Les Cahiers du GERAD

G-2014-16
March 2014

CITATION ORIGINALE / ORIGINAL CITATION

Audet, C., Le Digabel, S., Tribes, C., Dynamic scaling in the mesh adaptive direct search algorithm for blackbox optimization, Optimization and Engineering, 17(2), 333-358, 2016.

Les Cahiers du GERAD ISSN: 0711–2440

Dynamic Scaling in the Mesh Adaptive Direct Search Algorithm for Blackbox Optimization

C. Audet, S. Le Digabel,
C. Tribes

G–2014–16

March 2014

Dynamic Scaling in the Mesh Adaptive Direct Search Algorithm for Blackbox Optimization

Charles Audet
Sébastien Le Digabel
Christophe Tribes

GERAD & Polytechnique Montréal
C.P. 6079, Succ. Centre-ville
Montréal (Québec) Canada, H3C 3A7

charles.audet@gerad.ca
sebastien.le.digabel@gerad.ca
christophe.tribes@polymtl.ca

March 2014

Les Cahiers du GERAD
G–2014–16

Abstract: Blackbox optimization deals with situations in which the objective function and constraints are typically computed by launching a time-consuming computer simulation. The subject of this work is the Mesh Adaptive Direct Search (MADS) class of algorithms for blackbox optimization. We propose a way to dynamically scale the mesh, which is the discrete spatial structure on which MADS relies, so that it automatically adapts to the characteristics of the problem to solve. Another objective of the paper is to revisit the MADS method in order to ease its presentation and to reflect recent developments. This new presentation includes a non smooth convergence analysis. Finally, numerical tests are conducted to illustrate the efficiency of the dynamic scaling, both on academic test problems and on a supersonic business jet design problem.

Key Words: Blackbox optimization, Derivative-Free Optimization, Mesh Adaptive Direct Search, Dynamic Scaling.

Acknowledgments: Work of the first author was supported by NSERC grant 239436. The second author was supported by NSERC grant 418250. The first and second authors are supported by AFOSR FA9550-12-1-0198.

1 Introduction

Scaling variables can greatly affect the performance of an optimization algorithm [14, 15, 16, 22, 23]. The Mesh Adaptive Direct Search (MADS [6]) algorithm for constrained blackbox optimization is no exception. With the MADS default form, an initial scaling of the space of variables can be proposed explicitly by the user for each variable, or is obtained implicitly by considering the bounds on each variable and the magnitude of the components of the initial point.

The initial scaling is then used to construct a discretization of the space of variables, called the *mesh*. MADS algorithms are iterative, and each iteration attempts to improve the current best solution by generating trial points on the mesh. Unlike most meshes required to obtain numerical solutions of partial differential equations on complex physical domains (e.g., [20]), the mesh used by the MADS algorithm at a given iteration is simply defined by orthogonal coordinate lines that intersect at some points. For the Mads algorithm, the mesh is a convenient support for the set of all potential trial points (feasible or not) that can be evaluated in $\mathbb{R}^n$ at a given level of granularity, n being the dimension of the space of variables. The granularity of the mesh is updated at the end of each iteration. The coarseness or fineness of all subsequent meshes is dictated by a single strictly positive real-valued parameter called the *mesh size parameter*. Hence, with the MADS default form, the relative mesh aspect ratios remain invariant as the algorithm progresses.

In the event of a successful iteration, i.e., when a trial point turns out to be a new incumbent solution, then the mesh size parameter is increased and the search for a better solution is restarted. This allows the next iteration to generate trial points further away from this new incumbent solution. Alternatively, if none of the trial points generated during an iteration improves the incumbent, then the mesh size parameter is reduced. The set of trial points form some sort of discrete neighbourhood around the incumbent solution. Failing to improve the current solution implies that the incumbent solution is better than its neighbours, which is a crude sort of local optimality tied to the mesh size parameter.

The convergence analysis of the MADS algorithm studies convergent subsequences of incumbent solutions that were shown to be better than their neighbours, and for which the mesh size parameter goes to zero. These are called refining subsequences. At the limit of such a subsequence, the crude optimality conditions become the standard local optimality conditions. If the problem is sufficiently smooth, these conditions correspond to *(i)* a null gradient in the unconstrained case, and *(ii)* nonnegative directional derivatives in the directions tangent to the domain in the constrained case. In the nonsmooth case, these conditions state that *(iii)* zero belongs to the generalized gradient in the unconstrained case, and *(iv)* nonnegative directional generalized derivatives in the directions hypertangent to the domain in the constrained case. The details of this hierarchical convergence analysis based on decreasing requirements on the objective and constraints are found in [6].

These theoretical results hold as the number of function evaluations goes to infinity. In practice however, the number of function calls is always limited, often by a given time-related budget. Furthermore, if the scaling of the variables is inappropriate, the progress made by the algorithm within a budget of evaluations might be marginal.

The contribution of the present work is to propose a way to improve the MADS algorithm by considering a mesh size parameter specific to each variable. This allows to dynamically adapt the mesh in an anisotropic way: the relative mesh aspect ratios are allowed to vary. The main idea of the proposed contribution is that when a new incumbent solution is found, instead of coarsening the mesh for all variables using the same ratio, we coarsen it only with respect to the variables that were significantly modified in the new solution. Specifically, this is done by replacing the classical mesh size parameter in $\mathbb{R}_+$ that dictates the coarseness of the mesh, by a *mesh size vector* in $\mathbb{R}^n_+$, whose dimension equals the number of optimization variables. Since the entire convergence analysis of this type of methods is based on the mesh size parameter rather than its new vectorial expression, the convergence analysis needs to be entirely revisited.

The paper is divided as follows. Section 2 presents the MADS class of algorithm with dynamic scaling. The technical part of the nonsmooth convergence analysis is deferred to the appendix section. Section 3 discusses a first implementation of this new class of algorithms. Numerical experiments are conducted in

Section 4 on academic test problems and on a supersonic business jet design problem. Finally, the last section discusses future research directions.

Notation: Vectors can be considered row or column vectors according to context. The superscript $k \in \mathbb{N}$ is not an exponent, but an index associated to the iteration number. The value of a raised to the power b is written with parentheses: $(a)^b$ to avoid confusion with superscripts. If δ is a vector in $\mathbb{R}^n$, then $\operatorname{diag}(\delta)$ is the $n \times n$ diagonal matrix with $\delta \in \mathbb{R}^n$ as its diagonal. The set of m directions $D = \{d_1, d_2, \ldots, d_m\} \subset \mathbb{R}^n$ is said to be a positive spanning set if for any $v \in \mathbb{R}^n$, there exists an $\alpha \in \mathbb{R}^m_+$ such that $v = \sum_{i=1}^m \alpha_i d_i$. The unit sphere in $\mathbb{R}^n$ is denoted $\mathbb{S}^n = \{d \in \mathbb{R}^n : \|d\| = 1\}$.

2 Mads with dynamic scaling

Consider the general optimization problem

$$\min_{x \in \Omega} \quad f(x), \tag{1}$$

where f is a single-valued objective function, and Ω is the set of feasible solutions in $\mathbb{R}^n$. Blackbox optimization targets problems in which properties of the objective function f as well as the functions defining the feasible set are not easily exploitable. This typically occurs when the output of these functions are obtained by launching a computer code with some $x \in \mathbb{R}^n$ as input. These functions are typically nonsmooth, time-consuming to evaluate, contaminated by numerical noise, which renders the application of classical optimization methods difficult. Direct search optimization methods are designed to interact directly with the outputs of the blackbox. An introduction to derivative-free optimization can be found in [11].

In the present work, we focus on an extension of the Mads algorithm which dynamically scales the mesh for each variable independently. This new extension impacts the fundamental building blocks of the algorithm, and therefore we present it from scratch. This is also an opportunity to revisit some concepts and to present some aspects differently in order to ease the exposition.

2.1 High-level description of Mads

Mads is an iterative algorithm designed to solve problems of the form (1). It is launched from an initial point, or more generally from a finite collection of initial points $V^0 \subset \mathbb{R}^n$. The iteration number is represented by the superscript k, and at the start of iteration k, the set of trial points where the blackbox was previously evaluated is denoted by V^k. The current iterate x^k corresponds to the current best feasible point inside V^k and is called the incumbent solution, or the incumbent. It is simply a point of the set $\arg\min\{f(x) : x \in V^k \cap \Omega\}$. In the present paper, constraints are handled with the *Extreme Barrier* approach [6] which consists of launching the algorithm on the unconstrained extended valued function

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

The goal at each iteration of a Mads algorithm is to replace the incumbent by a better one. In order to achieve this, finitely many tentative trial points are generated on a discretization of the space of variables called the mesh M^k. The formal definition of the mesh requires additional notation, and is deferred to the next subsection. Some of these trial points may be far from the incumbent, but a subset of these trial points needs to be distributed nearby the incumbent solution. These neighbouring trial points are called the poll points while the remaining trial points are called the search points. More specifically, these two types of points are generated during two different steps of each iteration, called the search and the poll, also detailed in the next section.

At the end of iteration k, if a better incumbent solution was produced, then the iteration is declared successful and M^{k+1} is updated to be at least as coarse as M^k. The rationale behind this is to allow the next iteration to take larger steps. Alternately, iteration k failed at producing a better incumbent solution, and therefore the incumbent solution is better than its neighbouring poll points. This can be seen as some crude form of local optimality for the incumbent. At the conclusion of a failed iteration, the mesh is refined so that the poll points generated during the next iteration will be closer to the incumbent solution.

2.2 The SEARCH and POLL steps

Let us focus on two important parameters related to the high-level description of the MADS algorithm from the previous subsection. In previous work, the coarseness of the mesh was controlled by a single positive scalar. In the present work, we extend this by defining $\delta_j^k > 0$ as the *mesh size parameter for the j-th variable at iteration k*, for $j = 1, 2, \ldots, n$. Together with the set of trial points visited thus far V^k, the vector $\delta^k \in \mathbb{R}^n_+$, called the *mesh size vector*, allows to define the mesh at iteration k. The mesh is central to the algorithm since it contains every potential trial points:

$$M^k \quad = \quad V^k + \left\{ \operatorname{diag}(\delta^k) z \ : \ z \in \mathbb{Z}^n \right\}. \tag{2}$$

With this definition, it follows that t belongs to the mesh at iteration k if and only if there exists a trial point $v \in V^k$ at which the blackbox was previously evaluated, and an integer vector $z \in \mathbb{Z}^n$ such that $t = v + \operatorname{diag}(\delta^k) z$. Furthermore, by recursively applying this observation, it follows that for any $t \in M^k$, there exists an element x^0 from the set of starting points V^0 such that

$$t = x^0 + \sum_{\ell=0}^{k} \operatorname{diag}(\delta^\ell) z^\ell \tag{3}$$

for some integer vectors $\{z^0, z^1, \ldots, z^k\} \subset \mathbb{Z}^n$.

Iteration k attempts to improve the incumbent solution by performing a pair of steps called the SEARCH and the POLL. The SEARCH can be viewed as the diversification step of the algorithm since it attempts to improve the incumbent by generating finitely many trial points that are not required to be close to the incumbent. For doing this in practice, the user may indicate the trial points by exploiting some specific knowledge of the problem, or by using generic tools such as Latin Hypercube sampling, Variable Neighborhood Search [5], or any other method that he could come up with, as long as there are a finitely many candidates in M^k.

Note that the SEARCH is optional and not necessary for the convergence analysis. When the SEARCH succeeds at improving the incumbent solution, then the POLL is not executed. The POLL step may be viewed as the intensification step of the algorithm as it aims at finding local optima. It uses the second important parameter which delimits the region in which the neighboring poll points are generated. This parameter is $\Delta_j^k \geq \delta_j^k$ and is called the *poll size parameter for the j-th variable at iteration k*, for $j = 1, 2, \ldots, n$. Similarly to the mesh size vector, we denote the vector $\Delta^k \in \mathbb{R}^n_+$ as the *poll size vector*. This vector is used to delimit a region around the current incumbent solution $x^k \in V^k$ where the poll points will be selected. This region is called the *frame* [12] and is defined as follows.

Definition 2.1 *The frame at iteration k is $\{x \in \mathbb{R}^n : |x_j - x_j^k| \leq \Delta_j^k, \ \forall j = 1, 2, \ldots, n\}$.*

The POLL also generates a finite list of trial points, but more rigid rules are used than for the SEARCH. These poll points need to be distributed around the current incumbent solution in such a way that candidates are present in every half-spaces. Formally, the poll points at iteration k around the incumbent solution $x^k \in V^k$ is called the *poll set* and is given by:

$$P^k = \{x^k + \operatorname{diag}(\delta^k) d : d \in D^k\} \subseteq M^k$$

where $D^k \subset \mathbb{Z}^n$ is a positive spanning set of directions. The next subsection describes links between the mesh and poll size vectors, and gives a condition ensuring that any poll trial point $x \in P^k$ belongs to the frame.

2.3 Obtaining the poll size vector from the mesh size vector

At the first iteration, the poll size vector $\Delta^0 \in \mathbb{R}^n_+$ is either supplied by the user, or fixed by some procedure to reflect the scale of the variables. At the end of any iteration, the poll size parameter associated to the j-th variable is updated by multiplying the previous one by an integer power of some fixed real number $\beta_j > 1$, whose square is a rational number (the proof of Proposition A.1 uses this requirement on rationality). The power is greater than or equal to zero on successful iterations, and strictly negative on unsuccessful iterations. This implies that every poll size parameters decrease if and only if the iteration is unsuccessful. In the actual implementation described in Section 3, $\beta_j = 2$ for every j.

The following definition fixes the value of the mesh size parameter in terms of the poll size parameter.

Definition 2.2 *At iteration k, the mesh size parameter associated to the variable $j \in \{1, 2, \ldots, n\}$ is*

$$\delta^k_j = \frac{\left(\min\{\Delta^0_j,\ \Delta^k_j\}\right)^2}{\sqrt{n}\ \Delta^0_j} \ .$$

With this definition, $\delta^k_j \leq \Delta^k_j$ and furthermore, the initial mesh size vector is automatically fixed to $\delta^0 = \frac{\Delta^0}{\sqrt{n}}$. The term $\sqrt{n}$ in the denominator accounts for the fact that distances between objects grows with the dimension n. For example, the diameter of the unit hypercube in $\mathbb{R}^n$ is precisely $\sqrt{n}$.

A consequence of this way of updating the poll size vector is summarized in the next Lemma.

Lemma 2.3 *At iteration $k \geq 1$, and for every index $j \in \{1, 2, \ldots, n\}$, there exists an integer $r^k_j \in \mathbb{Z}^n$ such that the poll and mesh size parameters satisfy*

$$\Delta^k_j = \Delta^0_j(\beta_j)^{r^k_j} \qquad \text{and} \qquad \delta^k_j = \delta^0_j(\beta_j)^{r^k_j - |r^k_j|} \ \leq\ \delta^0_j. \tag{4}$$

Proof. Let j be an index in $\{1, 2, \ldots, n\}$. The result is clearly true at the initial iteration by setting $r^0_j = 0$. At the end of every iteration, the poll size parameter is multiplied by an integer power of β_j. It follows that there exists a sequence of integers r^k_j such that $\Delta^k_j = \Delta^0_j(\beta_j)^{r^k_j}$ for every k. Therefore

$$\begin{aligned} \delta^k_j &= \frac{\left(\min\left\{\Delta^0_j,\ \Delta^0_j(\beta_j)^{r^k_j}\right\}\right)^2}{\sqrt{n}\ \Delta^0_j} \\ &= \delta^0_j\left(\min\left\{1,\ (\beta_j)^{r^k_j}\right\}\right)^2 \\ &= \delta^0_j(\beta_j)^{\min\{0, 2r^k_j\}} \\ &= \delta^0_j(\beta_j)^{r^k_j - |r^k_j|}. \end{aligned}$$

□

The integer r^k_j from the previous Lemma is called the *mesh index*, and the last expression gives the mesh size parameter as a function of the mesh index. One can easily see that for a given value of j, the poll and mesh size parameters go simultaneously to zero:

$$\{\Delta^k_j\}_{k\in K} \to 0 \quad \Leftrightarrow \quad \{r^k_j\}_{k\in K} \to -\infty \quad \Leftrightarrow \quad \{\delta^k_j\}_{k\in K} \to 0.$$

But, since the mesh size parameter is constructed using the square of the poll size parameter, it will converge to zero faster. Furthermore, we impose that the algorithm possesses a mechanism ensuring that if one of the poll size parameters goes to zero, then the norm of the poll size vector also converges to zero:

$$\{\Delta^k_j\}_{k\in K} \to 0 \quad \Leftrightarrow \quad \{\|\Delta^k\|\}_{k\in K} \to 0 \tag{5}$$

for any $j \in \{1, 2, \ldots, n\}$ and subset of indices K. Such a mechanism is described in the implementation of Section 3.

Finally, the following proposition prescribes conditions on the directions used to construct the poll set Pk, ensuring that the poll points are within the frame.

Proposition 2.4 *Let $x \in P^k$ be a poll point at iteration k, and let $d \in D^k \subset \mathbb{Z}^n$ be such that $x = x^k + \mathrm{diag}(\delta^k)d$ where δ^k is the mesh size vector at iteration k. If*

$$|d_j| \quad \leq \quad (\beta_j)^{|r_j^k|} \tag{6}$$

for every $j = 1, 2, \ldots, n$, then x belongs to the poll frame, i.e., $|x_j - x_j^k| \leq \Delta_j^k$.

Proof. If $x = x^k + \mathrm{diag}(\delta^k)d$, then for each $j = 1, 2, \ldots, n$,

$$\begin{array}{rcll} |x - x^k|_j & = & \delta_j^k |d_j| & \\ & \leq & \delta_j^k (\beta_j)^{|r_j^k|} & \text{by (6)} \\ & = & \delta_j^0 (\beta_j)^{r_j^k - |r_j^k|} (\beta_j)^{|r_j^k|} & \text{by Lemma 2.3} \\ & = & \delta_j^0 (\beta_j)^{r_j^k} & \\ & = & \Delta_j^k & \text{by (4)} . \end{array}$$

□

2.4 The modified Mads algorithm

Algorithm 1 summarizes the main steps of the modified Mads method. Similarly to the original Mads, it must be seen as a general family of methods defining the blocks necessary for convergence. Different implementations can be defined, as the one presented in Section 3.

Algorithm 1: Mads with dynamic scaling

```
Input :  $V^0 \neq \emptyset$ the set of initial points in $\mathbb{R}^n$;
         $\Delta^0 \in \mathbb{R}^n_+$ the initial poll size vector;
         $\beta > 1 \in \mathbb{R}^n$ a vector of real numbers whose squares are rational;
for $k = 0, 1, 2, \ldots$ do
    Let $x^k \in \arg\min\{f_\Omega(x) : x \in V^k\}$ be the incumbent solution.
    Compute the mesh size vector $\delta^k$ from $\Delta^k$ using Definition 2.2.
    Perform the SEARCH and possibly the POLL steps (or only part of them if a new incumbent point
    $x^{k+1}$ is found).
       • OPTIONAL SEARCH: Test finitely many trial points in $M^k$.
       • LOCAL POLL: Test trial points in $P^k$.
    PARAMETER UPDATE: Update $\Delta^{k+1}$ through the mesh index $r^{k+1}$ as follows. If the iteration was
    unsuccessful, then set $r_j^{k+1} < r_j^k$ to an integer for each $j = 1, 2, \ldots, n$. Otherwise the iteration was
    successful: the incumbent $x^k$ was dominated by $x^{k+1}$. For each $j = 1, 2, \ldots, n$, set $r_j^{k+1} \geq r_j^k$ to an
    integer.
end
```

There are two important differences with the original Mads method. First there is no more the general direction matrix D (see [6]). Instead, it is implicitly fixed to $D = [-I_n \;\; I_n]$ where I_n is the identity matrix. In practice, it means that the mesh (2) is always based on orthogonal directions. Historically, the flexibility in the choice of the directions defining the mesh was introduced in the context of the *Generalized Pattern Search* [24] algorithm, in which a fixed number of polling directions was used. With Mads and its dense sets of directions, this flexibility has never been exploited in practice simply because it was not necessary. We abandon this requirement on D in order to simplify the presentation.

The second difference constitutes the main contribution of the present work. A generalization of the method is obtained by allowing the increase of selected mesh size parameters in the update step, at the end of a successful iteration. A practical rule for choosing which parameters to increase is presented in Section 3. It consists of allowing the mesh size parameters associated to the variables whose value have changed from iteration k to $k+1$ to increase: For each $j = 1, 2, \dots, n$, set $r_j^{k+1} = r_j^k$ if $x_j^{k+1} = x_j^k$ and set $r_j^{k+1} \geq r_j^k$ to an integer if $x_j^{k+1} \neq x_j^k$. In the case of a failed iteration, the behaviour remains the same as all the mesh size parameters must be decreased. A convergence analysis of MADS with dynamic scaling is presented in the appendix section. The main results are identical to those of MADS [6] but the analysis differs. The details of the convergence analysis are presented in the appendix.

3 Anisotropic MADS, an implementation of MADS with dynamic scaling

The previous section gave a general framework for the MADS algorithm with dynamic scaling. In the present section, we propose a first instantiation of this algorithm.

We make some algorithmic choices to simplify the presentation. For example, we set $\beta_j = 2$ for every index j.

The other details of the implementation are given in the following subsections. They include the way to initialize the poll size vector, how to update it after a failure and after a success, and how to define the poll directions.

3.1 Poll size initialization

The initial components of the poll size vector are a key point for the efficiency of the method. When no dynamic scaling is used, these values simply define the scaling of the problem, which is crucial as every practitioner knows. We must then choose values that reflect the best scaling that we can infer a priori. For that, our best guidelines are the lower and upper bounds on the variables, whose presence is always beneficial anyway with DFO methods. If the bounds are not available, then the initial point is considered. The proposed strategy for each $j = 1, 2, \dots, n$ is to set the initial poll size parameter as follows:

$$\Delta_j^0 = \begin{cases} \frac{u_j - l_j}{10} & \text{if variable } j \text{ is bounded by the finite values } l_j \text{ and } u_j, \\ \frac{|x_j^0 - b_j|}{10} & \text{if variable } j \text{ has only one bound } b_j \text{ which differs from } x_j^0, \\ \frac{|x_j^0|}{10} & \text{if variable } j \text{ has only one bound } b_j \text{ with } x_j^0 = b_j \neq 0, \\ \frac{|x_j^0|}{10} & \text{if variable } j \text{ has no bound and } x_j^0 \neq 0, \\ 1 & \text{otherwise.} \end{cases}$$

3.2 Updating the poll size vector at the end of an iteration

At the end of an unsuccessful iteration k, each direction of a positive spanning set D^k failed to improve an incumbent solution. Decrease each poll size parameter as follows:

$$\Delta_j^{k+1} = \frac{\Delta_j^k}{2},$$

i.e., by setting $r_j^{k+1} = r_j^k - 1$, for each j in $\{1, 2, \dots, n\}$.

At the end of a successful iteration k, a new incumbent solution was generated in a non-zero direction $d \in \mathbb{R}^n$. Algorithm 1 states that for each index $j = 1, 2, \dots, n$ the mesh size parameter can be increased or left to its current value, i.e., $r_j^{k+1} \geq r_j^k$. In the proposed implementation, we start by setting

$$r_j^{k+1} = \begin{cases} r_j^k + 1 & \text{if } |d_j| > \frac{1}{n}\|d\|_\infty, \\ r_j^k & \text{otherwise.} \end{cases}$$

With this way of modifying the parameters, it is possible that some of them became too small relative to the others. Let $\bar{r} = \max\{r_j^k : j = 1, 2, \dots n\}$ be the largest mesh index at iteration k.

Consider each index $j = 1, 2, \dots, n$. If the mesh index r_j^{k+1} is sufficiently negative so that it is less than twice the largest one, then increase it. Formally, we chose to implement the following rule: if $r_j^{k+1} < -2$ and if $r_j^{k+1} < 2\bar{r}$, then reset $r_j^{k+1} \leftarrow r_j^k + 1$.

3.3 Poll and mesh sizes example

Figure 1 illustrates the modifications of the poll and mesh size vectors on an unconstrained problem in $\mathbb{R}^2$ with starting point $x^0 = (10, 10)$. The procedure to compute the initial poll and mesh size vectors produces $\Delta^0 = (1, 1)$ and $\delta^0 = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$ with the mesh index vector $r^0 = (0, 0)$. The intersections of the horizontal and vertical lines represent the trial points on the mesh M^k and the shaded rectangle delimits the frame from Definition 2.1.

Iteration 0 is successful and generates a new incumbent solution $x^1 = (10, 10 + \frac{1}{\sqrt{2}})$. The success direction $d = x^1 - x^0 = (0, \frac{1}{\sqrt{2}})$ has a single nonzero entry, and therefore only one poll size parameter is increased: $\Delta^1 = (1, 2)$. The frame at iteration 1 is deformed, when compared to that of iteration 0. Due to the minimization operator in Definition 2.2, the mesh size parameter δ^1 remains the same as $\delta^0 = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$.

Iteration 1 is unsuccessful and both poll and mesh size parameters are decreased: $\Delta^2 = (\frac{1}{2}, 1)$ and the corresponding mesh size vector is $\delta^2 = (\frac{1}{4\sqrt{2}}, \frac{1}{\sqrt{2}})$. The new incumbent x^2 is equal to $x^1 = (10, 10 + \frac{1}{\sqrt{2}})$.

At iteration 0, the ratio between the two mesh size parameters was 1. At iteration 2, the only prior success was obtained by modifying the second variable. At this point the ratio between the two mesh size parameters is $\frac{1}{4}$. This ratio can be interpreted as a measure of the mesh anisotropy. The mesh is four times coarser with respect to the second variable.

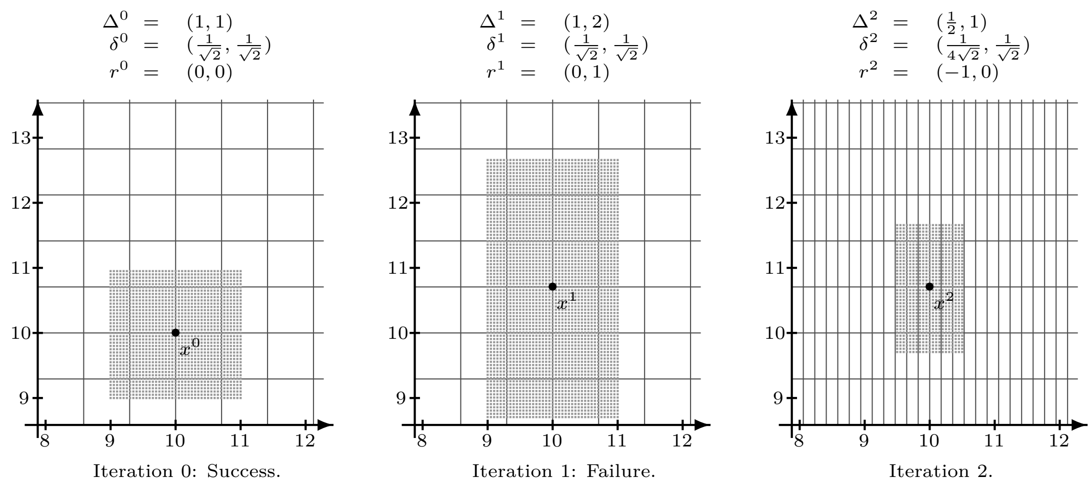

$\Delta^0 = (1,1)$
$\delta^0 = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$
$r^0 = (0,0)$
x^0
Iteration 0: Success.
$\Delta^1 = (1,2)$
$\delta^1 = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$
$r^1 = (0,1)$
x^1
Iteration 1: Failure.
$\Delta^2 = (\frac{1}{2},1)$
$\delta^2 = (\frac{1}{4\sqrt{2}}, \frac{1}{\sqrt{2}})$
$r^2 = (-1,0)$
x^2
Iteration 2.

Figure 1: The incumbent x^k is represented by the dot. M^k is represented by the intersection of the lines. The poll points need to be chosen on M^k within the frame, represented by the shaded box.

3.4 Poll set construction

This subsection describes a way to generate the poll directions. It is inspired from the OrthoMads implementation [1] except that it is no more restrained to the use of the quasi-random Halton sequence [18]. Another distinction is that because the anisotropic nature of the mesh, directions are no longer orthogonal. However, since the Householder orthogonal transformation is still used, we continue to qualify these directions as OrthoMads directions.

Let $\{u^\ell\}_{\ell=0}^{\infty}$ be a set of normalized directions in $\mathbb{R}^n$ such that their union is dense in the unit sphere. One option for this step is to consider random directions, which has been chosen for our numerical tests as detailed in Section 4. Another possibility is to consider the QR-MADS type of directions described in [26].

To ease the presentation, let us consider the iteration k and let v be one of these normalized directions. The Householder matrix

$$H = I - 2vv^T$$

is an orthonormal basis of $\mathbb{R}^n$. By multiplying the j-th row of H by Δ_j^k, we obtain a new basis $H' = \mathrm{diag}(\Delta^k)H$ of $\mathbb{R}^n$ scaled by the poll size parameter values. Next, let us round each entry of the new basis as follows: The i-th entry of the j-th column of the matrix H' can be written as $H'_{ij} = a_{ij} \times \delta_j^k$ for some real number a_{ij}. Let $\alpha_{ij} = \mathrm{round}(a_{ij})$ and set $H''_{ij} = \alpha_{ij} \times \delta_j^k$. Then the set $\{x^k + h : h \in H''\}$ belongs to M^k and the poll set P^k is defined to be $\{x^k \pm h : h \in H''\}$.

All of this can be compactly written as

$$H'' = [H''_{ij}] \text{ where } H''_{ij} = \mathrm{round}\left(\frac{\Delta_j^k \times H_{ij}}{\delta_j^k}\right) \times \delta_j^k,$$

and Figure 2 illustrates the different steps leading to the poll directions. The left part of the figure shows the $2n$ orthogonal directions forming a maximal positive basis. They correspond to the positive and negative columns of the Householder matrix H. The central part of the figure stretches these directions using the poll size vector. It represents the positive and negative columns of the matrix H'. It remains a maximal positive basis, but the directions are not orthogonal. The right part of the figure shows the actual poll points, obtained by rounding to the mesh.

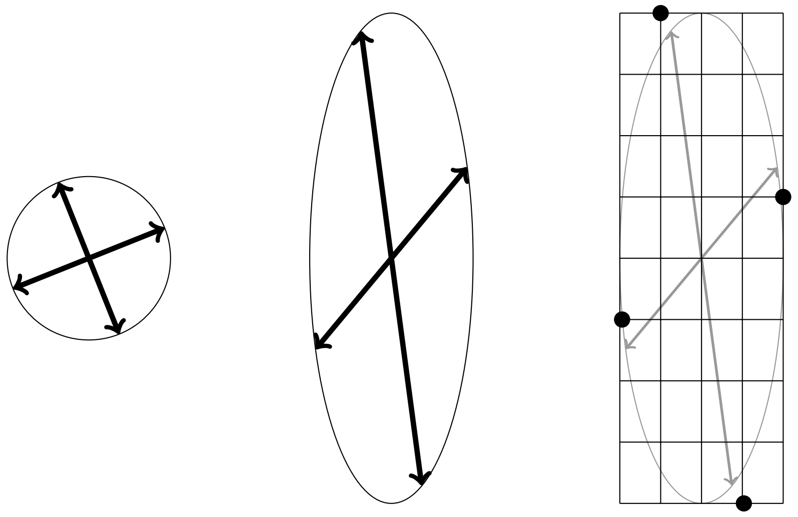


Figure 2: Stretching the set of polling directions and rounding to the mesh. An orthogonal positive basis $[-H\ H]$ is represented to the left. The basis is stretched in the center of the figure to $[-H'\ H']$. The stretched positive basis is rounded to the current mesh, and the resulting directions $[-H''\ H'']$ are represented in the right part of the figure by the dark circles.

4 Numerical results

Numerical tests are conducted using the beta version 3.7.0 of the NOMAD [19] software package freely available at www.gerad.ca/nomad. All tests use the OrthoMads strategy with $2n$ polling directions. The opportunistic strategy is enabled, which consists to sort a given list of poll trial points before it is evaluated, using the last direction of success, and to interrupt the evaluations as soon as a new improvement is made. Although quadratic models are used by default in NOMAD and are almost always beneficial in terms of performance for finding optimal solutions, they are disabled during the numerical tests to clearly isolate the sole effect of dynamic scaling.

In order to obtain a dense set of normalized directions $\{u^\ell\}_{\ell=0}^{\infty}$ as defined in Section 3.4, we randomly generate points in the unit sphere using n normal variables as follows:

$$u^\ell = \frac{X_1 + X_2 + \ldots + X_n}{\sqrt{X_1^2 + X_2^2 + \ldots + X_n^2}}$$

for some index ℓ and $X_j \sim \mathrm{N}(0,1)$ for all $j \in \{1, 2, \ldots, n\}$.

In practice, each normal random value is computed using the polar form of the Box-Muller transformation [10] based on draws from the uniform random generator found at madrabbit.org/∼ray/code/xorshf96.c. This strategy to generate a dense set of poll directions is used for both the new strategy with dynamic scaling and for the previous isotropic one. These two versions are called MADS 2n(aniso) and MADS 2n(iso), for the anisotropic and isotropic variants, respectively, In practice, this is set by the anisotropic_mesh (yes|no) parameter in NOMAD.

In the next two subsections, two settings are considered to test the efficiency of the new strategy: A set of analytical functions, and a realistic application. The stopping criteria is set to be the maximal number of evaluations while the minimal mesh size is left to the NOMAD default of 1E−13.

In both cases, many instances are employed allowing the presentation of results with *data profiles* inspired from [21]. In these graphs, the horizontal axis represents the convergence in terms of number of evaluations, while the vertical axis corresponds to the proportion of problems solved within a given relative tolerance τ. One curve is drawn for each algorithm, which allows to graphically compare the convergence of the two methods. For example, if one plot is always above the other, it means that the corresponding method dominated the other as it systematically solves more instances than the other for the same budget of function evaluations.

4.1 Test problems from the derivative-free optimization literature

We test 212 analytical problems from the derivative-free optimization literature [21]. These 212 instances are generated from 22 different CUTEst [17] functions with some different starting points and four variants (smooth, nonsmooth, deterministically noisy, and randomly noisy). The number of variables ranges from 2 to 12, and the problems are unconstrained except for 8 of them in which the variables are nonnegative.

Data profiles display the fraction of problems solved for a given tolerance depending on the progression of the algorithm. Here, the relative tolerance for matching the best obtained solution is fixed at τ =1E−3, and the convergence is represented by the equivalent number of simplex gradient evaluations, i.e., the number of groups of $(n + 1)$ calls to the simulation.

Preliminary tests have shown an influence of the random number generator seed on the data profiles. The random number generator is used to obtain the polling directions and it affects the solutions obtained for these problems. To make the data profiles less sensitive to the seed selection and more conclusive, a series of 10 different seeds are used for each problem and each algorithm. This leads to a total of 4240 executions, summarized in the data profiles of Figure 3.

The profiles show that there is a clear performance gain when the dynamic anisotropic mesh is enabled, since the entire plot associated with MADS 2n(iso) is dominated by the plot of MADS 2n(aniso). For a given budget of evaluations, the dynamic scaling solves approximately 10% more problems.

4.2 The Supersonic Business Jet Design example

The supersonic business jet design problem is presented in [3] and has been used to demonstrate the use of coordination algorithms such as the nonhierarchical analytical target cascading (ATC) formulation [25] for solving multidisciplinary design optimization (MDO) problems. The design problem involves the disciplines

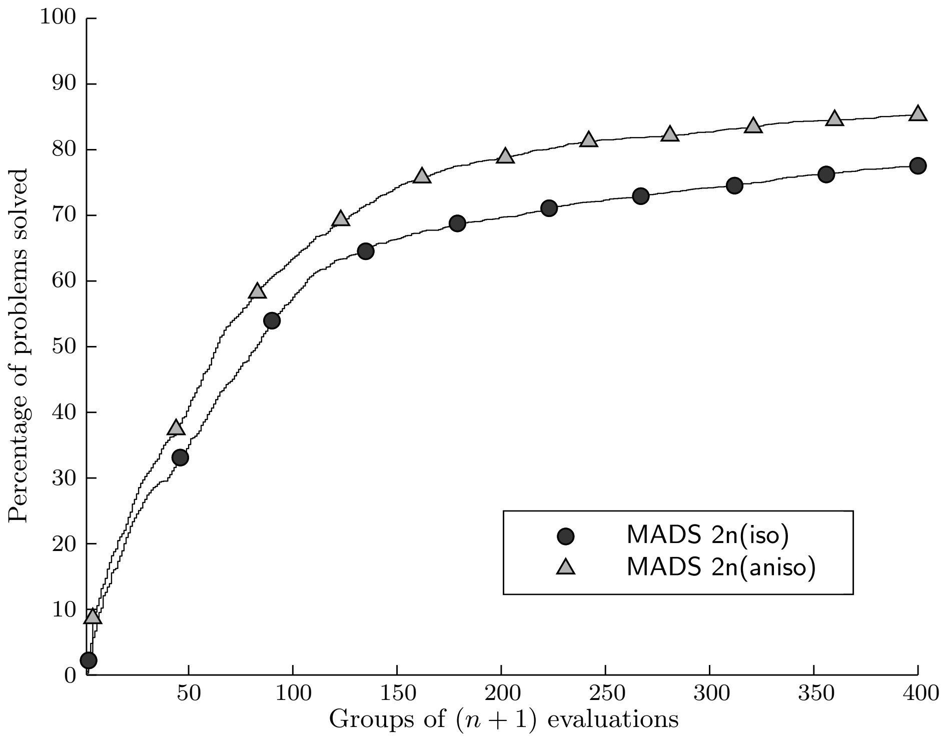


Figure 3: Data profiles with a relative tolerance of τ =1E−3 for 212 instances of test problems from the literature.

"Structure", "Aerodynamics", "Propulsion", and "Aircraft." In practice, each discipline analysis is calculated by a Matlab function that takes some arguments as input and evaluates a subset of the aircraft characteristics. The discipline analyses are interdependent: some discipline outputs are inputs of other disciplines. Only the multidisciplinary feasible optimization (MDF) approach [13] for solving the design problem is considered here. The multidisciplinary feasibility is achieved by a fixed-point iterative approach calling the Matlab functions in sequence until the maximum relative change of all outputs between two consecutive iterations is lower than a given precision of 1E−3. If this value is not reached after 50 fixed-point iterations, the multidisciplinary analysis is considered as failed.

The design objective is to minimize the weight of an aircraft while satisfying constraints from the disciplines. A description of the original optimization problem is available in [25]. Several optimal solutions can be obtained for this problem depending on the initial point considered. To better compare the algorithmic performances, a simpler instance of the problem has been considered. The original MDF optimization problem has 29 independent design variables and 46 design constraints. The objective, the design variables and constraints, the reference design values and the bound constraints considered in this work are the same as in [25] except that the design variables associated to the structural elements thicknesses (18 variables) are fixed to the optimal values reported and the reference value for the wing surface area is changed from 500 sq. ft. in the original problem to 650 sq. ft. in our modified version compared with the reported optimal value of 667 sq. ft. With these changes from the original problem, NOMAD is able to obtain more consistently solutions with objective in a narrower range. All of our numerical tests are conducted on this simpler instance.

The optimization process is launched from 100 randomly generated starting points, selected within ±10% of the reference values. The 46 design constraints are treated with the Extreme Barrier strategy. Hence, the infeasible solutions are simply rejected during optimization. The reference design being infeasible, an initial phase to find a feasible solution is conducted by NOMAD by minimizing the constraint violation function [9]. A maximal budget of 20,000 multidisciplinary analyses function calls is imposed.

Under these conditions, we observe that all solutions produced by NOMAD are strictly feasible, and the final objective function values range from 34,286 lbs (best obtained objective value) to 36,500 lbs. In most cases, NOMAD terminates because the maximum number of evaluations is reached.

Data profiles are used to display the fraction of problems solved within a given tolerance of the best obtained objective value with respect to progression of the algorithms represented by the number of evaluations. The results displayed in Figure 4 are obtained for a tolerance of τ =5E−2, which corresponds to the range [34,286;35,200] lbs reached by almost 75% of the runs. As for the analytic cases, the data profiles illustrate that the anisotropic mesh strategy outperforms the isotropic mesh strategy in reaching a solution within the given range even after a relatively low number of evaluations (> 2,000 evaluations).

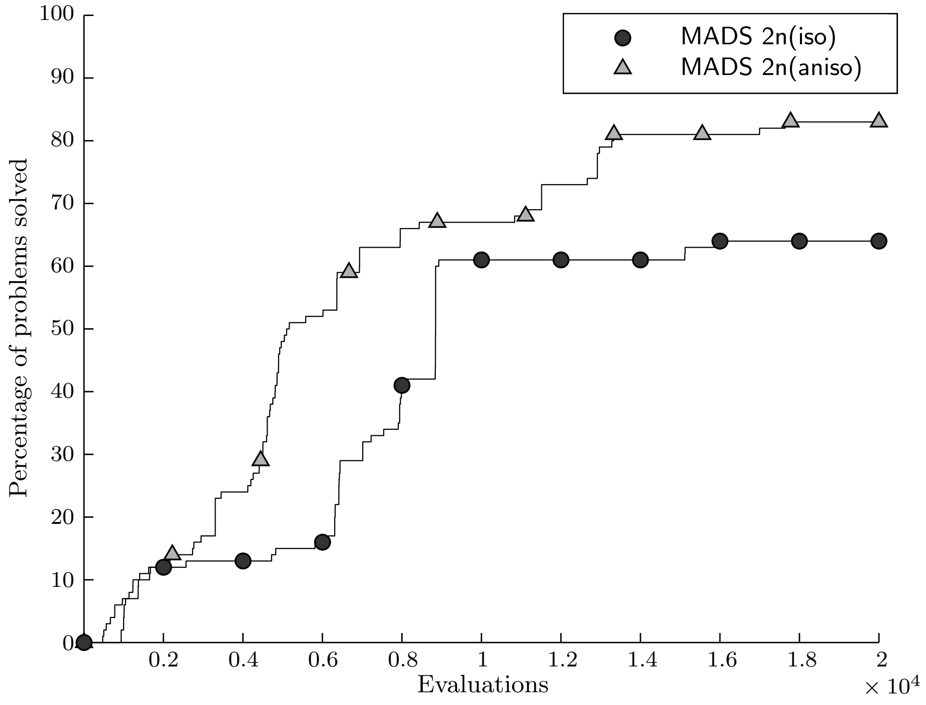


Figure 4: Data profiles with a relative tolerance of τ =5E−2 for the Supersonic Business Jet Design problem from 100 different starting points.

The histogram plot given in Figure 5 complements the data profile. It reveals that with a budget of 15,000 evaluations, for 21 of the 100 starting points, MADS 2n(iso) and MADS 2n(aniso) obtain solutions for which $|f_{\text{iso}} - f_{\text{aniso}}|$ <250 lbs, where f_{iso} and f_{aniso} denote the values of the solutions obtained by MADS 2n(iso) and MADS 2n(aniso), respectively. In other words, from these 21 starting points both algorithms have obtained comparable solution values. However, for 53 of the 100 starting points MADS 2n(aniso) obtained a better solution than MADS 2n(iso) and for the remaining 26 starting points MADS 2n(iso) obtained a better solution than MADS 2n(aniso). In addition, the maximum of the histogram plot is skewed to the right of the central bar which shows that the anisotropic strategy described in the paper is beneficial.

The above analyses pertain to 100 runs. Let us now focus on a single representative run from a feasible starting point that terminated with slightly less than 10,000 evaluations. Figure 6 shows the evolution of the anisotropy as the algorithms is deployed, by plotting the ratio of the minimum and maximum mesh size parameter values over all variables versus the number of blackbox evaluations on the runs MADS 2n(iso) and MADS 2n(aniso). Of course, the ratio is constant for the isotropic case and corresponds to the initial ratio for the anisotropic case. The value of this constant ratio is fixed by the poll size initialization, as described in Section 3.1. Here, there is a scaling difference of the order of 1E−3 between all the variables.

The figure shows that quite rapidly, MADS 2n(aniso) adapts the mesh size parameters to widely different magnitudes as the algorithm progresses. The bottom plot of the same figure shows the evolution of the best

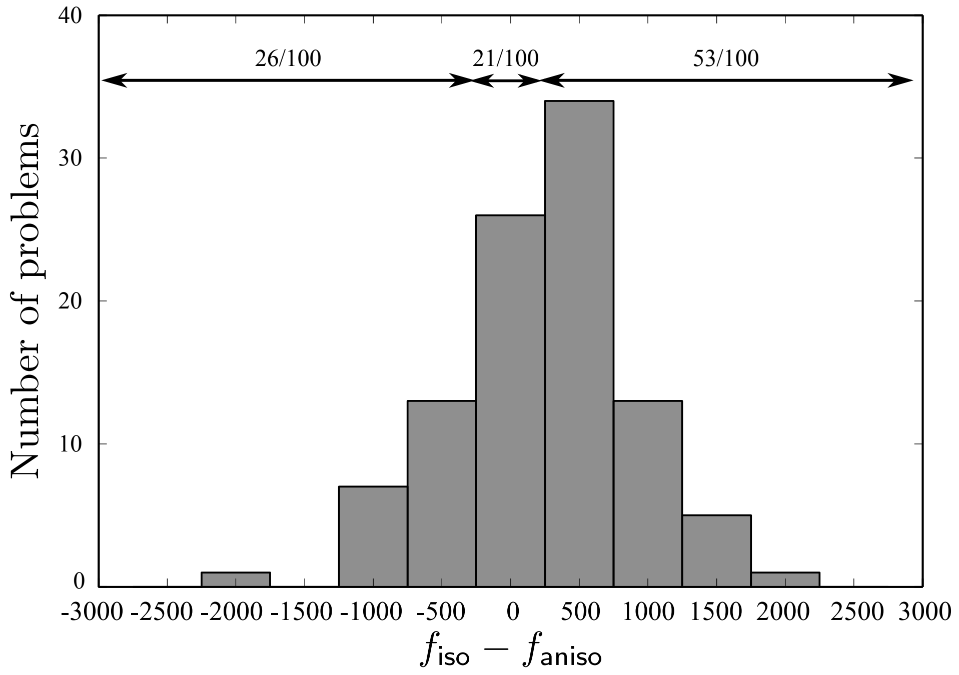


Figure 5: Histogram plot of the difference in objective functions obtained by MADS 2n(iso) and MADS 2n(aniso) after 15,000 evaluations for 100 starting points.

feasible objective function value for MADS 2n(aniso). Inspection of the log file reveals that the two downward spikes near evaluations 3,000 and 4,800 correspond to important decreases in the objective function values.

More precisely, between evaluations 4,803 and 4,816, five consecutive successful incumbents are obtained, the objective function value reduces from 34,265.09 lbs to 34,263,54 lbs, the maximum mesh size among all variables is increased from 1.4E−10 to 5.9E−7 and the minimum mesh size remains unchanged at 1.7E−15. These successes are obtained while some of the design variables are kept constant but others changed values.

A similar behaviour was observed around iteration 8,500, except that in this case the objective function value improved marginally. Furthermore, most of the reduction in the objective function value was accomplished during the first 1,000 evaluations, but all variables changed values, which explains why there is no important spikes in the left part of the top plot.

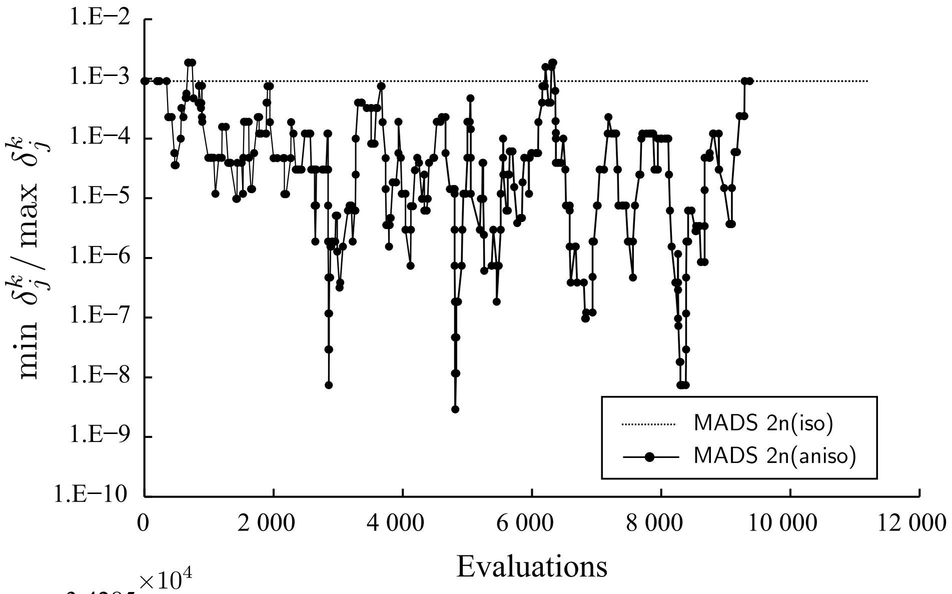
5 Discussion

This work introduces a new way of defining and updating the mesh, the spatial discretization on which the Mads algorithm for blackbox optimization is based. It allows a simpler, more intuitive, and more general description of the algorithm, including a simplified convergence analysis. In practice, the new strategy enables the use of more general types of directions, and an automatic and dynamic way of scaling the variables.

We have compared the previous isotropic Mads approach with the proposed one. On both a collection of standard test problems and on an engineering application, the data profiles showed a clear domination of the anisotropic approach. Further investigation on the engineering problem showed that the anisotropy is exploited and allows to find better solutions more rapidly.

The proposed new algorithmic variant will be available in the release 3.7.0 of the NOMAD software package available at www.gerad.ca/nomad.

Priority in future research will be given to adapt the Progressive Barrier [7] for constraints to the new framework. Another project will analyze links between the anisotropic mesh size parameters and the vari-



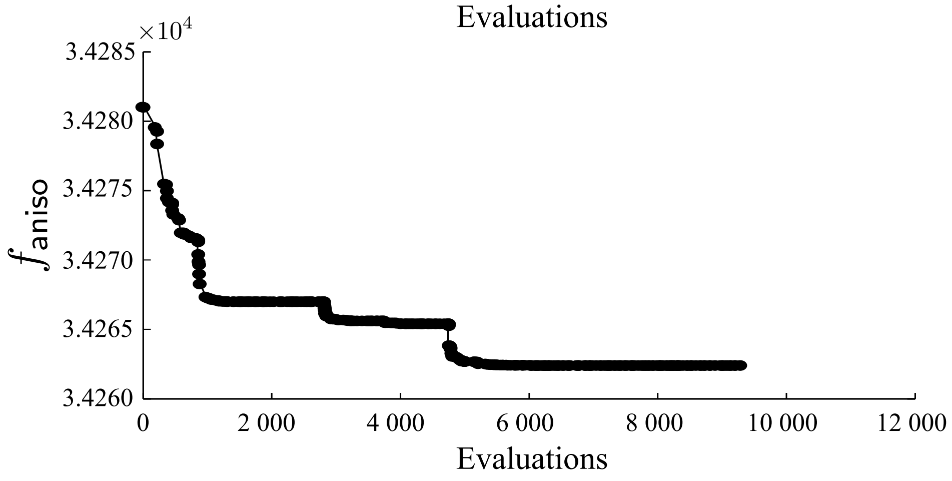


Figure 6: Variation of the mesh size min/max ratio for a single run of the Supersonic Business Jet Design problem and objective function value versus number of function evaluations.

able sensitivities, together with the exploitation of this information for the variable-based decomposition techniques of [2, 8].

Appendix: Convergence analysis of MADS with dynamic scaling

The convergence analysis of MADS with dynamic scaling follows the same two steps as in the previous isotropic case [6]. Under appropriate conditions, the first step proves the existence of a subsequence of iterations for which the norm of the mesh and poll size vectors converges to zero. The second step studies unsuccessful iterations, and considers the limit of a convergent subsequence of incumbent solutions x^k for which the mesh size parameters goes to zero.

The key element of this analysis is that $f_\Omega(x^k)$ is less than or equal to the extreme barrier objective function value evaluated at each neighbouring poll points of x^k. Now, as the iteration k goes to infinity, the distance from the poll points to the incumbent x^k gets arbitrarily small, which leads to the satisfaction of necessary optimality conditions at the limit points.

The convergence analysis below relies on the standard assumptions that there exists a starting point $x^0 \in V^0 \cap \Omega$ such that $f(x^0)$ is finite, and that all iterates $\{x^k\}$ produced by the MADS algorithm lie in a bounded set. Another algorithmic requirement stated in Section 2.3 is that the β_j parameters are such that

$(\beta_j)^2$ are rational numbers. This requirement imposed on the MADS algorithm is not an artifice of the proof since it cannot be relaxed as demonstrated in [4] on an example.

Proposition A.1 *The mesh and poll size vectors produced by a* MADS *instance satisfy*

$$\liminf_{k\to+\infty} \|\delta^k\| \;=\; \liminf_{k\to+\infty} \|\Delta^k\| \;=\; 0.$$

Proof. Let $j \in \{1,2,\ldots,n\}$ be the index of a variable. Suppose by way of contradiction that there exists a negative integer ρ for which the mesh index satisfies $r_j^k > \rho$ for all $k \geq 0$. It follows that the mesh size parameter $\delta_j^k = \delta_j^0(\beta_j)^{r_j^k - |r_j^k|}$ is bounded above by δ_j^0 and bounded below by $\delta_j^0(\beta_j)^{2\rho}$. Consequently, δ_j^k takes its value from the finite set $\{\delta_j^0(\beta_j)^{2\rho}, \delta_j^0(\beta_j)^{2\rho+2}, \ldots, \delta_j^0\}$ regardless of the iteration number $k \geq 0$.

But since $(\beta_j)^2$ is a rational number, there exists two integers p_j and q_j such that $(\beta_j)^2 = \frac{p_j}{q_j}$. It follows that for every $k \geq 0$, there exists some finite integer ζ_j^k between ρ and 0 such that $\delta_j^k = \delta_j^0 \frac{\zeta_j^k}{(q_j)^\rho}$. Therefore, any trial point $t \in M^k$ satisfies Equation (3) and thus

$$t_j \;=\; x_j^0 + \sum_{\ell=0}^{k} \delta_j^\ell z_j^\ell \;=\; x^0 + \frac{\delta_j^0}{(q_j)^\rho} \sum_{\ell=0}^{k} \zeta_j^\ell z_j^\ell$$

for some initial point $x^0 \in V^0$. In other words, the possible values taken by t_j belong to the unbounded set $\{x_j^0, x_j^0 \pm \frac{\delta_j^0}{(q_j)^\rho}, x_j^0 \pm \frac{2\delta_j^0}{(q_j)^\rho}, \ldots\}$ regardless of the iteration number k. But since all iterates produced by the MADS instance belong to a bounded set, then this implies that the trial points can take only finitely many different values in $\mathbb{R}^n$.

This contradicts the hypothesis that the mesh size parameter is bounded away from zero, implying, from the MADS PARAMETER UPDATE rule in Algorithm 1, that there are infinitely many successful iterations, and infinitely many different incumbent solutions. □

The second step of the convergence analysis studies optimality conditions at some accumulation points of the sequences of mesh local optimizers x^k. The following definition introduces refining sequences, refined points and refining directions.

Definition A.2 *Let U be the indices of the unsuccessful iterations. If K is an infinite subset of U such thatf $\lim_{k\in K} \|\delta^k\| = 0$ and the sequence $\{x^k\}_{k\in K}$ converges to some $x^* \in \mathbb{R}^n$, then the set of poll centers $\{x^k\}_{k\in K}$ is said to be a refining sequence, and x^* a refined point. Furthermore, if for each $k \in K$ there is a poll point $x^k + \operatorname{diag}(\delta^k)d^k$ for some direction $d^k \in D^k$ such that the sequence of normalized directions $\left\{\frac{v^k}{\|v^k\|}\right\}_{k\in K}$ (with $v^k = \operatorname{diag}(\delta^k)d^k$) converges to some $v^* \in \mathbb{S}^n$, then v^* is called a refining direction.*

Refined points are limits of mesh local optimizers on meshed that get infinitely fine.

Proposition A.3 *There exists a refining sequence and a refined point.*

Proposition A.1 ensures that there exists an infinite subset of indices K for which $\lim_{k\in K} \|\delta^k\| = 0$. The mesh size parameters are reduced only at the unsuccessful iterations, and therefore, it can be assumed without any loss of generality that $K \subseteq U$. Furthermore, all trial points are assumed to lie in a bounded set, and consequently we can also assume that the sequence $\{x^k\}_{k\in K}$ is convergent. □

The next result shows that the Clarke generalized derivative at a refined point is nonnegative in a refining direction.

Theorem A.4 *Let $x^* \in \Omega$ be a refined point, with $v^* \in T_\Omega^H(x^*)$ a refining direction. If f is Lipschitz near x^* then $f^\circ(x^*; v^*) \geq 0$.*

Proof. Let K be a subset of indices of unsuccessful iterations such that $\{x^k\}$ is a refining sequence converging to $x^* \in \Omega$ and $v^* \in T^H_\Omega(x^*)$ a refining direction.

By definition of the tangent cone, and since $\|v^k\| \leq \|\Delta^k\| \to 0$ for $k \in K$, then $x^k + v^k \in \Omega$ for all k sufficiently large. Therefore, since polling at $x^k + v^k$ was unsuccessful it follows that

$$\begin{aligned}
f^\circ(x^*; v^*) &= \limsup_{\substack{y \to x^*,\ y \in \Omega \\ t \downarrow 0,\ y + tv^* \in \Omega}} \frac{f(y+tv^*) - f(y)}{t} \\
&= \limsup_{\substack{y \to x^*,\ y \in \Omega, w \to v^* \\ t \downarrow 0,\ y + tw \in \Omega}} \frac{f(y+tw) - f(y)}{t} \\
&\geq \limsup_{\substack{k \in K, y = x^k, \\ w = \frac{v^k}{\|v^k\|}, t = \|v^k\|}} \frac{f(y+tw) - f(y)}{t} \\
&= \limsup_{k \in K} \frac{f(x^k + v^k) - f(x^k)}{\|v^k\|} \ \geq \ 0.
\end{aligned}$$

□

The next result constitutes the main convergence result. It states that if the mechanism used to produce the poll directions is sufficiently rich to generate a dense set of refining directions, then, under the constraint qualification that the hypertangent cone to Ω at a refined point $x^*$is non-empty, then x^* is a Clarke stationary point of the optimization problem.

Theorem A.5 *If the set of refining directions is dense in $\mathbb{S}$, and if f is Lipschitz near the refined point $x^* \in \Omega$ where $T^H_\Omega(x^*) \neq \emptyset$, then x^* is a Clarke stationary point, i.e., $f^\circ(x^*; v) \geq 0$ for every direction $v \in T^{Cl}_\Omega(x^*)$.*

If the set of refining directions is dense in the sphere $\mathbb{S}$, then any normalize Clarke-tangent direction to Ω at a refined point x^* is the limit of refining directions. The result follows from Theorem A.4. □

References

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

[2] L. Adjengue, C. Audet, and I. Ben Yahia. A variance-based method to rank input variables of the Mesh Adaptive Direct Search algorithm. *Optimization Letters*, 2014.

[3] J.S. Agte, J. Sobieszczanski-Sobieski, and R.R.J. Sandusky. Supersonic business jet design through bilevel integrated system synthesis. In *Proceedings of the World Aviation Conference*, volume SAE Paper No. 1999-01-5622, San Francisco, CA, 1999. MCB University Press, Bradford, UK.

[4] C. Audet. Convergence Results for Generalized Pattern Search Algorithms are Tight. *Optimization and Engineering*, 5(2):101–122, 2004.

[5] C. Audet, V. Béchard, and S. Le Digabel. Nonsmooth optimization through mesh adaptive direct search and variable neighborhood search. *Journal of Global Optimization*, 41(2):299–318, 2008.

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

[7] C. Audet and J.E. Dennis, Jr. A progressive barrier for derivative-free nonlinear programming. *SIAM Journal on Optimization*, 20(1):445–472, 2009.

[8] C. Audet, J.E. Dennis, Jr., and S. Le Digabel. Parallel space decomposition of the mesh adaptive direct search algorithm. *SIAM Journal on Optimization*, 19(3):1150–1170, 2008.

[9] C. Audet, J.E. Dennis, Jr., and S. Le Digabel. Globalization strategies for mesh adaptive direct search. *Computational Optimization and Applications*, 46(2):193–215, 2010.

[10] G.E.P. Box and M.E. Muller. A note on the generation of random normal deviates. *Annals of Mathematical Statistics*, 29:610–611, 1958.

[11] A.R. Conn, K. Scheinberg, and L.N. Vicente. *Introduction to Derivative-Free Optimization.* MOS/SIAM Series on Optimization. SIAM, Philadelphia, 2009.

[12] I.D. Coope and C.J. Price. Frame-based methods for unconstrained optimization. *Journal of Optimization Theory and Applications*, 107(2):261–274, 2000.

[13] E.J. Cramer, J.E. Dennis, Jr., P.D. Frank, R.M. Lewis, and G.R. Shubin. Problem formulation for multidisciplinary optimization. *SIAM Journal of Optimization*, 4(4):754–776, 1994.

[14] J.E. Dennis, Jr. and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Prentice-Hall, Englewood Cliffs, NJ, 1983. Reissued in 1996 by SIAM Publications, Philadelphia, as Vol. 16 in the series Classics in Applied Mathematics.

[15] R. Fletcher. *Practical Methods of Optimization*, volume 1: Unconstrained Optimization. John Wiley & Sons, 1980.

[16] P.E. Gill, W. Murray, and M.H. Wright. *Practical Optimization.* Academic Press, London, 1981.

[17] N.I.M. Gould, D. Orban, and Ph.L. Toint. CUTEst: A constrained and unconstrained testing environment with safe threads. Technical Report, *Les Cahiers du GERAD* G–2013–27, HEC Montréal, 2013.

[18] J.H. Halton. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerische Mathematik*, 2(1):84–90, 1960.

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

[20] V.D. Liseikin. *Grid Generation Methods.* Springer, 2009.

[21] J.J. Moré and S.M. Wild. Benchmarking derivative-free optimization algorithms. *SIAM Journal on Optimization*, 20(1):172–191, 2009.

[22] J. Nocedal and S.J. Wright. *Numerical Optimization.* Springer Series in Operations Research. Springer, New York, 1999.

[23] P.Y. Papalambros and D.J. Wilde. *Principles of optimal design: modeling and computation, 2nd edition.* Cambridge University Press, 2000.

[24] V. Torczon. On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, 7(1):1–25, 1997.

[25] S. Tosserams, L. Etman, and J. Rooda. A classification of methods for distributed system optimization based on formulation structure. *Structural and Multidisciplinary Optimization*, 39(5):503–517, 2009.

[26] B. Van Dyke and T.J. Asaki. Using QR decomposition to obtain a new instance of Mesh Adaptive Direct Search with uniformly distributed polling directions. *Journal of Optimization Theory and Applications*, 159(3):805–821, 2013.