Additive Acquisition Function Study on a Random Decomposable Bayesian Optimization method

Effectiveness of different acquisition functions on RDUCB

Lex Janssens
s2989344@umail.leidenuniv.nl
LIACS, Leiden University
Leiden, Netherlands

Abygaïl Stegenga
s2267063@umail.leidenuniv.nl
LIACS, Leiden University
Leiden, Netherlands

ACM Reference Format:
Lex Janssens and Abygaïl Stegenga . 2024. Additive Acquisition Function Study on a Random Decomposable Bayesian Optimization method: Effectiveness of different acquisition functions on RDUCB. In *Proceedings of Bayesian Optimization 2024-2025*. ACM, New York, NY, USA, 5 pages.

1 INTRODUCTION

Bayesian Optimization (BO) tackles expensive black-box optimization problems efficiently. However, even traditional BO methods seem to have difficulty with tasks of higher dimensionality. High-dimensional Bayesian Optimization algorithms tackle these kinds of problems with algorithms in their own way. One of these methods is via additive models. We will be looking at one additive model, RDUCB [11], and focus on its acquisition function. We will be modularly interchanging acquisition functions and testing the configurations on multiple BBOB functions from the COCO environment. The acquisition functions are adjusted such that we can use them in additive models. The BBOB functions provide full coverage of the different types of tasks one may expect in a BO setting, from separable problems to weakly structured multi-modality problems. We can conclude that Probability of Improvement does not perform well as an acquisition function for random tree decompositions due to its tendency to become stuck in local optima. We also see that Expected Improvement performs similarly to UCB, with Expected Improvement performing better in weakly structured functions.

2 BACKGROUND

In this paper, we are dealing with multiple black-box optimization algorithms. They are known to be derivative-free, and the structure remains unknown, unexploitable, and/or nonexistent. The task often proposed with such optimization algorithms is to find $\vec{x}^* \in \mathcal{X}$ for which $\vec{x}^* = \text{argmin}_{\vec{x}} f(\vec{x})$ or $\vec{x}^* = \text{argmax}_{\vec{x}} f(\vec{x})$, where $\mathcal{X} \subseteq \mathbb{R}^d$ is a search space. In the easiest case, no constraints apply. However, it may come by that $\vec{x}$ should suffice to some constraints $g_i(\vec{x}) \leq 0$. Finding $\vec{x}^*$ is a difficult task, especially when the black-box function is expensive. Common optimization methods such as grid search and random search are not applicable in this field. Therefore, Bayesian Optimization (BO) is used.

BO addresses the optimization of a black-box algorithm by picking $\vec{x}$ that is likely to be optimal and gaining as much information from some sampled $\vec{x}$ as possible to sample the next candidate. This is done by defining a surrogate model based on dataset $\mathcal{D} = \{(\vec{x}_i : f(\vec{x}_i)\}$, a collection of all sampled points and their objective values. The surrogate model is based on a multivariate Gaussian distribution and is usually a Gaussian Process (GP). Before fitting $\mathcal{D}$, it is called a prior. After fitting $\mathcal{D}$, it is called a posterior and is defined via a $\mu(\vec{x})$ and a $\sigma(\vec{x})$, being the mean and standard deviation on some sample $\vec{x}$. The process involves pairwise correlation between points in $\mathcal{D}$ via some kernel $k(\vec{x}, \vec{x'})$ and its mean $m(\vec{x})$, intending that samples close to one another may correlate. Both $m(\vec{x})$ and $k(\vec{x}, \vec{x'})$ make up the GP.

Lastly, given all this information, an Acquisition Function based on $\mu(\vec{x})$ and $\sigma(\vec{x})$ is defined. This function measures the value that would be generated by evaluation of the objective function $f(\vec{x})$ at a new point $\vec{x}$, based on the posterior $\mu(\vec{x})$ and $\sigma(\vec{x})$ that in turn is based on $\mathcal{D}$. Maximizing this function α yields $\vec{x}_t^*$, the most promising point given $\mu(\vec{x})$ and $\sigma(\vec{x})$. This sample is the next sample to be evaluated and added to $\mathcal{D}$. Picking such samples requires the balance of exploration and exploitation: sampled points should regard sampling unknown areas for better coverage and explore possible better optima, as well as sampling known areas to optimize and exploit knowledge gained to find some optima.

The workflow of Bayesian Optimization is described in algorithm 1. To avoid a cold-start problem in this workflow, one might include

Algorithm 1 Bayesian Optimization workflow

```
Assume a Bayesian prior on $f(\vec{x})$
Build the posterior distribution based on $\mathcal{D}$ and kernel $k(\vec{x}, \vec{x'})$
while budget not exhausted do
    Find $\vec{x}_t^*$ that maximizes acquisition function via $\vec{x}, \mu(\vec{x})$ and $\sigma(\vec{x})$.
    Find $f(\vec{x}_t^*)$ and add $(\vec{x}_t^* : f(\vec{x}_t^*))$ to $\mathcal{D}$
    Update the posterior distribution on $f(\vec{x})$
end while
```

a Design of Experiments (DoE). These are a set of samples, gathered via deterministic or stochastic structural methods, where another does not yet influence the samples via some acquisition function. As such, these are the initial samples, where the amount of samples is n_{init}. High-dimensional Bayesian Optimization (HDBO) deals with, as the name recalls, higher-dimensional problems. BO effectively deals with dimensions d where $d < 20$ [1], however, in tasks with higher dimensionality, it seems that regular BO methods tend to lack performance. Multiple classes of HDBO algorithms are found, each with its pros and cons [2].

In this paper, we will focus on an additive model method named Random Decomposition Upper Confidence Bound (RDUCB) [11]. Additive models address the challenge of HDBO by assuming that the black-box function $f(\vec{x})$ can be approximated by a sum of lower-dimensional functions, thereby reducing the complexity of the function. So, given some $f(\vec{x})$, additive models assume that an approximation of $f(\vec{x})$ may be $f(\vec{x}) = \sum_{i=1}^{m} f_i(\vec{x_{[i]}})$. Here, m is the number of additive components where $m \ll d$. Methods such as Tree [3] have shown that tree-structured (cycle-free pair-wise dimensional interactions) could represent black-box functions [3, 11]. This algorithm created tree-decompositions based on optimization via $\mathcal{D}$. RDUCB however continued to rely on data-independent decomposition rules.

3 METHODOLOGY

We will first dive deeper into RDUCB [11], its definition, and its workflow. We will then move on to adjustments to RDUCB, namely the acquisition functions we used. Lastly, we will go over our experimental setup, and give the details for reproduction.

3.1 RDUCB

RDUCB has properties similar to those of an additive method, as mentioned. First up, is the random tree decomposition. A random tree decomposition considers the interaction between two dimensions via components c. These components are established via algorithm 2. This algorithm ensures that the components are unique, acyclic (claimed to be effective by [3]), and dimensions may be regarded or disregarded based on the iteration. The decomposition changes in each iteration, ensuring effective coverage of interaction between dimensions. A decomposition g is thus a set of components c, such that g is approximated via $f(\vec{x}) = \sum_{c \in g} f_c(\vec{x_{[c]}})$, where $\vec{x_{[c]}}$ selects only the dimensions of $\vec{x}$ that appear in c.

Algorithm 2 Random tree decomposition sampling [6]

```
Require: $E$: number of edges in the decomposition with $0 < E < d$
  UF ← UnionFind data structure
  $e \leftarrow 0$
  while $e \leq E$ do
      $n_{in} \sim U(\{i | i \in \mathbb{N} \wedge 1 \leq i \leq d\})$
      $n_{out} \sim U(\{i | i \in \mathbb{N} \wedge 1 \leq i \leq d\})$
      if $n_{in}$ and $n_{out}$ not connected in UF then:
          Connect $n_{in}$ and $n_{out}$ in the tree
          Update UF accordingly
          $e \leftarrow e + 1$
      end if
  end while
```

Second, the kernel used to establish a covariance matrix between pairwise points is a squared exponential covariance kernel

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

They follow [9] that the mean function $m(\vec{x})$ in the GP is $m(\vec{x}) \equiv 0$. Furthermore, θ is a set of hyperparameters tuned by maximizing the data marginal [11]. Given the establishment of components c in decomposition g to approximate the black-box function $f(\vec{x})$, the additive kernel is now defined as

$$k^g(\vec{x}, \vec{x'}) = \sum_{c \in g} k_c(x_{[c]}, x'_{[c]})$$

Lastly, and most importantly for this paper, is the acquisition function. This function needs to have an additive structure since we are dealing with a decomposed function. From this decomposition, we have to build up knowledge gained from each $f_c(\vec{x_{[c]}})$ to find some promising $\vec{x}_t^*$ that may optimize our black-box function as a whole. The original paper insists on using message passing [10] to efficiently make use of the structural overlap of the components $c \in g$. The paper makes use of UCB as an additive acquisition function:

$$\alpha_t^{(add-UCB)}(\vec{x}|\mathcal{D}) = \sum_{c \in g} \alpha_{t,c}(\vec{x}|\mathcal{D}) = \sum_{c \in g} \mu_{t,c}(\vec{x}) + \beta_t \sigma_{t,c}(\vec{x})$$

where μ, σ are the posterior distribution on $\mathcal{D}$, and β an exploration hyperparameter.

The full workflow of RDUCB can be found in algorithm 3. The DoE used to populate $\mathcal{D}$ initially, are $n_{init} = 10$ stochastically Monte Carlo-sampled samples, regardless of the dimensionality of the problem.

Algorithm 3 RDUCB [11]

```
  Populate $\mathcal{D}$ with $n_{init}$ samples
  for $t = n_{init} + 1$ to $N$ do
      Sample tree decomposition $g$
      Fit a GP using $\mathcal{D}$ and kernel $k_g$
      Maximize $\alpha_t^{(add-UCB)}(\vec{x}|\mathcal{D})$ with message passing
      Find $f(\vec{x}_t^*)$ and add $(\vec{x}_t^* : f(\vec{x}_t^*))$ to $\mathcal{D}$
  end for
```

3.2 Acquisition functions

We will be testing several acquisition functions on RDUCB. All these acquisition functions have different ways of balancing exploration and exploitation and may be more complex than one another.

First of all, RDUCB already uses UCB: Upper Confidence Bound. This is one of the simplest acquisition functions and a way of balancing exploration and exploitation. A hyperparameter β controls this balance. Given the posterior defined by $\mu(\vec{x})$ and $\sigma(\vec{x})$, and decomposition g we can define the additive form of UCB as:

$$\alpha_t^{(add-UCB)}(\vec{x}|\mu_t, \sigma_t) = \sum_{c \in g} \mu_{t,c}(\vec{x}) + \beta_t \sigma_{t,c}(\vec{x})$$

Second, Probability of Improvement (PI) aims to find $\vec{x}$ that will lead to an improvement over the best observed y^*. Using random variable $\hat{y}(\vec{x})$, we can compute the probability of improvement upon y^* via $I(\vec{x}) = y^* - \hat{y}(\vec{x})$ as $P[I(\vec{x})] = \Phi(\frac{y^* - \mu(\vec{x})}{\sigma(\vec{x})})$, such that our additive PI function is defined as:

$$\alpha_t^{(add-PI)}(\vec{x}|\mu_t, \sigma_t) = \sum_{c \in g} \Phi\left(\frac{y_c^* - \mu_{t,c}(\vec{x})}{\sigma_{t,c}(\vec{x})}\right)$$

where Φ is the standard normal cumulative density function (CDF).

Third, is the Expected Improvement (EI) acquisition function. Similarly to PI, the acquisition function aims to find a point that maximizes the expected improvement (unlike the probability of improving). Improvement is now measured as a value, instead of a probability. Our additive EI function is thus defined as:

$$\alpha_t^{(add-EI)}(\vec{x}|\mu_t, \sigma_t) = \sum_{c\in g}$$

$$\begin{cases} (y_c^* - \hat{y}(\vec{x}))\Phi\left(\frac{y_c^* - \mu_{t,c}(\vec{x})}{\sigma_{t,c}(\vec{x})}\right) + \sigma_{t,c}(\vec{x})\phi\left(\frac{y_c^* - \mu_{t,c}(\vec{x})}{\sigma_{t,c}(\vec{x})}\right), \text{if } \sigma_{t,c}(\vec{x}) > 0 \\ 0, \text{if } \sigma_{t,c} = 0 \end{cases}$$

where Φ and ϕ are the CDF and probability density function (PDF) respectively.

3.3 Experimental setup

We will be using several BBOB functions from the COCO environment [4]. The functions are grouped into several categories. Separable functions are simple and require no interaction between variables. Low/moderate conditioned functions require interaction between variables but are well-behaved. High-conditioned functions have strong interactions between variables and are usually less well-behaved. Multi-model functions with an adequate global structure have many local optima but are well-behaved. Multi-model functions with weak global structures have no clear structure or patterns. From each of these groups, we will pick a function, in this case (following BBOB notation), F1; F8; F12; F15; and F21 respectively.

We will be running instances 0; 1; and 2 on each of those functions. Each instance may shift the optimal solution in $\mathcal{X}$-space or f-space, by simple shift operations or rotation operations. Each instance makes sure the shifts are randomly chosen anew [5]. Each experiment is repeated 5 times and the results are averaged. Dimensions d are picked such that $d \in \{2, 10, 40, 100\}$. The total budget is $10 \cdot d + 50$. The search space is bounded such that for $\vec{x} \in \mathcal{X}$, $\forall x_i \in \vec{x} : -5 \leq x_i \leq 5$.

The weight of the exploration parameter β is scaled as $\beta_t = 0.5 * \log(2t)$, indicating growth from exploitation to exploration over time t. This perhaps counterintuitive process is performed to fully map the most promising area and then explore within this region to conserve budget. In this trend, the jitter parameter of each acquisition function is set to 0.01 to slightly increase the explorative nature to balance the exploration parameter.

We stick to the DoE originally used in RDUCB [11]. These are $n_{init} = 10$ stochastically Monte Carlo-sampled samples, independent of the dimensionality of the task.

We will be using the GitHub repository for comparing HDBO algorithms on BBOB, where RDUCB is already a part of [8]. Our methods will be included in a copy of this repository [7][1].

4 RESULTS

The results of each of the experiments are subdivided into sections, with each section tackling a different BBOB function. Each section will focus on the four aforementioned dimensionalities as well as each instance of the problem in each of the dimensions, totalling 12 results for each BBOB function. Each plot includes the averaged results of each of the acquisition function configurations over 5 repetitions. We name the algorithms according to their acquisition function, such that we are experimenting with RDUCB, RDPI, and RDEI. The BBOB functions we are experimenting with and its results are:

- F1 - Sphere function; figure 1
- F8 - Rosenbrock function; figure 2
- F12 - Bent Cigar function; figure 3
- F15 - Schaffer's F7 Function, moderately ill-conditioned function; figure 4
- F21 - Gallagher's Gaussian 101-me Peaks function; figure 5

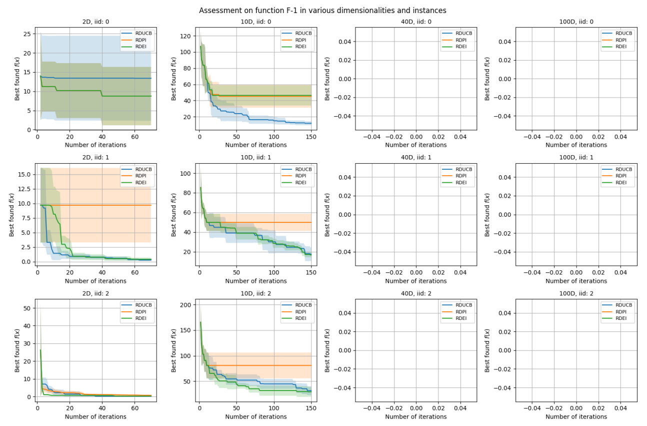

Figure 1: Results of BBOB function F-1, applied in multiple instances and multiple dimensions. Results are averaged (not smoothed) over 5 repetitions.

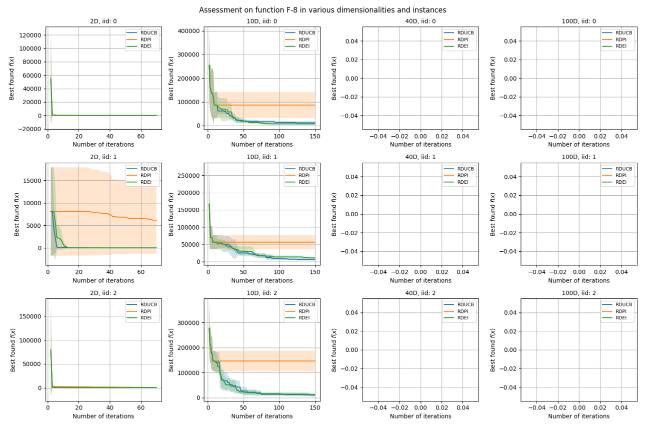

Figure 2: Results of BBOB function F-8, applied in multiple instances and multiple dimensions. Results are averaged (not smoothed) over 5 repetitions.

We observe that Upper Confidence Bound outperforms Probability of Improvement in lower dimensions (2D and 10D). Notably, RDPI appears to flatline for most optimization problems that were tested. The PI activation function works by sampling regions most

[1]The GitHub is public and can be found here: https://github.com/Lexpj/BO-FA

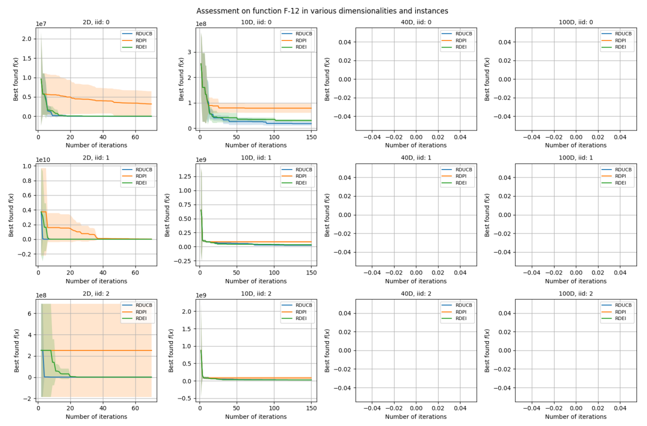

Figure 3: Results of BBOB function F-12, applied in multiple instances and multiple dimensions. Results are averaged (not smoothed) over 5 repetitions.

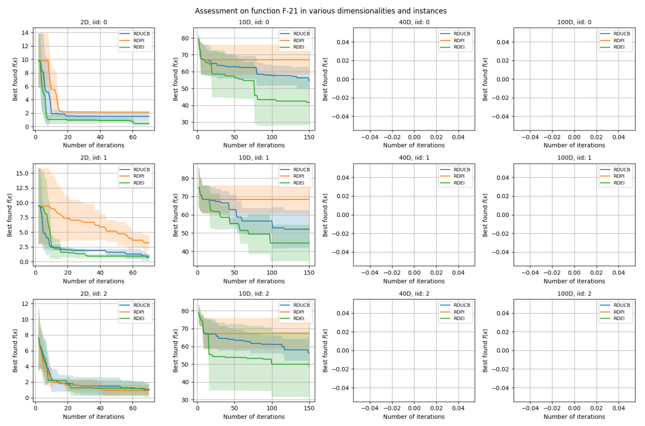

Figure 5: Results of BBOB function F-21, applied in multiple instances and multiple dimensions. Results are averaged (not smoothed) over 5 repetitions.

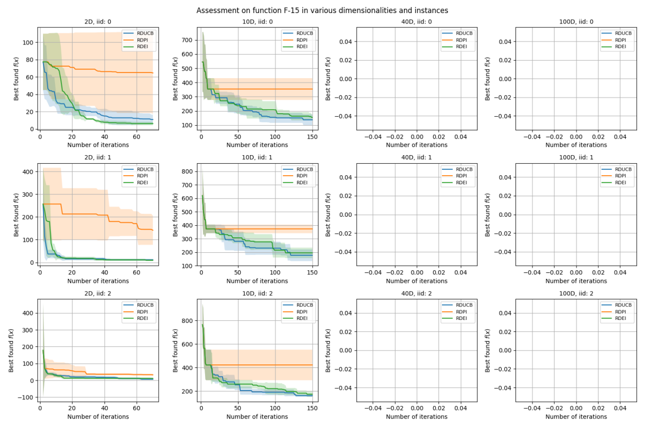

Figure 4: Results of BBOB function F-15, applied in multiple instances and multiple dimensions. Results are averaged (not smoothed) over 5 repetitions.

likely to improve over the best-found value so far by rewarding improvements with a constant value. Hence, PI does not take the extent of the improvement into account, meaning more promising regions may exist that are all treated equally by the acquisition function. As a result, the optimizer may become stuck in a local optimum which corresponds to the stagnation observed in the figures in section 4 [1]. This same phenomenon therefore also explains why Upper Confidence Bound outperforms Probability of Improvement. The fact that RDEI occasionally performs on-par with RDUCB for some iterations (figure 1 top left) can thus be explained by a "lucky" iteration where the PI does not get stuck in a more local optimum than UCB. Furthermore, RDUCB operates on the principle of randomized tree decompositions. The additive principle of this algorithm may cause problems when integrated with PI, considering that this acquisition function has increased complexity but with the assumption of constant reward. Hence, this incorrect assumption may lead to increased underperformance considering the context of the additive assumption of decomposition-based BO. In this regard, Thompson Sampling may perform well due to its robustness and hence this acquisition function is taken to be an interesting direction for future research. Another noteworthy result can be seen in figure 1, instance 0 in a 2-dimensional setting. We can see no setting performs well. Given the task, a simple, unshifted, unrotated sphere function, there is no interaction involved between the variables. These algorithms, in a 2-dimensional setting, always expect an interaction between the variables. A workaround has been done in the RDUCB paper (see [11]), but is not applied here. This wrong assumption of the task can be a conclusion of poor performance.

RDEI does take the flaw of constant rewards into account and includes rewards dependent on the level of improvement (hence, the name "Expected Improvement"). This is reflected in the results, where we observe that RDEI consistently outperforms RDPI. The exception to these observations is the first instance of the 10-dimensional F-1 function (figure 1, top right). Surprising, however, is the fact that Expected Improvement yields similar results to Upper Confidence Bound, even outperforming it for certain problem instances. The most striking example here is the F-21 problem, the most complex problem within the performed test with its multi-modality and weak global structure. Here, we observe a trend where RDEI appears to increasingly outperform RDUCB as the dimensionality increases, suggesting its usefulness for HDBO. A potential explanation could be the previously mentioned complex consideration that EI is able to make compared UCB. Due to the simplicity of UCB, the acquisition function may not be able to approximate the most promising points to sample considering the assumed additive nature of the interacting dimensions and the weak structure of the problem at hand. The F-15 problem is also multi-modal and does not exhibit the outperformance of RDEI (figure 4), suggesting that indeed a weakly-structured search space is where EI would be most useful compared to UCB.

5 DISCUSSION

In this paper, we have experimented with different acquisition functions for a random decomposable BO algorithm. We have seen that, although yet not all configurations have been run, picking the right acquisition function should be considered. The RDUCB paper mentions the use of UCB as an acquisition function due to its additive nature [11]. However, we have shown that other to-be-made additive acquisition functions would work well in place, and could outperform settings where UCB has been used.

Due to runtime and hardware constraints, we were unfortunately unable to test the effects of the various acquisition functions in higher dimensions. For instance, to run the experimental setup for a single algorithm variant in 40 dimensions a total of 18 hours of runtime was necessary. Sadly, hardware constraints did not realistically allow for problems of such a dimensionality or higher to be tackled, reducing the testing phase to non-HDBO results. For the sake of completeness, the setup and code to perform these experiments have been included in the project repository found in [7], in case the reader would like to run these. Instructions are included in the README.

Our original proposal, seen in appendix A, was more elaborate than we showed in this paper. However, the proposal is still relevant and will be considered for future work.

REFERENCES

[1] R. Garnett. *Bayesian Optimization.* 1 2023.

[2] M. González-Duque, R. Michael, S. Bartels, Y. Zainchkovskyy, S. Hauberg, and W. Boomsma. A survey and benchmark of high-dimensional bayesian optimization of discrete sequences, 2024.

[3] E. Han, I. Arora, and J. Scarlett. High-dimensional bayesian optimization via tree-structured additive models, 2020.

[4] N. Hansen, A. Auger, R. Ros, S. Finck, and P. Pošík. Comparing results of 31 algorithms from the black-box optimization benchmarking bbob-2009. In *Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '10, page 1689–1696, New York, NY, USA, 2010. Association for Computing Machinery.

[5] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Noisy functions definitions. 01 2009.

[6] Huawei-Noah. HEBO/RDUCB at master · huawei-noah/HEBO.

[7] Lexpj. Github - lexpj/bo-fa: Bo final assignment.

[8] MariaLauraSantoni. GitHub - MariaLauraSantoni/IOH-Profiler-HDBO-Comparison: Package to compare high-dimensional Bayesian Optimization (BO) algorithms by using IOH Profiler. Implementation for the TELO paper experiments.

[9] C. Rasmussen, O. Bousquet, U. Luxburg, and G. Rätsch. Gaussian processes in machine learning. *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures, 63-71 (2004)*, 3176, 09 2004.

[10] P. Rolland, J. Scarlett, I. Bogunovic, and V. Cevher. High-dimensional bayesian optimization via additive models with overlapping groups, 2018.

[11] J. Ziomek and H. Bou-Ammar. Are random decompositions all we need in high dimensional bayesian optimisation?, 2023.

A ORIGINAL PROPOSAL

RDUCB investigates the use of UCB as an acquisition function. We propose to swap this out for Thompson sampling and other additive acquisition(s), as the original paper does not investigate other possible acquisition functions and Thompson sampling is in this case compatible with the additive nature of the kernels (considering decomposition). The paper applied RDUCB to HEBO and still kept the multi-acquisition function, showing an improvement in performance. This is promising for the possibility of other acquisition functions potentially outperforming UCB.

Furthermore, TurBO makes use of trust regions, so we plan to insert RDUCB in the form of the random decompositions in TurBO at each iteration, creating a "double" dimensionality reduction as it were. We aim to do this by inserting the randomised decomposition model into TurBO at the sampling (and choosing) stage of the various trust regions. The paper itself indicates this as an area of future research and perhaps allows some insight into some of the "vague" or less explainable areas of RDUCB (e.g. its underperformance in lower-dimensional spaces (which may become relevant in problems of unknown dimensionality) and the potential presence of disconnected components within the tree decomposition).

As such and since TurBO and RDUCB are orthogonal to one another, we propose to consider a "combination" or "swapping out" of various components within each method to further enhance their performance in HDBO, in an ablation-style of experimenting.

Finally, we hope to find out what the strange jumps are that we observe within the RDUCB results as we investigate swapping out the various components of RDUCB and TurBO.