Blackbox optimization with MADS and NOMAD: Application to Hyperparameters Optimization (HPO)

Sébastien Le Digabel

GERAD
GROUPE D'ÉTUDES ET DE RECHERCHE EN ANALYSE DES DÉCISIONS

POLYTECHNIQUE MONTRÉAL
TECHNOLOGICAL UNIVERSITY

IREQ, 2020-10-14

Contributors and partners

CRSNG NSERC

InnovÉÉ Innovation en énergie électrique

IVADO

Hydro Québec

RioTinto

HUAWEI

Presentation outline

Introduction

The MADS algorithm and the NOMAD software package

Hyperparameters optimization

Conclusion and references

Introduction

The MADS algorithm and the NOMAD software package

Hyperparameters optimization

Conclusion and references

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)

$x \in \mathbb{R}^n$ →

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

→ $f(x)$, $x \in \Omega$?

- 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

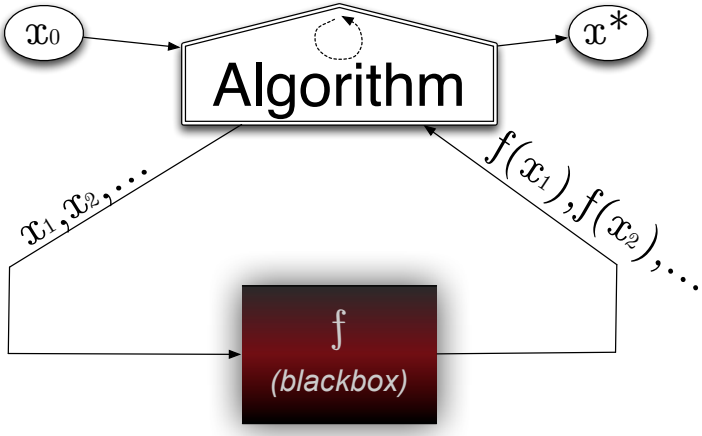
Introduction

The MADS algorithm and the NOMAD software package

Hyperparameters optimization

Conclusion and references

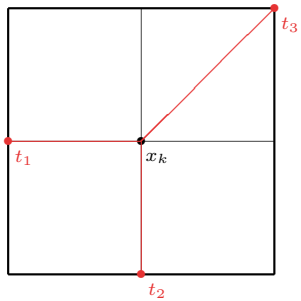
Typical setting



Unconstrained case, with one initial starting solution

MADS illustration with $n = 2$: Poll step

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



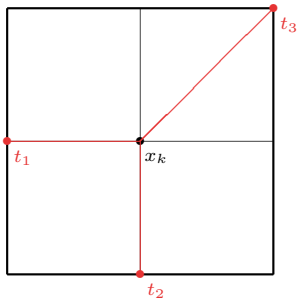
poll trial points$=\{t_1, t_2, t_3\}$

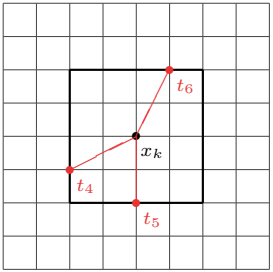
MADS illustration with $n = 2$: Poll step

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

$\Delta_{k+1}^m = 1/4$

$\Delta_{k+1}^p = 1/2$

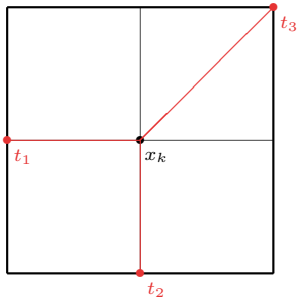




poll trial points$=\{t_1, t_2, t_3\}$ $= \{t_4, t_5, t_6\}$

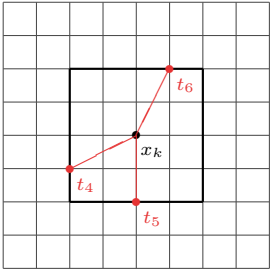
MADS illustration with $n = 2$: Poll step

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



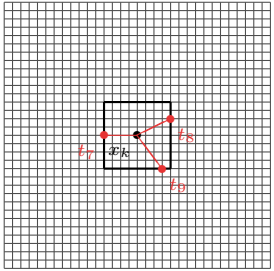
$\Delta_{k+1}^m = 1/4$

$\Delta_{k+1}^p = 1/2$



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

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



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

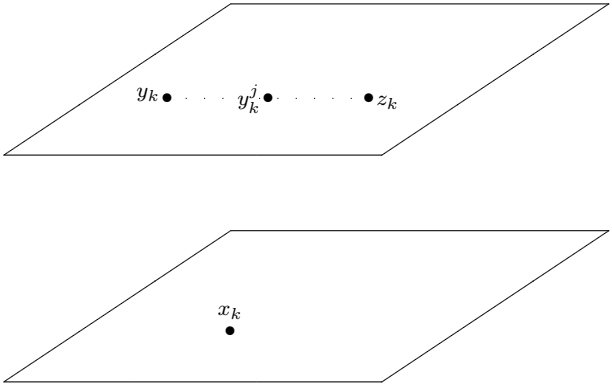
Special features of MADS

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

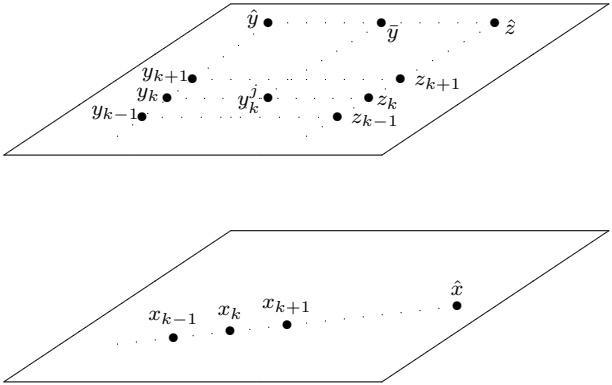
Types of variables in MADS

- MADS has been initially designed for continuous variables
- Some theory exists for categorical variables [Audet and Dennis, Jr., 2001, Abramson, 2004, Abramson et al., 2009]
- (Other discrete variables now considered in MADS: Integer, binary, granular [Audet et al., 2019b])
- Two kinds of "categorical" variables:
  - Non-orderable and unrelaxable discrete variables
  - An integer whose value changes the number of variables of the problem
- "Periodic" variables

MADS with categorical variables: Extended poll



MADS with categorical variables: Extended poll



NOMAD (Nonlinear Optimization with MADS)

- C++ implementation of the MADS algorithm
- Standard C++. Runs on Linux, Mac OS X and Windows
- Current version: 3.9 (June 2018); NOMAD v4 beta, 2020
- Parallel versions with MPI
- Multiple interfaces: Python, Julia, MATLAB, R, Excel
- Open and free – LGPL license
- Download at https://www.gerad.ca/nomad
- Support at nomad@gerad.ca
- Related article in ACM TOMS [Le Digabel, 2011] (WoS Highly Cited Paper)

Introduction

The MADS algorithm and the NOMAD software package

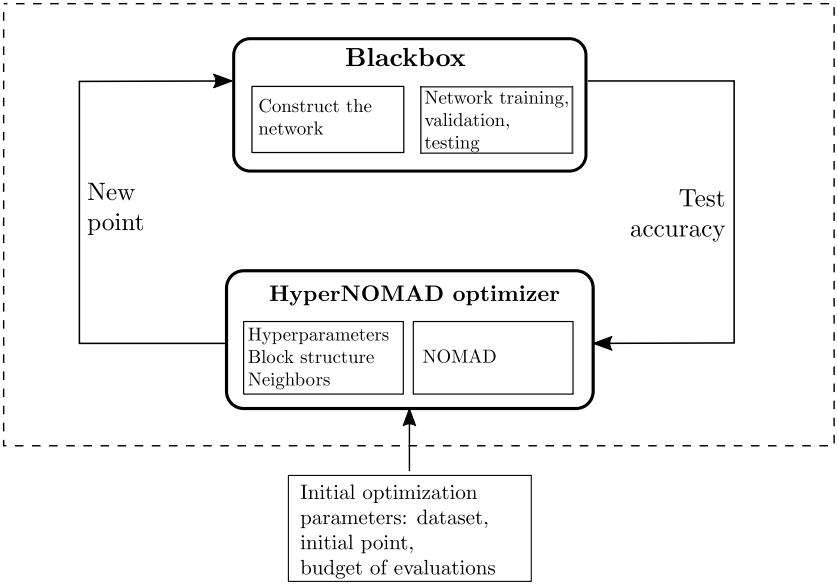
Hyperparameters optimization

Conclusion and references

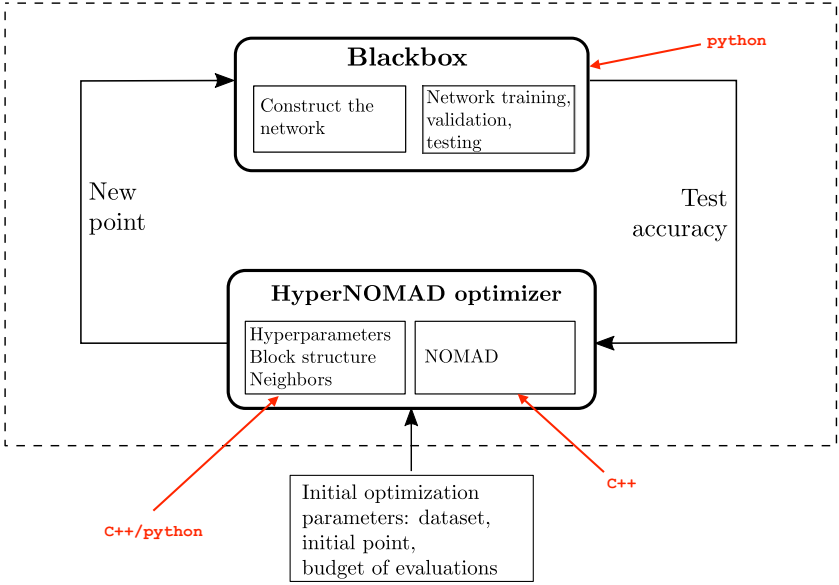
HPO with HyperNOMAD

- PhD project of Dounia Lakhmiri
- Technical report [Lakhmiri et al., 2019] in revision for ACM TOMS
- We focus on the HPO of convolutional 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
Test accuracy
HyperNOMAD optimizer
Hyperparameters
Block structure
Neighbors
NOMAD
Initial optimization parameters: dataset, initial point, budget of evaluations

Principle


Blackbox
python
Construct the network
Network training, validation, testing
New point
Test accuracy
HyperNOMAD optimizer
Hyperparameters
Block structure
Neighbors
NOMAD
C++/python
C++
Initial optimization parameters: dataset, initial point, budget of evaluations

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 two types of hyperparameters:
  - Architecture of the neural network
  - Optimizer

  (plus one hyperparameter for the size of mini-batches)

Network architecture

A convolutional neural network is a deep neural network consisting of a succession of convolutional layers followed by fully connected layers:

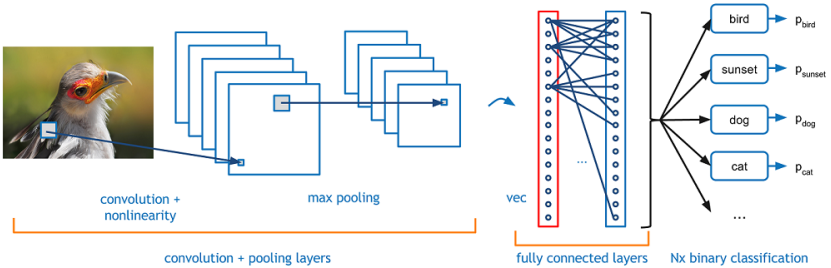


Image from [Deshpande, 2019]

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

Hyperparameter	Type	Scope
Number of convolutional layers (n_1)	Categorical	{0, 1, . . . , 20}
Number of output channels	Integer	{0, 1, . . . , 50}
Kernel size	Integer	{0, 1, . . . , 10}
Stride	Integer	{1, 2, 3}
Padding	Integer	{0, 1, 2}
Pooling size	Integer	{1, 2, . . . , 5}
Number of full layers (n_2)	Categorical	{0, 1, . . . , 30}
Size of the full layer	Integer	{0, 1, . . . , 500}
Dropout rate	Real	[0;1]
Activation function	Categorical	{ReLU, Sigmoid, Tanh}

Hyperparameters for the optimizer (5 parameters)

Optimizer	Hyperparameter	Type	Scope
SGD	η: Initial learning rate	Real	[0;1]
	μ : Momentum	Real	[0;1]
	δ: Dampening	Real	[0;1]
	λ: Weight decay	Real	[0;1]
Adagrad	η: Initial learning rate	Real	[0;1]
	Real	[0;1]	
	ϵ: Smoothing constant	Real	[0;1]
	λ: Weight decay	Real	[0;1]
RMSProp	η: Initial learning rate	Real	[0;1]
	μ: Momentum	Real	[0;1]
	γ: forgetting factor	Real	[0;1]
	λ: Weight decay	Real	[0;1]
Adam	η: Initial learning rate	Real	[0;1]
	Real	[0;1]	
	β_2	Real	[0;1]
	λ: Weight decay	Real	[0;1]

Additional hyperparameter for the training phase (1)

- The data is separated in 3 groups: Training, validation, testing
- During training, the network is fed with subsets (mini-batches) of the data
- Then it performs a forward pass, computes the prediction error and do a back-propagation in order to update the weights using the optimizer
- Size of the mini-batches: Integer hyperparameter that varies in $\{1, 2, \ldots, n_{train}\}$ where n_{train} is the size of the training data

Epochs

- When the network has been fed all of the training data, it is said to have performed an epoch
- Several epochs are needed to obtain good weights
- The number of epochs is not considered as a hyperparameter
- We consider a maximum value (typically 500), but we can interrupt the training if we detect that the current set of hyperparameters is not promising

Blocks of hyperparameters

- Convolution block: 2 convolutional layers with (number of output channels, kernel size, stride, padding, pooling) $= (16, 5, 1, 1, 0)$ and $(7, 3, 1, 1, 1)$:

2	16	5	1	1	0	7	3	1	1	1

- Fully connected block: 3 fully connected layers with sizes of output = 1200, 512, 20:

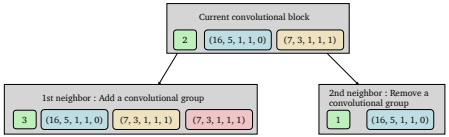
3	1200	512	20

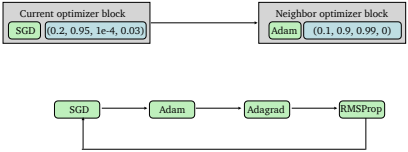
- Optimizer block: SGD with learning rate = 0.1, momentum = 0.9, dampening = 1e-4, and weight decay = 0:

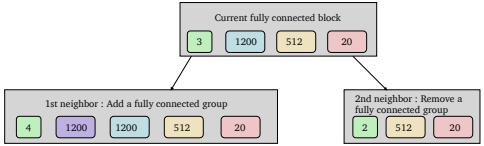
1	0.1	0.9	1e-4	0

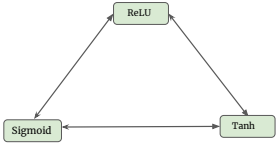
Blocks and neighbors

Illustration of how HyperNOMAD plays with neighbors of blocks

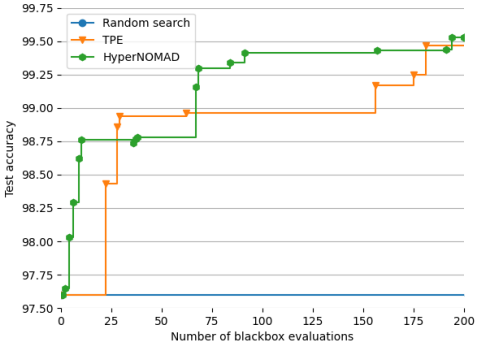


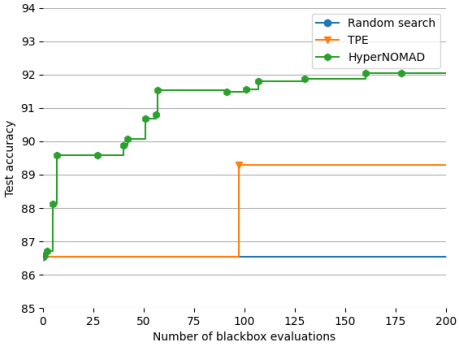






Results on MNIST (left) and Fashion-MNIST (right)

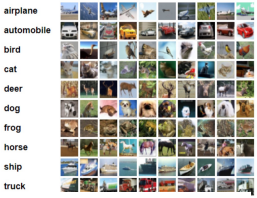


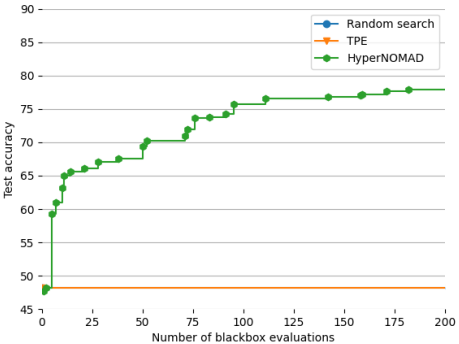


Comparison between HyperNOMAD, TPE and RS (Hyperopt) when launched from the default starting point of HyperNOMAD

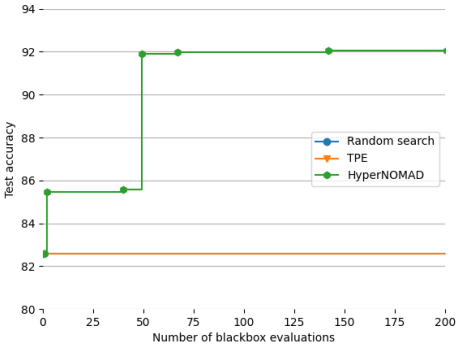
Results on CIFAR-10 (vs Hyperopt)

- 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

Perpectives for future work and collaboration

- Algorithmic improvements:
  - Dynamic handling of the number of epochs ($\rightarrow$ stochastic blackbox)
  - Other search space (as in EfficientNet)
  - New ways to handle categorical variables (MSc of Edward Hallé-Hannan)
- Applications:
  - Expand the library to other types of problems than classification, provide interfaces to other libraries
  - Other contexts: For example *Tuning a variational autoencoder for data accountability problem in the Mars Science Laboratory ground data system* [Lakhmiri et al., 2020]
  - Integration with Oríon and the bayesmark framework
  - Larger datasets such as ImageNet (need GPU server)
  - Applications at Hydro-Québec

Discussion

- HyperNOMAD: Library for the HPO problem
- Specialized for convolutional deep neural networks via the PyTorch library
- Key aspect: Optimize both the architecture and the optimization phase of a deep neural network
- Based on the blackbox optimization solver NOMAD and its ability to model categorical variables
- So far: Competitive results with state-of-the-art on the MNIST and CIFAR-10 datasets

Introduction

The MADS algorithm and the NOMAD software package

Hyperparameters optimization

Conclusion and references

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.

Abramson, M., Audet, C., Chrissis, J., and Walston, J. (2009).
Mesh Adaptive Direct Search Algorithms for Mixed Variable Optimization.
Optimization Letters, 3(1):35–47.

Alarie, S., Audet, C., Bouchet, P.-Y., and Digabel, S. L. (2019).
Optimization of noisy blackboxes with adaptive precision.
Technical Report G-2019-84, Les cahiers du GERAD.

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. (2001).
Pattern search algorithms for mixed variable programming.
SIAM Journal on Optimization, 11(3):573–594.

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.

References II

Audet, C., Dzahini, K., Kokkolaras, M., and Digabel, S. L. (2019a).
StoMADS: Stochastic blackbox optimization using probabilistic estimates.
Technical Report G-2019-30, Les cahiers du GERAD.

Audet, C., Le Digabel, S., and Tribes, C. (2019b).
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., Yamins, D., and Cox, D. (2013).
Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures.
In *Proceedings of the 30th International Conference on International Conference on Machine Learning*, volume 28 of *ICML'13*, pages I–115–I–123. JMLR.org.

Deshpande, A. (2019).
A Beginner's Guide To Understanding Convolutional Neural Networks.
https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks.

Lakhmiri, D., Alimo, R., and Digabel, S. L. (2020).
Tuning a variational autoencoder for data accountability problem in the Mars Science Laboratory ground data system.
Technical Report G-2020-31, Les cahiers du GERAD.

References III

Lakhmiri, D., Digabel, S. L., and Tribes, C. (2019).
HyperNOMAD: Hyperparameter optimization of deep neural networks using mesh adaptive direct search.
Technical Report G-2019-46, Les cahiers du GERAD.

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*, Montreal. GERAD.
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.