Are Random Decompositions all we need in High Dimensional Bayesian Optimisation?

Juliusz Ziomek [1] **Haitham Bou-Ammar** [1]

Abstract

Learning decompositions of expensive-to-evaluate black-box functions promises to scale Bayesian optimisation (BO) to high-dimensional problems. However, the success of these techniques depends on finding proper decompositions that accurately represent the black-box. While previous works learn those decompositions based on data, we investigate data-independent decomposition sampling rules in this paper. We find that data-driven learners of decompositions can be easily misled towards local decompositions that do not hold globally across the search space. Then, we formally show that a random tree-based decomposition sampler exhibits favourable theoretical guarantees that effectively trade off maximal information gain and functional mismatch between the actual black-box and its surrogate as provided by the decomposition. Those results motivate the development of the random decomposition upper-confidence bound algorithm (RDUCB) that is straightforward to implement - (almost) plug-and-play - and, surprisingly, yields significant empirical gains compared to the previous state-of-the-art on a comprehensive set of benchmarks. We also confirm the plug-and-play nature of our modelling component by integrating our method with HEBO (Cowen-Rivers et al., 2022), showing improved practical gains in the highest dimensional tasks from the Bayesmark problem suite.

1. Introduction

Although Bayesian optimisation (BO) demonstrated impressive successes in low-dimensional domains (Marchant & Ramos, 2012; Shahriari et al., 2015; Kandasamy et al., 2017; 2018; Grosnit et al., 2022), scaling BO to high-dimensional and expensive to evaluate black-box functions has proved challenging. Among the many proposed approaches ranging from linear to non-linear projections (Wang et al., 2016; Rana et al., 2017; Li et al., 2018; Tripp et al., 2020; Moriconi et al., 2020; Eriksson & Jankowiak, 2021; Grosnit et al., 2021b; Wan et al., 2021), decomposition methods that assume additively structured black-boxes emerged as a promising direction for high-dimensional BO (Kandasamy et al., 2015; Rolland et al., 2018; Han et al., 2021).

Given a decomposition in the dimensions of the problem, those techniques utilise additive kernel Gaussian processes (GPs) as surrogate models to trade off exploration and exploitation when suggesting novel queries to evaluate. Additive techniques uncover new inputs by maximising an acquisition function that is additive under the provided decomposition. Although first and second-order methods (Wilson et al., 2017) can be used to maximise acquisition functions, we adopt message-passing optimisers that can better exploit additive acquisition structures (Rolland et al., 2018).

The success of additive methods in high-dimensional BO depends on the correct choice of decompositions that must accurately mimic the inter-dimensional dependencies of the actual black-box function. Prior art empirically demonstrated that tree-structured decompositions (i.e., cycle-free pair-wise dimensional interactions) could effectively represent many black-box functions. The Tree algorithm (Han et al., 2021) discovers the best tree decomposition for the black-box based on the data collected during BO, by maximising a new GP marginal that includes decomposition parameters (encoding sparsity) and length scales.

At first glance, learning decompositions based on data and marginal likelihood is plausible and can yield promising optimisation results. While this is true when given a fixed dataset, dynamically acquired data during BO provides, at best, local (within the probed regions) function information, making it challenging to extrapolate dimensional interdependencies across the search space. In other words, such agents are easily misled by modifying the black-box function's factorisation based on regions of the search space (for example see Section 3).

[1]Huawei Noah's Ark Lab, London, UK. Correspondence to: Haitham Bou-Ammar <haitham [dot] ammmar (at) huawei {dot} com>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

Contributions: Of course, one can think of many directions to resolve this issue, like analysing distribution shifts (Kirschner et al., 2020), maximising estimators beyond marginals (Ziegel, 2003), or even designing novel Gaussian process kernels. In our work, however, we prefer to develop a simple (almost) plug-and-play approach that leads to empirical gains while adhering to rigorous theoretical guarantees. Therefore, rather than relying on data-driven learning, we *investigate adopting data-independent decomposition rules*. Our theoretical results indicate that random decomposition sampling strategies achieve the lowest expected mismatch to the black-box function when we fix the class of decompositions to trees, which allows us to favourably bound maximal information gain. Equipped with these results, we then propose the random decomposition upper-confidence bound (RDUCB) algorithm. RDUCB utilises a random tree sampler and an additive acquisition function to achieve superior empirical performance on a broad set of benchmarks compared to the prior state-of-the-art. The modelling component of RDUCB is simple to implement and thus easy to integrate on top of many existing BO frameworks. We support this claim by augmenting HEBO (Cowen-Rivers et al., 2022) - the winning submission of the NeurIPS 2020 black-box optimisation challenge (Turner et al., 2021) - with our random decomposition sampling strategy. We demonstrate that this adaptation, which we title RDHEBO, delivers improved performance on the set of highest dimensional Bayesmark (Asuncion & Newman, 2007; Turner et al., 2021) tasks.

2. Background

2.1. Bayesian Optimisation (BO)

We employ a sequential decision-making approach to the maximisation of expensive-to-evaluate black-box functions $f : \mathcal{X} \to \mathbb{R}$ over an input domain $\mathcal{X} \subseteq \mathbb{R}^d$. At each round t, we determine an input $\boldsymbol{x}_t \in \mathcal{X}$ and observe its black-box function value $f(\boldsymbol{x}_t)$. We allow noise-corrupted observations such that $y_t \sim \mathcal{N}(\boldsymbol{x}_t, \sigma_{\mathrm{n}}^2)$. Our goal is to approach the optimum, $\boldsymbol{x}^\star \equiv \arg\max_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})$, rapidly as a function of black-box queries. Given that both the black-box function and optima are unknown, BO solvers trade-off exploration and exploitation via a two-step procedure involving fitting a surrogate model and maximising an acquisition function. We detail each of those steps below.

Gaussian Process Surrogates (GPs): GPs allow us to place priors directly in the function space by specifying a mean function $m(\boldsymbol{x})$ and a covariance kernel $k(\boldsymbol{x}, \boldsymbol{x}')$ that encode our assumptions about the black-box. Following (Williams & Rasmussen, 2006), we assume a zero-mean $m(\boldsymbol{x}) \equiv 0$ and adopt a squared exponential covariance kernel:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}')^\mathsf{T}\mathrm{diag}(\boldsymbol{\theta}^2)^{-1}(\boldsymbol{x}-\boldsymbol{x}')\right),$$

where $\boldsymbol{\theta}$ is a set of hyper-parameters tuned by maximising the data marginal[1].

Given the data $\mathcal{D}_t = \{\boldsymbol{x}_i, y_i\}_{i=1}^t$ collected so-far during t BO rounds, we write the posterior at a point $\boldsymbol{x}$ as $p(f(\boldsymbol{x})|\mathcal{D}_t) \sim \mathcal{N}(\mu_t(\boldsymbol{x}), \sigma_t(\boldsymbol{x}))$ with:

$$\mu_t(\boldsymbol{x}) = \boldsymbol{k}_t^\mathsf{T}(\boldsymbol{x})\left(\boldsymbol{K}_t + \sigma_{\mathrm{n}}^2\boldsymbol{I}\right)^{-1}\boldsymbol{y}_t$$
$$\sigma_t(\boldsymbol{x})^2 = k(\boldsymbol{x},\boldsymbol{x}) - \boldsymbol{k}_t^\mathsf{T}(\boldsymbol{x})\left(\boldsymbol{K}_t + \sigma_{\mathrm{n}}^2\boldsymbol{I}\right)^{-1}\boldsymbol{k}_t(\boldsymbol{x}),$$

where $\boldsymbol{y}_t$ concatenates all observations $y_{1:t}$ in one vector and $\boldsymbol{I}$ represents an identity matrix. The matrix $\boldsymbol{K}_t$ evaluates the covariance kernel on all input pairs in $\mathcal{D}_t$. Finally, the vector $\boldsymbol{k}_t$ contains the kernel evaluation between $\boldsymbol{x}$ and all input data points from $\mathcal{D}_t$.

Acquisition Function Maximisation: Given the posterior predictive distribution above, we now discuss how to suggest novel query points that improve the guess of the optima. In BO, this process involves maximising an acquisition function $\alpha(\cdot|\mathcal{D}_t)$ that utilises the probabilistic model such that $\boldsymbol{x}_t \in \arg\max_{\boldsymbol{x}\in\mathcal{X}} \alpha_t(\boldsymbol{x}|\mathcal{D}_{t-1})$. While there exist many acquisitions ranging from myopic to non-myopic forms (Frazier et al., 2008; Grosnit et al., 2021a; Cowen-Rivers et al., 2022; Shahriari et al., 2015; Zhang et al., 2021), in this paper, we follow (Srinivas et al., 2009; 2012) and adapt the upper-confidence bound as the function of choice:

$$\alpha_t^{(\mathrm{UCB})}(\boldsymbol{x}|\mathcal{D}_{t-1}) = \mu_{t-1}(\boldsymbol{x}) + \beta_t\sigma_{t-1}(\boldsymbol{x}),$$

where β_t is a tuneable hyperparameter.

2.2. High-Dimensional BO with Decompositions

BO in high-dimensional spaces is an active area of research. In Section 6, we survey related methods to our work. In this section, we focus on decomposition-based strategies that promise to scale BO while accelerating acquisition optimisation. Before diving into the details of decomposition-based techniques, we now introduce the notion of decompositions of d-dimensional spaces as follows:

Definition 2.1. *A decomposition g of d-dimensions is a collection of sets c, called components, consisting of dimensions $i \in [1:d]$, i.e. $\forall_{c\in g}\forall_{i\in c} i \in [1:d]$.*

Decomposition-based BO assumes that the black-box function decomposes according to g: $f(\boldsymbol{x}) = \sum_{c\in g} f_c\left(\boldsymbol{x}_{[c]}\right)$, where $\boldsymbol{x}_{[c]}$ selects those dimensions of $\boldsymbol{x}$ that appear in c.

Additive GP Kernels: Compared to standard BO from Section 2.1, the first change decomposition methods employ is the usage of additive kernels (Duvenaud et al., 2011; Durrande et al., 2012; Qamar & Tokdar, 2014; Lu et al., 2022) that better suit a decomposable black-box function: $k^g(\boldsymbol{x}, \boldsymbol{x}') = \sum_{c\in g} k_c\left(\boldsymbol{x}_{[c]}, \boldsymbol{x}'_{[c]}\right)$. Significantly, if

[1] In our equation, we execute $\boldsymbol{\theta}^2$ element-wise.

two dimensions i and j do not appear together in any of the sets c, the kernel will not model interactions between them. Rolland et al. (2018) showed that the posterior of each component subfunction $f_c\left(\boldsymbol{x}_{[c]}\right)$ can be expressed as $p\left(f_c\left(\boldsymbol{x}_{[c]}\right)|\mathcal{D}_t\right) = \mathcal{N}\left(\mu_{t,c}\left(\boldsymbol{x}_{[c]}\right), \sigma^2_{t,c}\left(\boldsymbol{x}_{[c]}\right)\right)$, where:

$$\mu_{t,c}\left(\boldsymbol{x}_{[c]}\right) = \boldsymbol{k}^{\mathsf{T}}_{t,c}\left(\boldsymbol{x}_{[c]}\right)\left(\boldsymbol{K}_t + \sigma^2_{\mathrm{n}}\boldsymbol{I}\right)^{-1}\boldsymbol{y}_t$$
$$\sigma^2_{t,c}\left(\boldsymbol{x}_{[c]}\right) = k_c\left(\boldsymbol{x}_{[c]}, \boldsymbol{x}_{[c]}\right) - \boldsymbol{k}^{\mathsf{T}}_{t,c}\left(\boldsymbol{x}_{[c]}\right)\left(\boldsymbol{K}_t + \sigma^2_n\boldsymbol{I}\right)^{-1}\boldsymbol{k}_{t,c}\left(\boldsymbol{x}_{[c]}\right),$$

where $\boldsymbol{k}_{t,c}(\boldsymbol{x}_{[c]})$ is a vector of kernel evaluations between $\boldsymbol{x}$ and all inputs in $\mathcal{D}_t$ while only considering those dimensions $i \in c$ that appear in c.

If the size of each set c is much smaller than the total dimensionality d, such an approach will enjoy many benefits. First, we escape the curse of dimensionality, as we only need to consider interactions between a small number of dimensions. Second, if we utilise an acquisition function with an additive structure, such as the additive UCB: $\alpha_t^{(\text{add-UCB})}(\boldsymbol{x}|\mathcal{D}_{t-1}) = \sum_{c\in g}\alpha_{t,c}(\boldsymbol{x}|\mathcal{D}_{t-1}) = \sum_{c\in g}\mu_{t-1,c}(\boldsymbol{x}) + \beta_t\sigma_{t-1,c}(\boldsymbol{x})$ we can determine novel query points to evaluate very efficiently using message passing (Rolland et al., 2018).

However, this is only possible if we know the decomposition of the black box. Existing methods attempt to learn it from data using maximum likelihood (Kandasamy et al., 2015; Rolland et al., 2018; Han et al., 2021), but there are no guarantees regarding the correctness of such a learning procedure. We expand on the problems associated with this approach in the next section.

3. Misleading Decomposition Learners

As noted in the previous section, decomposition methods learn the optimal additive kernel structure by selecting the one which produces a model with the highest marginal likelihood. While plausible and generally adopted, we now point out some problems associated with likelihood maximisation when learning decompositions in BO. Before we underpin this issue from a theoretical perspective, we first provide an empirical example that demonstrates those challenges next.

3.1. Challenges to Decompositions Learners

Consider the maximisation of the function demonstrated by the heat plot shown in Figure 1. It is easy to see that while this function is not fully separable due to the correlated mode in the top-right corner, it can locally appear as if it was entirely separable.

Imagine that we collected several initial points from the region $0 \leq x \leq 600$ and $0 \leq y \leq 600$. Methods that rely entirely on data when learning the kernel structure (e.g., by maximising marginals) would exploit the local view of the function and falsely believe a complete separation in dimensions. We support this realisation in Figure 1 by running the state-of-the-art Tree algorithm from (Han et al., 2021) that learns decompositions via maximum likelihood.

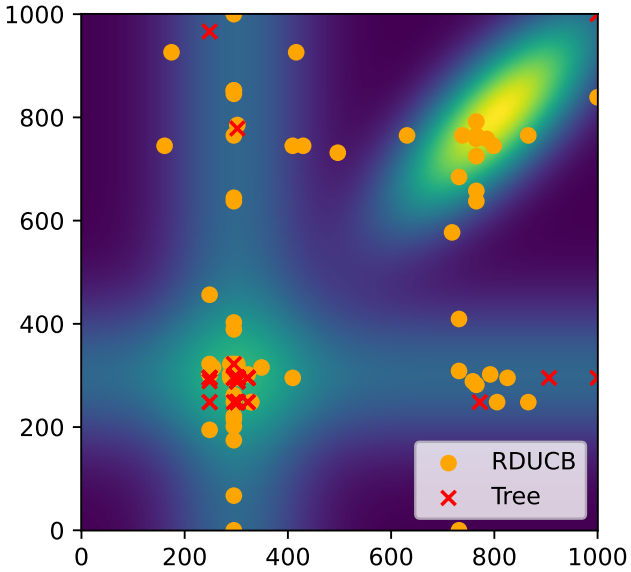


Figure 1. Comparison of points queried on a toy problem, given the same initial design. Tree refers to the method from (Han et al., 2021), while RDUCB is the algorithm we propose in this paper. It is clear that Tree gets stuck in the local mode, while RDUCB eventually arrives at the optimal mode. For details see App. D.

We see that the Tree algorithm gets stuck in the local mode because of erroneously learning that the function is fully decomposable. In contrast, our algorithm RDUCB, which we propose throughout the paper, circumvents this problem, whereby given the same initial design and settings of Tree, it can still find the optimal mode.

3.2. What Causes the Failure?

To better understand the above failure mode, we notice that approaches learning decompositions from data (e.g., Tree (Han et al., 2021)) assume that the function abides by one decomposition that does not vary across the search space, which is not the case in Figure 1.

To improve decomposition methods in high-dimensional BO, we wish to develop algorithms that enable varying decompositions across the search space. We could formalise this problem by imagining an adversary choosing a decomposition, g, and a corresponding black-box function, $f(\cdot)$, to optimise. This function must be a member of the reproducing kernel Hilbert space $\mathcal{H}^g$ defined by some kernel $k^g(\boldsymbol{x}, \boldsymbol{x}')$, following the selected decomposition g, unknown to the algorithm. It is important to note that such an adversary can select a black-box function that globally follows a decomposition g but locally appears to have fewer interaction components than g. As we cannot rely on locally

collected data, we wish to investigate data-independent rules while ensuring a rigorous theoretical understanding.

On Data Independent Kernel Updates: Although we are the first to propose data-independent updates to select decompositions in BO, it is worth noting that the work in (Berkenkamp et al., 2019) already considered data-independent rules but when tuning kernel hyper-parameters with no focus on decompositions. The authors demonstrated no-regret bounds for an algorithm that alters the GP kernel's hyperparameters according to a predefined scheme that does not rely on data gathered during BO.

However, the direct application of the work in (Berkenkamp et al., 2019) to learning decompositions is challenging for several reasons. First, our problem setting varies in that the authors in (Berkenkamp et al., 2019) consider low-dimensional BO that does not require learning kernel decompositions. Second and more importantly, the update rule in (Berkenkamp et al., 2019) keeps decreasing the length scales, expanding the kernel's complexity as measured by maximum information gain (Srinivas et al., 2009), which will play a critical role in trading-off complexity versus mismatch as we note in Section 4.1.

Next, we expand on how to design such data-independent decomposition schemes and note that sparse random trees serve as a simple, effective, and scalable strategy.

4. Decompositions Without Learning

To better understand what properties constitute good predefined decomposition rules, we begin with a theoretical study that hints at the necessary trade-offs our strategy needs to optimise. Let us introduce $S(t) : \mathbb{Z}^+ \to \mathcal{G}$ to be a predefined (data-independent) scheme that selects a decomposition from some class of decompositions $\mathcal{G}$ at each round t. Consequently, $S(t)$ defines the kernel $k_t(\boldsymbol{x}, \boldsymbol{x}') = k^{g_t}(\boldsymbol{x}, \boldsymbol{x}')$ that we use during that round t.

We wish to derive a high-probability regret bound in terms of 1) a quantity roughly measuring the kernel's complexity and 2) a notion of function mismatch between the true black-box and those functions spanned by our kernel $k_t(\boldsymbol{x}, \boldsymbol{x}')$. For kernel complexity, we follow (Srinivas et al., 2009) and use the maximum information gain γ_T of kernels defined by the decompositions proposed by our scheme[2]. Here, $\gamma_T = \max_{\boldsymbol{X} \subseteq \mathcal{X}, |\boldsymbol{X}|=T} I(\boldsymbol{f}_T, \boldsymbol{y}_T)$ denotes the maximum information gain after T steps, where $\boldsymbol{X}$ is a set of T selected points and $\boldsymbol{f}_T = (\hat{f}_1(\boldsymbol{x}_1), \ldots, \hat{f}_T(\boldsymbol{x}_T))$, where $\hat{f}_t = \sum_{c \in g_t} f_c$ and $f_c \sim \mathcal{GP}(0, k_c(\boldsymbol{x}, \boldsymbol{x}'))$. Concerning function mismatch, we define $\epsilon_t = |\hat{f}_t - f|_\infty$ such that $\hat{f}_t = \arg\min_{f' \in \mathcal{H}_t} |f' - f|_\infty$ is the function from the reproducing kernel Hilbert space (RKHS) $\mathcal{H}_t$ of kernel $k_t(\boldsymbol{x}, \boldsymbol{x}')$ that is closest to the black-box $f(\cdot)$ in terms of infinity norm. This way ϵ_t provides a notion of mismatch between the actual black-box and the closest function from the RKHS of $k_t(\boldsymbol{x}, \boldsymbol{x}')$. Now, if we run a UCB-style BO algorithm for T rounds, we obtain the following result.

[2]Of course in (Srinivas et al., 2009) decompositions are not considered. Here, we define a slight generalisation of maximal information gain to handle the case of a changing kernel.

Theorem 4.1. *Let the black-box function f be selected by an adversary from an RKHS $\mathcal{H}^g$ of kernel k^g, defined over some decomposition g that is also selected by an adversary. After T rounds, a UCB-style BO algorithm with an $S(t) : \mathbb{Z}^+ \to \mathcal{G}$ decomposition rule, incurs with a probability of at least $1 - \delta$ the following total cumulative regret R_T:*

$$R_T = \mathcal{O}\left(\sqrt{T\gamma_T}\left(B + \sqrt{\ln\frac{1}{\delta} + \gamma_T} + \sum_{t=1}^{T} \epsilon_t\right)\right),$$

where $B = \max_{t \in T} \|\hat{f}_t\|_t$ and $\|\cdot\|_t$ denotes the norm in $\mathcal{H}_t$.

Proof. (Sketch) We defer complete proof to Appendix A.1. Here, we provide a sketch of the main steps. BO under misspecification has been studied by (Bogunovic & Krause, 2021). In their setting, the authors assume fixed kernels in-between iterations. Hence, we need to adapt the proof from (Bogunovic & Krause, 2021) to our setting where the kernel and the mismatch vary. To do so, we observe that at any given time, the difference between $\max_{\boldsymbol{x} \in \mathcal{X}} \hat{f}_t(\boldsymbol{x})$ and $\max_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x})$ can be at most ϵ_t. Consequently, the difference between $\max_{\boldsymbol{x} \in \mathcal{X}} \hat{f}_t(\boldsymbol{x})$ and $\hat{f}_t(\boldsymbol{x}_t)$ becomes the new term that we bound. Here, we adapt a high probability bound to the case of a changing kernel. The final step of the proof is to express the bound in terms of maximum information gain γ_T, which easily follows from preceding BO literature (Srinivas et al., 2009). □

Note that ϵ_t is a random quantity. Thus to make this bound easier to analyse, we derive the following Corollary. We provide its proof in Appendix A.2.

Corollary 4.2. *Under the assumptions of Theorem 4.1, we have with probability at least $1 - \delta_A - \delta_B$, the cumulative regret R_T of a UCB-style BO algorithm utilising the decomposition suggesting scheme $S(t)$, incurs the following cumulative regret:*

$$R_T = \mathcal{O}\left(\sqrt{T\gamma_T}\left(B + \sqrt{\ln\frac{1}{\delta_A} + \gamma_T} + \frac{\mathbb{E}_S\left[\sum_{t=1}^{T} \epsilon_t\right]}{\delta_B}\right)\right).$$

4.1. Analysing Decomposition Rules

From Corollary 4.2, we notice that we need a decomposition rule $S(t)$ such that both γ_T and $\mathbb{E}\left[\sum_{t=1}^{T}\epsilon_t\right]$ are "small". To bound γ_T, we require a restriction on the class of decompositions our scheme can propose. If we choose this class to consist only of decompositions with pairwise components, we get the following result on γ_T, proven in Appendix A.3.

Proposition 4.3. *The maximum information gain for a squared exponential kernel following a decomposition only with pair-wise components in d dimensions is upper-bounded by $\gamma_T \leq \mathcal{O}(d(\log T)^3)$.*

In light of the result above, it might be tempting to try a decomposition with all possible pairwise components. However, given the empirical success of the Tree algorithm (Han et al., 2021), it appears that for many problems, it is sufficient to consider tree-structured decompositions, which we now proceed to define.

Definition 4.4. *A decomposition g is tree-based with E edges if it has E pair-wise components and the remaining components contain one dimension. Additionally, an undirected graph formed by edges corresponding to pair-wise interactions does not contain a cycle.*

If we restrict ourselves to the class of trees, we additionally reduce the maximum information gain, as stated by the next result, proven in Appendix A.4.

Proposition 4.5. *When $d > 2$, the maximum information gain for a kernel following a tree decomposition (possibly changing between timesteps) is always smaller than for a kernel containing all pairwise interactions.*

Bounding Expected Mismatches $\mathbb{E}\left[\sum_{t=1}^{T}\epsilon_t\right]$: Bounding $\mathbb{E}\left[\sum_{t=1}^{T}\epsilon_t\right]$ is far more challenging. To understand this difficulty, let us consider two extremes when suggesting decompositions: constant and adaptive decomposition selection rules $S(t)$. If $S(t)$ constantly suggests the same decomposition, the adversary efficiently exploits this strategy resulting in a constant high mismatch. While if we adaptively increase the kernel's complexity using procedures like those in (Berkenkamp et al., 2019), our method will indeed reduce mismatch regardless of the adversary's choices. However, as we introduce more interactions between dimensions, our decompositions eventually stop being trees and thus lead to increases in information gain γ_T.

Analysing the bound above, we observe a trade-off between maximal information gain and the risk of being exploited by an adversary. To resolve this problem, we propose to use a randomised tree sampling strategy that keeps the information gain small while circumventing adversaries. Importantly, we show that such an $S(t)$ exhibits the lowest expected mismatch within a class of tree-structured decompositions. We now state the formal result.

Theorem 4.6. *Let $\mathcal{G}$ be a class of d-dimensional tree-based decompositions with E edges. Let the adversary choose any function f from RKHS $\mathcal{H}_g$ defined by kernel k_g on a decomposition $g \in \mathcal{G}$ also chosen by the adversary, such that the infinity norm of each component function is bounded $\forall_{c\in g}|f_c|_\infty \leq M_c$ and $\sum_{c\in g} M_c \leq M$. We then have that for any data-independent scheme S, the expected sum of mismatches is at least as large as for the scheme S_r that selects a decomposition from $\mathcal{G}$ uniformly at random, i.e.,*

$$\forall_{S:\mathbb{Z}^+\to\mathcal{G}} \quad \mathbb{E}_S\left[\sum_{t=1}^{T}\epsilon_t\right] \geq \mathbb{E}_{S_r}\left[\sum_{t=1}^{T}\epsilon_t\right].$$

Proof. (Sketch) We defer complete proof to Appendix A.5. Observe that the optimal strategy for an adversary is to put all the mismatch M_c on the pair-wise component c least selected by the scheme $S(t)$. Since we will suffer a mismatch of M every time we do not select this component, the optimal scheme needs to maximise the expected number of times the least selected component is chosen. This happens when all pair-wise components have the same probability of being selected, proving the optimality of S_r. □

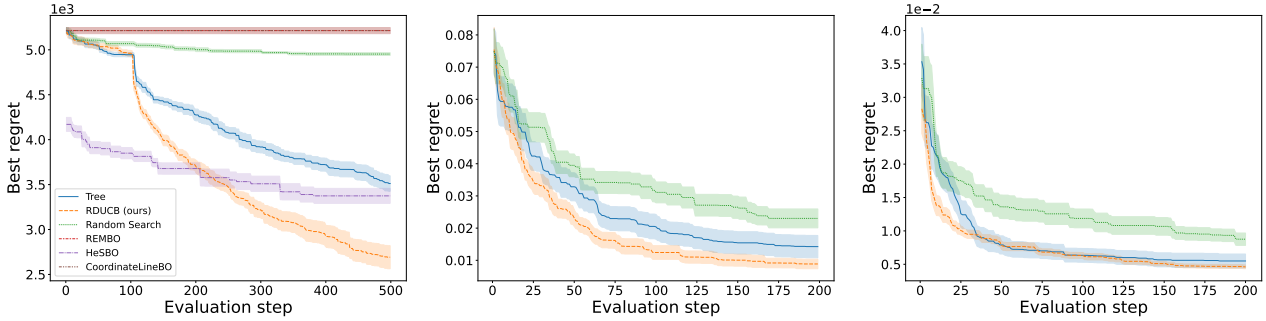
4.2. Practical Algorithm: Random Decompositions UCB

Equipped with the above results, this section presents a practical and effective algorithm for high-dimensional BO. Our algorithm, titled random decomposition upper confidence bound (RDUCB), adheres to the theoretical results from the previous section and allows a scalable implementation.

Algorithm 1 RDUCB

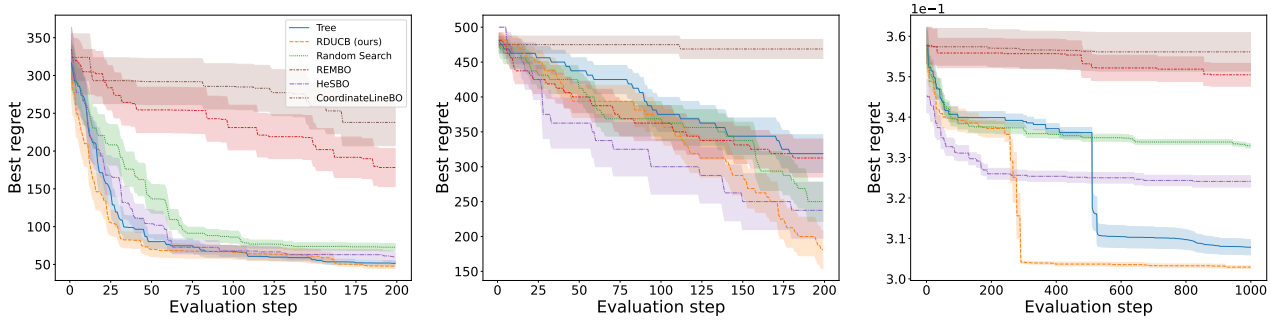
1: **Inputs:** Black-box function f, evaluation budget N, initial budget N_init, exploration bonuses $\{\beta_t\}_{t=1}^{N}$
2: Evaluate N_init random inputs in f & populate $\mathcal{D}_{N_\text{init}}$
3: **for** $t = N_\text{init} + 1$ **to** N **do**
4: Sample tree decomposition g (Alg. 2)
5: Fit a GP using $\mathcal{D}_{t-1}$ with the kernel $k_g(\cdot)$
6: Maximise $\alpha_t^{(\text{add-UCB})}(\boldsymbol{x}|\mathcal{D}_{t-1})$ with message passing
7: Evaluate f on the suggested query & add to $\mathcal{D}_{t-1}$
8: **end for**

Algorithm 1 is a pseudo code of RDUCB. At a high-level, our method follows any generic BO solver in that it first fits a probabilistic model (line 5) and then maximises an acquisition, as shown in line 6. Of course, we maximise additive UCB acquisitions as dictated by the sampled decomposition g. As noted earlier, decompositions g correspond to trees that we sample so that each edge has an equal probability of being selected. We present this sampler (referred to in line 4 of Algorithm 1) in Algorithm 2 in Appendix B.



(a) 250-d Stybtang Function (b) 9-d NAS Protein Dataset (c) 9-d NAS Parkinson Dataset

Figure 2. Best regret results from the 250-dimensional synthetic functions (averaged over 10 random seeds) and the NAS benchmark (averaged over 20 random seeds). We note that REMBO is not reported for the NAS (Protein and Parkinson) datasets as those involve discrete variables. Tree refers to the previous state-of-the-art decomposition technique from (Han et al., 2021). We observe that RDUCB outperforms Tree, REMBO and random search. It is also clear that as the dimensions increase, so does the performance of our algorithm.



(a) 74-d `qiu` MIP Task (b) 74-d `misc05inf` MIP Task (c) 180-d DNA LassoBench Dataset

Figure 3. Best regret results from tuning 74-d MIP problems and the 180-dimensional DNA dataset from the Lasso bench. We average the results over 40 seeds for `qiu` and 80 seeds for `misc05inf`. For the LassoBench task, we average over ten random seeds. Tree again refers to the previous state-of-the-art from (Han et al., 2021). We see that RDUCB outperforms others in best regret. Interestingly, the performance gap to other methods increases as the dimensions increase.

5. Empirical Evaluation

This section presents experimental results on multiple benchmarks. We compare RDUCB against well-established techniques, including tree-based learners (Han et al., 2021), random embedding BO (Wang et al., 2013), Hashing-enhanced Subspace Bayesian Optimization (HeSBO) (Nayebi et al., 2019), Coordinate Line BO (Kirschner et al., 2019) and random search[3]. The comparison against tree-learning-based BO allows us to gauge improvements compared to the prior state-of-the-art. In contrast, the comparison to REMBO, HeSBO and LineBO sheds light on the advantages we gain when operating in the original high-dimensional space.

In Appendix C, we provide all algorithm settings used in our experiments. We have open-sourced our code[4] to ease the reproducibility of our results.

5.1. Benchmark Datasets

Synthethic Functions: We test our method on 20-dimensional Rosebrock, 20-dimensional Hartmann, and 250-dimensional Styblinski-Tang (Stybtang) functions. We calculate the regret as the difference between queried function value and the theoretical minimum.

Neural Network Hyperparameter Tuning: In the second set of experiments, we consider the neural architecture search (NAS) hyperparameter tuning benchmark (Zela et al., 2020). This benchmark consists of precomputed validation mean-squared errors for different combinations of hyperparameters in a two-layer feed-forward neural network trained

[3]It is worth noting that REMBO, HeSBO and CoordinateLineBO are presented in all experiments except those involving neural architecture search (NAS). The reason is that NAS baselines operate in discrete search spaces for which these algorithms are not well-suited.

[4]https://github.com/huawei-noah/HEBO/tree/master/RDUCB

on four different datasets. This experiment considers *mixed* search spaces, e.g., learning and drop-out rates (continuous), the sizes of hidden layers and activation types (discrete).

Mixed Integer Programming: We consider the problem of tuning heuristic hyperparameters for the mixed integer programming (MIP) solver LPSolve (Berkelaar et al., 2015). This domain is high-dimensional with a 74-dimensional search space. We consider three MIP problems varying in difficulty with the `misc05inf` instance being the hardest. For each problem, the regret value is the duality gap of the best solution found after 5 seconds with given hyperparameters. We cap the maximum instantaneous regret at 500, and running the solver once is considered as one query.

Weighted Lasso Tuning: The last problem we consider is tuning the LassoBench (Šehić et al., 2022). LassoBench is a set of benchmark problems where BO tunes the weights for a weighted Lasso model. Hence, the number of dimensions of the search space scales with the number of features in the dataset. The highest dimension we study in this experiment is the 180 DNA data.

5.2. Regret Results

We run all algorithms varying the number of seeds per-each benchmark to allow for statistically significant results on the tasks introduced above. We report some of these results in this section (Figures 2 and 3) and defer others to Appendix E due to space constraints. Every algorithm started with 10 initial randomly sampled points. Note that for HeSBO, the initial points were sampled in the algorithm's lower-dimensional space (hence the initial regret is sometimes different from other algorithms).

In general, we notice that RDUCB outperforms other baselines. We see that our improvements are amplified in high dimensional settings, as reported in Figures 2(a) and Figures 3(a), 3(b) and 3(c). In three experiments, RDUCB draws in terms of final regret with Tree, and in two benchmarks RDUCB (minorly) underperforms compared to Tree (see Appendix E). We note that the cases when RDUCB draws or underperforms mostly correspond to low-dimensional settings. We expand on this in the next section.

5.3. Scalability as Dimensionalities Increase

We conduct an ablation study to test how our method scales with the number of dimensions and to understand its potential limitations. We choose one synthetic function and one real-world problem. For the synthetic function, we vary the dimensionality of the Stybtang function, while in the real-world case, we tackle the LassoBench DNA problem. For the latter, we produce different versions of this problem with varying dimensionalities. To achieve this, we turn some dimensions off by permanently setting them to zero and allowing the algorithm to vary only the remaining ones.

We compare the best regret achieved by the algorithm in (Han et al., 2021) (titled Tree in the figure) and RDUCB after 500 iterations for Stybtang and 700 iterations for LassoBench DNA, respectively. We show the results of ablation in Figure 4. Although for a small number of dimensions, Tree can slightly outperform RDUCB, we see that as the number of dimensions increases, RDUCB starts to outperform, and the gap widens for higher-dimensional problems.

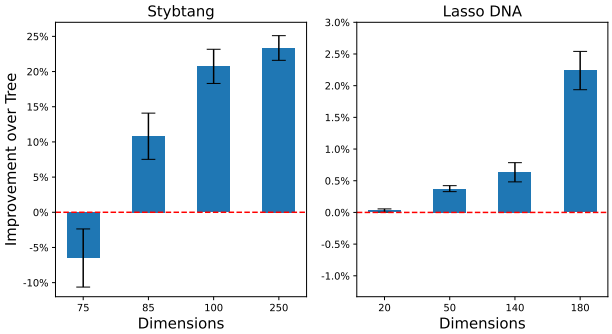


Figure 4. Improvement of RDUCB over Tree (Han et al., 2021) in terms of final best regret for different dimensionality of the problems. Error bars correspond to standard errors.

5.4. Plug-and-Play for RDHEBO

RDUCB presents a simple modification strategy to modelling using GPs, allowing it to plug on top of existing BO algorithms. This section supports our claim by introducing RDHEBO, an adaptation of HEBO (Cowen-Rivers et al., 2022) that uses our random decomposition scheme. We keep all the other components (warping functions and multi-objective acquisitions) of HEBO unchanged and present a comparison conducted on the highest dimensional Bayesmark tasks in Table 1. We can see that our modification has improved the performance of HEBO across those tasks, rising from a score of 92.68 to 93.67 in the MLP-Adam task, for instance.

Problem	**HEBO**	**RDHEBO**
MLP-Adam	92.68 ± 0.22	$\mathbf{93.67 \pm 0.30}$
MLP-SGD	90.66 ± 0.81	$\mathbf{91.65 \pm 0.10}$
DT	79.42 ± 0.45	$\mathbf{80.79 \pm 0.15}$
RF	84.97 ± 0.32	$\mathbf{87.64 \pm 2.00}$
Average	86.93 ± 0.45	$\mathbf{88.44 \pm 0.64}$

Table 1. Bayesmark normalised score with standard errors for HEBO and RDHEBO demonstrating that our modifications can improve HEBO's performance. We run 10 seeds for MLP-Adam and DT and 15 for MLP-SGD and RF.

6. Related Work

Random projection methods project the inputs from a high dimensional space to a space of lower dimensionality by randomised mappings. To do so, REMBO (Wang et al., 2013) uses a matrix with entries sampled from a normal distribution. Though successful, REMBO tends to over-explore regions that are on the boundary of the domain of the black-box function. The authors in (Letham et al., 2020) proposed ALEBO, an algorithm that addresses some of REMBO's problems, while Nayebi et al. (2019) use different projection types altogether. The algorithm they propose - HeSBO, initialises a randomised hashing function that maps back to the original space, leading to improved results. As opposed to all those techniques, RDUCB takes a different direction of decomposing the original space rather than projecting to lower dimensional manifolds.

LineBO methods (Kirschner et al., 2019) conduct optimisation over an affine, one-dimensional subspace. This subspace is chosen so that the best point found so far belongs to it. There are different variants of LineBO, where the direction of this one-dimensional space is chosen randomly (Random LineBO), aligned with a randomly chosen dimension (Coordinate LineBO) or along a gradient estimate (Descent LineBO). We chose to compare RDUCB with Coordinate LineBO, as in most cases it outperforms other LineBO variants.

Latent space methods perform the optimisation in some latent space and then utilise a generative model as a projection to original space. A commonly taken approach is to utilise the latent space of a Variational AutoEncoder (Kingma & Welling, 2013), pretrained on some larger batch of existing data. However, over the course of this pretraining, we need to ensure that the obtained latent space is well-suited for BO. There have been numerous VAE-based algorithms proposed (Eissman et al., 2018; Gómez-Bombarelli et al., 2018; Zhang et al., 2019; Tripp et al., 2020; Griffiths & Hernández-Lobato, 2020; Siivola et al., 2021; Grosnit et al., 2021b), employing various mechanisms to achieve this goal.

While using deep networks to scale BO is prominent and essential, RDUCB constitutes an orthogonal research direction where our discoveries, at least in the modelling component, can (easily) plug-and-play in latent spaces, e.g., when using deep kernel GPs (Wilson et al., 2016) in general. We instantiated one such plug-and-play use-case in this paper (see Section 5.4) and planned to further investigate such applications in the future.

Dropout methods (Li et al., 2017) randomly drop several dimensions and optimise only a subset at each step. When querying for new values of the black-box function, the value of dimensions that were not optimised at a given iteration are selected according to some filling strategy that can either copy the value of the best point so far or choose a random value. Though successful in isolated instances, dropout methods exhibit high-sample complexity.

SAASBO (Eriksson & Jankowiak, 2021) adopts a hierarchical Bayesian model, putting a prior on the length scale with a high probability mass concentrated at zero. Hence, a dimension will be effectively removed from the model unless there is enough evidence that it dramatically impacts the black-box function value.

Trust region methods tackle high dimensional spaces by optimising locally within a chosen trust region. The TuRBO algorithm (Eriksson et al., 2019) maintains several trust regions and selects one at a given time using a multi-armed bandit strategy. CASMOPOLITAN (Wan et al., 2021) extends this idea to handle categorical and mixed input spaces.

Our research in this paper is orthogonal to SAASBO and trust region methods. In the future, we plan to integrate our modelling component with the works in (Eriksson et al., 2019) and (Eriksson & Jankowiak, 2021).

7. Discussion & Future Work

This paper introduced random decompositions to scale BO to high-dimensional spaces. Our algorithm, RDUCB, is simple to implement, empirically effective and theoretically grounded. We compared our method against the prior decomposition state-of-the-art technique and showed improved best-regret results on a wide set of publicly available benchmarks. While this is the first step in understanding data-independent rules for high dimensional BO, there is a number of challenges that we now elaborate on.

Currently, RDUCB is not capable of handling non-numerical inputs (e.g., graphs or strings). This problem is not only tied to RDUCB but also challenges other decomposition-based techniques since we cannot easily define a notion of decomposition for such input types. We plan to investigate such decomposition rules in the future.

Apart from tackling technical challenges, in the future, we plan to scale RDUCB to thousands of dimensions. We hope to benefit from distributed and high-performance computing. Of course, this direction requires us to devise novel consensus-based acquisition optimisers, which have recently seen progress in non-convex settings (Zhang et al., 2022).

Although we applied the modelling component of RDUCB to HEBO, we believe many BO frameworks and algorithms can benefit from our decomposition scheme. Those include trust-region methods, latent space techniques, and combinatorial BO solvers. In the future, we will also investigate this direction in combination with scalability.

References

Asuncion, A. and Newman, D. Uci machine learning repository, 2007.

Berkelaar, M. et al. Package 'lpsolve', 2015.

Berkenkamp, F., Schoellig, A. P., and Krause, A. No-regret bayesian optimization with unknown hyperparameters. *arXiv preprint arXiv:1901.03357*, 2019.

Bogunovic, I. and Krause, A. Misspecified gaussian process bandit optimization. *Advances in Neural Information Processing Systems 34*, 2021.

Chowdhury, S. R. and Gopalan, A. On kernelized multi-armed bandits. In *International Conference on Machine Learning*, pp. 844–853. PMLR, 2017.

Cowen-Rivers, A. I., Lyu, W., Tutunov, R., Wang, Z., Grosnit, A., Griffiths, R. R., Maraval, A. M., Jianye, H., Wang, J., Peters, J., et al. Hebo: pushing the limits of sample-efficient hyper-parameter optimisation. *Journal of Artificial Intelligence Research*, 74:1269–1349, 2022.

Durrande, N., Ginsbourger, D., and Roustant, O. Additive covariance kernels for high-dimensional gaussian process modeling. In *Annales de la Faculté des sciences de Toulouse: Mathématiques*, volume 21, pp. 481–499, 2012.

Duvenaud, D. K., Nickisch, H., and Rasmussen, C. Additive gaussian processes. *Advances in neural information processing systems*, 24, 2011.

Eissman, S., Levy, D., Shu, R., Bartzsch, S., and Ermon, S. Bayesian optimization and attribute adjustment. In *Proc. 34th Conference on Uncertainty in Artificial Intelligence*, 2018.

Eriksson, D. and Jankowiak, M. High-dimensional bayesian optimization with sparse axis-aligned subspaces. In *Uncertainty in Artificial Intelligence*, pp. 493–503. PMLR, 2021.

Eriksson, D., Pearce, M., Gardner, J., Turner, R. D., and Poloczek, M. Scalable global optimization via local bayesian optimization. *Advances in neural information processing systems*, 32, 2019.

Frazier, P. I., Powell, W. B., and Dayanik, S. A knowledge-gradient policy for sequential information collection. *SIAM Journal on Control and Optimization*, 47(5):2410–2439, 2008.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

Griffiths, R.-R. and Hernández-Lobato, J. M. Constrained bayesian optimization for automatic chemical design using variational autoencoders. *Chemical science*, 11(2): 577–586, 2020.

Grosnit, A., Cowen-Rivers, A. I., Tutunov, R., Griffiths, R.-R., Wang, J., and Bou-Ammar, H. Are we forgetting about compositional optimisers in bayesian optimisation? *The Journal of Machine Learning Research*, 22(1):7183–7260, 2021a.

Grosnit, A., Tutunov, R., Maraval, A. M., Griffiths, R.-R., Cowen-Rivers, A. I., Yang, L., Zhu, L., Lyu, W., Chen, Z., Wang, J., et al. High-dimensional bayesian optimisation with variational autoencoders and deep metric learning. *arXiv preprint arXiv:2106.03609*, 2021b.

Grosnit, A., Malherbe, C., Tutunov, R., Wan, X., Wang, J., and Ammar, H. B. Boils: bayesian optimisation for logic synthesis. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1193–1196. IEEE, 2022.

Han, E., Arora, I., and Scarlett, J. High-dimensional bayesian optimization via tree-structured additive models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 7630–7638, 2021.

Kandasamy, K., Schneider, J., and Póczos, B. High dimensional bayesian optimisation and bandits via additive models. In *International conference on machine learning*, pp. 295–304. PMLR, 2015.

Kandasamy, K., Dasarathy, G., Schneider, J., and Póczos, B. Multi-fidelity bayesian optimisation with continuous approximations. In *International Conference on Machine Learning*, pp. 1799–1808. PMLR, 2017.

Kandasamy, K., Krishnamurthy, A., Schneider, J., and Póczos, B. Parallelised bayesian optimisation via thompson sampling. In *International Conference on Artificial Intelligence and Statistics*, pp. 133–142. PMLR, 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kirschner, J., Mutny, M., Hiller, N., Ischebeck, R., and Krause, A. Adaptive and safe bayesian optimization in high dimensions via one-dimensional subspaces. In *International Conference on Machine Learning*, pp. 3429–3438. PMLR, 2019.

Kirschner, J., Bogunovic, I., Jegelka, S., and Krause, A. Distributionally robust bayesian optimization. In *International Conference on Artificial Intelligence and Statistics*, pp. 2174–2184. PMLR, 2020.

Letham, B., Calandra, R., Rai, A., and Bakshy, E. Re-examining linear embeddings for high-dimensional bayesian optimization. *Advances in neural information processing systems*, 33:1546–1558, 2020.

Li, C., Gupta, S., Rana, S., Nguyen, V., Venkatesh, S., and Shilton, A. High dimensional bayesian optimization using dropout. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pp. 2096–2102, 2017.

Li, C., Gupta, S., Rana, S., Nguyen, V., Venkatesh, S., and Shilton, A. High dimensional bayesian optimization using dropout. *arXiv preprint arXiv:1802.05400*, 2018.

Lu, X., Boukouvalas, A., and Hensman, J. Additive gaussian processes revisited. In *International Conference on Machine Learning*, pp. 14358–14383. PMLR, 2022.

Marchant, R. and Ramos, F. Bayesian optimisation for intelligent environmental monitoring. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 2242–2249. IEEE, 2012.

Moriconi, R., Deisenroth, M. P., and Sesh Kumar, K. High-dimensional bayesian optimization using low-dimensional feature spaces. *Machine Learning*, 109(9): 1925–1943, 2020.

Nayebi, A., Munteanu, A., and Poloczek, M. A framework for bayesian optimization in embedded subspaces. In *International Conference on Machine Learning*, pp. 4752–4761. PMLR, 2019.

Nguyen, V., Gupta, S., Rana, S., Li, C., and Venkatesh, S. Regret for expected improvement over the best-observed value and stopping condition. In *Asian conference on machine learning*, pp. 279–294. PMLR, 2017.

Qamar, S. and Tokdar, S. T. Additive gaussian process regression. *arXiv preprint arXiv:1411.7009*, 2014.

Rana, S., Li, C., Gupta, S., Nguyen, V., and Venkatesh, S. High dimensional bayesian optimization with elastic gaussian process. In *International conference on machine learning*, pp. 2883–2891. PMLR, 2017.

Rolland, P., Scarlett, J., Bogunovic, I., and Cevher, V. High-dimensional bayesian optimization via additive models with overlapping groups. In *International conference on artificial intelligence and statistics*, pp. 298–307. PMLR, 2018.

Šehić, K., Gramfort, A., Salmon, J., and Nardi, L. Lassobench: A high-dimensional hyperparameter optimization benchmark suite for lasso. In *International Conference on Automated Machine Learning*, pp. 2–1. PMLR, 2022.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and De Freitas, N. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104 (1):148–175, 2015.

Siivola, E., Paleyes, A., González, J., and Vehtari, A. Good practices for bayesian optimization of high dimensional structured spaces. *Applied AI Letters*, 2(2):e24, 2021.

Srinivas, N., Krause, A., Kakade, S. M., and Seeger, M. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

Srinivas, N., Krause, A., Kakade, S. M., and Seeger, M. W. Information-theoretic regret bounds for gaussian process optimization in the bandit setting. *IEEE transactions on information theory*, 58(5):3250–3265, 2012.

Tripp, A., Daxberger, E., and Hernández-Lobato, J. M. Sample-efficient optimization in the latent space of deep generative models via weighted retraining. *Advances in Neural Information Processing Systems*, 33:11259–11272, 2020.

Turner, R., Eriksson, D., McCourt, M., Kiili, J., Laaksonen, E., Xu, Z., and Guyon, I. Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. In *NeurIPS 2020 Competition and Demonstration Track*, pp. 3–26. PMLR, 2021.

Wan, X., Nguyen, V., Ha, H., Ru, B., Lu, C., and Osborne, M. A. Think global and act local: Bayesian optimisation over high-dimensional categorical and mixed search spaces. *arXiv preprint arXiv:2102.07188*, 2021.

Wang, Z., Zoghi, M., Hutter, F., Matheson, D., De Freitas, N., et al. Bayesian optimization in high dimensions via random embeddings. In *IJCAI*, volume 13, pp. 1778–1784, 2013.

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

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

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. Deep kernel learning. In *Artificial intelligence and statistics*, pp. 370–378. PMLR, 2016.

Wilson, J. T., Moriconi, R., Hutter, F., and Deisenroth, M. P. The reparameterization trick for acquisition functions. *arXiv preprint arXiv:1712.00424*, 2017.

Zela, A., Siems, J., and Hutter, F. Nas-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search. *arXiv preprint arXiv:2001.10422*, 2020.

Zhang, M., Li, H., and Su, S. High dimensional bayesian optimization via supervised dimension reduction. *arXiv preprint arXiv:1907.08953*, 2019.

Zhang, Y., Zhang, X., and Frazier, P. Constrained two-step look-ahead bayesian optimization. *Advances in Neural Information Processing Systems*, 34:12563–12575, 2021.

Zhang, Y., Li, X., and Wang, L. Distributed h-infinity consensus of heterogeneous multi-agent systems with nonconvex constraints. *ISA transactions*, 131:160–166, 2022.

Ziegel, E. R. The elements of statistical learning, 2003.

A. Proofs of Theoretical Results

A.1. Proof of Theorem 4.1

Theorem 4.1. *Let the black-box function f be selected by an adversary from an RKHS $\mathcal{H}^g$ of kernel k^g, defined over some decomposition g that is also selected by an adversary. After T rounds, a UCB-style BO algorithm with an $S(t) : \mathbb{Z}^+ \to \mathcal{G}$ decomposition rule, incurs with a probability of at least $1-\delta$ the following total cumulative regret R_T:*

$$R_T = \mathcal{O}\left(\sqrt{T\gamma_T}\left(B + \sqrt{\ln\frac{1}{\delta} + \gamma_T} + \sum_{t=1}^{T}\epsilon_t\right)\right),$$

where $B = \max_{t\in T}\|\hat{f}_t\|_t$ and $\|\cdot\|_t$ denotes the norm in $\mathcal{H}_t$.

Proof. Without the loss of generality, we assume kernel for every decomposition is normalised, i.e. $\forall_{g\in\mathcal{G}} k^g(\boldsymbol{x},\boldsymbol{x}') \leq 1$ and that our function observations are corrupted by some σ_n-subgaussian noise. Our proof follows the idea of the proof of Theorem 1 from (Bogunovic & Krause, 2021), with some differences. Let $k_t = k^{g_t}$ be the kernel defined by decomposition g_t selected at time t. $\hat{f}^t = \text{argmin}_{f'\in\mathcal{H}^t}|f-f'|_\infty$ be a function living in RKHS $\mathcal{H}^t$ of kernel k_t that is closest to the true function f measured by the infinity norm. To avoid clutter we will write $\mu_t = \mu_t^{g_t}$ and $\sigma_t = \sigma_t^{g_t}$. Let $\boldsymbol{y}_t^*$ be the vector of corrupted observations generated by $\hat{f}_t$, i.e. $\boldsymbol{y}_t^* = (\hat{f}_t(\boldsymbol{x}_1)+n_1,\ldots,\hat{f}_t(\boldsymbol{x}_t)+n_t)$ for some σ_n-subgaussian noise n_t. Then let us define μ_t^* as the Gaussian Process posterior utilising kernel k_t, which uses observations $\boldsymbol{y}_t^*$, i.e.:

$$\mu_t^*(\boldsymbol{x}) = \boldsymbol{k}_t^\mathsf{T}(\boldsymbol{x})\left(\boldsymbol{K}_t + \sigma_n^2\boldsymbol{I}\right)^{-1}\boldsymbol{y}_t^*$$

In contrast, the mean computed by the algorithm $\mu_t(\boldsymbol{x})$ uses the full observations $\boldsymbol{y}_t = (f(\boldsymbol{x}_1)+n_1,\ldots,f(\boldsymbol{x}_t)+n_t)$ from the true black-box function f, without knowing which part came from $\hat{f}_t$. Let us assume, we run a UCB-style BO algorithm, with an acquisition function $\alpha_t(\boldsymbol{x})$ that we will define later. Let $\boldsymbol{x}^* = \text{argmax}_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})$ be the maximiser of black-box function. If we now look at the instantaneous regret, we get:

$$r_t = f(\boldsymbol{x}^*) - f(\boldsymbol{x}_t) = f(\boldsymbol{x}^*) - \max_{\boldsymbol{x}\in\mathcal{X}}\hat{f}_t(\boldsymbol{x}) + \max_{\boldsymbol{x}\in\mathcal{X}}\hat{f}_t(\boldsymbol{x}) - f(\boldsymbol{x}_t) \leq$$

$$\leq \left(f(\boldsymbol{x}^*) - \hat{f}_t(\boldsymbol{x}^*)\right) + \left(\hat{f}_t(\boldsymbol{x}^*) - \max_{\boldsymbol{x}\in\mathcal{X}}\hat{f}_t(\boldsymbol{x})\right) + \left(\max_{\boldsymbol{x}\in\mathcal{X}}\hat{f}_t(\boldsymbol{x}) - \hat{f}_t(\boldsymbol{x}_t)\right) + \left(\hat{f}_t(\boldsymbol{x}_t) - f(\boldsymbol{x}_t)\right) \leq$$

$$\leq 2\epsilon_t + \max_{\boldsymbol{x}\in\mathcal{X}}\hat{f}_t(\boldsymbol{x}) - \hat{f}_t(\boldsymbol{x}_t), \tag{1}$$

where the last inequality is due to the fact that $|f(\boldsymbol{x}) - \hat{f}_t(\boldsymbol{x})| \leq \epsilon_t$ and $\hat{f}_t(\boldsymbol{x}^*) \leq \max_{\boldsymbol{x}\in\mathcal{X}}\hat{f}_t(\boldsymbol{x})$. Consequently, $\max_{\boldsymbol{x}\in\mathcal{X}}\hat{f}_t(\boldsymbol{x}) - \hat{f}_t(\boldsymbol{x}_t)$ becomes the new term we need to bound. We now observe the following:

$$\hat{f}_t(\boldsymbol{x}) = \mu_{t-1}(\boldsymbol{x}) + \left(\hat{f}_t(\boldsymbol{x}) - \mu_{t-1}^*(\boldsymbol{x})\right) + \left(\mu_{t-1}^*(\boldsymbol{x}) - \mu_{t-1}(\boldsymbol{x})\right)$$

Hence, we would like to bound the differences $\hat{f}_t(\boldsymbol{x}) - \mu_t^*(\boldsymbol{x})$ and $\mu_t(\boldsymbol{x}) - \mu_t^*(\boldsymbol{x})$. To do so, we recall two lemmas from existing literature.

Lemma A.1 (Adapted Theorem 2 from (Chowdhury & Gopalan, 2017))**.** *Let $\mathcal{X}\subset\mathbb{R}^d$, and $\hat{f}_t : \mathcal{X}\to\mathbb{R}$ be a member of the RKHS of real-valued functions on $\mathcal{X}$ with kernel k_t defined by $g_t\in\mathcal{G}$, where g_t can be changed at each t, with RKHS norm bounded by $\|\hat{f}_t\|_t \leq B$. Let the observations $\boldsymbol{y}_t$ be corrupted by some σ-subgaussian noise. Then, with probability at least $1-\delta$, the following holds for all $\boldsymbol{x}\in\mathcal{X}$:*

$$|\mu_{t-1}^*(\boldsymbol{x}) - \hat{f}_t(\boldsymbol{x})| \leq \left(B + \sigma\sqrt{2\gamma_{t-1} + 1 + \ln(1/\delta))}\right)\sigma_{t-1}(\boldsymbol{x})$$

Proof. The proof is identical to the proof of Theorem 2 in (Chowdhury & Gopalan, 2017), except that now the kernel depends on time.

□

Lemma A.2 (Lemma 2 from (Bogunovic & Krause, 2021)). *For any $\boldsymbol{x} \in \mathcal{X}$ and $t \geq 1$, we have*

$$|\mu_{t-1}(\boldsymbol{x}) - \mu^*_{t-1}(\boldsymbol{x})| \leq \frac{\sigma_{t-1}(\boldsymbol{x})}{\sigma_n}\epsilon_t\sqrt{t},$$

where $\epsilon_t = \min_{f' \in \mathcal{H}^t} |f - f'|_\infty$.

We adopt the notation $\hat{\boldsymbol{x}}_t = \max_{\boldsymbol{x}\in\mathcal{X}} \hat{f}_t(\boldsymbol{x})$. Following up on Inequality 1, by Lemmas A.1 and A.2 with probability at least $1-\delta$, we get that the cumulative regret R_T admits the following upper bound:

$$R_T = \sum_{t=1}^T r_t \leq \sum_{t=1}^T 2\epsilon_t + \sum_{t=1}^T \hat{f}_t(\hat{\boldsymbol{x}}) - \hat{f}_t(\boldsymbol{x}_t) \leq$$

$$\leq \sum_{t=1}^T 2\epsilon_t + \sum_{t=1}^T \mu_{t-1}(\hat{\boldsymbol{x}}_t) + \beta_t\sigma_{t-1}(\hat{\boldsymbol{x}}_t) - (\mu_{t-1}(\boldsymbol{x}_t) - \beta_t\sigma_{t-1}(\boldsymbol{x}_t)),$$

where we have introduced β_t defined as:

$$\beta_t = B + \sigma\sqrt{2(\gamma_{t-1} + 1 + \ln(1/\delta))} + \frac{\epsilon_t\sqrt{t}}{\sigma_n}. \tag{2}$$

Let us now define the acquisition rule of our BO algorithm as $\max_{\boldsymbol{x}\in\mathcal{X}} \alpha_t(\boldsymbol{x})$, where

$$\alpha_t(\boldsymbol{x}) = \max_{\boldsymbol{x}\in\mathcal{X}} \mu_{t-1}(\boldsymbol{x}) + \beta_t\sigma_{t-1}(\boldsymbol{x})$$

By the acquisition rule, if point $\boldsymbol{x}_t$ was selected, then $\mu_{t-1}(\boldsymbol{x}_t) + \beta_t\sigma_{t-1}(\boldsymbol{x}_t) \geq \mu_{t-1}(\hat{\boldsymbol{x}}_t) + \beta_t\sigma_{t-1}(\hat{\boldsymbol{x}}_t)$ and so:

$$R_T \leq \sum_{t=1}^T 2\epsilon_t + \sum_{t=1}^T \mu_{t-1}(\hat{\boldsymbol{x}}_t) + \beta_t\sigma_{t-1}(\hat{\boldsymbol{x}}_t) - \mu_{t-1}(\boldsymbol{x}_t) + \beta_t\sigma_{t-1}(\boldsymbol{x}_t)$$

$$\leq \sum_{t=1}^T 2\epsilon_t + \sum_{t=1}^T \mu_{t-1}(\boldsymbol{x}_t) + \beta_t\sigma_{t-1}(\boldsymbol{x}_t) - \mu_{t-1}(\boldsymbol{x}_t) + \beta_t\sigma_{t-1}(\boldsymbol{x}_t) = \sum_{t=1}^T \epsilon_t + \sum_{t=1}^T 2\beta_t\sigma_{t-1}(\boldsymbol{x}_t)$$

Substituting the definition of β_t (Eq. 2) we get:

$$R_T \leq \sum_{t=1}^T 2\epsilon_t + \sum_{t=1}^T 2\left(B + \sigma\sqrt{2(\gamma_t + 1 + \ln(1/\delta))} + \frac{\epsilon_t\sqrt{t}}{\sigma_n}\right)\sigma_{t-1}(\boldsymbol{x}_t) \leq$$

$$\leq \sum_{t=1}^T 2\epsilon_t + 2\left(B + \sigma\sqrt{2(\gamma_T + 1 + \ln(1/\delta))}\right)\sum_{t=1}^T \sigma_{t-1}(\boldsymbol{x}_t) + \frac{\sqrt{T}}{\sigma_n}\sum_{t=1}^T \epsilon_t\sigma_{t-1}(\boldsymbol{x}_t)$$

We now observe that by Cauchy-Schwarz we have $\sum_{t=1}^T \sigma_{t-1}(\boldsymbol{x}_t) \leq \sqrt{T\sum_{t=1}^T \sigma^2_{t-1}(\boldsymbol{x}_t)}$ and $\sum_{t=1}^T \epsilon_t\sigma_{t-1}(\boldsymbol{x}_t) \leq \sqrt{\sum_{t=1}^T \epsilon_t^2}\sqrt{\sum_{t=1}^T \sigma^2_{t-1}(\boldsymbol{x}_t)}$. We thus obtain the following bound:

$$R_T \leq \sum_{t=1}^T 2\epsilon_t + 2\left(B + \sigma\sqrt{2(\gamma_T + 1 + \ln(1/\delta))}\right)\sqrt{T\sum_{t=1}^T \sigma^2_{t-1}(\boldsymbol{x}_t)} + \frac{1}{\sigma_n}\sqrt{\sum_{t=1}^T \epsilon_t^2}\sqrt{T\sum_{t=1}^T \sigma^2_{t-1}(\boldsymbol{x}_t)}$$

Observe that $\sigma^2_{t-1}(\boldsymbol{x}_t) \leq \sigma_n^2 C\log(1 + \sigma_n^{-2}\sigma^2_{t-1}(\boldsymbol{x}_t))$, where $C = \sigma_n^{-2}/\log(1+\sigma_n^{-2})$ and by Lemma 5.3 of (Srinivas et al., 2009), we have $\sum_{t=1}^T \log(1 + \sigma_n^{-2}\sigma^2_{t-1}(\boldsymbol{x}_t)) \leq 2\gamma_t$. We thus obtain:

$$R_T \leq \sum_{t=1}^T 2\epsilon_t + 2\left(B + \sigma\sqrt{2(\gamma_T + 1 + \ln(1/\delta))}\right)\sqrt{T\gamma_T} + \frac{\sqrt{T\gamma_T}}{\sigma_n}\sqrt{\sum_{t=1}^T \epsilon_t^2}$$

$$R_T = \mathcal{O}\left(\sqrt{T\gamma_T}\left(B + \sqrt{\ln\frac{1}{\delta} + \gamma_T} + \sum_{t=1}^{T}\epsilon_t\right)\right),$$

since due to $\forall_t \epsilon_t \geq 0$, we have $\sqrt{\sum_{t=1}^{T}\epsilon_t^2} \leq \sum_{t=1}^{T}\epsilon_t$.

□

A.2. Proof of Corollary 4.2

Corollary 4.2. *Under the assumptions of Theorem 4.1, we have with probability at least $1-\delta_A-\delta_B$, the cumulative regret R_T of a UCB-style BO algorithm utilising the decomposition suggesting scheme $S(t)$, incurs the following cumulative regret:*

$$R_T = \mathcal{O}\Bigg(\sqrt{T\gamma_T}\Bigg(B + \sqrt{\ln\frac{1}{\delta_A} + \gamma_T} + \frac{\mathbb{E}_S\left[\sum_{t=1}^{T}\epsilon_t\right]}{\delta_B}\Bigg)\Bigg).$$

Proof. Using Markov's inequality we have: $\mathbb{P}\text{r}\left(\sum_{t=1}^{T}\epsilon_t \geq \mathbb{E}_S\left[\sum_{t=1}^{T}\epsilon_t\right]/\delta_B\right) \leq \delta_B$. We now combine this fact with the bound developed in Theorem 4.1 and combine the probabilities using union bound to arrive at the corollary's statement. □

A.3. Proof of Proposition 4.3

Proposition 4.3. *The maximum information gain for a squared exponential kernel following a decomposition only with pair-wise components in d dimensions is upper-bounded by $\gamma_T \leq \mathcal{O}(d(\log T)^3)$.*

Proof. As shown in (Rolland et al., 2018), for an additive squared exponential kernel in A dimensional space, such that each subkernel operates on at most B dimensions, we get that the maximum information gain is bounded as:

$$\gamma_T \leq AB^B \log T^{B+1}$$

We observe that in our case $A = d$ and $B = 2$, which finishes the proof. □

A.4. Proof of Proposition 4.5

Proposition 4.5. *When $d > 2$, the maximum information gain for a kernel following a tree decomposition (possibly changing between timesteps) is always smaller than for a kernel containing all pairwise interactions.*

Proof. We will denote by $\mathcal{G}$ the class of all possible tree decompositions in d dimensions. As such $\bigcup_{g\in\mathcal{G}}$ is the set of all pairwise components. Additionally, we will introduce the following notation:

$$f^{\text{full}} = \sum_{c\in\bigcup_{g\in\mathcal{G}}} f_c \qquad y_t^{\text{full}} = f^{\text{full}}(\boldsymbol{x}_t) + \epsilon_t$$

$$\hat{f}_t = \sum_{c\in g_t} f_c \qquad y_t = \hat{f}_t(\boldsymbol{x}_t) + \epsilon_t,$$

where the quantities have the following distribution for all $c \in \bigcup_{g\in\mathcal{G}}$ and all $t > 0$:

$$f_c \sim \mathcal{GP}(0, k(\boldsymbol{x}, \boldsymbol{x}')) \qquad \epsilon_t \sim \mathcal{N}(0, \sigma_n^2).$$

For some fixed sequence $\boldsymbol{X}_T = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$, we will write $\boldsymbol{f}_T = (f_1(x_1), \ldots, \hat{f}_T(x_T))$ and $\boldsymbol{y}_T = (y_1, \ldots, y_T)$ and equivalently for $\boldsymbol{f}^{\text{full}}$ and $\tilde{\boldsymbol{y}}_T$. By properties of mutual information, we have:

$$I(\boldsymbol{y}_T^{\text{full}}, \boldsymbol{f}^{\text{full}}) - I(\boldsymbol{y}_T, \boldsymbol{f}_T) = H(\boldsymbol{y}_T^{\text{full}}) - H(\boldsymbol{y}_T^{\text{full}}|\boldsymbol{f}^{\text{full}}) - H(\boldsymbol{y}_T) + H(\boldsymbol{y}_T|\boldsymbol{f}_T) = H(\boldsymbol{y}_T^{\text{full}}) - H(\boldsymbol{y}_T),$$

where the last inequality is true, as the conditional distributions are the same. Using the formula for the entropy of multivariate Gaussian we get:

$$H(\boldsymbol{y}_T^{\text{full}}) - H(\boldsymbol{y}_T) = \frac{1}{2}\ln\left(\frac{\det(\mathcal{I}\sigma_n^2 + K_T^{\text{full}})}{\det(\mathcal{I}\sigma_n^2 + K_T)}\right) = \frac{1}{2}\sum_{t=1,\dots,T}\ln\left(\frac{\sigma_n^2 + \lambda_t(K_T^{\text{full}})}{\sigma_n^2 + \lambda_t(K_T)}\right),$$

where $\lambda_t(A)$ means the t-th largest eigenvalue of A and the covariance matrices are defined as $(K_T^{\text{full}})_{i,j} = \sum_{c\in\bigcup_{g\in\mathcal{G}}} k_c(x_i, x_j)$ and $(K_T)_{i,j} = \sum_{c\in g_i\cap g_j} k_c(x_i, x_j)$. We can thus write $K_T^{\text{full}} = K_T + \dot{K}_T$, where $(\dot{K}_T)_{i,j} = \sum_{c\notin g_i\cap g_j} k_c(x_i, x_j)$. One can easily see that $\dot{K}_T$ must be PSD, as such we have $\lambda_t(K_T^{\text{full}}) \geq \lambda_t(K_T)$, as adding a PSD matrix to another PSD matrix can never decrease its eigenvalues. This gives us:

$$\frac{1}{2}\sum_{t=1,\dots,T}\ln\left(\frac{\sigma_n^2 + \lambda_t(K_T^{\text{full}})}{\sigma_n^2 + \lambda_t(K_T)}\right) \geq \frac{1}{2}\sum_{t=1,\dots,T}\ln(1) = 0,$$

with equality if and only if $\forall_{t=1,\dots,T}\lambda_t(K_T^{\text{full}}) = \lambda_t(K_T)$, which can happen only if all eigenvalues of $\dot{K}_T$ are zero, meaning $\forall_{t>0} g_t = \bigcup_{g\in\mathcal{G}}$, i.e. all components are included at every step. This can only happen if there is only one decomposition in the class $|\mathcal{G}|$. As our class $\mathcal{G}$ consist of trees, this is only possible when $d \leq 2$. □

A.5. Proof of Theorem 4.6

Theorem 4.6. *Let $\mathcal{G}$ be a class of d-dimensional tree-based decompositions with E edges. Let the adversary choose any function f from RKHS $\mathcal{H}_g$ defined by kernel k_g on a decomposition $g \in \mathcal{G}$ also chosen by the adversary, such that the infinity norm of each component function is bounded $\forall_{c\in g}|f_c|_\infty \leq M_c$ and $\sum_{c\in g} M_c \leq M$. We then have that for any data-independent scheme S, the expected sum of mismatches is at least as large as for the scheme S_r that selects a decomposition from $\mathcal{G}$ uniformly at random, i.e.,*

$$\forall_{S:\mathbb{Z}^+\to\mathcal{G}} \quad \mathbb{E}_S\left[\sum_{t=1}^T \epsilon_t\right] \geq \mathbb{E}_{S_r}\left[\sum_{t=1}^T \epsilon_t\right].$$

Proof. We start by introducing the following lemma.

Lemma A.3. *Let g be the true tree-based decomposition of f and $\tilde{g} = g \setminus \tilde{C}$ for some set of omitted pair-wise components $\tilde{\mathcal{C}}$. Then we have:*

$$\min_{f'\in\mathcal{H}^{\tilde{g}}} |f - f'|_\infty \leq \sum_{c\in\tilde{C}} M_c$$

Proof.

$$\min_{f'\in\mathcal{H}^{\tilde{g}}} |f - f'|_\infty = \min_{f'\in\mathcal{H}^{\tilde{g}}} |\sum_{c\notin\tilde{\mathcal{C}}} f_c + \sum_{c\in\tilde{\mathcal{C}}} f_c - f'|_\infty \leq \min_{f'\in\mathcal{H}^{\tilde{g}}} |\sum_{c\notin\tilde{\mathcal{C}}} f_c - f'|_\infty + \sum_{c\in\tilde{\mathcal{C}}} |f_c|_\infty \leq \sum_{c\in\tilde{\mathcal{C}}} M_c$$

where the first inequality is due to the triangle inequality and second due to the fact that $|f_c|_\infty \leq M_c$ and $\sum_{c\notin\tilde{\mathcal{C}}} f_c \in \mathcal{H}_{\tilde{g}}$. □

Let us now define the probability of choosing a decomposition g at time t by our decomposition proposing scheme as $P_t^{\mathcal{G}}(g)$. A deterministic scheme will just be a special case of the probabilistic scheme, where all probability is concentrated on one decomposition. Instead of thinking about the adversary as selecting a function f, we can think about them as selecting the norm parameters for each pair-wise component M_c, with the constraint that $\sum_{c\in g} M_c \leq M$. Since the adversary knows the scheme, in the worst case they can select the decomposition and function so that the expected mismatch is maximal. This corresponds to:

$$\max_{g,f} \mathbb{E}_S\left[\sum_{t=1}^T \epsilon_t\right] = \max_{g,f} \sum_{g'\in\mathcal{G}} \sum_{t=1}^T \min_{f'\in\mathcal{H}^{g'}} |f - f'|_\infty P_t^{\mathcal{G}}(g') = \max_{g,\{M_c\}_{c\in g}} \sum_{t=1}^T \sum_{g'\in\mathcal{G}} \sum_{c\in g} M_c \mathbf{1}_{c\notin g'} P_t^{\mathcal{G}}(g'),$$

where the last equality is due to Lemma A.3 and the fact that the adversary, in the worst case, will choose a function with the highest possible mismatch. We can now exchange the order of summation to obtain:

$$\max_{g,f} \mathbb{E}_S \left[\sum_{t=1}^{T} \epsilon_t \right] = \max_{g,\{M_c\}_{c\in g}} \sum_{c\in g} M_c \sum_{g'\in\mathcal{G}} \mathbf{1}_{c\notin g'} \sum_{t=1}^{T} P_t^{\mathcal{G}}(g') = \max_{g,\{M_c\}_{c\in g}} \sum_{c\in g} M_c \mathbb{E}_S[N_{\neg c}],$$

where $\mathbb{E}_S[N_{\neg c}] = \sum_{g'\in\mathcal{G}} \mathbf{1}_{c\notin g'} \sum_{t=1}^{T} P_t^{\mathcal{G}}(g')$ is the expected number of time that a pair-wise component c is **not** included in the proposed decomposition. Note that the expression above is maximised subject to the constraint that $\sum_{c\in\mathcal{C}} M_c \leq M$. Thus the maximum is achieved when the biggest mismatch is placed on the pair-wise component that is on average least selected. Formally, let $c^* = \arg\max_{c\in\mathcal{C}} \mathbb{E}_S[N_{\neg c}]$, then the solution to the constrained maximization problem above is $M_c = M\mathbf{1}_{c=c^*}$ and g can be any decomposition including c^*. We thus obtain:

$$\max_{g,f} \mathbb{E}_S \left[\sum_{t=1}^{T} \epsilon_t \right] = M \max_{c\in\mathcal{C}} \mathbb{E}_S[N_{\neg c}].$$

This is minimised, when for the selected scheme S the expected number of times the least selected pair-wise component is not selected is minimal. This happens when the chance to include each of the pair-wise components is the same. Since all of the decompositions have the same number of pair-wise components, this corresponds to a uniform distribution over all trees. This proves the claim that a uniformly random scheme achieves the smallest expected mismatch. Under the uniformly random scheme, we select E pair-wise components at each time out of all possible $d(d-1)/2$, so the probability of any one component being selected is $\frac{2E}{d(d-1)}$ and the inverse event has a probability of $1 - \frac{2E}{d(d-1)}$. Since the scheme is the same across all timesteps we get $\mathbb{E}_S[N_{\neg c}] = T\left(1 - \frac{2E}{d(d-1)}\right)$.

□

B. Procedure for Sampling random Trees

Algorithm 2 Random Tree Sampler

```
Input: # of edges E, dimensionality of the problem d
Set L = [1, ..., d]
Create L_in and L_out by randomly permuting L
Initialise a Union-Find structure UF & empty graph g
for n_in ∈ L_in do
    for n_out ∈ L_out do
        if !UF.connected(n_in, n_out) then
            UF.union(n_in, n_out)
            g.add_edge(n_in, n_out)
        end if
        if g.number_of_edges() = E then
            Return g
        end if
    end for
end for
```

C. Algorithm Settings

In Table 2, we detail settings used by each algorithm. Those values are used for all of the experiments.

Algorithm	Setting	Value
Tree	Acquisition function Decomposition learning interval Gibbs sampling iterations	Additive UCB with $\beta_t = 0.5\log(2t)$ 15 100
RDUCB	Acquisition function Size of random tree	Additive UCB with $\beta_t = 0.5\log(2t)$ $\max\{\lfloor d/5 \rfloor, 1\}$
HeSBO	Acquisition function Size of embedding	EI (Nguyen et al., 2017) $\sqrt{d}$
REMBO/ CoordinateLineBO	All	Default values from here

Table 2. Hyperparameters used by algrotihms for all experiments. t denotes the timestep and d dimensionality of the problem.

C.1. Computing Resources

All experiments were run on machines with specifications described in Table 3.

Component	**Description**
CPU	Intel Core i9-9900X CPU @ 3.50GHz
GPU	Nvidia RTX 2080
Memory	64 GB DDR4

Table 3. Specifications of machines used to run experiments.

D. Toy problem details

In this section, we describe the details of the toy problem introduced in Section 3. For this experiment, we use algorithm setting as per Table 2. The function we use is three-dimensional, where the last dimension is redundant. Thus, in Figure 1 we only plot it as a function of two variables. We chose to add one redundant dimension as otherwise, RDUCB will always be sampling the same decomposition (size of random tree $E = 1$). The formula for the function is given below:

$$f(x, y, z) = w_1\mathcal{N}_{x,y}(\mu_1, \Sigma_1) + w_2\mathcal{N}_x(\mu_2, \Sigma_2) + w_3\mathcal{N}_y(\mu_3, \Sigma_3),$$

where $\mathcal{N}_c(\mu, \sigma)$ is a $|c|$-dimensional Gaussian PDF defined on dimensions in c with mean μ and covariance matrix Σ. For the toy problem, we used the numerical values shown below.

$$w_1 = 1/6 \quad w_2 = w_3 = 2.5/6$$

$$\mu_1 = (800, 800)^T \quad \mu_2 = \mu_3 = (300)$$

$$\Sigma_1 = \begin{pmatrix} 20000 & 15000 \\ 15000 & 20000 \end{pmatrix} \quad \Sigma_2 = \Sigma_3 = (10000)$$

Thus, there are two local optima for x and y, suboptimal at $(300, 300)$ and global at $(800, 800)$. Variable z can be set to any value, as it does not affect the function output. The initial points given to both Tree and RDUCB were exactly the same. We show them in Figure 5 below.

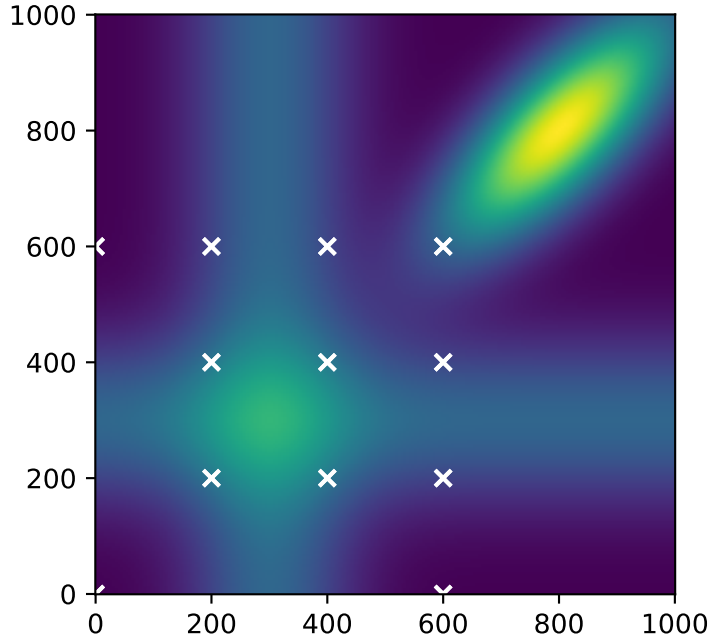
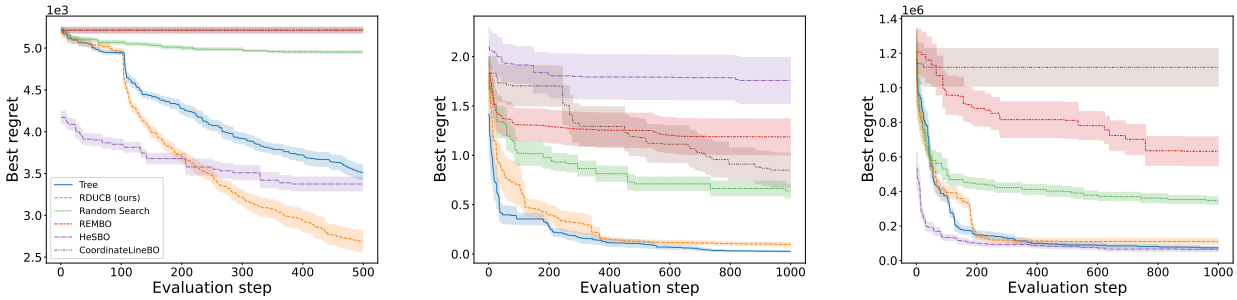


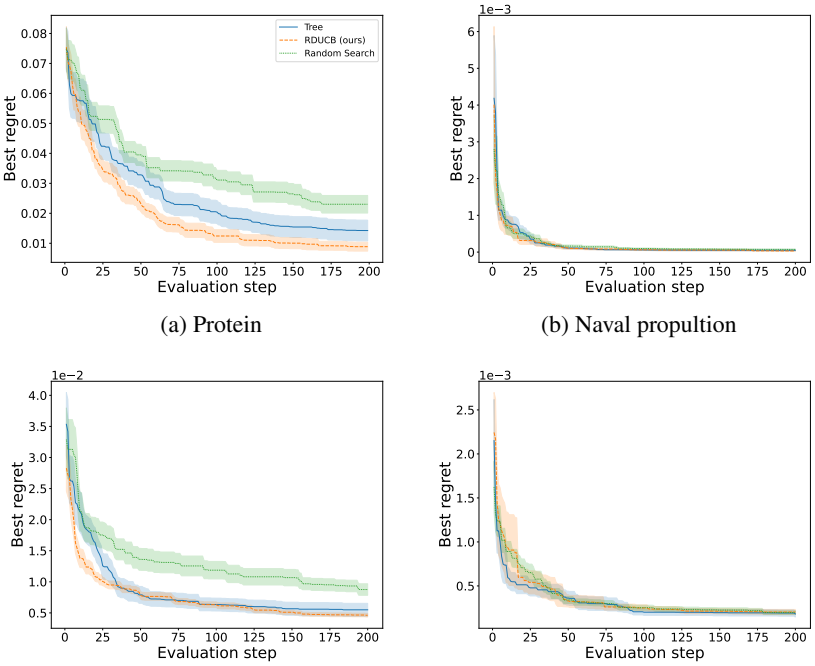
Figure 5. Initial points given to both Tree and RDUCB on the toy problem.

E. Additional Experimental Results



(a) Stybtang250 (b) Hartmann6+14 (c) Rosenbrock20

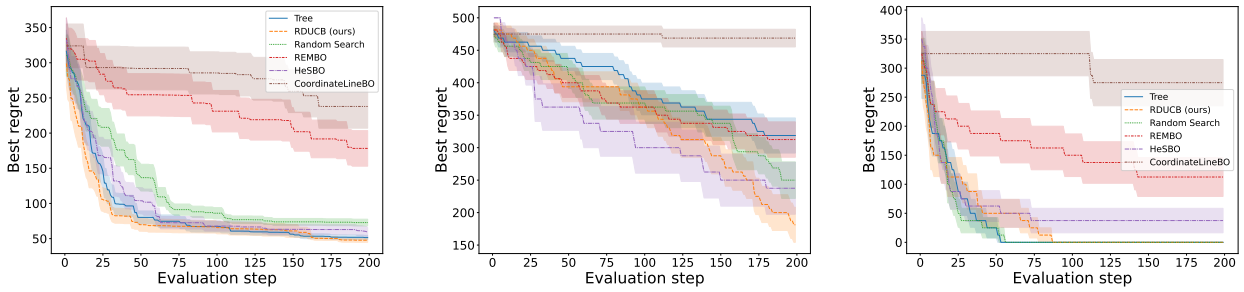
Figure 6. Performance comparison on selected synthetic functions. Solid lines are the mean values over 10 seeds, and shaded areas correspond to standard error.



(a) Protein (b) Naval propultion

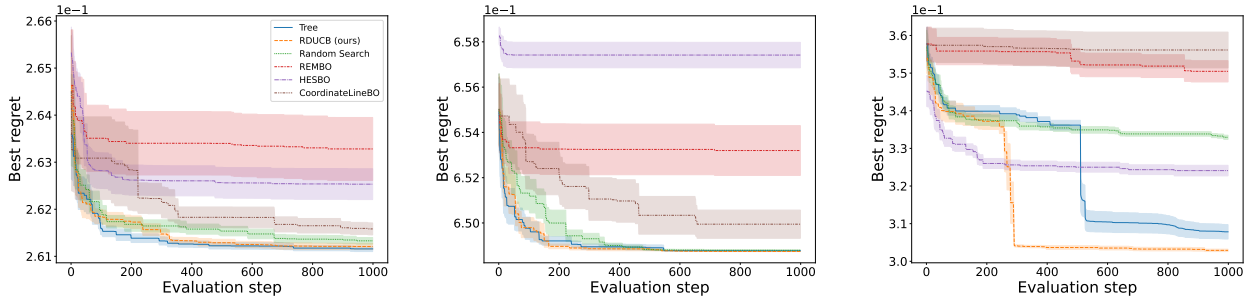
(c) Parkinson telemonitoring (d) Slice Locatization

Figure 7. Performance comparison on NAS benchmarks. Solid lines are the mean values over 20 seeds, and shaded areas correspond to standard error.



(a) qiu (b) misc05inf (c) mtest4ma

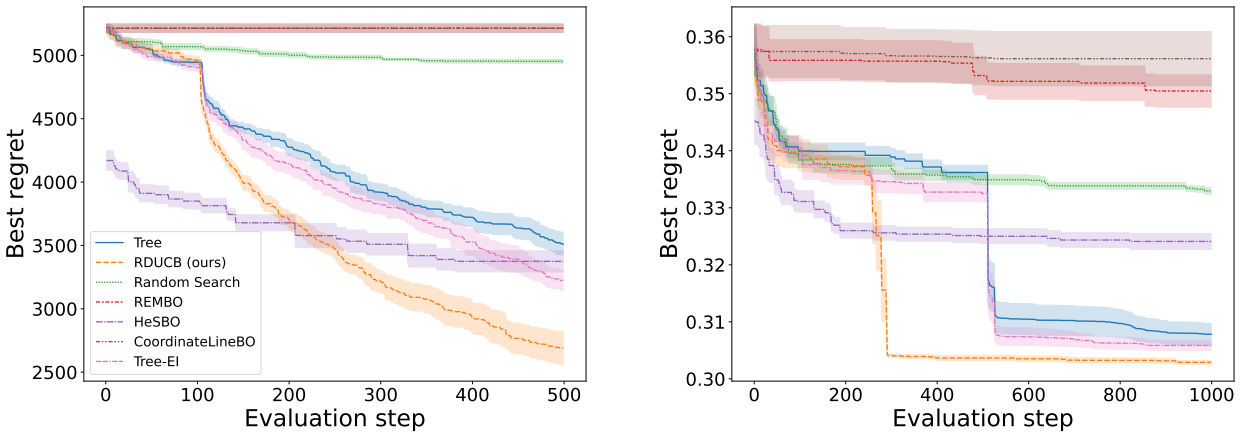
Figure 8. Performance comparison on selected MIP hyperparameter tuning problems. Solid lines are the mean values over 40 seeds and shaded areas correspond to standard error.



(a) Breast Cancer (10 dim.) (b) Diabetes (8 dim.) (c) DNA (180 dim.)

Figure 9. Performance comparison on LassoBench problems.

F. Comparing RDUCB to Tree with different acquisition functions



(a) Stybtang250

(b) Lasso DNA

Figure 10. Additional experiments over two highest dimensional tasks, comparing the improvement brought by RDUCB to the improvement brought by using a different acquisition function. Tree-EI stands for the Tree algorithm (Han et al., 2021) utilising the expected-improvement acquisition function (Nguyen et al., 2017). We develop an additive version of the expected improvement, where the term corresponding to component c is given by $\alpha_c(\boldsymbol{x}|\mathcal{D}_{t-1}) = (\mu_{t-1,c}(\boldsymbol{x}) - \mu_{t-1}(\boldsymbol{x}^+))\Phi(\frac{\mu_{t-1,c}(\boldsymbol{x})-\mu_{t-1,c}(\boldsymbol{x}^+)}{\sigma_{t-1,c}(\boldsymbol{x})}) + \sigma_{t-1,c}(\boldsymbol{x})\phi(\frac{\mu_{t-1,c}(\boldsymbol{x})-\mu_{t-1,c}(\boldsymbol{x}^+)}{\sigma_{t-1,c}(\boldsymbol{x})})$, in which x^+ is the best point found so far and $\Phi()$ and $\phi()$ are Gaussian CDF and PDF, respectively. Shaded areas correspond to standard errors over 10 seeds. We see that although choosing a different acquisition function might bring some improvement, it is smaller compared to the improvement delivered by RDUCB.