Honda Research Institute EU
Honda Research Institute Europe GmbH
https://www.honda-ri.de/

MF2: A Collection of Multi-Fidelity Benchmark Functions in Python

Sander van Rijn, Sebastian Schmitt

2020

Preprint:

This is an accepted article published in Journal of Open Source Software. The final authenticated version is available online at:
https://doi.org/10.21105/joss.02049

The Journal of Open Source Software

MF2: A Collection of Multi-Fidelity Benchmark Functions in Python

Sander van Rijn[1] **and Sebastian Schmitt**[2]

1 Leiden University, The Netherlands **2** Honda Research Institute Europe, Germany

DOI: 10.21105/joss.02049

Software

- Review
- Repository
- Archive

Editor: Melissa Weber Mendonça

Reviewers:

- @torressa
- @zbeekman

Submitted: 17 January 2020
Published: 25 August 2020

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

Summary

The field of (evolutionary) optimization algorithms often works with expensive black-box optimization problems. However, for the development of novel algorithms and approaches, real-world problems are not feasible due to their high computational cost. Instead, benchmark functions such as Sphere, Rastrigin, and Ackley are typically used. These functions are not only fast to compute, but also have known properties which are very helpful when examining the performance of new algorithms.

As only a limited set of benchmark functions are typically used in the literature, compiling a set of implementations for the most commonly used functions is warranted. This ensures correctness of the functions, makes any results directly comparable, and simply saves researchers time from not having to implement the functions themselves. An example of a commonly used benchmark suite for optimizing continuous problems is the COCO BBOB software by Hansen et al. (2019).

As simulation-based problems in engineering are requiring increasingly more time and computation power, a new sub-field of *multi-fidelity* optimization has gained popularity. A multi-fidelity problem is characterised by having multiple versions of an evaluation function that differ in their accuracy of describing the real objective. A real-world example would be the aerodynamic efficiency of an airfoil: A *low-fidelity* simulation would use a coarse mesh, and give lower accuracy, but be fast to calculate, while a *high-fidelity* simulation would use a much finer mesh and therefore be more accurate while taking longer to calculate. Multi-fidelity methods aim to combine these multiple information sources to obtain better results in equal or less time compared to only using a single information source.

To this end, new multi-fidelity benchmark functions have been introduced in the literature and are being adopted by other researchers. These multi-fidelity problems naturally benefit from having the different fidelities combined into a single 'problem'. The existing single-fidelity benchmark suites that exist cannot be used for this field: no existing suite of benchmark functions currently uses such a structure, or can easily accomodate it. Therefore, this new class of benchmark problems is best served by introducing a new implementation suite because their structure is inherently different from other benchmarks. A new suite additionally gives more freedom to adapt to new multi fidelity benchmarks as the field continues to evolve and new needs become apparent.

The `MF2` package provides a consistent Python implementation of a collection of these Multi-Fidelity Functions. It uses a standard interface that allows for querying single vectors or multiple row-vectors as a single matrix, relying on `numpy`'s optimized back-end to handle parallelization. It also offers a simple factory pattern interface for functions with parameters for e.g. correlation and dimensionality. A plot of how these implementations scale can be seen in Figure 1.

At this moment, `MF2` has collected functions from the following previous works:

- Forrester, Sóbester, & Keane (2007) introduced a simple 1D bi-fidelity function for mostly illustrative purposes.
- Surjanovic & Bingham (2013) have previously collected a small collection of MATLAB/R implementations for the Borehole, Currin and Park91 A and B functions.
- Dong, Song, Wang, & Huang (2015) introduced bi-fidelity versions of the Bohachevsky, Booth, Branin, Himmelblau and Six-hump Camelback functions.
- Toal (2015) introduced correlation-adjustable multi-fidelity versions of the Branin, Paciorek, Hartmann3 and Trid functions.

As no convenient existing implementations written in Python could be found during the authors' research on how the accuracy of multi-fidelity surrogate models depends on the number of samples per fidelity, which required the evaluation of many independent model training and test sets, the decision was made to standardize the implementations and make them available for wider use.

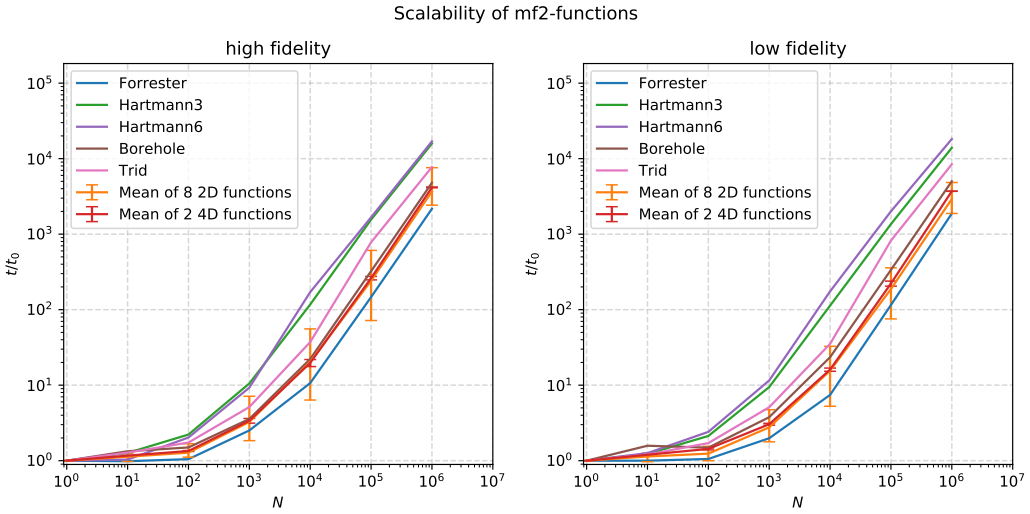


Figure 1: Scalability plot This plot shows how the evaluation time of high- and low-fidelity functions scales with the number of points N being passed in simultaneously. Running times were measured on a desktop PC with an Intel core i7 5820k 6-core CPU, with Python 3.6.3 and Numpy 1.18.4. The times are divided by the time needed for N=1 as a normalization. This is done independently for each function and fidelity level. Results are grouped by function dimensionality. If there are multiple functions, the mean is plotted with error bars indicating the minimum and maximum time. Note that the 3D Hartmann3 and 6D Hartmann6 function are significantly more computationally expensive than other functions by definition, as they requires multiple matrix multiplications.

Acknowledgements

This work is part of the research program DAMIOSO with project number 628.006.002, which is partly financed by the Netherlands Organisation for Scientific Research (NWO).

The first author would like to thank dr. Matthijs van Leeuwen, prof. dr. Thomas Bäck, and dr. Markus Olhofer for their supervision and involvement in the DAMIOSO project.

References

Dong, H., Song, B., Wang, P., & Huang, S. (2015). Multi-Fidelity Information Fusion Based

on Prediction of Kriging. *Struct. Multidiscip. Optim.*, *51*(6), 1267–1280. doi:10.1007/s00158-014-1213-9

Forrester, A. I. J., Sóbester, A., & Keane, A. J. (2007). Multi-Fidelity Optimization via Surrogate Modelling. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *463*(2088), 3251–3269. doi:10.1098/rspa.2007.1900

Hansen, N., Brockhoff, D., Mersmann, O., Tusar, T., Tusar, D., ElHara, O. A., Sampaio, P. R., et al. (2019). *COmparing Continuous Optimizers: numbbo/COCO on Github*. Zenodo. doi:10.5281/zenodo.2594848

Surjanovic, S., & Bingham, D. (2013). Virtual library of simulation experiments: Test functions and datasets. Retrieved October 10, 2017, from http://www.sfu.ca/~ssurjano.

Toal, D. J. J. (2015). Some Considerations Regarding the Use of Multi-Fidelity Kriging in the Construction of Surrogate Models. *Structural and Multidisciplinary Optimization*, *51*(6), 1223–1245. doi:10.1007/s00158-014-1209-5