Les Cahiers du GERAD ISSN: 0711–2440

Binary, unrelaxable and hidden constraints in blackbox optimization

C. Audet,
G. Caporossi, S. Jacquet

G–2019–76

October 2019

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2019
– Bibliothèque et Archives Canada, 2019

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2019
– Library and Archives Canada, 2019

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Binary, unrelaxable and hidden constraints in blackbox optimization

Charles Audet [a,b]
Gilles Caporossi [a,c]
Stéphane Jacquet [a,d]

[a] *GERAD, Montréal (Québec), Canada, H3T 2A7*

[b] *Department of Mathematics and Industrial Engineering, Polytechnique Montréal (Québec) Canada, H3C 3A7*

[c] *Department of Decision Sciences, HEC Montréal, Montréal (Québec), Canada, H3T 2A7*

[d] *Département d'informatique et de mathématique, UQAC, Chicoutimi (Québec), Canada, G7H 2B1*

charles.audet@gerad.ca
gilles.caporossi@gerad.ca
stephane.jacquet@gerad.ca

October 2019
Les Cahiers du GERAD
G–2019–76

Abstract: This work proposes strategies to handle three types of constraints in the context of blackbox optimization: binary constraints that simply indicate if they are satisfied or not; unrelaxable constraints that are required to be satisfied to trust the output of the blackbox; hidden constraints that are not explicitly known by the user but are triggered unexpectedly. Using tools from classification theory, we build surrogate models of those constraints to guide the MADS algorithm. Numerical results are conducted on three engineer problems.

Keywords: Blackbox optimization, derivative-free optimization, surrogate functions, progressive barrier, k-nearest-neighbors

Acknowledgments: Work of the first author was supported by NSERC Discovery Grant #2015–05311.

1 Introduction

Derivative-free optimization (DFO) studies problems for which there no access to the derivatives of the functions defining the problem, even if they may exist. However, the structure can be known and exploited. Blackbox optimization is a part of DFO where there is no known structure of the functions used in the optimization problem definition. Those optimization problems can happen when output of the blackbox is given from a laboratory experiment or a computer simulation.

Consider the following constrained blackbox optimization problem

$$\begin{cases} \min\limits_{x \in X} & f(x) \\ \text{subject to} & b_i(x) \leq 0, \, \forall i \in [\![1, q]\!] \\ & c_i(x) \leq 0, \, \forall i \in [\![1, m]\!] \end{cases} \tag{1}$$

where for all $i \in [\![1; q]\!]$, $b_i : \mathbb{R}^n \mapsto \{0; 1\}$, for all $i \in [\![1; m]\!]$, $c_i : \mathbb{R}^n \mapsto \mathbb{R}$, $f : \mathbb{R}^n \mapsto \mathbb{R}$ and $X \subseteq \mathbb{R}^n$. The output of the blackbox from an input $x \in \mathbb{R}^n$ is composed of all the $c_i(x)$, $b_i(x)$ and $f(x)$. The set

$$\Omega = \{ \, x \in X \; : \; \forall i \in [\![1, q]\!], b_i(x) \leq 0; \; \forall i \in [\![1, m]\!], c_i(x) \leq 0 \, \}$$

denotes the feasible region.

This optimization problem contains different kinds of constraints and we will use the terminology from [12] to describe them. First of all, the constraints $c_i(x) \leq 0$ and $b_i(x) \leq 0$ are "relaxable". That means that if a point violates one of these constraints, the user does not need to lose the trust in the other functions involved in the problem. Secondly, the constraint $x \in X$ contains two types of constraints: "unrelaxable" and "hidden" constraints. If an unrelaxable constraint is violated, then the user cannot trust the rest of the output. A point that violates an unrelaxable constraint cannot be exploited as well as a boint that violates only unrelaxable constraints. Hidden contraints, first defined in [8], are constraints that are not explicitly described by the user. It can occur for example when a negative value is used in a log from one of the functions involved. If a point violates a hidden constraint, the blackbox fails to return an output for that point. The article [16] offers a treatment of hidden constraints by modifying the "DIRECT" algorithm. It uses sub-dividing steps in case the center of a hyper-rectangle violates a hidden constraint. All unrelaxable and hidden constraints are defined in the set X.

The Mads algorithm [6] was designed to solve such blackbox optimization problems. This algorithm creates a discretization of the input space $\mathbb{R}^n$ called a "mesh". In Mads, each type of constraints is handled in its own way. Especially, the points that violate an unrelaxable or a hidden constraint are rejected. Despite the binary constraints, as defined in the optimization problem being relaxable, Mads handles them as if they were unrelaxable or hidden. The main objective of this work it to propose alternative ways to handle those three types of constraints.

The paper is structured as follows. Section 2 describes how Mads handles constraints. Section 3 describes how surrogates of binary, unrelaxable and hidden constraints can be calculated and used within Mads. Section 4 shows numerical results to see the gain of this new approach. Section 5 concludes the work of the article and suggests future work.

2 Opportunistic strategy and constraints management in Mads

Mads is an iterative direct search optimization algorithm, that generates trial points on a discretization of the space of variables called the mesh. The objective and constraint functions are sequentially evaluated at these trial points. These evaluations belong to two possible steps. (i) The first one is the "poll step". It is a local search in order to find local optimum. The convergence analysis of Mads relies on this step [6]. (ii) The second possible step is the "search step" and is optional. It can be any methods the user wants to suggest new points on the mesh for evaluation in order to find a global optimum. It can be, for example, a "Variable Neighborhood Search" [2] or a Nelder-Mead [7] search

step. At each step, MADS suggests a list of candidate points $\mathcal{L}^k$ that will be given to the blackbox for evaluation. However, terminating the step as soon as a new incumbent is found may reduce the overall computational time.

This section is explains some features of MADS. It includes how MADS sorts the elements of $\mathcal{L}^k$ with surrogate functions and how MADS handles the constraints.

2.1 Ordering strategy and surrogate functions

MADS uses the opportunistic strategy to reduce the number of evaluations. It can be used both at the poll and the search step. At iteration $k \in \mathbb{N}$, the opportunistic strategy consists of stopping the evaluation of the elements of $\mathcal{L}^k$ as soon as an element of $\mathcal{L}_k$ is shown to be feasible and improves the current best objective function value. Ordering $\mathcal{L}_k$ with the objective to find such element as soon as possible is called using an "ordering strategy". If this element is found early, then the remaining elements of $\mathcal{L}_k$ will not be evaluated and the budget of evaluation will be then preserved. The work from Sarrazin-Mc Cann [17] shows the importance of the opportunistic strategy and ordering strategies in DFO.

To order $\mathcal{L}_k$, surrogates from the objective function and the constraints can be used. There are static surrogates which can be simplified versions of the problem given by the user. Dynamic surrogates are surrogates that depend on the elements already evaluated through the solving process. For example, it can be quadratic models [9]. This work proposes a surrogate functions for binary, unrelaxable and hidden constraints. Those surrogates will give an approximation of the objective function and of the constraints. Thus they will give an idea of which elements of $\mathcal{L}_k$ have the more chances to be feasible and to improve the objective function value. The next sections describes the different ways MADS handles constraints and how it uses surrogates.

2.2 Constraints management

The first way to handle constraints is through the so-called extreme barrier approach [6]. Instead of minimizing f with all the constraints, a function $f_\Omega(x)$ is defined such that all infeasible points have an infinite value through $f_\Omega(x)$:

$$f_\Omega(x) = \begin{cases} f(x) & \text{if } x \in \Omega \\ +\infty & \text{if } x \notin \Omega. \end{cases}$$

The optimization is then conducted on the unconstrained minimization of the barrier function f_Ω.

Since the extreme barrier rejects all the infeasible point, the progressive barrier was introduced in [4] to bring more flexibility. It uses the constraint violation function [10] which quantifies how much the constraints are violated.

Definition 2.1 *The constraint violation function h is defined as:*

$$h(x) = \begin{cases} \sum_{i=1}^{m} \max(0, c_i(x))^2 & \text{if } x \in X \\ +\infty & \text{otherwise.} \end{cases}$$

The errors are squared so that no additional non-smoothness is introduced. The progressive barrier uses both f and h to choose the next list of points to evaluate $\mathcal{L}^k$. All the details are given in [4]. This work considers surrogates of binary, unrelaxable and hidden constraints. It has an impact on the ordering strategy, thus on the order in which elements will be evaluated. This has no impact on the convergence analysis from [4].

The function h being defined, if surrogate functions of the constraint functions c_i, $i \in [\![1; m]\!]$ are available, then a surrogate of h, noted $\tilde{h}$, can be defined. This surrogate, in addition to a surrogate of f, allows the following ordering strategy: let us define $x, y \in \mathbb{R}^n$, then x will be evaluated before y if $(\tilde{f}(x) \leq \tilde{f}(y)$ and $\tilde{h}(x) < \tilde{h}(y))$ or if $(\tilde{f}(x) < \tilde{f}(y)$ and $\tilde{h}(x) \leq \tilde{h}(y))$.

3 Treatment of binary, unrelaxable and hidden constraints

Consider the optimization problem (1) from in which the set X contains all the non-relaxable constraints but also the hidden constraints. The constraints, for all $i \in [\![1; q]\!]$, $b_i \leq 0$ and, for all $i \in [\![1; m]\!]$, $c_i \leq 0$ are relaxable constraints.

Unrelaxable, hidden and binary constraints $b_i(x) \leq 0$ are treated with the extreme barrier by MADS. In this work, the unrelaxable constraint $x \in X$ is treated as a binary constraint, and all binary constraints will have surrogates $\tilde{b}_i$. Those $\tilde{b}_i$ will then be used to construct a surrogate constraint violation function $\tilde{h}$. Since the unrelaxable constraints and hidden constraints are considered as one binary constraint.

3.1 Different formulation of the optimization problem

All unrelaxable constraints and hidden constraints are combined into a single binary constraint. By construction, that new constraint is necessarily unquantifiable, as described in [12]. The constraint is either satisfied or it is not, without being able to give further information. Thus, the fusion of those constraints in a binary constraint is meaningful. In order to do that, $x \in X$ will be substituted by the binary constraint $b_0(x) \leq 0$ where $b_0 : \mathbb{R}^n \mapsto \{0; 1\}$ with:

$$b_0(x) = \begin{cases} 0 & \text{if } x \in X \\ 1 & \text{if } x \notin X. \end{cases}$$

Problem (1) may be reformulated as:

$$\begin{cases} \min\limits_{x \in \mathbb{R}^n} & f(x) \\ \text{subject to} & b_i(x) \leq 0, \, \forall i \in [\![0, q]\!] \\ & c_i(x) \leq 0, \, \forall i \in [\![1, m]\!]. \end{cases} \tag{2}$$

The progressive barrier allows a more flexible way to handle the constraints than the extreme barrier. The progressive barrier uses the constraint violation function, but only for the relaxable constraints. So the formula remains unchanged despite the appearance of the new constraint $b_0(x) \leq 0$. For all $x \in \mathbb{R}^n$,

$$h(x) = \begin{cases} \sum\limits_{i=1}^{q} \max(0, b_i(x))^2 + \sum\limits_{i=1}^{m} \max(0, c_i(x))^2 & \text{if } b_0(x) \leq 0 \\ +\infty & \text{otherwise.} \end{cases}$$

which can be re-written even more simply by using properties of binary functions.

$$h(x) = \begin{cases} \sum\limits_{i=1}^{q} b_i(x) + \sum\limits_{i=1}^{m} \max(0, c_i(x))^2 & \text{if } b_0(x) \leq 0 \\ +\infty & \text{otherwise.} \end{cases}$$

Different surrogate functions of h will be suggested. They are defined, for all $x \in \mathbb{R}^n$, by

$$\tilde{h}_1(x) = \sum_{i=0}^{q} \max\left(0, \tilde{b}_i(x)\right)^2 + \sum_{i=1}^{m} \max\left(0, \tilde{c}_i(x)\right)^2; \tag{2}$$

$$\tilde{h}_2(x) = \sum_{i=0}^{q} \max\left(\tilde{b}_0(x), \tilde{b}_i(x)\right)^2 + \sum_{i=1}^{m} \max\left(0, \tilde{c}_i(x)\right)^2; \tag{3}$$

$$\tilde{h}_3(x) = \left(1 + \tilde{b}_0(x)\right) \left(\sum_{i=1}^{q} \max(0, \tilde{b}_i(x))^2 + \sum_{i=1}^{m} \max(0, \tilde{c}_i(x))^2 \right). \tag{4}$$

The function $\tilde{h}_1$ is the natural prolongation of the default version of $\tilde{h}$ while taking into account the binary constraints with their surrogates of binary constraints $\tilde{b}_i$, $i \in [\![1;q]\!]$ and taking into account the unrelaxable and the hidden constraints with $\tilde{b}_0$.

The function $\tilde{h}_2$ is very similar to Equation 2 by replacing to $\max(0,\tilde{b}_i(x))^2$ by $\max(\tilde{b}_0,\tilde{b}_i(x))^2$. This attempts to favor points who have a chance to satisfy the constraint b_0.

The function $\tilde{h}_3$ is the natural prolongation of the default version of $\tilde{h}$ by adding a multiplicative penalization for the b_0 constraint.

The constraint functions b_i and c_i are not necessarily taking values of the same magnitude. The binary constraints only take the values 0 and 1. However, the other constraints can take a wide range of values. If a function c_i takes much higher values than 1, then it will penalize a lot more in $\tilde{h}_1$, $\tilde{h}_2$, $\tilde{h}_3$ than any binary constraints. The technical report [3] suggests definitions of other surrogates of h. Those surrogates will use scalings of the output.

Using the expressions of the surrogates of h given by $\tilde{h}_1$, $\tilde{h}_2$ and $\tilde{h}_3$, other definitions of surrogates of h can be given using scaling.

$$\tilde{h}_4(x) = \sum_{i=0}^{q} \max\left(0,\tilde{b}_i(x)\right)^2 + \sum_{i=1}^{m} \max\left(0,\frac{\tilde{c}_i(x)}{a_i^k}\right)^2; \tag{5}$$

$$\tilde{h}_5(x) = \sum_{i=0}^{q} \max\left(\tilde{b}_0(x),\tilde{b}_i(x)\right)^2 + \sum_{i=1}^{m} \max\left(0,\frac{\tilde{c}_i(x)}{a_i^k}\right)^2; \tag{6}$$

$$\tilde{h}_6(x) = \left(1+\tilde{b}_0(x)\right)\left(\sum_{i=1}^{q} \max\left(0,\tilde{b}_i(x)\right)^2 + \sum_{i=1}^{m} \max\left(0,\frac{\tilde{c}_i(x)}{a_i^k}\right)^2\right). \tag{7}$$

Since the constraints are being scaled, $\tilde{h}_5$ can be adapted to compare $\tilde{b}_0$ with all $\tilde{c}_i$.

$$\tilde{h}_7(x) = \sum_{i=0}^{q} \max\left(\tilde{b}_0(x),\tilde{b}_i(x)\right)^2 + \sum_{i=1}^{m} \max\left(\tilde{b}_0(x),\frac{\tilde{c}_i(x)}{a_i^k}\right)^2. \tag{8}$$

At the beginning of iteration $k \in \mathbb{N}$, the coefficient a_i^k, $i \in [\![1;m]\!]$ will be equal to the highest violation of the constraint c_i as suggested by the conclusions of [3]:

$$a_i^k = \begin{cases} 1 & \text{if } \{c_i(x) : x \in V^k, c_i(x) > 0\} = \emptyset \\ \max\limits_{x \in V^k} c_i(x) & \text{otherwise} \end{cases}$$

where V^k is the list of all the points from $\mathbb{R}^n$ that have been evaluated up to the beginning of iteration k.

With $\tilde{f}$ and $\tilde{h}$, the ordering strategy sorts the points according to the following rule: a trial point $x \in \mathcal{L}_k$ is given to the blackbox before $y \in \mathcal{L}_k$ if and only if $\tilde{f}(x) \leq \tilde{f}(y)$ and $\tilde{h}(x) < \tilde{h}(y)$ or $\tilde{f}(x) < \tilde{f}(y)$ and $\tilde{h}(x) \leq \tilde{h}(y)$.

3.2 Surrogates with weighted k-nearest-neighbors (WKNN)

The surrogates of the constraint functions b_i, $i \in [\![0;q]\!]$ are computed using the weighted k-nearest-neighbors algorithm. It relies on doing a weighted average of the classes of the k nearest neighbors from x^0 [11]. This weighted average will give the desired estimation.

$$\tilde{b}(x^0) = \frac{1}{k}\sum_{i=1}^{k} \frac{w_i b(x^i)}{\sum\limits_{j=1}^{k} w_j}.$$

The weights w_i can be determined with a non-negative kernel $K : \mathbb{R}^n \mapsto (\mathbb{R}_+)^n$:

$$\tilde{b}(x^0) = \frac{1}{k} \sum_{i=1}^{k} \frac{K\left(\frac{||x^0 - x^i||}{\lambda}\right) b(x^i)}{\sum_{j=1}^{k} K\left(\frac{||x^0 - x^j||}{\lambda}\right)}.$$

where $\lambda > 0$ is called the "bandwidth". The kernel that will be used when making surrogates for binary, unrelaxable and hidden constraints is the Epanechnikov kernel because of its optimal properties [13]. The general expression of the Epanechnikov kernel is [19]:

$$K(u) = \begin{cases} \frac{3}{4}(1-u^2) & \text{if } u \in [-1,1] \\ 0 & \text{otherwise.} \end{cases}$$

4 Numerical results

In this section, numerical results will be generated using 30 instances from each of the three following blackboxes:

- The "Styrene" problem [2] contains four binary constraints and hidden constraints. The hidden constraints are important here since approximately 14% of the evaluated points fail to satisfy one of the hidden constraints [5]. There are also seven relaxable constraints.
- The original formulation of the "MDO" problem [18] has no binary, hidden nor unrelaxable constraints. To observe the behaviour of the algorithm with $\tilde{h}$ considering binary, hidden and unrelaxable constraints, the first constraint of MDO is transformed into a binary constraint, while the second constraint is considered as an unrelaxable constraint. The last constraint is unchanged and is the only relaxable constraint here.
- Just like for MDO, "Lockwood" [14] has its first constraint replaced by a binary constraint and the second changed to an unrelaxable constraint. The two other constraints are unchanged and are the only relaxable constraints.

Tests are done with Nomad 3.8.0 using OrthoMads to generate the directions [1], quadratic models [9] are activated to get the surrogates $\tilde{f}$ and $\tilde{c}_i$. The budget of evaluations is set to 1500. The substitute functions $\tilde{b}_i$, $i \in [\![1; q]\!]$ using WKNN with the Epanechnikov kernel and a number of neighbors equals to $\left\lceil \sqrt{|\mathcal{V}^k|} \right\rceil$ where $k \in \mathbb{N}$ is the number of the iteration and $\mathcal{V}^k \subset \mathbb{R}^n$ the elements evaluated at the beginning of iteration k. The bandwidth λ is chosen to be equal to the distance to the $\left\lceil \sqrt{|\mathcal{V}^k|} \right\rceil$-th neighbor. To compare those different versions, data profiles [15] are produced. Those rely on the following convergence test

$$f(x_0) - f(x) \leq (1 - \tau)(f(x_0) - f_L), \tag{9}$$

where $x_0 \in \mathbb{R}^n$ is a feasible starting point, f_L is the best value found by all the compared versions, given a budget of evaluation, and $\tau > 0$ is the wished precision of the test of convergence. The abscissa of data profiles show the number of evaluations divided by $n + 1$. The ordinate shows the ratio of problems that satisfy the test of convergence given at precision τ The first tests compare the default version using the extreme barrier for binary, unrelaxable and hidden constraints, and the others versions using $\tilde{h}_1$, $\tilde{h}_2$ and $\tilde{h}_3$.

Figure 1 shows that each of the three different ways to calculate $\tilde{h}$ (with $\tilde{h}_1$, $\tilde{h}_2$ and $\tilde{h}_3$) solves more problems than the default method of calculating $\tilde{h}$. Despite the fact there are few differences between those three ways, it seems that the treatment with the function h_1 leads to better results.

Figure 2 shows that, once again, using surrogate functions of binary, unrelaxable and hidden constraints gives better solutions according to the data profiles at precisions $\tau = 10^{-2}$ and $\tau = 10^{-4}$. $\tilde{h}_4$, $\tilde{h}_5$, $\tilde{h}_6$ and $\tilde{h}_7$ all dominate the default version of Nomad 3.8.0 at those precisions. However, in that

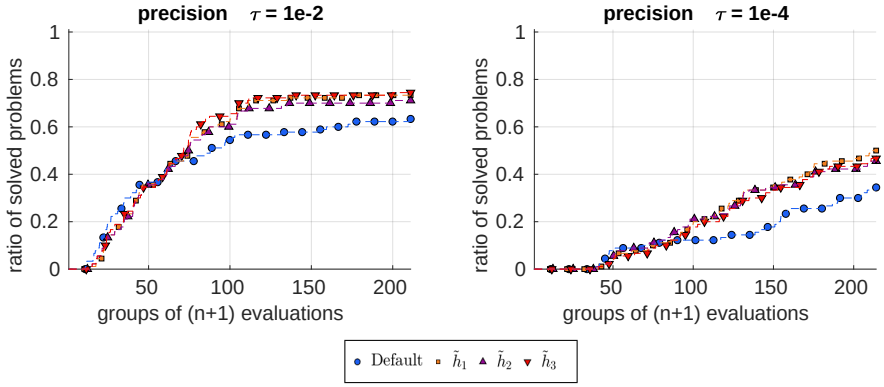

Figure 1: Data profiles with $\tau = 10^{-2}$ and $\tau = 10^{-4}$ for blackboxes with surrogates that do not scale the constraints.

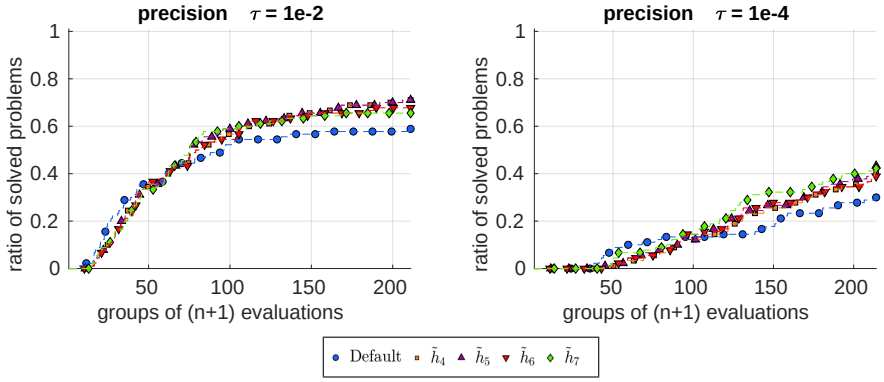

Figure 2: Data profiles with $\tau = 10^{-2}$ and $\tau = 10^{-4}$ for blackboxes with surrogates that scale the constraints.

situation, weighting did not bring much improvements. An explanation could be that those blackboxes are already well balanced.

The next results compare the best definition of $\tilde{h}$ without scaling and with scaling, which are $\tilde{h}_1$ and $\tilde{h}_7$. This is done in order to offer the reader what version should be preferred. Figure 3 shows two data profiles with $\tilde{h}_1$ and $\tilde{h}_7$ in the same conditions as previously. Those profiles show only small differences between those two variations. However, $\tilde{h}_7$, with the squares, dominates at precision $\tau = 10^{-2}$ and seem to dominate at precision $\tau = 10^{-4}$ after 180 groups of $n + 1$ evaluations.

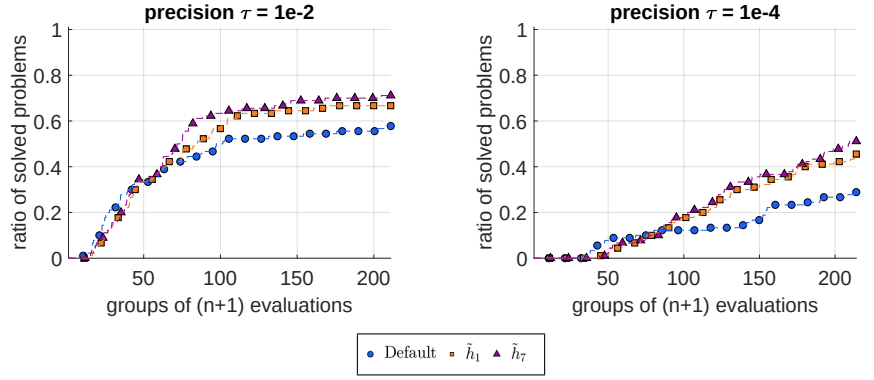

Figure 3: Data profiles with $\tau = 10^{-2}$ and $\tau = 10^{-4}$ for blackboxes comparing the default surrogate and the best surroagte with and without scaling.

The numerical results show that the recommended surrogate of the constraint violation function is given by $\tilde{h}_7$. Overall, that means that using surrogates for the binary, the unrelaxable and the hidden constraints give on average better solutions and that scaling is recommended.

5 Discussion

Prior to this work, binary, unrelaxable and hidden constraints were handled by MADS by the extreme barrier. This work offers different ways to deal with them. First, binary constraints are modeled through k-nearest-neighbor in order to have an estimation of the probability to satisfy the constraint. This estimation is used through $\tilde{h}$, the surrogate of the constraints violation function. The unrelaxable and hidden constraints are considered as binary constraints. Thus, they take advantage of WKNN to have a surrogate of them in $\tilde{h}$. It provides a way to use the information given by those constraints. It is also suggested that the constraints could be suffering from scaling issues. This is why weighting of the constraints using the maximum violation is used. Numerical results showed that making surrogates with WKNN of those constraints was useful compared to the extreme barrier and that the differences between all the versions were minimal. The weighting did bring little improvement, even if it is not significant. It is therefore recommenced to use surrogates function of those functions and to use weightings within $\tilde{h}$.

There is also another use for the surrogate of the binary relaxable constraints. In the constraint violation function, the true value of the function is used. With a binary relaxable constraint, only the values 0 and 1 would be used. However, the surrogate function gave an estimation of the probability to not satisfy the constraint. So, it gives the kind of information interesting for the constraint violation function. If done so, true values and surrogate values would cohabit in the constraint violation function.

References

[1] Abramson, M. A., Audet, C., Dennis, Jr., J. E., Le Digabel, S., Jan. 2009. Orthomads: A deterministic mads instance with orthogonal directions. SIAM Journal on Optimization 20(2), 948–966.

[2] Audet, C., Béchard, V., Le Digabel, S., Jun. 2008. Nonsmooth optimization through mesh adaptive direct search and variable neighborhood search. Journal of Global Optimization 41(2), 299–318. http://dx.doi.org/10.1007/s10898-007-9234-1

[3] Audet, C., Caporossi, G., Jacquet, S., september 2019. Constraint scaling in the Mesh Adaptative Direct Search algorithm. Tech. rep., Les Cahiers du GERAD G–2019–65, GERAD, HEC Montrál, Canada.

[4] Audet, C., Dennis, J., 2009. A progressive barrier for derivative-free nonlinear programming. SIAM Journal on Optimization 20(1), 445–472. https://doi.org/10.1137/070692662

[5] Audet, C., Dennis, J. E., Le Digabel, S., Jun 2010. Globalization strategies for mesh adaptive direct search. Computational Optimization and Applications 46(2), 193–215. https://doi.org/10.1007/s10589-009-9266-1

[6] Audet, C., Dennis, Jr., J. E., Jan. 2006. Mesh adaptive direct search algorithms for constrained optimization. SIAM Journal on Optimization 17(1), 188–217.

[7] Audet, C., Tribes, C., June 2018. Mesh-based Nelder–Mead algorithm for inequality constrained optimization. Computational Optimization and Applications. https://doi.org/10.1007/s10589-018-0016-0

[8] Choi, T. D., Eslinger, O. J., Kelley, C. T., David, J. W., Etheridge, M., Jun 2000. Optimization of automotive valve train components with implicit filtering. Optimization and Engineering 1(1), 9–27. https://doi.org/10.1023/A:1010071821464

[9] Conn, A., Le Digabel, S., 2013. Use of quadratic models with mesh-adaptive direct search for constrained black box optimization. Optimization Methods and Software 28(1), 139–158. http://dx.doi.org/10.1080/10556788.2011.623162

[10] Fletcher, R., Leyffer, S., Jan 2002. Nonlinear programming without a penalty function. Mathematical Programming 91(2), 239–269. https://doi.org/10.1007/s101070100244

[11] Hastie, T., Tibshirani, R., Friedman, J., 2009. The elements of statistical learning: data mining, inference and prediction, 2nd Edition. Springer. http://www-stat.stanford.edu/~tibs/ElemStatLearn/

[12] Le Digabel, S., Wild, S., 2015. A Taxonomy of Constraints in Simulation-Based Optimization. Tech. Rep. G–2015–57, Les cahiers du GERAD. http://www.optimization-online.org/DB_HTML/2015/05/4931.html

[13] Ledel, T., 2004. Kernel density estimation: Theory and application in discriminant analysis. Austrian Journal of Statistics 33(2), 267–279.

[14] Matott, L. S., Leung, K., Sim, J., 2011. Application of MATLAB and python optimizers to two case studies involving groundwater flow and contaminant transport modeling. Computers & Geosciences 37(11), 1894–1899.

[15] Moré, J. J., Wild, S. M., 2009. Benchmarking derivative-free optimization algorithms. SIAM Journal on Optimization 20(1), 172–191. https://doi.org/10.1137/080724083

[16] Na, J., Lim, Y., Han, C., 2017. A modified direct algorithm for hidden constraints in an lng process optimization. Energy 126, 488–500. http://www.sciencedirect.com/science/article/pii/S0360544217304164

[17] Sarrazin-Mc Cann, L. A., mai 2018. Opportunisme et ordonnancement en optimisation sans dérivées. Master's thesis, École Polytechnique de Montréal. https://publications.polymtl.ca/3099/

[18] Tribes, C., Dubé, J.-F., Trépanier, J.-Y., 2005. Decomposition of multidisciplinary optimization problems: formulations and application to a simplified wing design. Engineering Optimization 37(8), 775–796. http://dx.doi.org/10.1080/03052150500289305

[19] Zambom, A. Z., Dias, R., 2013. A review of kernel density estimation with applications to econometrics. International Econometric Review (IER) 5(1), 20–42. http://EconPapers.repec.org/RePEc:erh:journl:v:5:y:2013:i:1:p:20-42