The Journal of Open Source Software

PyDDSBB: A Python Package for Simulation-Optimization using Data-Driven Branch-and-Bound Techniques

Jianyuan Zhai [1], **Suryateja Ravutla** [1], **and Fani Boukouvala** [1¶]

1 Department of Chemical and Biomolecular Engineering, Georgia Institute of Technology, Atlanta, GA ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

High-fidelity (HF) computer simulations are essential for quantitative analysis and decision-making (i.e., design or inverse optimization), however, often the objective functions and the constraints reliant on a simulation are only available as black-box functions (Bhosekar & Ierapetritou, 2018; Boukouvala et al., 2016; McBride & Sundmacher, 2019; Rios & Sahinidis, 2013). Sample generation using HF simulations is often expensive due to computational cost and time. Most of the equation/derivative-based optimization solvers cannot be directly applied to optimize problems with embedded HF simulations due to lack of closed-form equations and gradient information. As an alternative, machine learning (ML) surrogate models are often employed to approximate HF simulation data and expedite the optimization search(Bhosekar & Ierapetritou, 2018). However, training and building accurate models requires large amount of data model parameterizations are highly subjected to the available data. One may arrive at different solutions because the ML models may be nonconvex and the surrogate model parameterizations can be different due to re-initialization and slight changes in the data set. Finally, most derivative-free or black-box methods offer no information regarding the quality of the incumbent optimal solution (e.g., upper/lower bound on the optimum) (Amaran et al., 2016; Boukouvala et al., 2016; Larson et al., 2019; Zhai et al., 2021; Zhai & Boukouvala, 2022). To tackle these challenges, we recently proposed a data-driven equivalent of the spatial branch-and-bound (DDSBB) algorithm based on the concept of constructing convex underestimators of HF data from simulations and low-fidelity (LF) data from ML model approximations (Zhai et al., 2021; Zhai & Boukouvala, 2022).

Statement of need

Black-box optimization is a challenging problem due to lack of analytic equations and derivatives. Such problems arise in many fields of science and engineering, such as chemical process design, oilfield operations, protein folding, aircract/vehicle design, and many more (McBride & Sundmacher, 2019). Developing novel and improved data-driven approaches for efficient optimization of such systems will help in quantitative analysis and improved decision making. Although many methods exist, one drawback of many sampling- or ML-based optimization methods is the high dependency on initial sampling and selection of the ML model type. PyDDSBB is a Python package for the proposed Data-Driven Spatial Branch-and-Bound Algorithm(Zhai et al., 2021; Zhai & Boukouvala, 2022). Instead of only using samples, or directly relying on a single ML model fit, pyDDSBB develops convex underestimators as relaxations of data generated from HF models and/or ML surrogates. These relaxations are embedded into the branch and bound algorithm, and the search space is progressively partitioned by branching and pruning heuristics. This approach allows for estimation of upper

and lower bounds on the optimum solution, which help in pruning spaces that have a very low probability of containing a better optimum. Samples are added adaptively in the non-pruned subspaces(Rios & Sahinidis, 2013; Zhai et al., 2021; Zhai & Boukouvala, 2022). Through benchmarking of pyDDSBB on a large pool of problems with dimensions 1-10, we have shown that the solver can locate the global optimum for >90% of the problems with 2-3 dimensions and >70% of the problems with 4-10 dimensions. Most importantly pyDDSBB provides valid bounds for overall >90% of the problems(Zhai & Boukouvala, 2022).

Overview and Description

The PyDDSBB algorithm has been constructed using object-oriented programming, with an intent of easier incorporation of further additions or extensions to the algorithm. The algorithm has the option of employing Support Vector Regression models (trained by HF data) to generate LF data that also inform the underestimators. PyDDSBB utilizes the Scikit-learn (Pedregosa et al., 2011) package for constructing and optimizing the surrogate models for LF data generation and also offers the capability of user-based additions of new surrogate models. The algorithm utilizes the package – PYOMO(Hart et al., 2011) to construct the convex quadratic underestimators and also provides an option for integration of other type of underestimators. Latin Hypercube Sampling (LHS) technique is used for generating the samples and the samples are generated adaptively in the non-pruned subspaces by utilizing the bounding information. Two possible paths for sampling and constructing the underestimators in algorithm are shown in Figure 1.

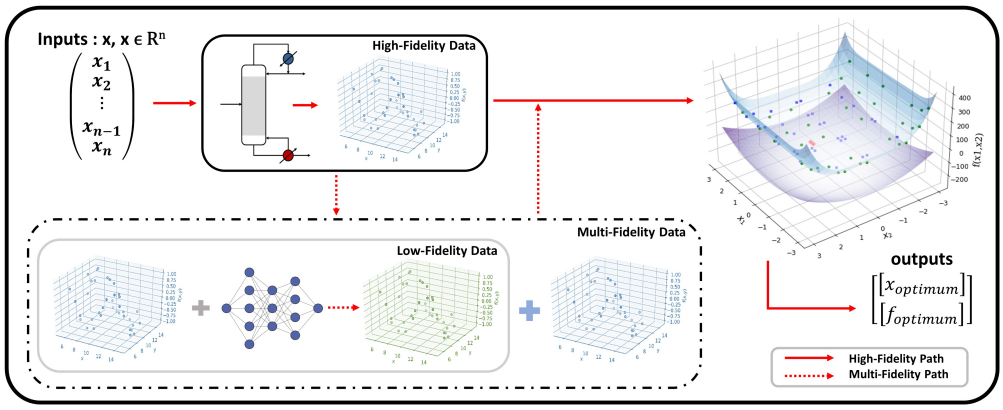

Figure 1: Overview of PyDDSBB sampling and understimator construction. Solid red line shows the HF path where only HF samples obtained from black-box simulation are used in constructing underestimators. Dotted red line shows the path for MF approach, in which HF samples are used to generete LF samples and, combined HF and LF samples are used to construct underestimators

While the objective function is assumed to be simulation-based (black-box), PyDDSBB allows the user to include simulation-based constraints (unknown or black-box) and equation-based (known) constraints into the formulation. A variety of branching techniques (e.g., equal bisection, longest side, ML-based prioritizing important variables, etc.)(Zhai et al., 2021) and branch-and-bound heuristics are considered, and default options are recommended for the case of box-constrained and general constrained problems. The HF simulation can be in the form of a python function, or any external platform that could be connected via an API. The ultimate goal of this software is to provide a user-friendly simulation-based optimization framework for both expert and non-expert users, benefiting from the high-level features of Python. PyDDSBB follows the object-oriented programming paradigm and is designed to allow easy extension of the core functionality by users and developers. The optimization of an example benchmark

black-box function and visualization of the outputs in shown in Figure 2

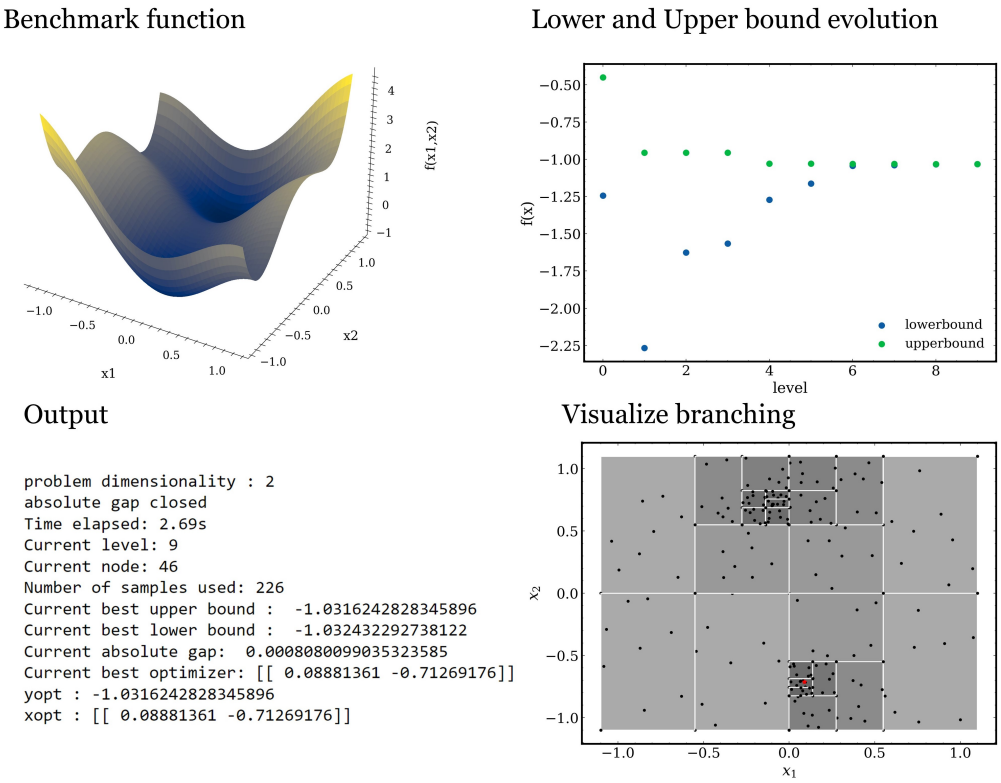

Figure 2: Optimization of sample benchmark function and visualizing the output. **Top left:** A non-convex benchmark function $f(x_1, x_2) = 4x_1^2 - 2.1x_1^4 + 0.3333x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$, available as black-box model. **Top right:** Evolution of upper and lower bounds found during the branching process of the sample space. **Bottom right:** Sampling and branching in the sample space visualized for the branch-and-bound process. Darker regions correspond to a higher level of branching and the red dot represents the optimum solution found. **Bottom left:** Algorithm output after termination with some information on sampling and optimum solution found

Acknowledgements

The authors acknowledge financial support from the National Science Foundation (NSF-1805724 & GR00003313) and DOE/AICHE RAPID Institute "Synopsis" project grant (GR10002225)

References

Amaran, S., Sahinidis, N. V., Sharda, B., & Bury, S. J. (2016). Simulation optimization: a review of algorithms and applications. *Annals of Operations Research*, *240*(1), 351–380.

Bhosekar, A., & Ierapetritou, M. (2018). Advances in surrogate based modeling, feasibility analysis, and optimization: A review. *Computers & Chemical Engineering*, *108*, 250–267.

Boukouvala, F., Misener, R., & Floudas, C. A. (2016). Global optimization advances in mixed-integer nonlinear programming, MINLP, and constrained derivative-free optimization, CDFO. *European Journal of Operational Research*, *252*(3), 701–727.

Hart, W. E., Watson, J.-P., & Woodruff, D. L. (2011). Pyomo: modeling and solving mathematical programs in Python. *Mathematical Programming Computation*, *3*(3), 219–260.

Larson, J., Menickelly, M., & Wild, S. M. (2019). Derivative-free optimization methods. *Acta Numerica*, *28*, 287–404.

McBride, K., & Sundmacher, K. (2019). Overview of surrogate modeling in chemical process engineering. *Chemie Ingenieur Technik*, *91*(3), 228–239.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., & Dubourg, V. (2011). Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, *12*, 2825–2830.

Rios, L. M., & Sahinidis, N. V. (2013). Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, *56*(3), 1247–1293.

Zhai, J., & Boukouvala, F. (2022). Data-driven spatial branch-and-bound algorithms for box-constrained simulation-based optimization. *Journal of Global Optimization*, *82*(1), 21–50.

Zhai, J., Shirpurkar, S., & Boukouvala, F. (2021). Data-driven Branch-and-bound Algorithms for Constrained Simulation-based Optimization. In *Computer aided chemical engineering* (Vol. 50, pp. 1067–1072). Elsevier. ISBN: 1570-7946