Gradient-free optimization study. Full-Low Evaluation Method

Elizaveta Zarezina
Optimization Class Project. MIPT

Introduction

Derivative free optimization (DFO) is used when information about the derivative of the objective function is unavailable, unreliable or impractical to obtain. Many DFO methods are aimed at narrow classes of problems. For instance, they outperform acceptable results in cases when the objective function is smooth, convex, continuous, separable, when there are no constraints and when there is no noise. Some of them don't have proofs of convergence, robustness, efficiency, *etc.*. [2], [3], [4]. Full-Low Evaluation method is expected to be efficient and robust for functions of all types under different noise regimes [1]. The project aims is to ensure this.

Full-Low Evaluation: Algorithm

Choose an initial iterate x_0. Set iteration i-type(0) = Full-Eval.

1: **for** $k = 0, 1, \ldots$ **do**
2: **if** i-type(k) = Full-Eval **then**
3: attempt to compute a Full-Eval step
4: **if** success **then**
5: update x_{k+1} and set i-type($k+1$) = Full-Eval
6: **else**
7: $x_{k+1} = x_k$ and i-type($k+1$) = Low-Eval
8: **end if**
9: **end if**
10: **if** i-type(k) = Low-Eval **then**
11: compute a Low-Eval step, update x_{k+1}, choose i-type($k+1$)
12: **end if**
13: **end for**

The reviewed method is organized around two iteration types and a switch condition. The Full iteration type is expensive in function evaluations, but exhibits good performance in the smooth cases. It consists of a line search based on a gradient, approximated via finite differences, and the direction calculated by a quasi-Newton step (BFGS). The Low iteration type is cheap in function evaluations, yet more robust in the presence of or non-smoothness. It consists of probabilistic direct search. [1]

Set of problems

Test functions given herein represent various challenging tasks that the DFO method has to face; they are useful to evaluate its characteristics.

- Rastrigin function
- Ackley function
- Rosenbrock function
- Bukin N6 function
- Easom function
- Schaffel N2 function

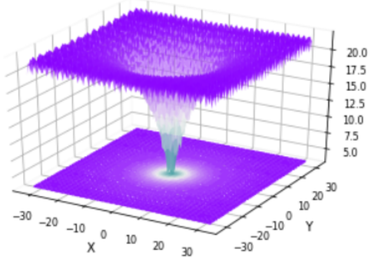

Figure 1: Ackley Function

Classical methods of DFO

The DFO methods given herein are listed as a reference. The implementations are from *scipy.optimize*, *optuna* and etc.

- BFGS-FD
- Powell's method
- Bayessian optimization
- Nelder-Mead method
- Simulated annealing

Numerical experiment

Consider a numerical example with the N-dimensional objective function from the set of problems, N varies. The unconstrainted optimization problem is standard: $f(x) : \mathbf{R}^n \to \mathbf{R}$,

$$f(x) \to \min_{x \in \mathbf{R}^n} \tag{1}$$

The aims of the experiment are to compare DFO methods' results using quality metrics and to draw conclusions about Full-Low Evaluation method's efficiency, robustness and competitiveness.

Results

Effectiveness $= \|x - x^*\|$, where x is an optimum's location, obtained via one of considered DFO methods; x^* is an analytical solution.

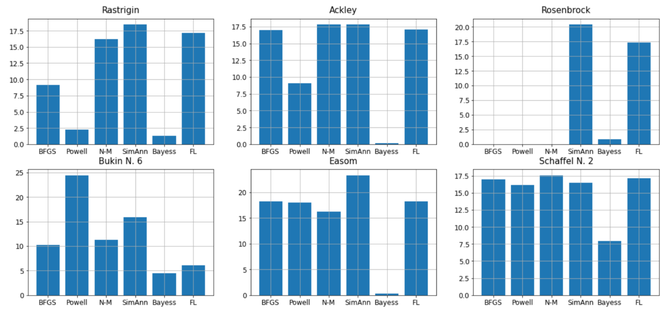

Figure 2: Effectiveness, N = 2

Effeciency $= \dfrac{T_{FLE}}{T_{DFO}} \approx 10^2 - 10^3$, where T_{FLE} is operating time for Full-Low method, T_{DFO} is operating time for classical DFO methods.
Function's value $= f(x)$, where x is an optimum's location, obtained via one of considered DFO methods, f is a test function.

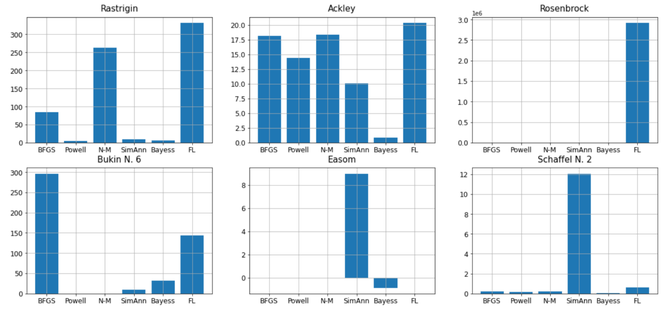

Figure 3: Function's value, N = 2

Results. Noisy Case

Full-Low Evaluation Method's behavior in noisy cases was also investigated. It shows tolerance to normally distributed random noise because these test functions are difficult to optimize with DFO methods even if there is no noise. In other words, the method's results in noisy and non-noisy cases are approximately the same. Operating time in all cases does not exceed $4 \cdot 10^{-2}$ seconds.

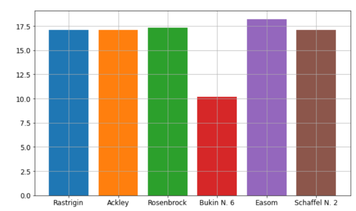

Figure 4: Noisy Effectiveness, N=2

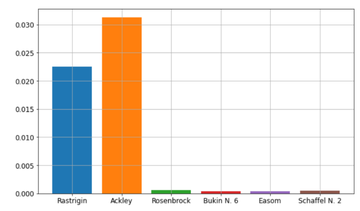

Figure 5: Noisy Duration (sec), N = 2

Conclusion

In case of complicated test problems zero-order methods show unreliable results, and considered Full-Low Evaluations method's results are relatively the same. However, it is sufficiently more expensive in time because of Python implementation. An important inference is that the initial point which was generated randomly greatly influences DFO methods' results. The method under consideration showed comparable behavior in non-noisy and noisy cases, but in general results are not reliable and stable. Such outcome is probably a consequence of improper implementation of Full-Low algorithm.

Acknowledgements

This material is based upon work supported by Cesar William Alvarenga and Axel Thevenot.

References

[1] Albert S. Berahas, Oumaima Sohab, and Luis Nunes Vicente. Full-low evaluation methods for derivative-free optimization. *arXiv*, 2021.

[2] Sahinidis N.V. Rios, L.M. Derivative-free optimization: a review of algorithms and comparison of software implementations. *J Glob Optim*, (56), 2013.

[3] Oliver Kramer, David Echeverría Ciaurri, and Slawomir Koziel. *Derivative-Free Optimization*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[4] Hao-Jun Michael Shi, Melody Qiming Xuan, Figen Oztoprak, and Jorge Nocedal. On the numerical performance of derivative-free optimization methods based on finite-difference approximations. *arXiv*, 2021.

Realization: Code on GitHub