controlling the fail tolerance

As the dimensionality grows, the GP needs more data to model the objective function accurately. We partition the evaluation budget into chunks of increasing sizes such that the sizes are proportional to the target dimensionalities. Each target dimensionality then has an individual evaluation budget. Our strategy is to control the budget that BAXUS has in each target dimensionality by the fail tolerance. We choose this indirect approach as we do not want to force BAXUS to increase the current target dimensionality if it makes considerable progress. The rule we propose is based on the fail tolerance $\tau_{\text{fail}}$ required for the increasing embedding to reach the input dimensionality $D$, under the assumption that BAXUS always fails in making progress. We calculate the number of splits $n$ required to reach $D$ by

$$D \leq d_{\text{init}} \cdot (b + 1)^n \Rightarrow n = \left\lfloor \log_{b+1} \frac{D}{d_{\text{init}}} \right\rceil, \tag{3}$$

with $\lfloor \cdot \rceil$ indicating rounding to the nearest integer. We assign each split $i$ a *split budget* $m_i^s$ that is proportional to $d_i$, such that $\sum_{i=1}^{n} m_i^s = m_D$, where $m_D$ is the evaluation budget until $D$ is reached: $m_i^s = \left\lfloor -\frac{b \cdot m_D \cdot d_i}{1-(b+1)^{n+1}} \right\rceil$. With that budget, we set the *raw adjusted fail tolerance* in the $i$-th split as $\tilde{\tau}_{\text{fail}}^i = \left\lfloor \frac{m_i^s}{k} \right\rceil$, with $k = \left\lfloor \log_{1/2} \frac{L_{\min}}{L_{\text{init}}} \right\rceil$ being the number of TR shrinkages until the TR is restarted.

We aim to reduce the fail tolerance to enable BAXUS to reach higher target dimensions when necessary. We are not interested in having even higher fail tolerances than the TURBO algorithm, which could occur with this rule. Therefore, we set the fail tolerance by taking the minimum of the raw adjusted fail tolerance and the TURBO fail tolerances for single batches, $\tau^i_{\text{fail}} = \max\left(1, \min\left(\tilde{\tau}^i_{\text{fail}}, d_i\right)\right)$, where $d_i$ is the target dimension in the $i$-th split.

**Setting the initial target dimension.** If the target dimensionality is fixed, increasing the target dimensionality at an exponential rate can lead to cases where none of the target dimensionalities $d_k$ are close to the input dimensionality $D$. For example, for a 600-dimensional problem, with $d_{\text{init}} = 1$, and $b = 3$, the difference between $d_k$ and $D$ is at least 344. Since the final target dimensionality of BAXUS is at most $D$, such cases are in disagreement with the exponential growth of the target dimensionality: Either the final target dimensionality is much less than the input dimensionality, or the last split adds only a few new dimensions. In the first case, we potentially reduce the success probability. In the latter case, we assign a too large budget to the final split. Therefore, we set the initial target dimensionality such that the final target dimensionality is as close to $D$ as possible:

$$d_{\text{init}} = \underset{i \in \{1,\ldots,b\}}{\arg\min} \min_{n \in \mathbb{N}_+} \left| i \cdot (b+1)^n - D \right| . \tag{4}$$

The optimal $n$ is given by Eq. (3).

# 4 Experimental evaluation

In this section, we evaluate the performance of BAXUS on a 388D hyperparameter optimization task, 124D design problem, and a collection of tasks that exhibit an active subspace. The code is available at `hiddenforanonymization`.

**The experimental setup.** We benchmark against TURBO [19], SAASBO [17], ALEBO [36], random search [2], CMA-ES [22], and HESBO [45], using the implementations provided by the respective authors with their settings, unless stated otherwise. For CMA-ES, we use the PYCMA [23] implementation. To show the effect of different choices of $d$, we run HESBO with $d = 10$ and $d = 20$. For ALEBO, we run the optimization with $d = 10$. We observed that ALEBO and SAASBO are constrained by their high runtime and memory consumption. The available hardware allowed up to 100 evaluations for SAASBO and 500 evaluations for ALEBO. Larger sampling budgets or higher target dimensions for ALEBO resulted in out-of-memory errors. We point out that limited scalability was expected for these methods, whereas the other methods scaled to considerably larger budgets, as required for scalable BO. We initialize each optimizer with ten initial samples and BAXUS with $b = 3$ and $m_D = 1000$. Plots show the mean performance with one standard error.

**The benchmarks.** We evaluate the selected algorithms on six benchmarks that differ considerably in their characteristics. Following [63], we augment the BRANIN2 and HARTMANN6 functions with additional dummy dimensions that have no influence on the function value. We use the 388D SVM benchmark and the 124D soft-constraint version of the MOPTA08 benchmark proposed in [17]. We set a budget of 1000 evaluations for MOPTA08, BRANIN2, and HARTMANN6 and of 2000 evaluations for the other benchmarks. Moreover, we stopped for BRANIN2 and HARTMANN6 when the simple regret dropped below .001. We also tested the algorithms on the 300D LASSO-HIGH and the 1000D LASSO-HARD benchmarks from LASSOBENCH [53]. These benchmarks have an effective dimensionality of $5\%$ of the input dimensionality, i.e., the LASSO-HIGH and LASSO-HARD benchmarks have 15 and 50 effective dimensions, respectively. To study the robustness to observational noise, we tested on noisy variants of LASSO-HARD and LASSO-HIGH.

## 4.1 Experimental results

We begin with the six noise-free benchmarks. Fig. 3 summarizes the performances. On MOPTA08, a 124D vehicle design problem, SAASBO makes slightly faster initial progress than BAXUS. We suspect that this benchmark has high effective dimensionality, such that BAXUS first needs to adapt the target dimensionality to make further progress. On the 388D SVM benchmark, BAXUS adapts to the appropriate target dimensionality where it can reach good function values faster than TURBO and CMA-ES. For this benchmark, Eriksson and Jankowiak [17] reported that SAASBO learned

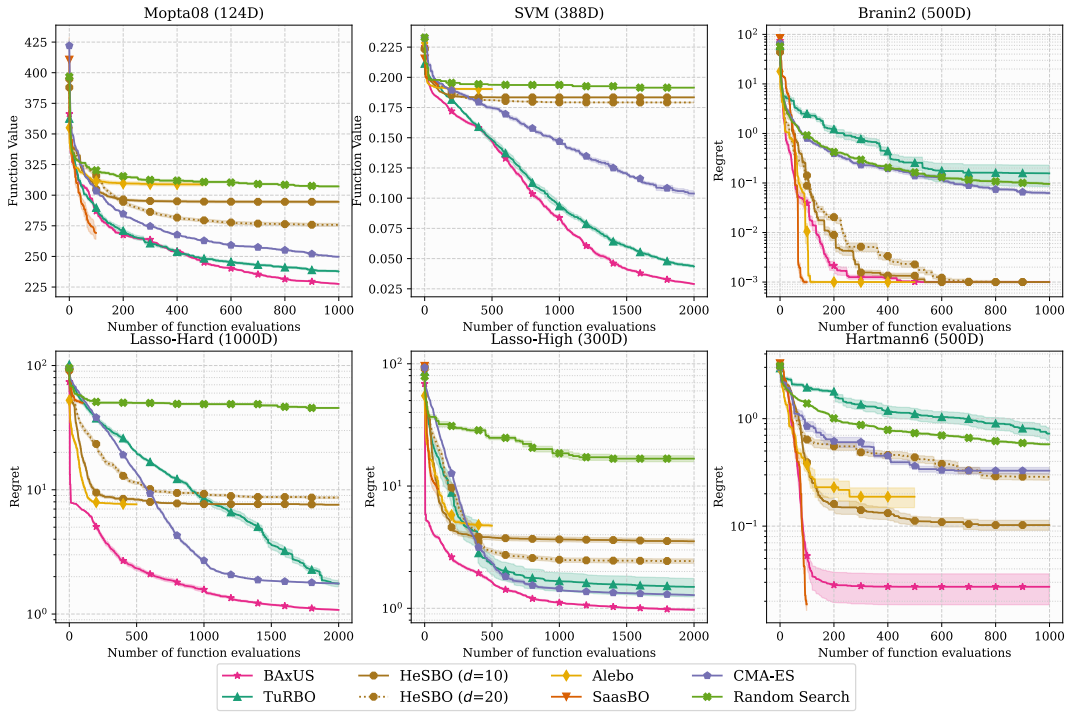


Figure 3: Top row: **124D MOPTA08 (l)**: BAXUS obtains the best solutions, followed by TURBO and CMA-ES. **388D SVM (c)**: BAXUS outperforms the other methods from the start. **500D BRANIN (r)**: SAASBO, BAXUS, ALEBO, and HESBO find an optimum; SAASBO and ALEBO converge fastest. Bottom row: **100D LASSO-HARD (l)** and **300D LASSO-HIGH (c)**: BAXUS outperforms the baselines. SAASBO, ALEBO, and HESBO struggle. **500D HARTMANN6 (r)**: SAASBO performs best, closely followed by BAXUS. The other methods show only slow progress or stagnate.

three effective dimensions. However, the fact that ALEBO and HESBO seem to stagnate after a few hundred evaluations, while BAXUS, TURBO, and CMA-ES find better solutions, indicates that optimizing more of the 385 kernel length scales of the SVM benchmark allows for better solutions. On the 500D HARTMANN6, SAASBO performs best, closely followed by BAXUS. ALEBO and HESBO are competitive initially but converge to suboptimal solutions. HESBO, BAXUS, SAASBO, ALEBO all find excellent solutions on the BRANIN benchmark, with the latter algorithms converging faster.

Next, we examine the performances on the 1000D LASSO-HARD and the 300D LASSO-HIGH that exhibit active subspaces, here without observational noise. BAXUS achieves considerably better solutions than all state-of-the-art methods. We also note that TURBO and CMA-ES perform better than SAASBO and ALEBO. While one may expect BAXUS to outperform TURBO and CMA-ES on these tasks with high input dimensions, it is surprising that SAASBO, HESBO, and ALEBO are not able to benefit from the present active subspace. Here BAXUS's strategy to adaptive expand the nested subspace is superior. Another crucial observation is that performances of BAXUS vary only slightly across runs. Thus, BAXUS is robust despite the stochastic construction of the embedding.

Across the broad collection of benchmarks, BAXUS is the only method to achieve constantly high performance.

**Noisy benchmarks.** We evaluate the algorithms also for tasks with observational noise. Fig. 4 summarizes the results. We observe that BAXUS achieves considerably better solutions for any number of observations than the competitors. Moreover, we note that the performances of SAASBO, CMA-ES, and HESBO ($d = 20$) degrade considerably on the LASSO-HIGH task compared to the noise-free formulation of the task studied above. BAXUS's performance is equally strong as for the noise-free case and keeps making progress after 1000 observations.

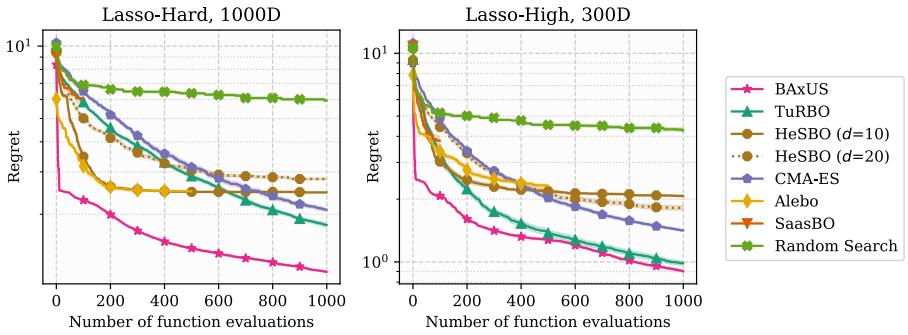


Figure 4: BAXUS outperforms the SOTA and in particular proves to be robust to noise on the **1000D LASSO-HARD (l)** and the **300D LASSO-HIGH (r)**. Note that CMA-ES performs considerably worse than on the noise-free versions of the benchmarks.

**BAXUS embedding ablation study.** To investigate whether the proposed family of nested random subspaces contributes to the superior performance of BAXUS, we replaced the new embedding with a similar family of nested HESBO embeddings. The results show that the proposed embedding provides a significant performance gain. Due to space constraints, the results were moved to Appendix A.

## 5 Discussion

High-dimensional Bayesian optimization is aspiring to unlock impactful applications broadly in science and industry. However, state-of-the-art methods suffer from limited scalability or, in some cases, require practitioners to 'guess' certain hyperparameters that critically impact the performance. This paper proposes BAXUS that works out-of-the-box and achieves considerably better performance for high-dimensional problems, as the comprehensive evaluation shows. A key idea is to scale up the dimensionality of the target subspace that the algorithm optimizes over. We apply a simple strategy that we find to work well across the board. However, we expect substantial headroom in tailoring this strategy to specific applications, either using domain expertise or by a more sophisticated data-driven approach that, for example, learns a suitable target space. Moreover, future work will explore extending BAXUS to structured domains, in particular the combinatorial spaces common in materials sciences and drug discovery.

**Societal impact.** Bayesian optimization has recently become a popular tool for tasks in drug discovery [46], chemical engineering [8, 24, 27, 52, 54], materials science [20, 25, 26, 47, 57, 61], aerospace engineering [1, 35, 39], robotics [9, 10, 37, 41, 49], and many more. This speaks to the progress that Bayesian optimization has made in becoming a robust and reliable 'off-the-shelf solver.' However, this promise is not yet fulfilled for the newer high-dimensional Bayesian optimization field that allows optimization over hundreds of 'tunable levers.' The applications mentioned above benefit from incorporating more such levers in the optimization: it allows for more detailed modeling of an aerospace design or a more granular control of a chemical reaction, to give some examples. The evaluation shows that the performance of state-of-the-art methods degenerates drastically for such high dimensions if the application does not meet specific requirements. Adding insult to the injury, such requirements like the dimensionality of an active subspace cannot be determined beforehand.

The proposed algorithm achieves a robust performance over a broad collection of tasks and thus will become a 'goto' optimizer for practitioners in other fields. Therefore, we release the BAXUS code.

# References


[1] R. Baptista and M. Poloczek. Bayesian optimization of combinatorial structures. In *International Conference on Machine Learning*, pages 462–471. PMLR, 2018.

[2] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[3] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for Hyper-Parameter Optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 24. Curran Associates, Inc., 2011.

[4] M. Binois. *Uncertainty quantification on Pareto fronts and high-dimensional strategies in Bayesian optimization, with applications in multi-objective automotive design*. PhD thesis, Ecole Nationale Supérieure des Mines de Saint-Etienne, 2015.

[5] M. Binois, D. Ginsbourger, and O. Roustant. A warped kernel improving robustness in Bayesian optimization via random embeddings. In *International Conference on Learning and Intelligent Optimization (LION)*, pages 281–286. Springer, 2015.

[6] 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, 2020.

[7] M. A. Bouhlel, N. Bartoli, R. G. Regis, A. Otsmane, and J. Morlier. Efficient global optimization for high-dimensional constrained problems by using the Kriging models combined with the partial least squares method. *Engineering Optimization*, 50(12):2038–2053, 2018.

[8] B. Burger, P. M. Maffettone, V. V. Gusev, C. M. Aitchison, Y. Bai, X. Wang, X. Li, B. M. Alston, B. Li, R. Clowes, et al. A mobile robotic chemist. *Nature*, 583(7815):237–241, 2020.

[9] R. Calandra, N. Gopalan, A. Seyfarth, J. Peters, and M. P. Deisenroth. Bayesian Gait Optimization for Bipedal Locomotion. In P. M. Pardalos, M. G. Resende, C. Vogiatzis, and J. L. Walteros, editors, *Learning and Intelligent Optimization*, pages 274–290, Cham, 2014. Springer International Publishing.

[10] R. Calandra, A. Seyfarth, J. Peters, and M. P. Deisenroth. Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence*, 76(1):5–23, 2016.

[11] A. Candelieri, R. Perego, and F. Archetti. Bayesian Optimization of Pump Operations in Water Distribution Systems. *J. of Global Optimization*, 71(1):213–235, may 2018. ISSN 0925-5001.

[12] M. Charikar, K. Chen, and M. Farach-Colton. Finding Frequent Items in Data Streams. In P. Widmayer, S. Eidenbenz, F. Triguero, R. Morales, R. Conejo, and M. Hennessy, editors, *Automata, Languages and Programming*, pages 693–703, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. ISBN 978-3-540-45465-6.

[13] J. Chen, G. Zhu, C. Yuan, and Y. Huang. Semi-supervised Embedding Learning for High-dimensional Bayesian Optimization. *arXiv preprint arXiv:2005.14601*, 2020.

[14] P. G. Constantine. *Active subspaces: Emerging ideas for dimension reduction in parameter studies*. SIAM, 2015.

[15] Z. Cosenza, R. Astudillo, P. Frazier, K. Baar, and D. E. Block. Multi-Information Source Bayesian Optimization of Culture Media for Cellular Agriculture. *Biotechnology and Bioengineering*, 2022.

[16] A. Ejjeh, L. Medvinsky, A. Councilman, H. Nehra, S. Sharma, V. Adve, L. Nardi, E. Nurvitadhi, and R. A. Rutenbar. HPVM2FPGA: Enabling True Hardware-Agnostic FPGA Programming. In *Proceedings of the 33rd IEEE International Conference on Application-specific Systems, Architectures, and Processors*, 2022.

[17] D. Eriksson and M. Jankowiak. High-dimensional Bayesian optimization with sparse axis-aligned subspaces. In C. de Campos and M. H. Maathuis, editors, *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pages 493–503. PMLR, 27–30 Jul 2021.

[18] D. Eriksson and M. Poloczek. Scalable Constrained Bayesian Optimization. In A. Banerjee and K. Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 730–738. PMLR, 13–15 Apr 2021.

[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 (NeurIPS)*, pages 5496–5507, 2019.

[20] P. I. Frazier and J. Wang. *Bayesian Optimization for Materials Design*, pages 45–75. Springer International Publishing, Cham, 2016. ISBN 978-3-319-23871-5.

[21] J. Gardner, C. Guo, K. Weinberger, R. Garnett, and R. Grosse. Discovering and exploiting additive structure for Bayesian optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 1311–1319, 2017.

[22] N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In *Proceedings of IEEE International Conference on Evolutionary Computation (ICEC)*, pages 312–317, 1996.

[23] N. Hansen, Y. Akimoto, and P. Baudis. CMA-ES/pycma on Github. Zenodo, DOI:10.5281/zenodo.2559634, Feb. 2019. Last accessed: 05/09/2022. License: BSD-3-Clause.

[24] F. Hase, L. M. Roch, C. Kreisbeck, and A. Aspuru-Guzik. Phoenics: a Bayesian optimizer for chemistry. *ACS central science*, 4(9):1134–1145, 2018.

[25] F. Häse, M. Aldeghi, R. J. Hickman, L. M. Roch, and A. Aspuru-Guzik. Gryffin: An algorithm for Bayesian optimization of categorical variables informed by expert knowledge. *Applied Physics Reviews*, 8(3):031406, 2021.

[26] H. C. Herbol, W. Hu, P. Frazier, P. Clancy, and M. Poloczek. Efficient search of compositional space for hybrid organic–inorganic perovskites via Bayesian optimization. *npj Computational Materials*, 4(1):1–7, 2018.

[27] J. M. Hernández-Lobato, J. Requeima, E. O. Pyzer-Knapp, and A. Aspuru-Guzik. Parallel and Distributed Thompson Sampling for Large-scale Accelerated Exploration of Chemical Space. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1470–1479. PMLR, 06–11 Aug 2017.

[28] Z. E. Hughes, M. A. Nguyen, J. Wang, Y. Liu, M. T. Swihart, M. Poloczek, P. I. Frazier, M. R. Knecht, and T. R. Walsh. Tuning materials-binding peptide sequences toward gold-and silver-binding selectivity with Bayesian optimization. *ACS nano*, 15(11):18260–18269, 2021.

[29] C. Hvarfner, D. Stoll, A. Souza, L. Nardi, M. Lindauer, and F. Hutter. PiBO: Augmenting Acquisition Functions with User Beliefs for Bayesian Optimization. In *International Conference on Learning Representations*, 2022.

[30] D. R. Jones. Large-scale multi-disciplinary mass optimization in the auto industry. In *MOPTA 2008 Conference (20 August 2008)*, 2008.

[31] K. Kandasamy, J. Schneider, and B. Póczos. High dimensional Bayesian optimisation and bandits via additive models. In *International conference on machine learning (ICML)*, pages 295–304, 2015.

[32] K. Kandasamy, W. Neiswanger, J. Schneider, B. Poczos, and E. P. Xing. Neural Architecture Search with Bayesian Optimisation and Optimal Transport. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31. Curran Associates, Inc., 2018.

[33] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[34] A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter. Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets. In A. Singh and J. Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 528–536. PMLR, 20–22 Apr 2017.

[35] R. Lam, M. Poloczek, P. Frazier, and K. E. Willcox. Advances in Bayesian optimization with applications in aerospace engineering. In *2018 AIAA Non-Deterministic Approaches Conference*, page 1656, 2018.

[36] B. Letham, R. Calandra, A. Rai, and E. Bakshy. Re-Examining Linear Embeddings for High-Dimensional Bayesian Optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 1546–1558. Curran Associates, Inc., 2020.

[37] D. J. Lizotte, T. Wang, M. H. Bowling, D. Schuurmans, et al. Automatic Gait Optimization With Gaussian Process Regression. In *IJCAI*, volume 7, pages 944–949, 2007.

[38] X. Lu, J. Gonzalez, Z. Dai, and N. D. Lawrence. Structured Variationally Auto-encoded Optimization. In J. G. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3273–3281. PMLR, 2018.

[39] T. W. Lukaczyk, P. Constantine, F. Palacios, and J. J. Alonso. Active subspaces for shape optimization. In *10th AIAA multidisciplinary design optimization conference*, page 1171, 2014.

[40] N. Maus, H. T. Jones, J. S. Moore, M. J. Kusner, J. Bradshaw, and J. R. Gardner. Local latent space Bayesian optimization over structured inputs. *arXiv*, 2022.

[41] M. Mayr, F. Ahmad, K. I. Chatzilygeroudis, L. Nardi, and V. Krüger. Skill-based Multi-objective Reinforcement Learning of Industrial Robot Tasks with Planning and Knowledge Integration. *CoRR*, abs/2203.10033, 2022.

[42] R. Moriconi, M. P. Deisenroth, and K. S. Sesh Kumar. High-dimensional Bayesian optimization using low-dimensional feature spaces. *Machine Learning*, 109(9):1925–1943, Sep 2020. ISSN 1573-0565.

[43] M. Mutny and A. Krause. Efficient high dimensional Bayesian optimization with additivity and quadrature Fourier features. *Advances in Neural Information Processing Systems*, 31, 2018.

[44] L. Nardi, D. Koeplinger, and K. Olukotun. Practical design space exploration. In *2019 IEEE 27th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 347–358. IEEE, 2019.

[45] A. Nayebi, A. Munteanu, 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 (PMLR)*, pages 4752–4761. PMLR, 09–15 Jun 2019.

[46] D. M. Negoescu, P. I. Frazier, and W. B. Powell. The Knowledge-Gradient Algorithm for Sequencing Experiments in Drug Discovery. *INFORMS Journal on Computing*, 23(3):346–363, 2011.

[47] D. Packwood. *Bayesian optimization for materials science*. Springer.

[48] G. Pedrielli and S. H. Ng. G-STAR: A new kriging-based trust region method for global optimization. In *2016 Winter Simulation Conference (WSC)*, pages 803–814. IEEE, 2016.

[49] A. Rai, R. Antonova, S. Song, W. Martin, H. Geyer, and C. Atkeson. Bayesian optimization using domain knowledge on the ATRIAS biped. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1771–1778. IEEE, 2018.

[50] R. G. Regis. Trust regions in Kriging-based optimization with expected improvement. *Engineering optimization*, 48(6):1037–1059, 2016.

[51] B. Ru, X. Wan, X. Dong, and M. Osborne. Interpretable Neural Architecture Search via Bayesian Optimisation with Weisfeiler-Lehman Kernels. In *International Conference on Learning Representations*, 2021.

[52] A. M. Schweidtmann, A. D. Clayton, N. Holmes, E. Bradford, R. A. Bourne, and A. A. Lapkin. Machine learning meets continuous flow chemistry: Automated optimization towards the Pareto front of multiple objectives. *Chemical Engineering Journal*, 352:277–282, 2018.

[53] K. Šehić, A. Gramfort, J. Salmon, and L. Nardi. LassoBench: A High-Dimensional Hyperparameter Optimization Benchmark Suite for Lasso. In *First Conference on Automated Machine Learning (Main Track)*, 2022.

[54] B. J. Shields, J. Stevens, J. Li, M. Parasram, F. Damani, J. I. M. Alvarado, J. M. Janey, R. P. Adams, and A. G. Doyle. Bayesian reaction optimization as a tool for chemical synthesis. *Nature*, 590(7844):89–96, 2021.

[55] J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian Optimization of Machine Learning Algorithms. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[56] F. J. Solis and R. J.-B. Wets. Minimization by Random Search Techniques. *Mathematics of Operations Research*, 6(1):19–30, 1981. ISSN 0364765X, 15265471.

[57] A. Souza, L. B. Oliveira, S. Hollatz, M. Feldman, K. Olukotun, J. M. Holton, A. E. Cohen, and L. Nardi. DeepFreak: Learning crystallography diffraction patterns with automated machine learning. *arXiv preprint arXiv:1904.11834*, 2019.

[58] L. Tallorin, J. Wang, W. E. Kim, S. Sahu, N. M. Kosa, P. Yang, M. Thompson, M. K. Gilson, P. I. Frazier, M. D. Burkart, et al. Discovering de novo peptide substrates for enzymes using machine learning. *Nature communications*, 9(1):1–10, 2018.

[59] W. R. Thompson. On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples. *Biometrika*, 25(3/4):285–294, 1933. ISSN 00063444.

[60] A. Tripp, E. Daxberger, and J. M. Hernández-Lobato. Sample-Efficient Optimization in the Latent Space of Deep Generative Models via Weighted Retraining. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 11259–11272. Curran Associates, Inc., 2020.

[61] T. Ueno, T. D. Rhone, Z. Hou, T. Mizoguchi, and K. Tsuda. COMBO: An efficient Bayesian optimization library for materials science. *Materials Discovery*, 4:18–21, 2016. ISSN 2352-9245.

[62] X. Wan, V. Nguyen, H. Ha, B. Ru, C. Lu, and M. A. Osborne. Think Global and Act Local: Bayesian Optimisation over High-Dimensional Categorical and Mixed Search Spaces. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10663–10674. PMLR, 18–24 Jul 2021.

[63] Z. Wang, F. Hutter, M. Zoghi, D. Matheson, and N. de Feitas. Bayesian optimization in a billion dimensions via random embeddings. *Journal of Artificial Intelligence Research (JAIR)*, 55: 361–387, 2016.

[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*, pages 745–754, 2018.

[65] C. K. Williams and C. E. Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[66] J. Zhou, Z. Yang, Y. Si, L. Kang, H. Li, M. Wang, and Z. Zhang. A trust-region parallel bayesian optimization method for simulation-driven antenna design. *IEEE Transactions on Antennas and Propagation*, 69(7):3966–3981, 2020.

## Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 5.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix C.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix F.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Appendix F.
   (b) Did you mention the license of the assets? [Yes] See Appendix F.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

# A Ablation study for the BAxUS embedding

We conduct an ablation study to investigate the difference between the BAxUS and HeSBO embeddings. We run TuRBO, a third-party optimizer by [19], in an embedded subspace with the two different embeddings. We use a version of Ackley10 (ten effective dimensions, i.e., $d_e = 10$), where we shift the optimum away from the origin with a random vector $\boldsymbol{\delta} \in [-32.768, 32.768]^{d_e}$ with $\delta_i \sim \mathcal{U}(-32.768, 32.768)$. The function we optimize is then

$$f_{\text{ShiftedAckley10}}(\boldsymbol{x}) = f_{\text{Ackley10}}(\boldsymbol{x} + \boldsymbol{\delta}).$$

We adjust the boundaries of the search space such that $f_{\text{ShiftedAckley10}}$ is evaluated on the domain $\mathcal{X} = [-32.768, 32.768]^{d_e}$. The reason for shifting the optimum is that the original Ackley function has its optimum at the origin. In that case, any count-sketch type embedding can represent this optimum, even if all the important input dimensions are mapped to the same target dimension.

We add 10 dummy dimensions, such that $D = 30$ and set $d = 20$. With this problem-setting, the BAxUS and HeSBO embeddings have a probability of approximately 0.27 and 0.07 to contain the optimum, respectively.

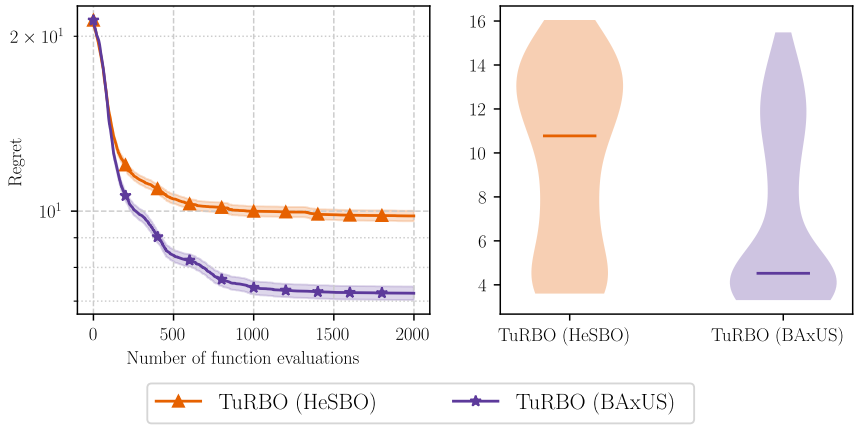


Figure 5: **Left**: the BAxUS embedding gives better optimization performance on the shifted Ackley10 function: TuRBO in embedded subspaces of the BAxUS and HeSBO embeddings. The BAxUS embedding has a higher probability to contain the optimum. **Right**: the distribution of the final incumbents (lower the better). The horizontal bars show the median.

The left side of Figure 5 shows the incumbent mean for TuRBO in the two different embedded subspaces. The shaded regions show one standard error. TuRBO in an BAxUS embedding has significantly better optimization performance than in a HeSBO embedding. The right side of Figure 5 shows the distributions of the final incumbents and their median. The BAxUS embedding has a significantly lower median and only rarely a similarly bad embedding as the HeSBO method.

We perform a two-sided Wilcoxon rank-sum statistical test to check the difference between the best observed function values for the two embeddings. The difference is significant with $p \approx 0.00001$.

The performance difference between the two embeddings depends on the characteristics of the function and the different dimensionalities, the input dimensionality $D$, the target dimensionality $d$, and the effective dimensionality $d_e$. For problems with a few effective dimensions and many input dimensions, the BAxUS and HeSBO embeddings become more similar (see Figure 2). However, by Corollary 2, the BAxUS embedding is, in expectation and in terms of success probability, always better than the HeSBO embedding for arbitrary sparse axis-aligned functions.

For functions having the optimum at the origin, both embeddings contain that optimum regardless of $d$: Even if all important input dimensions are mapped to the same target dimension, the optimum in the input space can be reached by "setting" this particular target dimension to zero.

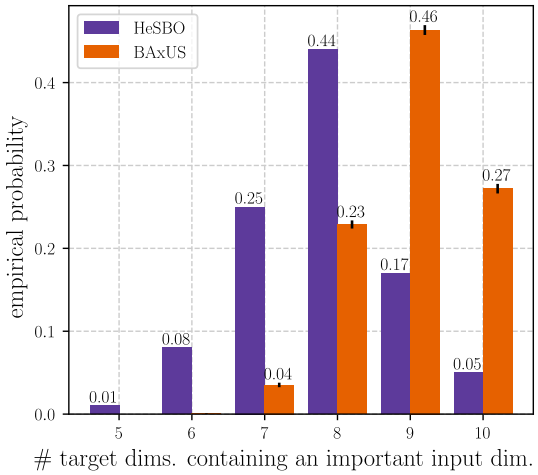


Figure 6: TuRBO in the BAxUS and HeSBO embeddings on the shifted Ackley10 function with input dimensionality $D = 30$, target dimensionality $d = 20$, and effective dimensionality $d_e = 10$: The $x$-axis shows in how many different target dimensions the important input dimensions lie on average (more: better). The black lines at the bar tips give the standard error.

**Distribution of important input dimensions.** The success probability studied in Section 3 and above only considers the case where all the important input dimensions are mapped to different target dimensions. It might not be necessary to map all the important input dimensions to different target dimensions, but it is generally beneficial if they spread out. For example on Mopta08 or Svm in Figure 3 (top left, top center), BAxUS makes initial progress, then seems to be in a too low embedding before the target dimensionality is increased and BAxUS makes further progress. If only the case where all $d_e$ important input dimensions are mapped to individual target dimensions allowed to make progress, BAxUS would not make any progress until it reaches a suitable embedding.

Even though it is often possible to make progress without mapping all important input dimensions to individual target dimensions, it is undoubtedly beneficial if, for a given target dimensionality, they map to more individual target dimensions; The more the important input dimension spread out, the more the effective subspace can be covered.

Figure 6 shows a histogram of the number of individual target dimensions the $d_e = 10$ important input dimensions map to in the example from above for the BAxUS and HeSBO embeddings. The bars on the very right approximately give the success probability for the HeSBO and BAxUS embeddings - they correspond to a success (see Definition 2). The BAxUS embedding maps the 10 important input dimensions to 9 or 10 individual target dimensions in around 74% of the cases; in the HeSBO embedding, this happens in less than half as many cases (22%).

**TuRBO with BAxUS embedding vs. BAxUS.** We compare the simple idea of running TuRBO in a BAxUS embedding with BAxUS. We run this simple approach for 11 different target dimensionalities $d$ $(2, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100)$ on the Lasso-Hard benchmark and show the results with a sequential color map in Figure 7. Only the first $d = 2$-dimensional embedding achieves the same initial speedup as BAxUS, which is expected as BAxUS starts in a similarly low-dimensional initial embedding. However, the fixed embedding cannot explore the input space sufficiently and has the worst final solution. High-dimensional fixed embeddings have more freedom in exploring the input space; however, they suffer from slower initial optimization performance.

BAxUS has the same initial speedup as the two-dimensional fixed embedding but can explore the space further by increasing the dimensionality of its embedding.

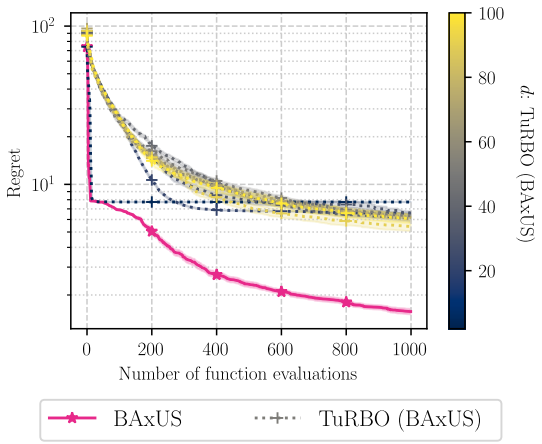


Figure 7: BAxUS and TuRBO with BAxUS embeddings of different target dimensionalities on Lasso-Hard: We run TuRBO with the BAxUS embedding for target dimensionalities $d = 2, 10, 20, 30, \ldots, 100$ and compare to BAxUS.

## B Additional Lasso benchmark

In addition to the synthetic Lasso-High and Lasso-Hard benchmarks studied in Section 4, we evaluate BAxUS on the Lasso-DNA benchmark from LassoBench [53]. The Lasso-DNA benchmark is a biomedical classification task, taking binarized DNA sequences as input [53].

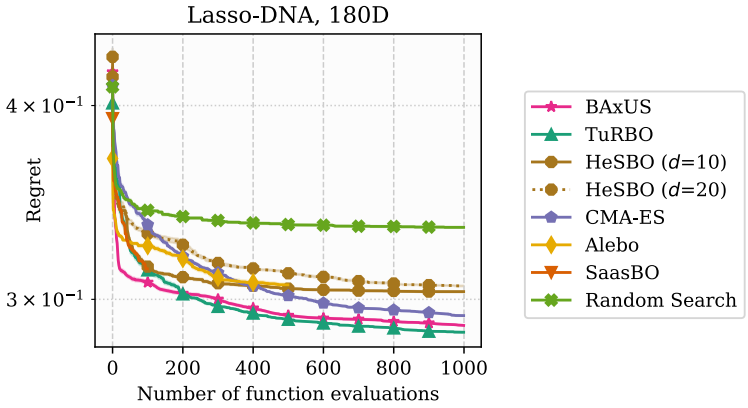


Figure 8: BAxUS and baselines on Lasso-DNA. As before, BAxUS makes considerable progress in the beginning and converges faster than TuRBO and CMA-ES.

Figure 8 shows the mean performance of BAxUS on the Lasso-DNA. Each line shows the incumbent mean; the shaded regions around the lines show one standard error. We see the same qualitative behavior as discussed in Section 4: BAxUS reaches a good initial solution faster than any other method and converges quickly.

## C Theoretical foundation for the BAxUS embedding

**Definition 3 (Optima-preserving count-sketch type embedding).** An embedding of the count-sketch type is *optima-preserving* if each target dimension contains at most one important input dimension.

**Definition 4 (Sparse axis-aligned function).** A function $f : \mathcal{X} \to \mathbb{R}$ with $\mathcal{X} = [-1, 1]^D$ is called *sparse*, if there exist an effective subspace $\mathcal{Y} \subseteq \mathbb{R}^{d_e}$ with $d_e < D$ and a projection matrix

$S^\intercal \in \mathbb{R}^{d_e \times D}$, such that for any $\boldsymbol{x} \in \mathcal{X}$ there exists a $\boldsymbol{y} \in \mathcal{Y}$ so that $f(\boldsymbol{x}) = f(S^\intercal \boldsymbol{y})$. It is further called *axis-aligned*, if $S^\intercal$ is of count-sketch type and $\mathcal{Y} = [-1, 1]^{d_e}$.

### C.1 Proof of Lemma 1

**Lemma 1** (**Success probability of the BAxUS embedding**). Let $D$ be the input dimensionality and $d \geq d_e$ the dimensionality of the embedding. Let $\beta_{\text{small}} = \lfloor \frac{D}{d} \rfloor$ and $\beta_{\text{large}} = \lceil \frac{D}{d} \rceil$ be the small and large bin sizes. Then, the probability of $Y^*$ (see Definition 2) with the BAxUS embedding is

$$p_A(Y^*; D, d, d_e) = \frac{\sum_{i=0}^{d_e} \binom{d(1+\beta_{\text{small}})-D}{i} \binom{D-d\beta_{\text{small}}}{d_e-i} \beta_{\text{small}}^i \beta_{\text{large}}^{d_e-i}}{\binom{D}{d_e}}. \tag{5}$$

*Proof.* The assignment of input dimensions to target dimensions and the signs of the input dimensions fully define the HESBO and BAxUS embeddings. Note that the signs do not have any effect on $p_A(Y^*; D, d, d_e)$ because they only correspond to "flipping" the input dimension in the target space.

If and only if an assignment is optima-preserving, it is possible to find a point in $\mathcal{Y}$ that maps to an optimum in $\mathcal{X}$ for any $f$. The "only if" is true because $f$ is assumed to be sparse and axis-aligned with $d_e$ effective dimensions. This means that the optima in $\mathcal{X}$ only change their function values along the $d_e$ effective dimensions. Suppose it is possible to find a point in $\mathcal{Y}$ that maps to an arbitrary optimum in $\mathcal{X}$. In that case, the assignment has to be optima-preserving because it has to be able to adjust all the $d_e$ effective dimensions in $\mathcal{X}$ individually. However, this requires each important input dimension to be mapped to a different target dimension. Otherwise, there would be at least two important input dimensions that cannot be changed independently. Therefore, the probability of $Y^*$ is equal to the probability of an optima-preserving assignment.

As all assignments are equally likely, the probability of an assignment to be optima-preserving is equal to the number of possible optima-preserving assignments divided by the total number of assignments.

Let us first assume that $\beta_{\text{small}} = \beta_{\text{large}} := \beta$, i.e., all target dimensions have the same number of input dimensions and $d$ divides $D$. There are $\binom{d}{d_e}$ ways of assigning $d_e$ important dimensions to $d$ different target dimensions. Given one effective dimension, there are $\beta$ ways in which this dimension can map to the target dimension. Therefore, for the case of homogeneous bin sizes, the success probability is given by

$$p_A(Y^*; D, d, d_e) = \frac{\beta^{d_e} \binom{d}{d_e}}{\binom{D}{d_e}}. \tag{6}$$

Next, we generalize Eq. (6) for cases where $d$ does not divide $D$. Then, one must consider all possible splits of target dimensions between small and large numbers of contributing input dimensions. This gives Eq. (5).

It is easy to see that Eq. (5) is equivalent to the uniform formulation in Eq. (6). Consider the fact that $\binom{0}{0} = 1$ and $\binom{0}{x} = 0$ for all $x \in \mathbb{N}_{++}$. The quantities $d(1 + \beta_{\text{small}}) - D$ and $D - d\beta_{\text{small}}$ can be interpreted as the number of target bins with fewer and more contributing input dimensions, respectively. Then, for the uniform case $\beta_{\text{small}} = \beta_{\text{large}} = \beta$, $d(1 + \beta_{\text{small}}) - D = d$, and $D - d\beta_{\text{small}} = 0$. Therefore, the success probability for the almost-uniform case is given by

$$\begin{aligned} p_A(Y^*; D, d, d_e) &= \frac{\sum_{i=0}^{d_e} \binom{d(1+\beta_{\text{small}})-D}{i} \binom{D-d\beta_{\text{small}}}{d_e-i} \beta_{\text{small}}^i \beta_{\text{large}}^{d_e-i}}{\binom{D}{d_e}} \\ &= \frac{\sum_{i=0}^{d_e} \binom{d}{i} \binom{0}{d_e-i} \beta^i \beta^{d_e-i}}{\binom{D}{d_e}} \\ &= \frac{\beta^{d_e} \sum_{i=0}^{d_e} \binom{d}{i} \binom{0}{d_e-i}}{\binom{D}{d_e}} = \frac{\beta^{d_e} \binom{d}{d_e}}{\binom{D}{d_e}}, \end{aligned}$$

where the last equality is true because the sum is zero unless $i = d_e$.

□

### C.2 Proof of Corollary 1

**Corollary 1.** In the limit, the success probability of the BAXUS embedding is

$$\lim_{D\to\infty} p_A(Y^*; D, d, d_e) = \frac{d!}{(d-d_e)!d^{d_e}} \tag{7}$$

and hence matches HESBO's success probability $p_H(Y^*; d, d_e)$.

*Proof.* The HESBO embedding is constructed by choosing a target dimension and a sign uniformly at random for each input dimension. With this construction, the HESBO embedding is independent of $D$. [36] give an expression for the HESBO success probability:

$$p_H(Y^*; d, d_e) = \frac{d!}{(d-d_e)!d^{d_e}}. \tag{8}$$

The validity of this expression can be seen by considering the probability that $d_e$ important input dimensions are mapped to $d$ target dimensions without collision if $d \geq d_e$:

$$p_H(Y^*; d, d_e) = \frac{d}{d} \cdot \frac{d-1}{d} \cdot \ldots \cdot \frac{d-d_e+1}{d} = \prod_{i=0}^{d_e-1}\left(1-\frac{i}{d}\right) = \frac{d!}{(d-d_e)!d^{d_e}} \tag{9}$$

Clearly, as $D \to \infty$, the HESBO embedding has uniform bin sizes; intuitively, the HESBO embedding and our formulation are equivalent in the limit.

We consider $\lim_{\beta\to\infty}$, which is equivalent to $\lim_{D\to\infty}$ as $\beta = \frac{D}{d}$ and $d$ is fixed. We further use the uniform formulation in Eq. (6) as a starting point because all bins are of infinite size in the limit.

$$\begin{aligned}
\lim_{\beta\to\infty} \frac{\beta^{d_e}\binom{d}{d_e}}{\binom{D}{d_e}} &= \lim_{\beta\to\infty} \frac{\beta^{d_e} d!(D-d_e)!}{D!(d-d_e)!} && (10)\\
&= \lim_{\beta\to\infty} \frac{\beta^{d_e} d!(\beta d-d_e)!}{(\beta d)!(d-d_e)!} && (11)\\
&= \lim_{\beta\to\infty} \frac{d!}{(d-d_e)!}\frac{\beta^{d_e}(\beta d-d_e)!}{(\beta d)!} && (12)\\
&\overset{\text{Stirling}}{=} \lim_{\beta\to\infty} \beta^{d_e}\frac{d!}{(d-d_e)!}\frac{\sqrt{2\pi(\beta d-d_e)}\left(\frac{\beta d-d_e}{e}\right)^{\beta d-d_e}}{\sqrt{2\pi\beta d}\left(\frac{\beta d}{e}\right)^{\beta d}} && (13)\\
&= \lim_{\beta\to\infty} \beta^{d_e}\frac{d!}{(d-d_e)!}\sqrt{\frac{\beta d-d_e}{\beta d}}e^{d_e}\frac{(\beta d-d_e)^{\beta d-d_e}}{(\beta d)^{\beta d}} && (14)\\
&= \lim_{\beta\to\infty} \beta^{d_e}\frac{d!}{(d-d_e)!}\underbrace{\sqrt{\frac{\beta d-d_e}{\beta d}}}_{\to 1}e^{d_e}\underbrace{\left(\frac{\beta d-d_e}{\beta d}\right)^{\beta d}}_{\to e^{-d_e}}\frac{1}{(\beta d-d_e)^{d_e}} && (15)\\
&= \lim_{\beta\to\infty} \frac{d!}{(d-d_e)!}\underbrace{\sqrt{\frac{\beta d-d_e}{\beta d}}}_{\to 1}e^{d_e}\underbrace{\left(\frac{\beta d-d_e}{\beta d}\right)^{\beta d}}_{\to e^{-d_e}}\underbrace{\frac{\beta^{d_e}}{(\beta d-d_e)^{d_e}}}_{\to \frac{1}{d^{d_e}}} && (16)\\
&= \frac{d!}{(d-d_e)!d^{d_e}} && (17)
\end{aligned}$$

Eq. (17) is the failure probability of the HESBO embedding [36]. □

### C.3 Proof of Corollary 2

**Corollary 2.** With the same input, target, and effective dimensionalities ($D$, $d$, and $d_e$), no count-sketch type embedding has a higher success probability than the BAXUS embedding.

*Proof.* By Definition 1, an embedding matrix $S \in \{0, \pm 1\}^{D \times d}$ is of the count-sketch type if each row in $S$ has precisely one non-zero entry. Such an embedding can always be interpreted as disjoint sets of signed input dimensions that are assigned to different target dimensions: For the $i$-th input dimension, find the column where the non-zero entry lies. The number of the respective column gives the target dimension; the entry in the matrix itself gives the sign. Conversely, each target dimension has a set of contributing input dimensions, and we call the set of input dimensions mapping to a target dimension a "bin". The sign of the input dimensions does not influence the success probability as it does not influence the ability of an embedding to contain the optimum.

We will prove that the BAxUS embedding is optimal because every other embedding of the count-sketch type has a success probability that is lower or equal to the success probability of the BAxUS embedding. We start the proof by giving the success probability for arbitrary bin sizes.

Let $i_0 = 0$. Let $\beta_p$ be the bin size of the $p$-th bin. The success probability for arbitrary bin sizes is given by

$$p_{\text{general}}(Y^*; D, d, d_e) = \frac{\sum_{i_1=0}^{1} \sum_{i_2=0}^{1} \cdots \sum_{i_d=0}^{1} \mathbb{1}_{(\sum_{n=1}^{d} i_n)=d_e} \prod_{n=1}^{d} \beta_n^{i_n}}{\binom{D}{d_e}}, \tag{18}$$

with $\beta_i > 0$, $\beta_d = D - \sum_{n=1}^{d-1} \beta_n$, and $d \geq d_e$.

Eq. (18) considers all assignments of the $d_e$ effective dimensions to the $d$ different bins. The indicator function considers only cases where the important input dimensions mapped to the bins sum up to $d_e$. Next, if the $n$-th bin has one or more contributing input dimensions, we consider all the locations $\beta_n$ in which the important input dimension can lie. Otherwise, the $\beta$-term of this bin is always 1, the neutral element of multiplication.

Eq. (18) is the general failure probability of any embedding of the count-sketch type as the success probability only depends on the assignment of important input dimensions to target dimensions.

We now prove that any count sketch embedding has a success probability that is less or equal to the success probability of the BAxUS embedding.

$$p_{\text{general}}(Y^*; D, d, d_e) = \frac{\overbrace{\sum_{i_1=0}^{1} \sum_{i_2=0}^{1} \cdots \sum_{i_d=0}^{1} \mathbb{1}_{(\sum_{n=1}^{d} i_n)=d_e}}^{=\binom{d}{d_e}} \overbrace{\prod_{n=1}^{d} \beta_n^{i_n}}^{\leq \beta^{d_e}}}{\binom{D}{d_e}} \leq \frac{\beta^{d_e} \binom{d}{d_e}}{\binom{D}{d_e}}, \tag{19}$$

where $\beta = D/d$. The first term in the nominator of Eq. (18) is equal to $\binom{d}{d_e}$ because it counts all cases in which $d_e$ important input dimensions are assigned to $d$ different bins in an arbitrary manner. For the inequality $\prod_{n=1}^{d} \beta_n^{i_n} \leq \beta^{d_e}$, note that $\sum_{n=1}^{d} \beta_n = D$. Then, this inequality holds by the inequality of arithmetic and geometric means (AM-GM inequality):

$$\frac{a_1 + a_2 + \ldots + a_n}{n} \geq \sqrt[n]{a_1 a_2 \ldots a_n} \quad \text{for } a_i > 0,$$

with equality if and only if $a_i = a_j$ for $i, j \in \{1, \ldots, n\}$.

Finally, observe that $\prod_{n=1}^{d} \beta_n^{i_n}$ is a convex function because all factors in the product are $\in \mathbb{N}_{++}$. Therefore, the more the bin size deviates from $D/d$, the lower the failure probability (and equality only holds if all bin sizes are equal by AM-GM).

By construction, the BAxUS embedding is as close to homogeneous bin sizes as possible. It corresponds to the RHS of Eq. (19) if $d$ divides $D$; otherwise, it is as close as possible to it as the bin sizes differ by at most 1. Therefore, the BAxUS embedding is optimal in terms of success probability. □

# D Global convergence of BAxUS

We will now prove the global convergence of BAxUS. The proof idea is similar to Eriksson and Poloczek [18] but relaxes the assumption of a unique global minimizer. Since, based on our

assumptions, $f$ is sparse and axis-aligned (see Definition 4), i.e., there exists a set of dimensions of $f$ that do not influence the function value whatsoever, it is crucial to make this relaxation. BAxUS does not have to converge to a particular configuration of these dimensions — they can take any arbitrary value.

**Theorem 2** (**BAxUS convergence**). *Under the following assumptions:*

1. *The initial points for* BAxUS *are chosen such that* $\forall \delta > 0$ *and* $\boldsymbol{x} \in [-1, 1]^D$, $\exists\, \nu(\boldsymbol{x}, \delta) > 0$, *such that the probability that at least one point in* $\{\boldsymbol{x}_i\}$ *ends up in a ball centered at* $\boldsymbol{x}$ *with radius* $\delta$ *is at least* $\nu(\boldsymbol{x}, \delta)$*;*

2. *The objective* $f$ *is bounded in* $\mathcal{X}$*, i.e.,* $|f(\boldsymbol{x})| < C$*, where* $\boldsymbol{x} \in \mathcal{X}$ *and* $C \in \mathbb{R}_{++}$*;*

3. BAxUS *considers any sampled point an improvement only if it improves the current best solution by at least some constant* $\gamma \in \mathbb{R}_{++}$*;*

4. $D$ *is finite;*

5. *At least one of the minima of* $f$ *is not singularly discontinuous;*

BAxUS *converges to a global minimizer* $\boldsymbol{x}^* \in \arg\min_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x})$ *for a noiseless function* $f$.

*Proof.* By Assumption (4), the fail tolerance is always finite since it is always bounded by the target dimension ($\forall i\, \tau^i_{\text{fail}} \leq d_i$) which is at most equal to $D$ ($\forall i\, d_i \leq D$). By Assumption (3), we need to make a progress of at least $\gamma$ to consider a new point an improvement. By this construction and the fact that $f$ is bounded (Assumption (2)), BAxUS increases the target dimensionality in a finite number of function evaluations and eventually reaches the input dimensionality $D$.

Once BAxUS reaches $D$, it behaves like TuRBO [19]. Eriksson and Poloczek [18] proved global convergence for TuRBO under the assumption of the existence of only one global minimizer. However, by the convergence theorem for global search by Solis and Wets [56], page 20, BAxUS converges to some global minimizer that lies in a continuous region with non-zero probability (Assumption (5)). □

# E The nested family of random embeddings

We describe the method for increasing the target dimensionality under the retention of the observations. Suppose that we have collected $n$ observations and are in target dimension $d$ when Algorithm 2 is invoked. Algorithm 2 loops over the target dimensions $1, \ldots, d$. For each target dimension, the contributing input dimensions are re-assigned to new bins. If the number of contributing input dimensions is less than $b + 1$ ($b$: number of new bins), then it is not possible to re-assign the contributing input dimensions to $b + 1$ bins. Therefore we re-assign the contributing input dimensions to $\hat{b} = \min(b, l_s - 1)$ bins, where $l_s$ is the number of contributing input dimensions to the $s$-th target dimension. This also ensures that the target dimension never grows larger than $D$ in the BAxUS embedding. We evenly distribute the $l_s$ contributing input dimensions across the $\hat{b}$ bins by again using the BAxUS embedding. This gives a smaller projection matrix $\tilde{S}^\intercal$ which we finally use to update $S^\intercal$:

**Algorithm 2** Observation-preserving embedding increase

**Input:** transposed embedding matrix $S^\intercal$, number of new bins per latent dimension $b$, observed points $Y \in [-1, 1]^{n \times d}$.

**Output:** updated transposed embedding matrix $S^\intercal$ and updated observation matrix $Y$

**for** $s \in \{1, \dots, d\}$ **do**

  $\boldsymbol{D}^s \leftarrow$ contributing input dimensions of $s$-th latent dimension.

  $l_s \leftarrow |\boldsymbol{D}^s|$

  $\hat{b} \leftarrow \min(b, l_s - 1)$ ▷ If $l_s - 1 < b$, we can at most create $l_s - 1$ new bins.

  Copy and append $s$-th column of $Y$ $\hat{b}$ times at the end of $Y$.

  Add $\hat{b}$ zero columns at the end of $S^\intercal$.

  $\boldsymbol{\sigma} \leftarrow$ signs of dimensions $\in \boldsymbol{D}^s$.

  $\tilde{S}^\intercal \leftarrow$ `BAxUS-Embedding`$(l_s, \hat{b} + 1)$ ▷ Re-assign input dims. homogen., $\tilde{S}^\intercal \in \{0, \pm 1\}^{l_s \times \hat{b}+1}$

  **for** $i \in \{1, \dots, l_s\}, j \in \{1, \dots, \hat{b} + 1\}$ **do**

    **if** $\tilde{S}^\intercal_{ij} \neq 0$ **then**

      **if** $j > 1$ **then** ▷ Move values that fall into new bins to end of $S^\intercal$.

        $S^\intercal_{\boldsymbol{D}^s_i, \hat{d} - \hat{b} - 1 + j} \leftarrow \boldsymbol{\sigma}_i$ ▷ $\hat{d}$: columns of $S^\intercal$

        $S^\intercal_{\boldsymbol{D}^s_i, s} \leftarrow 0$ ▷ Set value in "old" column to zero.

**Return** $S^\intercal$ and $Y$.

# F Additional implementation details

We benchmark against SAASBO, TuRBO, HeSBO, Alebo, and CMA-ES:

- For SAASBO, we use the implementation from [17] (`https://github.com/martinjankowiak/saasbo`, license: none, last accessed: 05/09/2022).
- For TuRBO, we use the implementation from [19] (`https://github.com/uber-research/TuRBO`, license: `Uber`, last accessed: 05/09/2022).
- For HeSBO and Alebo, we use the implementation from [36] (`https://github.com/facebookresearch/alebo`, license: `CC BY-NC 4.0`, last accessed: 05/09/2022).
- For the Lasso benchmarks, we use the implementation from [53] (`https://github.com/ksehic/LassoBench`, license: `MIT` and `BSD-3-Clause`, last accessed: 05/09/2022).

We use `GPyTorch` to train the GP with the following setup: We place a top-hat prior one the Gaussian likelihood noise, the signal variance, and the length scales of the ARD kernel. The interval for the noise is $[0.005, 0.2]$, for the signal variance $[0.05, 20]$, and for the lengthscales $[0.005, 10]$.

We evaluate on the synthetic Branin2[1] and Hartmann6[2] functions. Since we augment the function with dummy dimensions, we use the same domain for $x_1$ and $x_2$, namely $[-5, 15]^D$ for Branin2 and $[0, 1]^D$ for Hartmann6.

Similar to TuRBO, we sample a $\min(100d_n, 5000)$-element Sobol sequence on which we minimize the posterior sample. To maximize the marginal log-likelihood of the GP, we sample 100 initial hyperparameter configurations. The ten best samples are further optimized using the Adam optimizer for $50$ steps.

We ran the experiments for approximately $7,000$ core hours on Intel Xeon Gold 6130 CPUs provided by a compute cluster.

[1] See `https://www.sfu.ca/~ssurjano/branin.html`, last accessed: 05/09/2022

[2] See `https://www.sfu.ca/~ssurjano/hart6.html`, last accessed: 05/09/2022