Comparison of High-Dimensional Bayesian Optimization Algorithms on BBOB

Description

This repository contains the code and results showcased in the TELO paper `Comparison of High-Dimensional Bayesian Optimization Algorithms on BBOB`. The code for reproducing the experiments is also 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 [1] as 'HDBO' dataset.

The selected algorithms for comparison are:

- **Sparse Axis Aligned Subspace Bayesian Optimization (SAASBO)** [2],
- **Random Decomposition Upper-Confidence Bound (RDUCB)** [3],
- **PCA-assisted Bayesian Optimization (PCA-BO)** [4],
- **Kernel PCA-assisted Bayesian Optimization (KPCA-BO)** [5],
- **Trust Region Bayesian Optimization (TuRBO)** [6].

The collected data also includes results for the **Heteroscedastic and Evolutionary Bayesian Optimisation (HEBO) solver [7]** and **Ensemble Bayesian Optimization (EBO)** [8] but their results are not shown in this paper due to reasons explained in the main text.

The work compares the algorithms on the 24 noiseless functions of COCO's BBOB suite [9], accessed via IOHprofiler [10]. For each function, dimensions 10, 20, 40, and 60 are considered. For SAASBO complete experiments for dimensions 20, 40, and 60 are missing due to time and memory constraints. 10 independent runs of each algorithm are conducted for three different instances (instance ID 0-2) of the 24 BBOB functions, with the exception of SAASBO. For the SAASBO algorithm:

- Data is available for functions $f1$-$f24$ at dimension 10,
- Data is available for functions $f15$-$f24$ at dimensions 20 and 40,
- No data is available for dimension 60 (due to infeasible computational time and memory requirements).

The HDBO algorithms are compared with a vanilla BO from [11], a default CMA-ES [12], and random search [13]. For each run, the total evaluation budget is set to $10D + 50$ 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 selected algorithm default settings for their hyperparameters are used. Algorithm performance is evaluated in terms of (1) loss (defined as the target precision, i.e., the absolute difference between the best-so-far value and the value of the global minimum) and (2) CPU time.

[1] https://github.com/MariaLauraSantoni/IOH-Profiler-HDBO-Comparison

Project Structure

The project is structured with a folder for each tested algorithm. Within each algorithm folder, there are subfolders corresponding to each experiment. Each experiment folder is named based on the experiment details, including the function, dimension, algorithm, and repetition information. For example, the naming convention for the experiment folders is `BOdim60i0_Opt-BO_sklearn_F-15_Dim-60_Rep-2_Id-2-0`. This indicates the name of the experiment, the optimization algorithm used, the function being optimized, the dimensionality, and the repetition number. The data folders are structured according to the IOHanalyzer format. Inside each experiment folder, there are two files:

- A Metadata file, such as `IOHprofiler_f1.info`, serves to summarize the algorithmic performance for each problem instance. Within these files, essential information is encapsulated, providing a concise overview of the results obtained from individual experiments. The file adopts a structured format consisting of three lines. The initial line captures meta-information about the experiment through (key, value) pairs, delineated by commas. Notably, three keys—`funcId`, `DIM`, and `algId`—are case-sensitive and indispensable components of this structure. Additionally, the `maximization` key is set to `F` (False) indicating the focus on minimization tasks in this work. The second line always starts with a single %, indicating that what follows this symbol should be the general comments from the user on this experiment. By default, it is left empty. The instance number can be found on the third line before the colon.

- A Raw-data (e.g., IOHprofiler_f1_DIM60.dat) is a CSV-like file that contains performance information. The columns and header of this format are as follows: (1) the column labeled `function evaluation` indicates the current number of function evaluations, (2) the column labeled `current f(x)` denotes the loss value observed when the corresponding number of function evaluations is consumed, (3) the column labeled `best-so-far f(x)` tracks the best loss value observed since the beginning of one run, (4) the values stored under `current af(x)+b` and `best af(x)+b` represent the raw function values. For BO-based algorithms, three additional columns are included: `acq_opt_time`, `model_fit_time`, and `cum_iteration_time`. These columns store the CPU time required for specific tasks during the optimization process. Specifically, `acq_opt_time` represents the time taken to optimize the acquisition function, `model_fit_time` indicates the time required to fit the model, and `cum_iteration_time` records the cumulative iteration time at each corresponding evaluation. The results plotted in the paper specifically focus on the loss values extracted from the `best-so-far f(x)` column and the CPU time values.

References

[1] H. Wang, D. Vermetten, F. Ye, C. Doerr, and T. Bäck, "IOHanalyzer: Detailed Performance Analyses for Iterative Optimization Heuristics," *ACM Trans. Evol. Learn. Optim.*, vol. 2, no. 1, pp. 3:1–3:29, 2022.

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

[3] J. K. Ziomek and H. Bou Ammar, "Are random decompositions all we need in high dimensional Bayesian optimisation?," in *Proceedings of the 40th International Conference on Machine Learning* (A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, eds.), vol. 202 of *Proceedings of Machine Learning Research*, pp. 43347–43368, PMLR, 23–29 Jul 2023.

[4] E. Raponi, H. Wang, M. Bujny, S. Boria, and C. Doerr, "High dimensional bayesian optimization assisted by principal component analysis," in *Proc. of Parallel Problem Solving from Nature (PPSN)*, vol. 12269 of *LNCS*, pp. 169–183, Springer, 2020.

[5] K. Antonov, E. Raponi, H. Wang, and C. Doerr, "High dimensional bayesian optimization with kernel principal component analysis," in *Proc. of Parallel Problem Solving from Nature (PPSN)*, vol. 13398, pp. 118–131, Springer, 2022.

[6] D. Eriksson, M. Pearce, J. Gardner, R. D. Turner, and M. Poloczek, "Scalable global optimization via local Bayesian Optimization," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[7] A. I. Cowen-Rivers, W. Lyu, R. Tutunov, Z. Wang, A. Grosnit, R. R. Griffiths, A. M. Maraval, H. Jianye, J. Wang, J. Peters, and H. B. Ammar, "Hebo pushing the limits of sample-efficient hyperparameter optimisation," 2022.

[8] Z. Wang, C. Gehring, P. Kohli, and S. Jegelka, "Batched large-scale bayesian optimization in high-dimensional spaces," in *International Conference on Artificial Intelligence and Statistics*, pp. 745–754, PMLR, 2018.

[9] N. Hansen, A. Auger, R. Ros, O. Mersmann, T. Tusar, and D. Brockhoff, "COCO: A platform for comparing continuous optimizers in a black-box setting," *Optimization Methods and Software*, vol. 36, pp. 114–144, 2021.

[10] C. Doerr, H. Wang, F. Ye, S. van Rijn, and T. Bäck, "IOHprofiler: A benchmarking and profiling tool for iterative optimization heuristics," *arXiv preprint arXiv:1810.05281*, 2018. https://iohprofiler.github.io/.

[11] T. Head, M. Kumar, H. Nahrstaedt, G. Louppe, and I. Shcherbatyi, "scikit-optimize: Sequential model-based optimization in python," 2017.

[12] N. Hansen, Y. Akimoto, and P. Baudis, "Cma-es/pycma on github. zenodo, doi: 10.5281/zenodo. 2559634.(feb. 2019)," 2019.

[13] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, Sept. 2020.