Comparison of Bayesian Optimization Algorithms for BBOB Problems in Dimensions 10 and 60

Maria Laura Santoni, Elena Raponi, Renato de Leone, Carola Doerr

Comparison of Bayesian Optimization Algorithms for BBOB Problems in Dimensions 10 and 60

Maria Laura Santoni
Sorbonne Université, CNRS, LIP6
Paris, France

Elena Raponi
TUM School of Engineering and Design, TU Munich
Munich, Germany

Renato De Leone
School of Science and Technology, University of Camerino
Camerino, Italy

Carola Doerr
Sorbonne Université, CNRS, LIP6
Paris, France

ABSTRACT

Bayesian Optimization (BO) is a class of black-box, surrogate-based heuristics that can efficiently optimize problems that are expensive to evaluate and therefore allow only small evaluation budgets. Regardless of the size of the budget, high dimensionality also poses a challenge to BO, whose performance reportedly often suffers when the dimension exceeds 15 variables. Many new algorithms have been proposed to address this problem. However, it is not well understood which one is the best for which optimization scenario.

In this work, we compare five state-of-the-art high-dimensional BO algorithms, with vanilla BO and CMA-ES on the 24 BBOB functions of the COCO environment at two dimensionalities, 10 and 60 variables. Our results confirm the superiority of BO over CMA-ES for limited evaluation budgets and suggest that the most promising approach to improve BO at high dimensionality is the use of trust regions. However, we also observe significant performance differences for different function landscapes and budget exploitation phases, indicating improvement potential, e.g., through hybridization of algorithmic components.

KEYWORDS

Black-box optimization, Bayesian Optimization, High-dimensional Bayesian Optimization, Benchmarking

1 INTRODUCTION

One of the most commonly used surrogate-based optimization methods is Bayesian Optimization (BO) [6]. Through careful modeling based on Gaussian process regression (GPR) and intelligent search for candidate solutions through the optimization of an acquisition function, BO algorithms can deliver impressive optimization performance even with small evaluation budgets. However, when the dimensionality of the problem exceeds 15 variables, the performance of BO deteriorates due to the so-called *curse of dimensionality*. Scaling BO to higher-dimensional spaces is challenging due to its high statistical and computational complexity: the number of points queried to satisfactorily cover the search space increases exponentially with dimension, and optimizing the acquisition function requires more and more computational power, being a non-convex optimization problem on the same design space itself.

Related work: In recent years, characterized by increasingly complex systems and large and high-dimensional data, great efforts have been made to extend BO to higher dimensions, and various strategies have been proposed. According to [2], these are mainly based on (but not limited to) one of the following methods: Variable selection, additive models, linear and nonlinear embeddings, and trust regions. All of these strategies have advantages and disadvantages [2], and it is not clear which one is best for which optimization scenario. In Sec. 2 we discuss these categories further and give some examples. However, for a more comprehensive overview, we refer the reader to available reviews [2, 10], even though they are purely informative and lack an experimental study comparing the methods. Comparative studies have also been conducted to present new algorithms such as SMAC [8], TuRBO [5], SAASBO [4], etc., but they are either outdated or limited to showing the potential of the proposed algorithm rather than presenting a comprehensive and unbiased performance comparison. Our work aims to fill this gap.

Disclaimer: In line with [2] we refer to the setting with dimension 60 as *high-dimensional*, even if BO-approaches for problems with several thousands of variables have been studied [14].

Our contribution: In this article, we present the results of a benchmarking study that follows the standardized, well-established guidelines for unbiased performance comparison in numerical black-box optimization. With the goal to obtain a first overview over which high-dimensional BO (HDBO) approaches to favor for which problem characteristics, we benchmark five BO variants that are specifically designed for high-dimensional, low-budget optimization problems with two standard solvers, vanilla BO and the Covariance Matrix Adaptation Evolution Strategy (CMA-ES).

To obtain interpretable results, we focus on the 24 functions of the Black-Box Optimization Benchmarking (BBOB) suite from the COCO benchmarking environment [7] and compare algorithm performance for small evaluation budgets. Key findings from our experiments are that (1) Vanilla BO performs better than CMA-ES for small dimensions and low budget, (2) many of the algorithms that aim to scale BO to higher-dimensional spaces outperform vanilla BO and CMA-ES, and (3) among the compared algorithms, the BO variant using trust regions performs particularly well on a large number of function, dimension, and budget combinations.

Reproducibility: Our code for reproducing the experiments is available on GitHub, in the public repository `IOH-HDBO-Comparison`[1]. The project data is also available for interactive analysis and visualization on the IOHanalyzer platform [12] as 'HDBO' dataset.

[1] https://anonymous.4open.science/r/IOH-Profiler-HDBO-Comparison-ECF8

2 HIGH-DIMENSIONAL BAYESIAN OPTIMIZATION

BO is a sample-efficient framework for the global optimization of expensive-to-evaluate black-box functions. It involves two main components: a method for statistical inference, usually a GPR model, and an acquisition function to sample new points. Both components become more complex as the dimensionality of the problem increases. Here, we test and compare algorithms representing the main categories of algorithms for solving the problem of high dimensionality in BO: Variable selection, additive models, linear embeddings, nonlinear embeddings, and trust regions [2]. Each category has advantages and disadvantages, and the performance is often influenced by the available evaluation budget and problem structure. For example, the first two categories assume an underlying structure on the objective function, e.g., intrinsic lower dimensionality or additive structures, i.e., a decomposition of the objective function into a sum of lower-dimensional components. In the selection of the algorithms, we prefer the most recent ones, namely those that provide a Python implementation. For the variable selection approach, we focus on **Sparse Axis Aligned Subspace Bayesian Optimization (SAASBO)** [4]. Among the algorithms that use additive models, we analyze **Ensemble Bayesian Optimization (EBO)** [13]. Among the linear and nonlinear embeddings approaches, we focus on **PCA-assisted Bayesian Optimization (PCA-BO)** [11] and **Kernel PCA-assisted Bayesian Optimization (KPCA-BO)** [1], respectively. For trust-region-based approaches, we consider **Trust Region Bayesian Optimization (TuRBO)** [5].

3 EXPERIMENTAL SETUP

We compare algorithms on the 24 functions of the BBOB suite using their definition from IOHprofiler [3]. The BBOB functions on which we base our evaluation are divided into five groups: separable functions (f1-f5), functions with low or moderate conditioning (f6-f9), functions with high conditioning and unimodal structure (f10-f14), multimodal functions with appropriate global structure (f15-f19), and multimodal functions with weak global structure (f20-f24). The problems belonging to the last two groups are most representative of real-world problems, as they have a more complex landscape characterized by high nonlinearity and roughness, with multiple peaks or valleys. For each function, we consider dimensions 10 and 60. For dimension 60, we did not run complete experiments for some algorithms due to time and memory constraints. Thus, we performed 10 independent runs of each algorithm on the first 3 instances (instance ID 0-2) of the 24 BBOB functions, with the following exceptions for D = 60: (1) We did not run experiments at all with SAASBO, because the algorithm requires about 1 week to complete a single run and a large amount of memory; (2) for EBO, we only have results for functions f15-f24, because the algorithm requires about 2 weeks for a run.

We compare the HDBO algorithms with a vanilla BO and a default CMA-ES. The general settings of the experiments follow. For each run, the total evaluation budget is set to $10 \times D + 50$ function evaluations. For BO-based algorithms, the initial DoE size is set to D. By default for the BBOB suite, the domain is set to $[-5, 5]^D$. For each algorithm introduced in Sec. 2, we use default settings for their hyperparameters. The implementation of vanilla BO is taken from the Python module `scikit-learn`[2], choosing Expected Improvement (EI) as the acquisition function and a noise level equal to 0.01. The implementation for CMA-ES is the one in the pycma package, available from the GitHub repository `pycma`[3]. The code for SAASBO is taken from the GitHub repository `saasbo`[4]. The EBO code is taken from the GitHub repository `Ensemble-Bayesian-Optimization`[5], but we redefine the acquisition function as the EI, because the default implementation uses a global minimum value that is assumed to be known, while we assume that it works in a complete black-box scenario. We compare two different implementations of EBO: EBO and EBO_B. They differ for the value of the hyperparameter B that represents the number of query points selected at each iteration. We use $B = 1$ and $B = 10$, respectively. To use the same total budget, EBO_B runs for budget/10 iterations. The code for PCA-BO and KPCABO is taken from the GitHub repository `Bayesian-Optimization`[6]. The TuRBO code is taken from the GitHub repository `TuRBO`[7]. In our experiments, two different implementations of TuRBO are compared: TuRBO1 and TuRBOm. They differ in the number of trust regions used by the algorithm: $tr = 1$ and $tr = \lfloor D/5 \rfloor$, respectively, where tr denotes the number of trust regions and $\lfloor \cdot \rfloor$ denotes the floor function. The code to run the experiments is a modular framework compatible with IOHprofiler. It is available on GitHub in the repository `IOH-HDBO-Comparison`. We used Python post-processing libraries to present our results.

4 RESULTS

Dimension D = 10. Fig. 1 compares the convergence behavior of all algorithms at dimension 10, on all BBOB functions from f1 to f24. For a small evaluation budget, vanilla BO always performs better than CMA-ES, as we can see in particular on f2-f4, f12, and f19. However, at the end of the budget, CMA-ES outperforms or is comparable to vanilla BO in many cases. Overall, we see a good performance of vanilla BO, which is due to the still low dimensionality. BO is always among the best or at least comparable to the other algorithms, with a few exceptions, reaching a particularly good performance on f5, f6, f22, and f24. In Fig. 1, algorithm performance is not stable across functions. However, SAASBO and TuRBO tend to predominate. Specifically, TuRBO finds excellent loss values on f2, f3, f13, f14, f16, f18, f20, and f22 (here together with SAASBO). Stagnation at a very low budget is a common behavior of linear PCA-BO, KPCA-BO, and both EBO and EBO_B (f1, f5, f6, and f20-f22). Finally, if we focus on f5 (linear slope), we can observe the extremely good performance of BO and SAASBO, which are able to immediately find the global optimum due to the unimodal and monotonic landscape of this function (this observation also holds for D = 60).

After computing the CPU time in seconds to complete the entire run, we can also claim that SAASBO is the most expensive strategy in terms of total CPU time with an average of 5284.51 seconds. The two versions of TuRBO are the fastest, TuRBO1 has an average of 9.80 seconds and TuRBOm of 35.49 seconds. They are followed by EBO_B (843.03 s), linear PCA-BO (115.59 s), and vanilla BO (168.01

[2] https://scikit-optimize.github.io/stable/auto_examples/bayesian-optimization.html
[3] https://github.com/CMA-ES/pycma
[4] https://github.com/martinjankowiak/saasbo
[5] https://github.com/zi-w/Ensemble-Bayesian-Optimization
[6] https://github.com/wangronin/Bayesian-Optimization
[7] https://github.com/uber-research/TuRBO

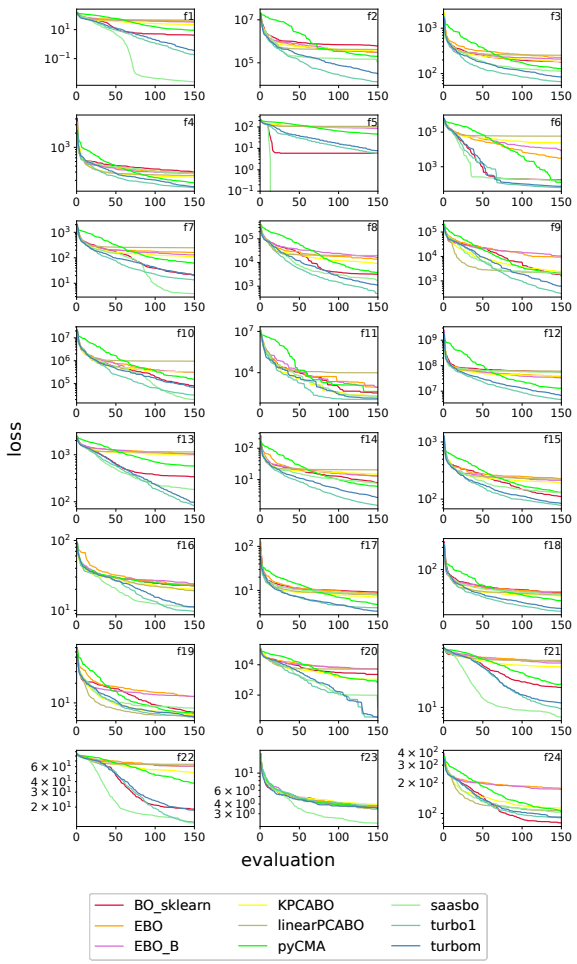

Figure 1: The best-so-far target gap for dimension 10.

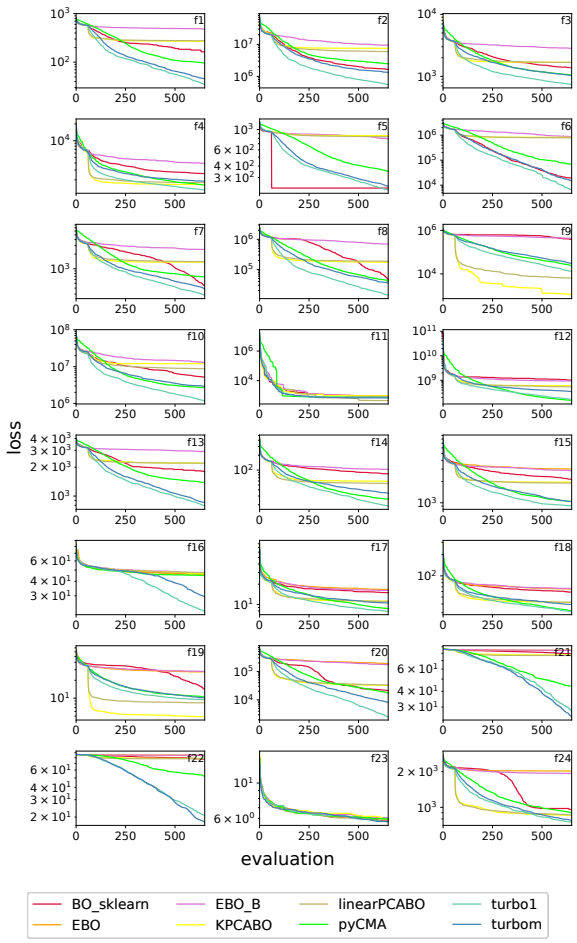

Figure 2: Best-so-far target gap for dimension 60.

s). From the analyses of convergence and CPU time at 10D, TuRBO1 and TuRBOm seem to perform best.

Dimension D = 60. Fig. 2 compares the convergence behavior of the algorithms at dimension 60 for all BBOB functions from 1 to 24. Due to computational constraints, some algorithms are missing, as explained in Sec. 3. The figure demonstrates how BO suffers from a lower convergence rate at high dimensionality. In all cases except for f5, here better convergence capabilities of CMA-ES are evident. BO suffers from premature stagnation, which is due to its higher computational complexity at dimension 60. We can observe the same behavior for some HDBO methods, such as EBO and, in some cases, linear and kernel PCA-BO. The performance of BO decreases even on f24, where it was the best solver at D = 10. Both versions of TuRBO show very good performance for f1, f13, f21, and f22, and there is a statistically significant difference between them and the other algorithms, which is confirmed by a Wilcoxon signed-rank test. Moreover, TuRBO is the only algorithm that continues to improve as the number of evaluations increases, while the other algorithms stagnate easily. Finally, we note that the difference in performance between TuRBO1 and TuRBOm becomes clearer (f2, f3, f8, f10, f16, f20), which suggests the use of just one trust region for high-dimensional problems. Nonetheless, we can observe an interesting performance of PCA-BO and KPCA-BO on f9, f11, f18-f20, and f24, where they either rank first or show initial speedups that make them good candidates for high-dimension/low-budget optimization problems. EBO and EBO_B perform poorly. We attribute this to the choice of their hyperparameters as the default ones, which might not be ideal for the function landscapes addressed in this study. As for the CPU time to complete the entire run, EBO (732050.72 s) and KPCA-BO (145660.00 s) are the most expensive, while TuRBO1 (874.75 s) is significantly faster than the other algorithms, followed by TuRBOm (5297.99 s).

Further Discussion. For an in-depth comparison, we also present in Fig. 3 the convergence evolution of the algorithms compared on f24 at dimension 60, by freezing it at two different budgets: 200 and 600 evaluations. The figure gives an idea of the ranking of the algorithms in different phases of the optimization runs and clearly shows in which context one of the algorithms is preferable to the others. For a limited budget, Fig. 3 (left) shows that linear PCA-BO and KPCA-BO find the lowest loss values. We attribute this to their better ability to find good solutions in a lower-dimensional

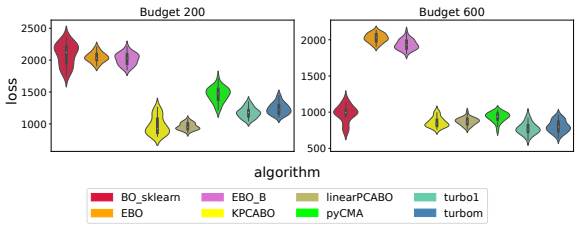

Figure 3: Violin plots showing loss values at two different budgets (200 and 600 function evaluations) for function 24.

manifold of the original search space. However, this often drives the search towards local optima. At the end of the run, we observe in Fig. 3 (right) that linear PCA-BO and KPCA-BO are outperformed by TuRBO1 and TuRBOm, which better balance between exploration and exploitation of the domain by using dynamic trust regions and multiple restarts. Based on the Wilcoxon signed-rank test, these results are statistically significant.

Therefore, based on the analysis, we believe that it would be interesting to explore the possibility of merging the concepts behind PCA-BO and TuRBO, by building local low-dimensional embeddings. In this way, one can benefit from both the flexibility of the trust regions and the lower complexity of the manifolds with reduced dimension. We leave this to our future research.

5 CONCLUSION AND FUTURE PERSPECTIVES

We conducted an experimental study comparing the performance of vanilla BO, CMA-ES, and five BO-based (HDBO) algorithms on the 24 noiseless BBOB functions in dimensions 10 and 60. Our results confirm good performance of BO in the 10D case. But this performance deteriorates as the dimensionality of the problem increases to 60D. Here, CMA-ES performs better, especially for larger budgets. However, the average observed performance of CMA-ES is worse than that of the HDBO algorithms. Although we observe different performances for different function landscapes and budget utilization phases, TuRBO seems to be the most promising algorithm, both in terms of convergence trend and CPU time. However, linear PCA-BO and KPCA-BO also show potential for small evaluation budgets, with fast convergence towards a near-optimal solution.

Further work is planned to develop a hybrid algorithm combining linear PCA-BO and TuRBO. This algorithm could avoid the stagnation of linear PCA-BO by using restarts and trust regions, while still benefiting from a linear, low-dimensional embedding. This could lead to a very competitive algorithm for optimizing expensive black-box functions at high dimensionality. In addition, we aim to create a modular framework that allows for choosing a convenient strategy for the different optimization phases. The choice of modules would concern the model fitting, the acquisition function (AF), the optimizer to search the AF, the use of trust regions, etc. We also found that the poor performance of some algorithms, especially EBO, could be due to poor initialization of their hyperparameters. Therefore, we plan to investigate how the investigated algorithms could benefit from a dedicated hyperparameter optimization [9]. We expect that these results could significantly change the comparison of HDBO algorithms, since some of them were not specifically designed for general optimization scenarios, but rather for applications in machine learning.

Acknowledgments. The work leading to this publication was supported by the Sorbonne Center for Artificial Intelligence (SCAI) (IDEX SUPER 11-IDEX-0004), by ANR project ANR-22-ERCS-0003-01, by CNRS INS2I project *IOHprofiler*, and by the PRIME programme of the German Academic Exchange Service (DAAD) with funds from the German Federal Ministry of Education and Research (BMBF).

REFERENCES

[1] Kirill Antonov, Elena Raponi, Hao Wang, and Carola Doerr. 2022. High Dimensional Bayesian Optimization with Kernel Principal Component Analysis. In *Proc. PPSN*, Vol. 13398. Springer, 118–131. https://doi.org/10.1007/978-3-031-14714-2_9

[2] Mickaël Binois and Nathan Wycoff. 2022. A Survey on High-dimensional Gaussian Process Modeling with Application to Bayesian Optimization. *ACM Trans. Evol. Learn. Optim.* 2, 2 (2022), 8:1–8:26. https://doi.org/10.1145/3545611

[3] Carola Doerr, Hao Wang, Furong Ye, Sander van Rijn, and Thomas Bäck. 2018. IOHprofiler: A benchmarking and profiling tool for iterative optimization heuristics. *arXiv preprint arXiv:1810.05281* (2018). https://iohprofiler.github.io/.

[4] David Eriksson and Martin Jankowiak. 2021. High-dimensional Bayesian optimization with sparse axis-aligned subspaces. In *Uncertainty in Artificial Intelligence*. PMLR, 493–503.

[5] David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. 2019. Scalable global optimization via local Bayesian Optimization. *Advances in Neural Information Processing Systems* 32 (2019).

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

[7] N. Hansen, A. Auger, R. Ros, O. Mersmann, T. Tušar, and D. Brockhoff. 2021. COCO: A Platform for Comparing Continuous Optimizers in a Black-Box Setting. *Optimization Methods and Software* 36 (2021), 114–144. Issue 1. https://doi.org/10.1080/10556788.2020.1808977

[8] Frank Hutter, Holger Hoos, and Kevin Leyton-Brown. 2013. An Evaluation of Sequential Model-Based Optimization for Expensive Blackbox Functions. In *Proc. GECCO (Companion)*. ACM, 1209–1216. https://doi.org/10.1145/2464576.2501592

[9] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Thomas Stützle, and Mauro Birattari. 2011. *The irace package, Iterated Race for Automatic Algorithm Configuration*. Technical Report TR/IRIDIA/2011-004. IRIDIA, Université Libre de Bruxelles, Belgium. http://iridia.ulb.ac.be/IridiaTrSeries/IridiaTr2011-004.pdf

[10] Mohit Malu, Gautam Dasarathy, and Andreas Spanias. 2021. Bayesian Optimization in High-Dimensional Spaces: A Brief Survey. In *International Conference on Information, Intelligence, Systems & Applications (IISA)*. IEEE, 1–8. https://doi.org/10.1109/IISA52424.2021.9555522

[11] Elena Raponi, Hao Wang, Mariusz Bujny, Simonetta Boria, and Carola Doerr. 2020. High Dimensional Bayesian Optimization Assisted by Principal Component Analysis. In *Proc. PPSN (LNCS, Vol. 12269)*. Springer, 169–183. https://doi.org/10.1007/978-3-030-58112-1_12

[12] Hao Wang, Diederick Vermetten, Furong Ye, Carola Doerr, and Thomas Bäck. 2022. IOHanalyzer: Detailed Performance Analyses for Iterative Optimization Heuristics. *ACM Trans. Evol. Learn. Optim.* 2, 1 (2022), 3:1–3:29. https://doi.org/10.1145/3510426

[13] Zi Wang, Clement Gehring, Pushmeet Kohli, and Stefanie Jegelka. 2018. Batched large-scale Bayesian optimization in high-dimensional spaces. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 745–754.

[14] Ziyu Wang, Frank Hutter, Masrour Zoghi, David Matheson, and Nando de Freitas. 2016. Bayesian Optimization in a Billion Dimensions via Random Embeddings. *arXiv:1301.1942* (2016).