Blackbox optimization with the MADS algorithm and the NOMAD software

Sébastien Le Digabel

CRM-McGill Applied Mathematics Seminar, 2022-02-14

Contributors and partners

CRSNG
NSERC

InnovÉÉ
Innovation en énergie électrique

IVADO

Hydro Québec

RioTinto

HUAWEI

Research team

Professors (SLD and C. Audet)

Research associate (C. Tribes)

Postdocs / Students

Graduates

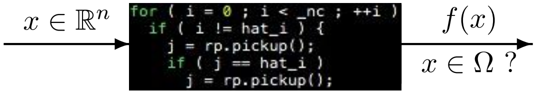
Presentation outline

Blackbox / Derivative-Free Optimization

We consider

$$\min_{x\in\Omega} \quad f(x)$$

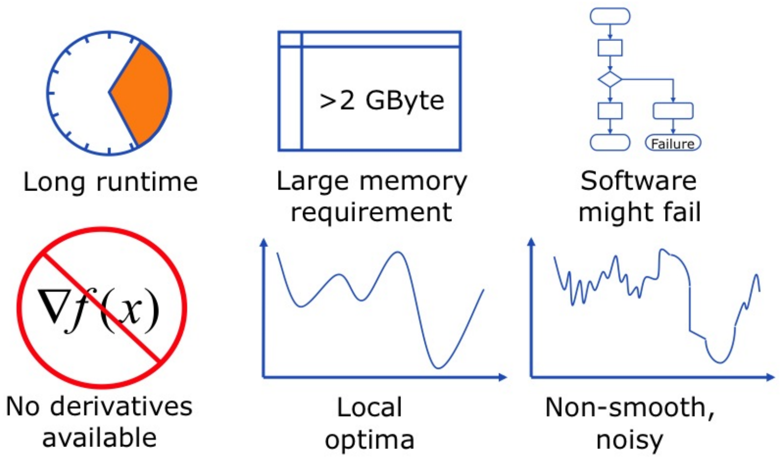
where the evaluations of f and the functions defining Ω are the result of a computer simulation (a blackbox)



```
for ( i = 0 ; i < _nc ; ++i )
  if ( i != hat_i ) {
    j = rp.pickup();
    if ( j == hat_i )
      j = rp.pickup();
```

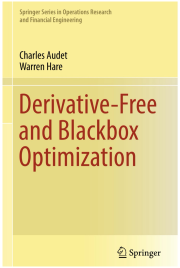
- Each call to the simulation may be expensive
- The simulation can fail
- Sometimes $f(x) \neq f(x)$
- Derivatives are not available and cannot be approximated

Blackboxes as illustrated by a Boeing engineer



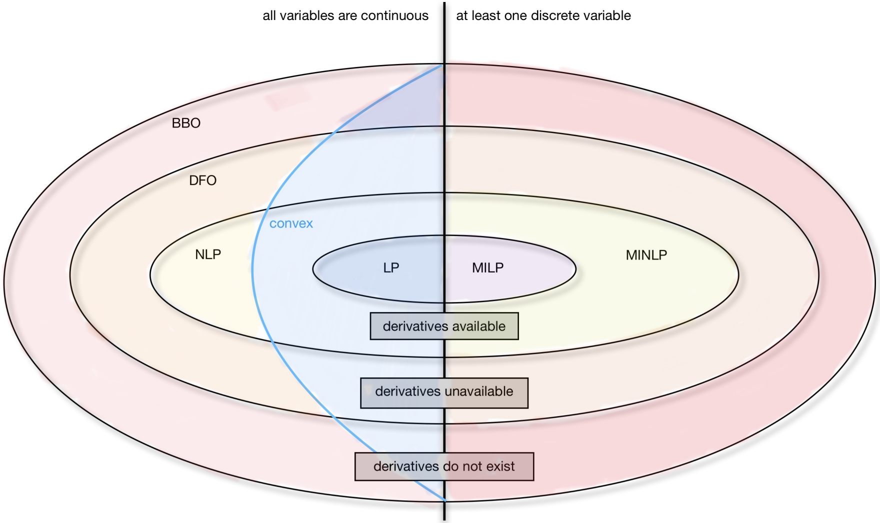
Terms

- *"Derivative-Free Optimization (DFO) is the mathematical study of optimization algorithms that do not use derivatives"* [Audet and Hare, 2017]
  - Optimization without using derivatives
  - Derivatives may exist but are not available
  - Obj./constraints may be analytical or given by a blackbox

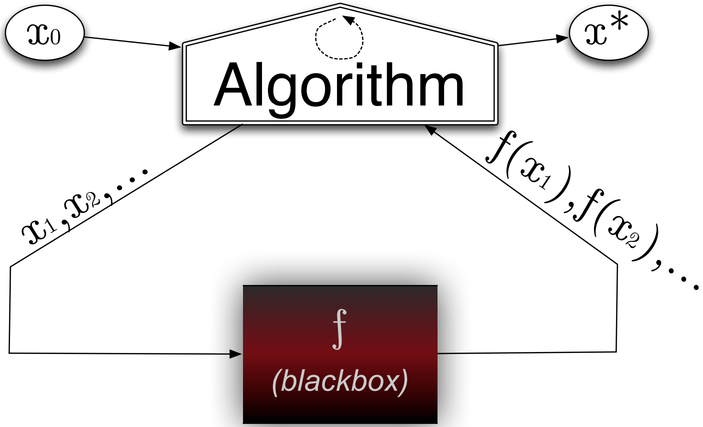

Springer Series in Operations Research and Financial Engineering
Charles Audet
Warren Hare
Derivative-Free and Blackbox Optimization
Springer

- *"Blackbox Optimization (BBO) is the study of design and analysis of algorithms that assume the objective and/or constraints functions are given by blackboxes"* [Audet and Hare, 2017]
  - A simulation, or a blackbox, is involved
  - Obj./constraints may be analytical functions of the outputs
  - Derivatives may be available (ex.: PDEs)
  - Sometimes referred as *Simulation-Based Optimization (SBO)*

Optimization: Global view



Typical setting



Unconstrained case, with one initial starting solution

Algorithms for blackbox optimization

A method for blackbox optimization should ideally:

- Be efficient given a limited budget of evaluations
- Be robust to noise and blackbox failures
- Natively handle general constraints
- Deal with multiobjective optimization
- Deal with integer and categorical variables
- Easily exploit parallelism
- Have a publicly available implementation
- Have convergence properties ensuring first-order local optimality in the smooth case – otherwise why using it on more complicated problems?

Families of methods

- *"Computer science"* methods:
  - Heuristics such as genetic algorithms
  - No convergence properties
  - Cost a **lot** of evaluations
  - Should be used only in last resort for desperate cases
- Statistical methods:
  - Design of experiments
  - Bayesian optimization: EGO algorithm based on surrogates and expected improvement
  - Still limited in terms of dimension
  - Does not natively handle constraints
  - Better to use these tools in conjonction with DFO methods
- Derivative-Free Optimization methods (DFO)

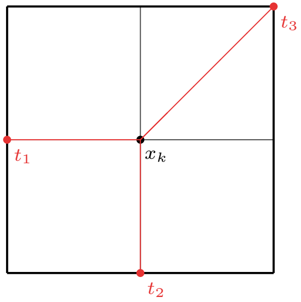
DFO methods

- **Model-based methods:**
  - Derivative-Free Trust-Region (DFTR) methods.
  - Based on quadratic models or radial-basis functions
  - Use of a trust-region
  - Better for $\{$ DFO $\setminus$ BBO $\}$
  - Not resilient to noise and *hidden constraints*
  - Not easy to parallelize
- **Direct-search methods:**
  - Classical methods: Coordinate search, Nelder-Mead – the *other* simplex method
  - Modern methods: Generalized Pattern Search (GPS), Generating Set Search (GSS), Mesh Adaptive Direct Search (MADS)

So far, the size of the instances (variables and constraints) is typically limited to $\simeq 50$, and we target local optimization

MADS illustration with $n = 2$: Poll step

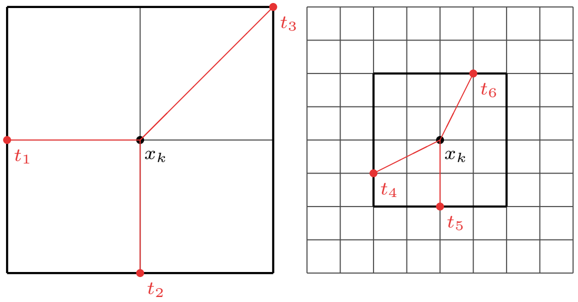
$$\Delta_k^m = \Delta_k^p = 1$$



poll trial points$=\{t_1, t_2, t_3\}$

MADS illustration with $n = 2$: Poll step

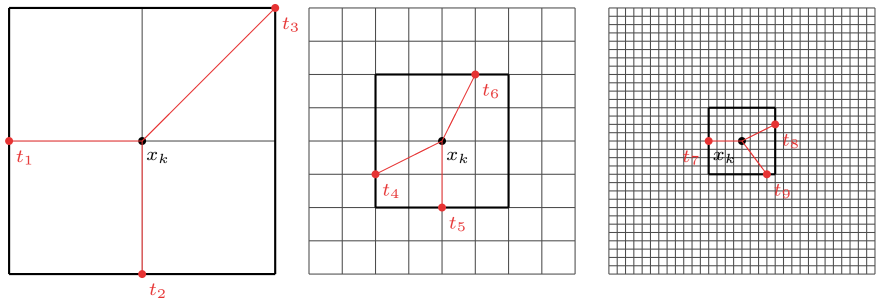
$$\Delta_k^m = \Delta_k^p = 1 \qquad \Delta_{k+1}^m = 1/4$$
$$\Delta_{k+1}^p = 1/2$$



poll trial points$=\{t_1, t_2, t_3\}$ $\qquad = \{t_4, t_5, t_6\}$

MADS illustration with $n = 2$: Poll step

$\Delta_k^m = \Delta_k^p = 1$ $\Delta_{k+1}^m = 1/4$ $\Delta_{k+1}^p = 1/2$ $\Delta_{k+2}^m = 1/16$ $\Delta_{k+2}^p = 1/4$



poll trial points$=\{t_1, t_2, t_3\}$ $= \{t_4, t_5, t_6\}$ $= \{t_7, t_8, t_9\}$

```
[0] Initializations  ($x_0$, $\Delta_0$ )
[1] Iteration $k$
        [1.1] Search
                select a finite number of mesh points
                evaluate candidates opportunistically
        [1.2] Poll (if Search failed)
                construct poll set $P_k = \{x_k + \Delta_k d : d \in D_k\}$
                sort($P_k$)
                evaluate candidates opportunistically
[2] Updates
        if success
                $x_{k+1} \leftarrow$ success point
                increase $\Delta_k$
        else
                $x_{k+1} \leftarrow x_k$
                decrease $\Delta_k$
        $k \leftarrow k + 1$, stop or go to [1]
```

The MADS algorithm [Audet and Dennis, Jr., 2006]

Special features of MADS

- Constraints handling with the Progressive Barrier technique [Audet and Dennis, Jr., 2009]
- Surrogates [Talgorn et al., 2015]
- Categorical variables [Abramson, 2004]
- Granular and discrete variables [Audet et al., 2019]
- Global optimization [Audet et al., 2008a]
- Parallelism [Le Digabel et al., 2010, Audet et al., 2008b]
- Multiobjective optimization [Audet et al., 2008c]
- Sensitivity analysis [Audet et al., 2012]
- Handling of stochastic blackboxes [Alarie et al., 2021, Audet et al., 2021a]

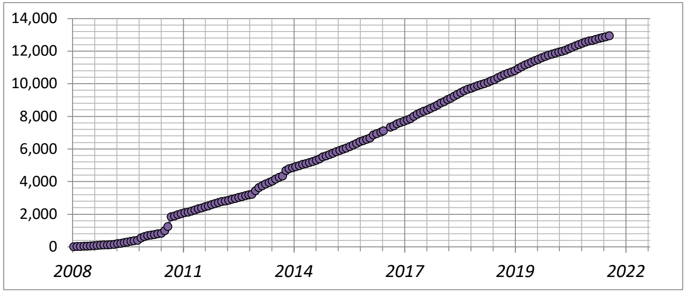
NOMAD (Nonlinear Optimization with MADS)

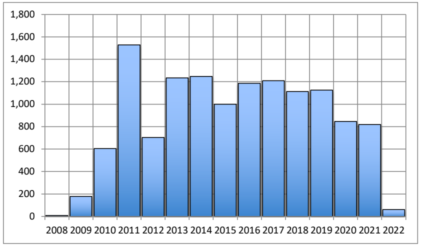
- C++ implementation of the MADS algorithm [Audet and Dennis, Jr., 2006]
- Standard C++. Runs on Linux, Mac OS X and Windows
- Parallel versions with MPI
- MATLAB versions; Multiple interfaces (Python, Excel, etc.)
- Open and free – LGPL license
- Download at `https://www.gerad.ca/nomad`
- Support at `nomad@gerad.ca`
- Related article in TOMS [Le Digabel, 2011] (WoS Highly Cited Paper), and [Audet et al., 2021b]

NOMAD: History and team

- Developed since 2000
- Current versions: 3.9 (June 2018) and 4.2.0 (Feb. 2022)
- Algorithm designers, developers:
  - M. Abramson, C. Audet, G. Couture, J. Dennis, S. Le Digabel, V. Rochon-Montplaisir, C. Tribes
- Developers:
  - Versions 1 and 2: G. Couture
  - **Version 3 (2008)**: S. Le Digabel, C. Tribes
  - **Version 4 (2021)**: V. Rochon-Montplaisir, C. Tribes

$\simeq$13,000 certified downloads since 2008





Main functionalities (1/2)

- Single or biobjective optimization
- Variables:
  - Continuous, integer, binary, categorical, granular
  - Periodic
  - Fixed
  - Groups of variables
- Searches:
  - Latin-Hypercube
  - Variable Neighborhood Search
  - Nelder-Mead Search
  - Quadratic models
  - Statistical surrogates
  - User search

Main functionalities (2/2)

- Constraints treated with 4 different methods:
  - Progressive Barrier (default)
  - Extreme Barrier
  - Progressive-to-Extreme Barrier
  - Filter method
- Several direction types:
  - Coordinate directions
  - LT-MADS
  - OrthoMADS
  - Hybrid combinations
- Sensitivity analysis

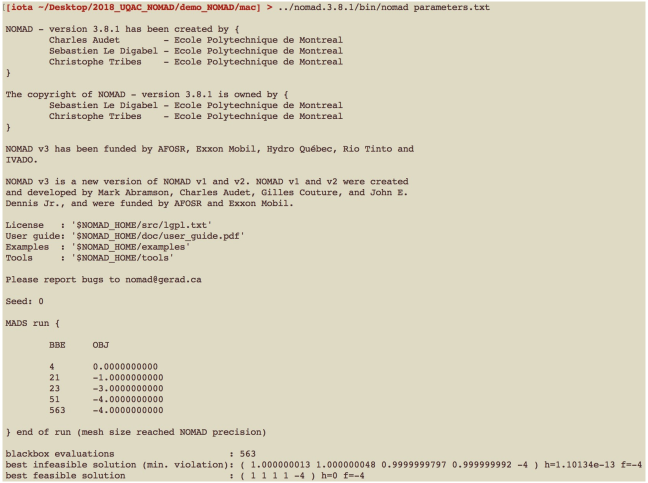
(all items correspond to published or submitted papers)

Blackbox conception (batch mode)

- Command-line program that takes in argument a file containing x, and displays the values of $f(x)$ and the $c_j(x)$'s
- Can be coded in any language
- Typically: `> bb.exe x.txt` displays `f c1 c2` (objective and two constraints)

Run NOMAD

```
> nomad parameters.txt
```

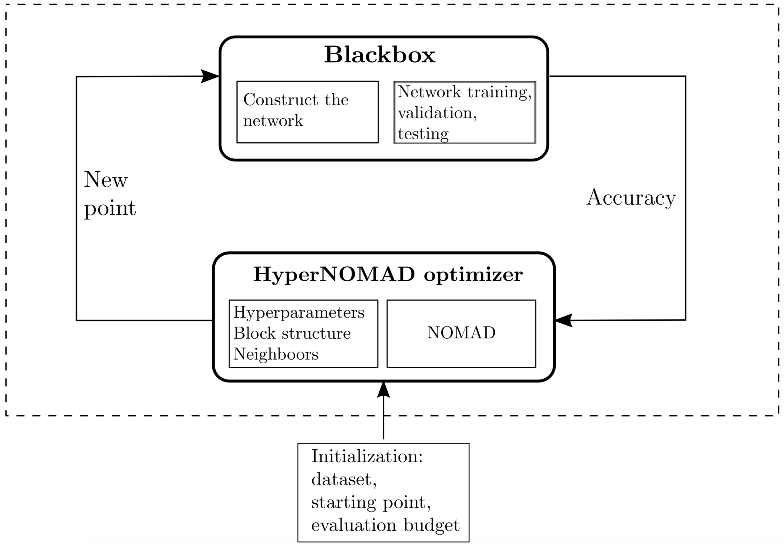


```
[iota ~/Desktop/2018_UQAC_NOMAD/demo_NOMAD/mac] > ../nomad.3.8.1/bin/nomad parameters.txt

NOMAD - version 3.8.1 has been created by {
        Charles Audet        - Ecole Polytechnique de Montreal
        Sebastien Le Digabel - Ecole Polytechnique de Montreal
        Christophe Tribes    - Ecole Polytechnique de Montreal
}

The copyright of NOMAD - version 3.8.1 is owned by {
        Sebastien Le Digabel - Ecole Polytechnique de Montreal
        Christophe Tribes    - Ecole Polytechnique de Montreal
}

NOMAD v3 has been funded by AFOSR, Exxon Mobil, Hydro Québec, Rio Tinto and
IVADO.

NOMAD v3 is a new version of NOMAD v1 and v2. NOMAD v1 and v2 were created
and developed by Mark Abramson, Charles Audet, Gilles Couture, and John E.
Dennis Jr., and were funded by AFOSR and Exxon Mobil.

License   : '$NOMAD_HOME/src/lgpl.txt'
User guide: '$NOMAD_HOME/doc/user_guide.pdf'
Examples  : '$NOMAD_HOME/examples'
Tools     : '$NOMAD_HOME/tools'

Please report bugs to nomad@gerad.ca

Seed: 0

MADS run {

        BBE     OBJ

        4       0.0000000000
        21      -1.0000000000
        23      -3.0000000000
        51      -4.0000000000
        563     -4.0000000000

} end of run (mesh size reached NOMAD precision)

blackbox evaluations                    : 563
best infeasible solution (min. violation): ( 1.000000013 1.000000048 0.9999999797 0.999999992 -4 ) h=1.10134e-13 f=-4
best feasible solution                  : ( 1 1 1 1 -4 ) h=0 f=-4
```

HPO with HyperNOMAD

- PhD project of Dounia Lakhmiri
- Published in TOMS [Lakhmiri et al., 2021]
- We focus on the HPO of deep neural networks
- Our advantages:
    - Blackbox optimization problem:
      One blackbox call = Training + validation + test, for a fixed set of hyperparameters
    - Presence of categorical variables *(ex.: number of layers)*
    - Existing methods are mostly heuristics
      (grid search, random search, GAs, etc.)
- Based on the NOMAD implementation of MADS

Principle


Blackbox
Construct the network
Network training, validation, testing
New point
Accuracy
HyperNOMAD optimizer
Hyperparameters
Block structure
Neighboors
NOMAD
Initialization: dataset, starting point, evaluation budget

HyperNOMAD

- HyperNOMAD is the interface between NOMAD and a deep learning platform
- Based on the PyTorch library
- Works with preexisting datasets such as MNIST or CIFAR-10, or on custom data
- Available at `https://github.com/bbopt/HyperNOMAD`
- We consider three types of hyperparameters:
    - Architecture of the neural network
    - Optimizer
    - Plus one for the size of mini-batches

Hyperparameters for the architecture $(5n_1 + n_2 + 4)$

Hyperparameter	**Type**	**Scope**
Number of convolutional layers (n_1)	Categorical	[0;20]
Number of output channels	Integer	[0;50]
Kernel size	Integer	[0;10]
Stride	Integer	[1;3]
Padding	Integer	[0;2]
Do a pooling	Boolean	0 or 1
Number of full layers (n_2)	Categorical	[0;30]
Size of the full layer	Integer	[0;500]
Dropout rate	Real	[0;1]
Activation function	Categorical	ReLU, Sigmoid, Tanh

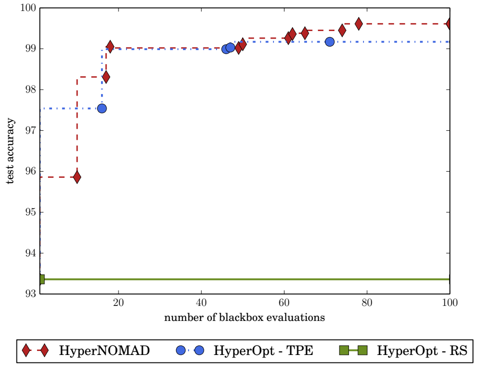
Hyperparameters for the optimizer (5)

Optimizer	Hyperparameter	Type	Scope
Stochastic Gradient Descent (SGD)	Learning rate	Real	[0;1]
	Momentum	Real	[0;1]
	Dampening	Real	[0;1]
	Weight decay	Real	[0;1]
Adam	Learning rate	Real	[0;1]
	β_1	Real	[0;1]
	β_2	Real	[0;1]
	Weight decay	Real	[0;1]
Adagrad	Learning rate	Real	[0;1]
	Learning rate decay	Real	[0;1]
	Initial accumulator	Real	[0;1]
	Weight decay	Real	[0;1]
RMSProp	Learning rate	Real	[0;1]
	Momentum	Real	[0;1]
	α	Real	[0;1]
	Weight decay	Real	[0;1]

Average results on MNIST

Algorithm	Avg accuracy on validation set	Avg accuracy on test set
Rand. search [Bergstra and Bengio, 2012]	94.02	89.07
SMAC [Hutter et al., 2011]	95.48	97.54
RBFOpt [Diaz et al., 2017]	95.66	97.93
NOMAD	**96.81**	**97.98**

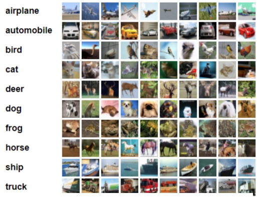
MNIST results with HyperNOMAD

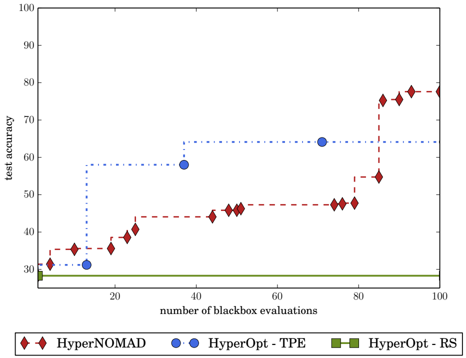


Comparison between HyperNOMAD, TPE and RS when launched from the default starting point of HyperNOMAD, on the MNIST data set. Best solution with HyperNOMAD: 99.61%

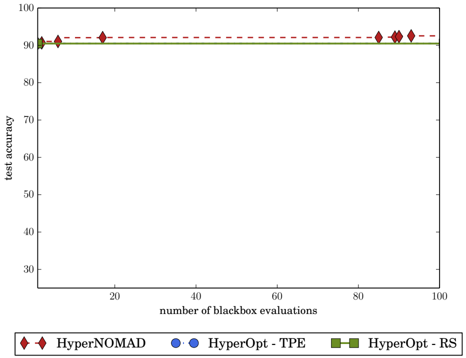
Results on CIFAR-10 (vs Hyperopt)

- Training with 40,000 images, validation/test on 10,000 images
- One evaluation (training+test) $\simeq$ 2 hours (i7-6700@3.4 GHz, GeForce GTX 1070)





(a) Default starting point



(b) From a VGG architecture

Summary

- Blackbox optimization motivated by industrial applications
- Algorithmic features backed by mathematical convergence analyses and published in optimization journals
- NOMAD: Software package implementing MADS
- Open source; LGPL license
- Features: Constraints, biobjective, global optimization, surrogates, several types of variables, parallelism
- HyperNOMAD: Library for the HPO problem
- Fast support at nomad@gerad.ca
- NOMAD has become a baseline for benchmarking DFO algorithms

References I

Abramson, M. (2004).
Mixed Variable Optimization of a Load-Bearing Thermal Insulation System Using a Filter Pattern Search Algorithm.
Optimization and Engineering, 5(2):157–177.

Alarie, S., Audet, C., Bouchet, P.-Y., and Le Digabel, S. (2021).
Optimisation of stochastic blackboxes with adaptive precision.
SIAM Journal on Optimization, 31(4):3127–3156.

Audet, C., Béchard, V., and Le Digabel, S. (2008a).
Nonsmooth optimization through Mesh Adaptive Direct Search and Variable Neighborhood Search.
Journal of Global Optimization, 41(2):299–318.

Audet, C. and Dennis, Jr., J. (2006).
Mesh Adaptive Direct Search Algorithms for Constrained Optimization.
SIAM Journal on Optimization, 17(1):188–217.

Audet, C. and Dennis, Jr., J. (2009).
A Progressive Barrier for Derivative-Free Nonlinear Programming.
SIAM Journal on Optimization, 20(1):445–472.

Audet, C., Dennis, Jr., J., and Le Digabel, S. (2008b).
Parallel Space Decomposition of the Mesh Adaptive Direct Search Algorithm.
SIAM Journal on Optimization, 19(3):1150–1170.

Audet, C., Dennis, Jr., J., and Le Digabel, S. (2012).
Trade-off studies in blackbox optimization.
Optimization Methods and Software, 27(4–5):613–624.

References II

Audet, C., Dzahini, K., Kokkolaras, M., and Le Digabel, S. (2021a).
Stochastic mesh adaptive direct search for blackbox optimization using probabilistic estimates.
Computational Optimization and Applications, 79(1):1–34.

Audet, C. and Hare, W. (2017).
Derivative-Free and Blackbox Optimization.
Springer Series in Operations Research and Financial Engineering. Springer, Cham, Switzerland.

Audet, C., Le Digabel, S., Rochon Montplaisir, V., and Tribes, C. (2021b).
NOMAD version 4: Nonlinear optimization with the MADS algorithm.
Technical Report G-2021-23, Les cahiers du GERAD.

Audet, C., Le Digabel, S., and Tribes, C. (2019).
The Mesh Adaptive Direct Search Algorithm for Granular and Discrete Variables.
SIAM Journal on Optimization, 29(2):1164–1189.

Audet, C., Savard, G., and Zghal, W. (2008c).
Multiobjective Optimization Through a Series of Single-Objective Formulations.
SIAM Journal on Optimization, 19(1):188–210.

Bergstra, J. and Bengio, Y. (2012).
Random search for hyper-parameter optimization.
Journal of Machine Learning Research, 13:281–305.

Diaz, G., Fokoue, A., Nannicini, G., and Samulowitz, H. (2017).
An effective algorithm for hyperparameter optimization of neural networks.
IBM Journal of Research and Development, 61(4):9:1–9:11.

References III

Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2011).
Sequential model-based optimization for general algorithm configuration.
In *International Conference on Learning and Intelligent Optimization*, pages 507–523. Springer.

Lakhmiri, D., Le Digabel, S., and Tribes, C. (2021).
HyperNOMAD: Hyperparameter Optimization of Deep Neural Networks Using Mesh Adaptive Direct Search.
ACM Transactions on Mathematical Software, 47(3).

Le Digabel, S. (2011).
Algorithm 909: NOMAD: Nonlinear Optimization with the MADS algorithm.
ACM Transactions on Mathematical Software, 37(4):44:1–44:15.

Le Digabel, S., Abramson, M., Audet, C., and Dennis, Jr., J. (2010).
Parallel Versions of the MADS Algorithm for Black-Box Optimization.
In *Optimization days*, Montréal.
Slides available at `https://www.gerad.ca/Sebastien.Le.Digabel/talks/2010_JOPT_25mins.pdf`.

Talgorn, B., Le Digabel, S., and Kokkolaras, M. (2015).
Statistical Surrogate Formulations for Simulation-Based Design Optimization.
Journal of Mechanical Design, 137(2):021405–1–021405–18.