egobox, a Rust toolbox for efficient global optimization

Rémi Lafage[1]

1 ONERA, Université de Toulouse, France

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor:

Submitted: 20 April 2022
Published: unpublished

Summary

Efficient global optimization algorithms are used to tackle design optimization problems that involve computationally costly models (Jones et al., 1998). Basically, the idea is to use gaussian process regression to approximate the objective function and use probabilistic information to select the promising next point where the optimum could be located. Starting from that, one still has to tackle numerous challenges regarding the dimensionality, multi-modality and computation time. Different algorithms may be develop to overcome these hurdles (Bartoli et al., 2019; Dubreuil et al., 2020).

A key component of such surrogate-based algorithm is the existence of an implementation of gausian process regression also known as kriging method (Bouhlel et al., 2016). The Surrogate Modeling Toolbox (Bouhlel et al., 2019; smtorg, 2018) library addresses such concerns using the Python programming language focusing on various surrogate modeling methods and derivatives.

Started by porting relevant parts from the SMT Python library, the egobox library aims at providing building blocks useful to implement EGO-like algorithms and take advantage of the Rust programming language.

Statement of need

Research scientists reach for prototyping programming language such as Python to develop new methods. Thanks to performant easy-to-use libraries like `numpy`, `scipy`, `scikit-learn` the Python language has become popular in science computing. The ability of Python to glue together different codes explains Python ecosystem is now a de-facto open source standard for scientific software. Nevertheless, one might notice that the performances of above Python libraries relies extensively on C/Fortran compiled code.

Thus, in the long run, a language like Python well-suited for prototyping may not be suitable for maintainability or performance even with a strong software development discipline. Performance issues may arise as new algorithms are built on top of the others which at some point may not be tractable with such interpreted programming language. As stated above, it is common to resort to compiled languages like C/C++ or Fortran to implement computation-intensive methods or for embeding them in more constrained environments where a Python runtime does not fit.

Library features

With the above need in mind, the Rust programming language appears to be of interest with its selling points being performance, reliability and productivity. The language is meant to challenge C as a system language but also rely on strong typing, and high level features such as functional programming, algebric data types, module management. It has a strict approach regarding memory-safety management and benefits from a state of the art tooling for software development.

The Rust community has developed scientific libraries like `ndarray`, `ndarray-linalg` which can be seen as the `numpy` and `scipy` Rust counterparts. Last but not least, the `linfa` project addresses the machine-learning domain with the purpose of being the Rust equivalent of `scikit-learn`.

The egobox library relies on the above thriving Rust machine learning ecosystem and focuses on providing some building blocks to implement efficient global optimization algorithms. The library is organized in four sub-packages as follows:

- `doe`: sampling methods implementing Latin Hypercube sampling, popular sampling method used to create design of experiments,
- `gp`: gaussian process regression also known as kriging algorithm used as surrogate models for computationally costly black-box functions,
- `moe`: mixture of experts which aims at increasing the accuracy of a surrogate approximation by clustering the design space, training and selecting the best surrogate models on each cluster,
- `ego`: an efficient global optimization implementation with handling of inequality constraints and mixed integer optimization through continuous relaxation.

Finally thanks to the `PyO3` project, the Rust language is well-suited to create Python extensions which benefits from Rust strenghts while being integrated in the Python ecosystem.

In order to increase the dissemination among the scientific community and demonstrate actual optimization capabilities based on the library, we implemented a Python module `egobox` as the binding of the implemented EGO-like Rust optimizer, namely `Egor`.

Acknowledgements

I would like to thank my colleagues Nathalie Bartoli, Thierry Lefebvre and Sylvain Dubreuil as their work on surrogate-based adaptative optimization has fueled this software development.

References

Bartoli, N., Lefebvre, T., Dubreuil, S., Olivanti, R., Priem, R., Bons, N., Martins, J. R. R. A., & Morlier, J. (2019). Adaptive modeling strategy for constrained global optimization with application to aerodynamic wing design. *Aerospace Science and Technology*, *90*, 85–102. https://doi.org/https://doi.org/10.1016/j.ast.2019.03.041

Bouhlel, M. A., Bartoli, N., Otsmane, A., & Morlier, J. (2016). Improving kriging surrogates of high-dimensional design models by partial least squares dimension reduction. *Structural and Multidisciplinary Optimization*, *53*(5), 935–952.

Bouhlel, M. A., Hwang, J. T., Bartoli, N., Lafage, R., Morlier, J., & Martins, J. R. R. A. (2019). A python surrogate modeling framework with derivatives. *Advances in Engineering Software*, 102662. https://doi.org/https://doi.org/10.1016/j.advengsoft.2019.03.005

Dubreuil, S., Bartoli, N., Gogu, C., & Lefebvre, T. (2020). Towards an efficient global multidisciplinary design optimization algorithm. *Structural and Multidisciplinary Optimization*, *62*(4), 1739–1765.

Jones, D. R., Schonlau, M., & Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, *13*(4), 455–492.

smtorg. (2018). Surrogate modeling toolbox. In *GitHub repository*. GitHub. https://github.com/SMTOrg/smt