Optimization of noisy blackboxes with adaptive precision *

Stéphane Alarie† Charles Audet‡ Pierre-Yves Bouchet‡ Sébastien Le Digabel‡

November 13, 2019

Abstract:
In derivative-free and blackbox optimization, the objective function is often evaluated through the execution of a computer program seen as a blackbox. It can be noisy, in the sense that its outputs are contaminated by random errors. Sometimes, the source of these errors is identified and controllable, in the sense that it is possible to reduce the standard deviation of the stochastic noise it generates. A common strategy to deal with such a situation is to monotonically diminish this standard deviation, to asymptotically make it converge to zero and ensure convergence of algorithms because the noise is dismantled. This work presents MpMads, an algorithm which follows this approach. However, in practice a reduction of the standard deviation increases the computation time, and makes the optimization process long. Therefore, a second algorithm called DpMads is introduced to explore another strategy, which does not force the standard deviation to monotonically diminish. Although these strategies are proved to be theoretically equivalents, tests on analytical problems and an industrial blackbox are presented to illustrate practical differences.
Keywords: derivative-free, blackbox, stochastic, noisy, adaptive precision, tunable precision, direct-search, Monte-Carlo simulation.

1 Introduction

This work consider the unconstrained problem

$$\min_{x \in D_f \subseteq \mathbb{R}^n} \quad f(x) \tag{1}$$

under the framework of *blackbox optimization* (BBO) and *derivative-free optimization* (DFO). In those frameworks is required almost no structure on the problem and f. It is frequently considered that f is continuous on its domain $D_f \subset \mathbb{R}^n$ unknown *a priori*, but its derivatives may not exist (in BBO) or be impossible to evaluate (in DFO). Usually, a *blackbox* is a complex computer program, computationally intensive to run. This high level of complexity makes the derivatives nonexistent or difficult to estimate. Therefore, algorithms to minimise a blackbox problem do not rely on any gradient-based processes

*This work is supported by the NSERC CRD RDCPJ 490744-15 grant and by an InnovÉÉ grant, both in collaboration with Hydro-Québec and Rio Tinto.

†Expertise – Science des données et calcul haute performance, Institut de recherche d'Hydro-Québec, Varennes, Canada.

‡GERAD and Département de mathématiques et génie industriel, Polytechnique Montréal, Montréal, Canada.

and use only models of the objective function or comparison of previously computed values of f to decide the quality of a given point.

However, in some contexts the values of f cannot be computed exactly. In the simulations performed by a program, it is possible that some stochasticity appears. For example, if the blackbox encodes a Monte-Carlo estimation of the value of interest, two executions of the program with the same input parameter x may return different values. Then, the true value of $f(x)$ is unknown, as any attempt to compute it returns a value affected by some error. This source of errors makes any deterministic algorithm prone to failure, because it is assumed in their design that the computation of $f(x)$ is possible and accurate. When the source of stochasticity is known and its implementation in the program is intentional, it is sometimes possible to control its *magnitude* through the standard deviation of the random error on the returned value. For example, when the program performs a Monte-Carlo estimation of a value, improving the number of Monte-Carlo draws used in the estimation statistically improves the quality of the returned estimate and reduces its standard deviation. This work refers to this situation as an *adaptive precision program*. In this document, an *adaptive precision blackbox* denotes a deterministic function f which cannot be computed, but may be approximated by an adaptive precision program. The computation time of an adaptive precision program depends on the magnitude of errors it ensures: the lower the standard deviation is, the higher the computation time is. In Monte-Carlo estimations, the time grows as the inverse of the square of the standard deviation: the total computation time for N Monte-Carlo draws is roughly $t \propto N$ while the standard deviation is of the form $\sigma \propto 1/\sqrt{N}$. One may consider that as a trade-off: at any execution of the program, any standard deviation can be guaranteed, but the cost can be prohibitive. Three paradigms tackle this added layer of complexity. Specific algorithms exist on each, most of these being extensions of existing deterministic strategies (overviews are given in [7, 15]).

The first possibility is to not control the magnitude of noise during the optimization, but rather to decide it before the optimization starts. Under this strategy, any algorithm which deals with uncertainty can be used. Notably, algorithms designed for situations where the noise is not adaptive. However, it should be noted that deterministic algorithms have no guarantee to work, even with this strategy. One can consider the Robust-Mads algorithm [8] which modifies the Mads algorithm [3] to create a smooth representation of the function, knowing noisy estimates. Mads is also adapted as Stoch-Mads in [6], an algorithm using probabilistic estimates to ensure convergence of a noisy problem where the noise variance is nor known neither adaptive. Various techniques from surface response design can also be used to dynamically generate a sequence of functions approximating f: for example, the Phoenics solver [18] which uses Bayesian kernel density estimations, or kriging, studied by Sasena [29]. A line search algorithm is developed by Paquette and Scheinberg [25]. Also, some algorithms exploit trust-region principles, like Astro-Df in [30].

The second possibility is to lead the optimization on a high magnitude of noise, and reduces it monotonically during the optimization process. The algorithm from Polak and Wetter [27] adapts the mechanics from the Gps algorithm [31] in the case where errors have a controllable upper bound. Chen and Kelley [13] propose a way to reduce the magnitude which can be used in direct-search algorithms when the adaptive precision program performs a Monte-Carlo estimation. They extend this strategy with the addition of a smoothing effect in [32]. A trust-region algorithm handling constraints and using Gaussian models, SNowPaC, is proposed in [9]. Rivier and Congedo proposes in [28] a multi-objective

framework. In [24] is proposed a Delaunay triangulation in $\mathbb{R}^n$, refined jointly with the grow in precision. Heuristics are also used, for example the modification of the Nelder-Mead algorithm [23] proposed by Chang in [12].

The last possibility is to adapt more frequently the magnitude of the noise, reducing it when necessary and augmenting it when possible. Picheny *et al.* [26] propose an adaptation of the EGO algorithm [19] which uses adaptive precision programs with errors given by centred normal laws with controllable magnitude. This strategy is also used in multi-fidelity optimization, for example by Frandi and Papini in [17] which uses a direct-search algorithm to a multi-fidelity context. Multi-fidelity optimization is also named *simulation optimization* in some works, as in the review [2].

The present work introduces two modifications of the MADS algorithm [3] called MPMADS (monotonic precision) and DPMADS (dynamic precision) to handle an adaptive precision blackbox problem with the last two paradigms. The paper is divided as follows. Section 2 introduces the notations and summarises the pertinent elements from the MADS algorithm to introduce DPMADS and MPMADS. The section then presents the two algorithmic variants and concludes by discussing practical implementation issues. A convergence analysis is provided in Section 3. The main result provides necessary conditions that ensure that the algorithm produces, with probability one, a point at which the Clarke generalised directional derivatives are non-negative. Computational experiments are performed in Section 4 on two analytical problems as well as on a real industrial problem. The results demonstrates that DPMADS can considerably reduce the overall computational effort.

2 Two precision-adapting algorithms

This section introduces notations and proposes the monotone and dynamic precision algorithms DPMADS and MPMADS.

2.1 Notations and mathematical optimization problem

This work aims to solve the unconstrained optimization Problem (1). The domain D_f on which f is defined is unknown *a priori*. As capturing this domain is part of the problem, one may consider the extreme barrier formulation of the objective proposed in [3]. Denoting $\overline{\mathbb{R}} = \mathbb{R} \cup \{\pm\infty\}$, Problem (1) can be reformulated with an extended definition of f:

$$\min_{x\in\mathbb{R}^n} f(x), \text{ with } f : \left\{ \begin{array}{ccc} \mathbb{R}^n & \longrightarrow & \overline{\mathbb{R}} \\ x & \longmapsto & \left\{ \begin{array}{ll} f(x) & \text{if } x \in D_f \\ +\infty & \text{otherwise.} \end{array} \right. \end{array} \right. \tag{2}$$

In the context of this work, exact values of f cannot be obtained. Only approximations may be computed, because of a *noise* which alters the value $f(x)$ during the numerical evaluation. It is assumed that a *stochastic noise* η is added to the numerical value: one observe $f(x) + \eta$ instead of computing $f(x)$. More precisely, while $\eta_\sigma \sim \mathcal{N}(0, \sigma^2)$ follows a centred normal law with standard deviation σ independent from the point x and the objective $f(x)$, the noise is represented as $f_\sigma(x) = f(x) + \eta_\sigma$, a random variable following $\mathcal{N}(f(x), \sigma^2)$. Any attempt to compute $f(x)$ returns $f_\sigma(x;\omega)$, an observation ω of $f_\sigma(x)$. Therefore, two consecutive attempts to evaluate $f(x)$ may return two different outputs $f_\sigma(x;\omega_1) \neq f_\sigma(x;\omega_2)$. In addition to the normal behaviour of the noise, the standard

deviation σ is assumed to be *adaptive*. This signifies that its value is controllable by the one who attempts to evaluate $f(x)$. The value σ may be modified for any evaluation, even if x remains unchanged. Modifying the value σ alters the random variable $f_\sigma(x)$ used to obtain approximation $f_\sigma(x;\omega)$ of $f(x)$. As a lower standard deviation σ gives fewer probability that the estimate $f_\sigma(x;\omega) = f(x) + \eta_\sigma(\omega)$ highly differs from $f(x)$, the present work uses *higher precision* as a synonym for *lower standard deviation*. Furthermore, at an infinite precision (equivalently, a null standard deviation) $f_\sigma(x)$ converge to a Dirac measure centred on value $f(x)$: $\mathbb{P}\left(\lim_{\sigma\to 0} f_\sigma(x) = f(x)\right) = 1$.

The precision required at any estimation of any $f(x)$ is determined by a so-called *precision index* denoted $r \in \mathbb{R}$. This index is used to set the value of the standard deviation through the mapping function ρ: $\sigma = \rho(r)$. This function is assumed to be positive, upper bounded by a finite σ_{max}, and decreasing. Under these hypotheses, the index r can be interpreted as the precision, and $\rho(r)$ its associated standard deviation. Via ρ, a high value of precision index r corresponds to a low standard deviation σ.

The DpMads and MpMads algorithms presented in Section 2.3 follow an iterative mechanic. The optimization process exploits, at any iteration denoted k, the computations performed by earlier iterations. The *historic* at a point $x \in \mathbb{R}^n$, or *cache at* x, is defined as follows: $\mathcal{V}^k(x) = \{(f_\sigma(x;\omega), \sigma) \mid f_\sigma(x;\omega) \text{ has been observed during an iteration } i \leq k\}$. Each element of this set is a couple where the first element is estimated value $f_\sigma(x;\omega)$ of $f(x)$ obtained from a noisy observation $f_\sigma(x)$, while the standard deviation σ of the noise is the second element of the couple. If x has not been evaluated up to iteration k, then $\mathcal{V}^k(x) = \emptyset$ is void. The set $\mathcal{V}^k(x)$ can be interpreted as the full historic at a given point x at a given iteration k. One can also define the *cache* at iteration k: $\mathcal{V}^k = \left\{\left(x, \mathcal{V}^k(x)\right) \mid x \in \mathbb{R}^n\right\}$, which links a point x with the historic at x up to iteration k.

These notations can be abusively extended in the following way: $\mathcal{V}^k(x)$ is a function which returns the cache at x. This function searches in $\mathcal{V}^k$ for the couple formerly denoted $(x, \mathcal{V}^k(x))$, and returns the second element. Then, $\mathcal{V}^k(x)$ returns the empty set $\emptyset$ if x has never been evaluated up to iteration k, and the full historic at x otherwise.

Given the cache $\mathcal{V}^k(x)$ at iteration k and point x, it is possible to construct an estimation of the objective function on x. This estimate is denoted $f^k(x)$ and its statistical standard deviation $\sigma^k(x)$. Various techniques exist to create those, such as the maximum likelihood used in this work. The value $f^k(x)$ is the best estimation of $f(x)$ that can be proposed up to iteration k. It is defined as the most plausible value of the mean of all the normal laws $f_\sigma(x)$ for which an observation $f_\sigma(x;\omega)$ is contained in $\mathcal{V}^k(x)$. The value $f^k(x)$ is given by the formula below, where (λ, σ) are elements of $\mathcal{V}^k(x)$, of the form $(f_\sigma(x;\omega), \sigma)$. Then, the statistical standard deviation of $f^k(x)$ can be computed as $\sigma^k(x)$, and $f^k(x)$ statistically follows a law $\mathcal{N}\left(f(x), \sigma^k(x)^2\right)$. No predicted or estimated values are proposed at non-evaluated points (points x such that $\mathcal{V}^k(x) = \emptyset$). The estimates are:

$$\begin{cases} f^k(x) &= \dfrac{\sum_{(\lambda,\sigma)\in\mathcal{V}^k(x)} \lambda/\sigma^2}{\sum_{(\lambda,\sigma)\in\mathcal{V}^k(x)} 1/\sigma^2} \quad \text{if } \mathcal{V}^k(x) \neq \emptyset, \ +\infty \text{ otherwise,} \\ \sigma^k(x) &= \sqrt{\dfrac{1}{\sum_{(\lambda,\sigma)\in\mathcal{V}^k(x)} 1/\sigma^2}} \quad \text{if } \mathcal{V}^k(x) \neq \emptyset, \ +\infty \text{ otherwise.} \end{cases} \tag{3}$$

These estimates are used to define the *incumbent* at iteration k as $x_*^k \in \underset{x\in\mathbb{R}^n}{\arg\min} f^k(x)$.

The original Problem (2) can be reformulated with no use of the values of $f(x)$ (values of the true objective function, which cannot be computed):

$$\min_{x\in\mathbb{R}^n} \lim_{\#\mathcal{V}^k(x)\to+\infty} f^k(x). \tag{4}$$

Optima are unchanged, because of the almost-sure equality provided by strong law of large numbers which ensures that for all points x estimated infinitely often as $k \to +\infty$, it is almost sure that the maximum likelihood $f^k(x)$ converges to $f(x)$:

$$\mathbb{P}\left(\lim_{k\to+\infty} f^k(x) = f(x) \mid \#\mathcal{V}^k(x) \underset{k\to+\infty}{\longrightarrow} +\infty\right) = 1, \quad \forall x \in \mathbb{R}^n.$$

2.2 Adaptation of MADS for adaptive precision control

This work proposes two algorithms exploiting the spatial exploration mechanics given by the MADS algorithm to solve the noisy Problem (4). Recall that MADS uses of discretisations of the space $\mathbb{R}^n$ named *meshes of size δ_m centred on x*. Such a mesh is defined as the set $\mathcal{M}_{\delta_m}(x) = \{x + \delta_m z \mid z \in \mathbb{Z}^n\}$. At iteration k, MADS defines the mesh of size δ_m^k centred on its current incumbent x_*^{k-1}: $\mathcal{M}^k = \mathcal{M}_{\delta_m^k}(x_*^{k-1}) = \{x_*^{k-1} + \delta_m^k z \mid z \in \mathbb{Z}^n\}$. From $\mathcal{M}^k$ is extracted a *poll* $\mathcal{P}^k$, a set of *candidates* points to be evaluated. The candidates remain close to the incumbent, on a *frame* of size δ_p^k : $\|x_*^{k-1} - x\|_\infty \leq \delta_p^k,\ \forall x \in \mathcal{P}^k$. Values of δ_m^k are chosen so that $\forall k,\ \delta_m^k \leq \delta_p^k$, and therefore the rule $\delta_m^k = \min(\delta_p^k, (\delta_p^k)^2)$ from [3] is chosen. A common strategy to efficiently explore the neighbourhood of the incumbent is to create a positive basis of $\mathbb{R}^n$ (denoted $\mathcal{D}^k$) such that $x_*^{k-1} + d \in \mathcal{M}^k$ *and* $\|d\|_\infty \leq \delta_p^k,\ \forall d \in \mathcal{D}^k$, as proposed in the ORTHOMADS algorithm in [1].

As the precision is adaptive in this work, a cornerstone of both algorithms is the way they modify that precision. The precision r can grow arbitrarily high to ensure a standard deviation σ as low as desired. However, it impacts the computational cost per evaluation. Therefore, There is a trade-off to exploit in the best way: how to choose r at any iteration and any point, to ensure the convergence within a computational effort as low as possible.

The first algorithm is MPMADS (*Monotonic Precision* MADS), a generalisation of the work of Polak and Wetter [27]. Its behaviour gives a monotonic control of the precision: the precision increases during the optimization process, as slow as possible to avoid over-consumption of computational budget, but fast enough to ensure that the noise never impacts the convergence. At the end of any iteration k, MPMADS checks the quality of the estimates. If they are sufficiently accurate, the precision index $r^{k+1} = r^k$ is left unchanged. Otherwise, it is increased ($r^{k+1} > r^k$) so that the standard deviation $\sigma^{k+1} = \rho(r^{k+1})$ is sufficiently low to avoid the algorithm being misled by the noise.

The second algorithm is DPMADS (*Dynamic Precision* MADS), with a different control of the precision. "Dynamic" means that the precision is not forced to increase. In DPMADS, $r^{k+1} < r^k$ is possible. This deteriorates the quality of future estimates, but DPMADS ensures that the uncertainty coming from this reduction is sufficiently low to avoid biased convergence. At any iteration k, DPMADS attempts to set the precision r^k at the lowest value possible which ensures that the standard deviation $\sigma^k = \rho(r^k)$ is sufficiently low to prevent biased decisions. In the algorithms, the $UpdateR(r^k, p^k)$ function modifies r^k, given p^k an indicator of estimates quality. Detailed expressions of $UpdateR$ are given in Section 2.4.

The standard deviation σ^k is used during iteration k in the following way. DpMads and MpMads start the iteration k from their current incumbent solution (a point which have the lowest estimate: $x_*^{k-1} \in \arg\min\{f^{k-1}(x) \mid x \in \mathbb{R}^n\}$), and generate a *poll* set $\mathcal{P}^k$ of candidates around x_*^{k-1} (following the mechanics of Mads). Incumbent, as well as all the candidates, are evaluated so that $\sigma^k(x) \leq \sigma^k,\ \forall x \in \mathcal{P}^k \cup \{x_*^{k-1}\}$.

The poll step on the set $\mathcal{P}^k$ is implemented in DpMads and MpMads via Algorithm 1:

Algorithm 1 Poll step algorithm for non-deterministic objective

1: **function** $Poll(x_*^{k-1},\ \delta_p^k,\ r^k,\ \mathcal{V}^{k-1})$
2: $\quad \sigma^k \leftarrow \rho(r^k)$
3: $\quad \delta_m^k \leftarrow \min\left(\delta_p^k, \left(\delta_p^k\right)^2\right)$
4: $\quad \mathcal{M}^k \leftarrow \left\{x_*^{k-1} + \delta_m^k z \mid z \in \mathbb{Z}^n\right\}$
5: $\quad \mathcal{D}^k \leftarrow$ *positive basis of* $\mathbb{R}^n$ *such that* $x_*^{k-1} + d \in \mathcal{M}^k$ *and* $\|d\|_\infty \leq \delta_p^k,\ \forall d \in \mathcal{D}^k$
6: $\quad \mathcal{P}^k \leftarrow \left\{x_*^{k-1} + d \mid d \in \mathcal{D}^k\right\}$
7: $\quad$ **for** $x \in \mathcal{P}^k \cup \left\{x_*^{k-1}\right\}$ *such that* $\sigma^{k-1}(x) > \sigma^k$ **do**
8: $\quad\quad$ Compute some $(\sigma_1, \sigma_2, \ldots, \sigma_u)$ so that $\sigma^k(x) = \left(\frac{1}{\sigma^{k-1}(x)^2} + \sum_{i=1}^u \frac{1}{\sigma_i^2}\right)^{-1/2} \leq \sigma^k$
9: $\quad\quad$ Generate $(f_{\sigma_i}(x))_{1\leq i\leq u}$, observe $(f_{\sigma_i}(x;\omega_i))_{1\leq i\leq u}$, update $\mathcal{V}^{k-1}(x)$ as $\mathcal{V}^{k-1}_{updated}(x)$
10: $\quad \mathcal{V}^k \leftarrow \left\{(x, \mathcal{V}^{k-1}_{updated}(x)) \mid x \in \mathbb{R}^n\right\}$
11: $\quad$ Choose $x_c^k \in \arg\min\left\{f^k(x) \mid x \in \mathcal{P}^k\right\}$
12: $\quad S^k \leftarrow \begin{cases} Success \text{ if } f^k(x_c^k) < f^k(x_*^{k-1}) \\ Barrier \text{ if } P^k \cap D_f = \emptyset \\ Failure \text{ otherwise} \end{cases}$
13: $\quad$ **return** $x_c^k, S^k, \mathcal{V}^k$

Algorithm 1 implements the poll step using the mechanics from Mads, and returns:

- x_c^k, the best candidate found during the search,
- S^k, an indicator of the quality of this candidate: $S^k = Success$ if it appears better than the incumbent, $S^k = Barrier$ if no point of $\mathcal{P}^k$ belongs to the feasible domain D_f, and $S^k = Failure$ if no feasible candidate have an estimate lower than $f^k(x_*^{k-1})$,
- $\mathcal{V}^k$, the former cache updated with the evaluations performed during the poll.

Observe that Line 8 indicates to compute some values $(\sigma_i)_{1\leq i\leq u}$ so that the standard deviation $\sigma^k(x) = \left(\frac{1}{\sigma^{k-1}(x)^2} + \sum_{i=1}^u \frac{1}{\sigma_i^2}\right)^{-1/2}$ of estimate $f^k(x)$ satisfies $\sigma^k(x) \leq \sigma^k$. This means that the *Poll* checks the quality of the estimate $f^{k-1}(x)$ by comparing its current standard deviation $\sigma^{k-1}(x)$ to σ^k. If it is higher than σ^k, then the algorithm produces one or many standard deviations $(\sigma_i)_{1\leq i\leq u}$ and observations $(f_{\sigma_i}(x;\omega_i))_{1\leq i\leq u}$ such that the new estimate $f^k(x)$ have a standard deviation satisfying $\sigma^k(x) \leq \sigma^k$. A simple strategy consists of the following: if $\sigma^{k-1}(x) \leq \sigma^k$, no new observation is performed, but if $\sigma^{k-1}(x) > \sigma^k$ an unique ($u = 1$) standard deviation is produced at an high value (σ such that $\sigma^k(x) = \sigma^k$, or $\sigma = \sigma_{max}$ if the former equation leads to a solution σ higher than σ_{max}).

DpMads and MpMads also allow the optional "search step" from Mads to be used at the beginning of iteration k, with the $Search(\mathcal{V}^{k-1}, r^k)$ function. It improves the estimates of all or some points x in $\mathcal{M}^k$ with an observation of $f_{\sigma_s}(x)$ for a given σ_s. To avoid this step being too costly, this standard deviation σ_s can be set at a high value. In the following, it is set to $\sigma_s^k = \rho(r^k - r_s)$ for a given $r_s > 0$. *Search* creates $\mathcal{V}_{search}^{k-1}$, the cache $\mathcal{V}^{k-1}$ updated with the addition of the estimates computed by the function, then returns a minimiser x_s^k of f^{k-1} over $\mathcal{V}_{search}^{k-1}$. If this step is disabled, $\mathcal{V}_{search}^{k-1} = \mathcal{V}^{k-1}$ and $x_s^k = x_*^{k-1}$ are unchanged. Altrough it does not impact convergence, it is possible that x_s^{k-1} differs from x_*^{k-1}.

2.3 Mads algorithm with monotonic and dynamic precision control

The comprehension of the behaviour of both algorithms is easier if one recall how to decide if estimates are sufficiently accurate. At the end of iteration k, DpMads and MpMads consider their incumbent x_*^k as a point which have the lowest objective function value estimate among the set of evaluated points: $x_*^k \in \arg\min\{f^k(x) \mid \mathcal{V}^k(x) \neq \emptyset\}$. However, because of the noise, it is uncertain that this incumbent also minimises f over the evaluated points. This uncertainty is quantified as follows. One can compare the best candidate $x_c^k \in \arg\min\{f^k(x) \mid x \in \mathcal{P}^k\}$ to x_s^k by computing the statistical *p-value* p^k of the hypothesis "$f(x_c^k) < f(x_s^k)$", knowing $\mathcal{V}^k$. If p^k is close to 1, then "$f(x_c^k) < f(x_s^k)$" is highly plausible and the estimates $f^k(\cdot)$ are considered sufficiently accurate. If p^k is close to 0, the estimates are considered accurate because the opposite hypothesis "$f(x_c^k) \geq f(x_s^k)$" is highly plausible. Then, the estimates are considered inaccurate if p^k is too close to 0.5. Figure 1 illustrates the hypothesis "$f(x_2) < f(x_1)$" on three scenarios. The values p^k are computed analytically, using the cumulative distribution function of the $\mathcal{N}(0,1)$ law denoted Φ.

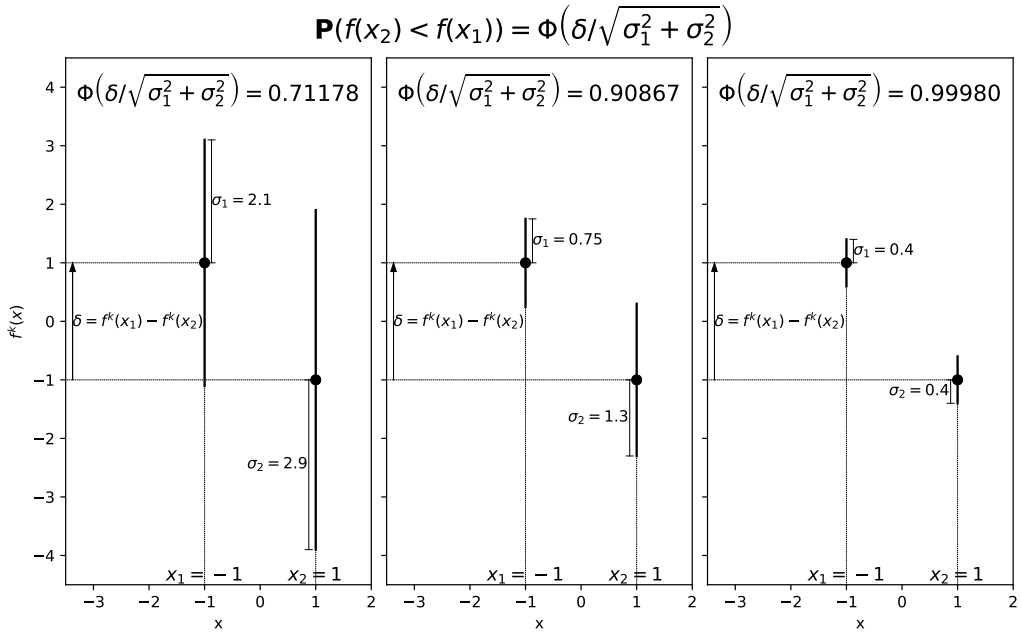

Figure 1: Comparison of two estimates in three situations. The dots represent the estimated objective function values and the vertical lines the standard deviations.

MpMads is designed to avoid the situations where an assumption is not highly plausible. It improves its precision index r^k if p^k or $1-p^k$ lies in the interval $[\beta_\ell, \beta_u] = [0.03\%, 99.7\%]$. DpMads is more tolerant, as it also considers the computational cost required to reach low standard deviations. It improves the precision if p^k or $1-p^k$ is in $[\beta_\ell, \beta_u] = [15\%, 85\%]$ but can reduce it if plausibility becomes too high (considering that plausibility of, say, 99.7% is more than strictly required to avoid biased convergence, because 90% plausibility is almost as viable and is cheaper to reach). With these precision ranges, MpMads only accepts the hypothesis "$f(x_2) < f(x_1)$" on the rightmost scenario of Figure 1, while DpMads accepts it on the two rightmost ones. These thresholds are chosen in concordance with the computation of uncertainty performed by the p-value: the difference δ between two estimates $f^k(x)$ and $f^k(y)$ is seen as an observation of an equivalent normal law centred on 0 and with standard deviation $\sigma = \sqrt{\sigma^k(x)^2 + \sigma^k(y)^2}$, and p^k is the probability to observe $x \leq \delta$ within this law. With the chosen thresholds, MpMads accepts a new incumbent only when $|\delta| \geq 3\sigma$, while DpMads accepts it when $|\delta| \geq \sigma$.

Let also C_{Dp} and C_{Mp} be two logical conditions about the precision index r^k:

$$C_{\text{Dp}}: \left\{p^k \in [\beta_\ell, \beta_u] \implies r^{k+1} > r^k\right\} \text{ and } C_{\text{Mp}}: \left\{ \begin{array}{c} C_{\text{Dp}} \textit{ is satisfied} \\ p^k \notin [\beta_\ell, \beta_u] \implies r^{k+1} = r^k \end{array} \right\}. \quad (5)$$

The first one, C_{Dp}, is the minimal requirement to make the algorithms working. It forces the precision index r^k to increase as soon as the indicator of the quality of the estimates (p^k) shows that the estimates are not precise enough. However, it does not dictate any behaviour in the other case $p^k \notin [\beta_\ell, \beta_u]$. Then, the precision could either increase, decrease or remain constant in this situation. The second condition, C_{Mp}, is more stringent. It also imposes to r^k to strictly increase if $p^k \in [\beta_\ell, \beta_u]$ but in addition, it forces r^k to remain constant in the other situation. Therefore, under the condition C_{Mp} the precision grows monotonically during the optimization process, while it is not necessarily the case when only C_{Dp} is satisfied.

The only difference between MpMads and DpMads relies on the $UpdateR(r^k, p^k)$ function which leads the evolution of the precision index r. At any iteration, in DpMads the function $UpdateR$ only needs to satisfy C_{Dp}, while in MpMads it needs to satisfy C_{Mp}. Therefore, MpMads is a specific variant of DpMads. With both algorithms the precision r^k is forced to increase as soon as the p-value p^k belongs to $[\beta_\ell, \beta_u]$. When p^k lies outside this range, MpMads forces r^{k+1} to remain unchanged (equal to r^k), while in DpMads the precision can also be increased, or even reduced. In other words, in MpMads the precision index remains constant until it is uncertain that the candidate x_c^k is better than x_s^k or not. In DpMads, regardless of the certainty of this comparison, the precision index can either increase or decrease. It increases, not necessarily by $r^{k+1} = r^k + 1$, if p^k is too close to 0.5, and decreases if it is too far from 0.5. Also, default values of β_ℓ and β_u are not the same on both algorithms. MpMads uses restrictive thresholds (0.03% and 99.7%) while in DpMads there is more flexibility (15% and 85% as default values).

DpMads and MpMads are formulated under these notations and concepts in Algorithm 2.

Algorithm 2 DpMads and MpMads structure

function Mads(f; x_*^0; ρ, β_ℓ, β_u)
  Initialise : $r^0 = 0, \mathcal{V}^0 = \emptyset, \delta_p^0 = 1, k = 1$
  while not(*Stopping criteria reached*) **do**
    $x_s^k, \mathcal{V}_{search}^{k-1} \leftarrow Search(\mathcal{V}^{k-1}, r^k)$
    $x_c^k, S^k, \mathcal{V}^k \leftarrow Poll\left(x_s^k, \delta_p^k, r^k, \mathcal{V}_{search}^{k-1}\right)$
    $p^k \leftarrow PValue(x_c^k, x_s^k, \mathcal{V}^k)$
    if $S^k = Success$ **then**
      $\delta_p^{k+1} \leftarrow \begin{cases} 2\delta_p^k & \text{if } p^k > \beta_u \\ \delta_p^k & \text{otherwise} \end{cases}$
      $r^{k+1} \leftarrow UpdateR(r^k, p^k)$, *satisfying* C_{MP} *for* MpMads, *or* C_{DP} *for* DpMads
    else if $S^k = Failure$ **then**
      $\delta_p^{k+1} \leftarrow \begin{cases} \delta_p^k/2 & \text{if } p^k < \beta_\ell \\ \delta_p^k & \text{otherwise} \end{cases}$
      $r^{k+1} \leftarrow UpdateR(r^k, p^k)$, *satisfying* C_{MP} *for* MpMads, *or* C_{DP} *for* DpMads
    else ($S^k = Barrier$)
      $\delta_p^k \leftarrow \delta_p^k/2$
      $r^{k+1} \leftarrow r^k$
    $x_*^k \in \arg\min_{y \in \mathcal{P}^k}(f^k(y))$
    $k \leftarrow k + 1$
  return x_*^k

2.4 Practical implementations of the DpMads and MpMads algorithms

This section gives additional information necessary to implement the algorithms from Section 2.3. As the adaptive precision program may be unable to propose an arbitrarily high standard deviation, then one may define $\sigma_{max} = \sup_{\sigma \geq 0}\{\sigma \mid f_\sigma(x) \text{ can be observed}\}$. One can also propose $\sigma_{min} > 0$ if one does not want the algorithms to ask for arbitrarily costly computation. Then, ρ has to satisfy $\lim_{r \to -\infty} \rho(r) = \sigma_{max}$ and $\lim_{r \to +\infty} \rho(r) = \sigma_{min}$. A parameter $\theta > 0$ is proposed to control the decrease rate (it can be seen as the attenuation of the noise magnitude, given in decibel). Also, a reference index r_0 is defined, such that $\rho(r_0) = \frac{\sigma_{min}+\sigma_{max}}{2}$. Then, the ρ function is given by

$$\rho : \begin{cases} \mathbb{R} & \longrightarrow & [\sigma_{min}; \sigma_{max}] \\ r & \longmapsto & \begin{cases} \sigma_{min} + \dfrac{\sigma_{max} - \sigma_{min}}{2} 10^{-(r-r_0)\theta} & \text{if } r \geq r_0 \\ \sigma_{min} + \dfrac{\sigma_{max} - \sigma_{min}}{2}\left(2 - 10^{(r-r_0)\theta}\right) & \text{if } r < r_0. \end{cases} \end{cases}$$

and is represented in Figure 2. Default values are $\sigma_{max} = 1, \sigma_{min} = 0, r_0 = 0$.

The optional function *Search* is disabled for MpMads. In DpMads, it is called with the internal parameter $r_s = -5$, and re-estimates only the points which have an high enough plausibility to appear better than the incumbent. In other words, DpMads's *Search* function at iteration k uses an observation with $\sigma_s = \rho(r^k - r_s)$ to update the estimates of

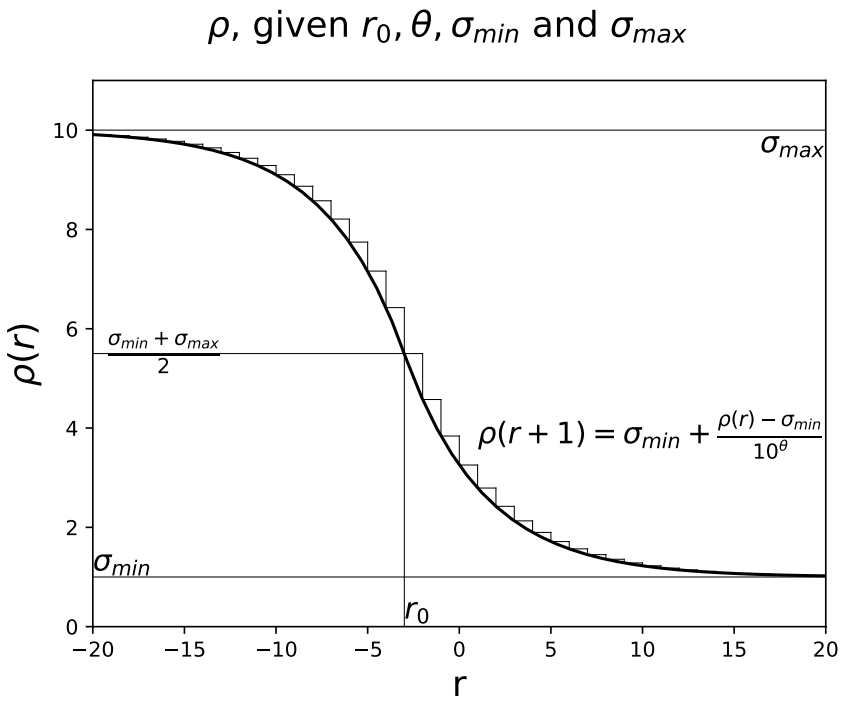

Figure 2: The ρ function with parameters $r_0 = -3$, $\theta = 0.1$, $\sigma_{min} = 1$, $\sigma_{max} = 10$.

all the points in the following set:

$$\left\{x \in \mathcal{V}^{k-1} \mid PValue\left(x, x_*^{k-1}, \mathcal{V}^{k-1}\right) \geq \tau\right\} \text{ (with by default } \tau = 0.25).$$

The last function needing to be described is the $UpdateR$ function. For DpMads, the only theoretical requirement is C_{Dp} : $r^{k+1} = UpdateR(r^k, p^k) > r^k$ if $p^k \in [\beta_\ell, \beta_u]$. An acceptable function is represented in Figure 3, using various thresholds on p to control the variations on r. For MpMads, $r^{k+1} \neq r^k \iff p^k \in [\beta_\ell, \beta_u]$ and if so, $r^{k+1} > r^k$ is mandatory. In Figure 3, the proposed function computes $r^{k+1} = r^k + 1$ as soon as $p^k \in [\beta_\ell, \beta_u]$, regardless of its value. However, some thresholds could be proposed to allow a non-unitary increasing of the precision index.

Modifying these practical parameters requires precaution. Some values of the σ_{min} parameter may lead the algorithms to fail it the *Search* function is disabled. Due to the algorithmic mechanics, an estimate $f^k(x)$ satisfying $\sigma^k(x) < \sigma^k$ is not re-evaluated by the *Poll* step. If the *Search* is enabled, it will perform such a re-evaluation. However, if it is disabled, the *Poll* step cannot reduce the standard deviation $\sigma^k(\cdot)$ lower than σ_{min}, making the convergence impossible to achieve. Then, σ_{min} have to be forced to 0 if the *Search* is disabled. This remark is especially important for MpMads, as the *Search* is disabled by default. Also, the parameter β_ℓ has to be strictly positive: $0 < \beta_\ell \leq 0.5$.

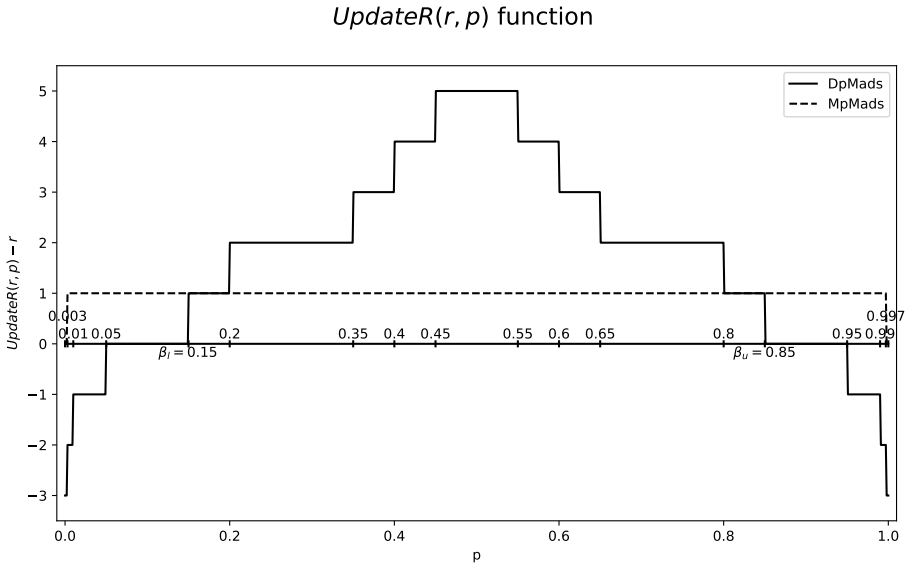

Figure 3: Default *UpdateR* function, given β_ℓ, β_u and some thresholds.

3 Convergence analysis

This section studies the convergence of both algorithms over the Problem (4), under an additional assumption that f is defined everywhere: $D_f = \mathbb{R}^n$, and has all its level sets bounded. It is also assumed that $\sigma_{min} = 0$ and that the sequence of precision indexes $(r^k)_k$ satisfies the condition C_{DP} from (5). The main idea of the proof is that in the absence of a stopping criteria, the algorithms generates a sequence of estimated optima x_*^k, with an accumulation point denoted x_*. This point almost surely satisfies local conditions based on the Clarke derivatives (defined in [14]) of the true objective function f.

3.1 Technical lemmas

This section gives some useful technical results. The first one defines an optimization problem to provide an upper bound for a quantity which appears in the proof of Theorem 3.7.

Lemma 3.1 (Maximum of a sum of products of variables)**.** *Let $n \geq 2$ be an integer and M the set of $n \times n$ matrices with all entries in the interval $[0, 1]$. Denote $\Phi_{i,j} \in M$ such a matrix (this notation is chosen in concordance with the proof of Theorem 3.7, on which some cumulative distribution of the centred reduced normal law are calculated). The problem*

$$\max_{\Phi \in M} \sum_{j=1}^{n} \prod_{i=1}^{n} \Phi_{i,j}$$
$$\text{subject to} \begin{cases} \Phi_{i,i} = 1 & \forall i \in [\![1; n]\!] \\ \Phi_{i,j} = 1 - \Phi_{j,i} & \forall i, j \in [\![1; n]\!]^2 \mid i > j. \end{cases}$$

has an optimal objective value equal to 1*.*

Proof. Denote $E = \{(i, j) \in \mathbb{N}^2 \mid 1 \leq i < j \leq n\}$ and f the objective function. The variables $\Phi_{i,j}$ with $(i, j) \notin E$ can be removed from the problem using the equality constraints, leading

to a bound-constrained reformulation:

$$\max_{\Phi_{i,j}\in[0,1]\ \forall(i,j)\in E} f(\Phi) = \sum_{j=1}^{n} \prod_{i<j} \Phi_{i,j} \prod_{i>j} 1-\Phi_{j,i}.$$

Denote also $\Pi_j = \prod_{i<j} \Phi_{i,j} \prod_{i>j} 1-\Phi_{j,i},\ \forall j \in [\![1;n]\!]$. The problem is therefore

$$\max_{\Phi_{i,j}\in[0,1]\ \forall(i,j)\in E} \sum_{j=1}^{n} \Pi_j.$$

First, assume all the variables are either 0 or 1. If any of the $\Pi_j = 1$ (say, $\Pi_b = 1$), all the other $\Pi_j, j \neq b$, are necessarily 0:

$$\begin{array}{rcl} \Pi_b = 1 & \Rightarrow & \Phi_{i,b} = 1\ \forall i < b \text{ and } \Phi_{b,i} = 0\ \forall i > b \\ & \Rightarrow & \Pi_j = 0\ \forall j \neq b. \end{array}$$

This proves that if all the variables are either 0 or 1, objective value is at most 1.

Second, one can prove that a solution with some variables in $]0,1[$ cannot be optimal. Denoting $\pi_{i,j} = \prod_{k<j,k\neq i} \Phi_{k,j} \prod_{k>j,k\neq i} 1-\Phi_{j,k},\ \forall i,j \in [\![1;n]\!]^2$, the partial derivatives of f are:

$$\frac{\partial f}{\partial \Phi_{i,j}} = \pi_{j,i} - \pi_{i,j}.$$

This does not depend on $\Phi_{i,j}$, so f is necessarily not optimal until every variables $\Phi_{i,j}$ are set to one of their bounds (0 or 1 depending on the sign of $\pi_{j,i} - \pi_{i,j}$). □

The two following Lemma 3.2 and 3.3 ensure that the estimated values f^k of the function f, over a finite set, respect the partial order defined by f:

Lemma 3.2 (Localisation of a minimum is consistent on finite sets)**.** *Let E be a set with a finite number of elements. Let $f : E \to \mathbb{R}$ be a function and $\forall x \in E, f^i(x)$ the maximum likelihood value of $f(x)$ constructed from the set $\mathcal{V}^i(x)$, under the assumption that it contains i elements. If $x_* \in \arg\min_{x\in E} f$ is the unique minimiser of f on E, then*

$$\mathbb{P}\left(\exists I \in \mathbb{N} : \forall i \geq I,\ x_* \in \arg\min_{x\in E} f^i(x)\right) = 1.$$

Proof. Denoting $E = \{x_1, \ldots, x_e, x_*\}$ (thus making $\#E = e+1$), one defines the optimality gaps $\mu_i = f(x_i) - f(x_*) > 0,\ \forall i \in [\![1;e]\!]$.

The strong law of large numbers ensures the convergence of any estimate to the true value it approximates: $\forall x \in E,\ \mathbb{P}\left(f^k(x) \underset{k\to+\infty}{\longrightarrow} f(x)\right) = 1$. It follows that

$$\begin{cases} \forall i \in [\![1;e]\!], & \mathbb{P}\left(\exists K_i : \forall k \geq K_i,\ f^k(x_i) > f(x_i) - \mu_i/2\right) = 1, \\ \text{with } \eta = \min_{i\in[\![1;e]\!]} \{\mu_i\}, & \mathbb{P}\left(\exists K_\eta : \forall k \geq K_\eta,\ f^k(x_*) < f(x_*) + \eta/2\right) = 1. \end{cases}$$

Then, the constant $K = \max\{K_1, \dots, K_e, K_\eta\}$ exists almost surely (as a maximum of a finite number of constants which all exists almost surely) and satisfies:

$$\forall i \in [\![1; e]\!],\ \forall k \geq K,\ f^k(x_i) > f(x_i) - \mu_i/2 \geq f(x_*) + \eta/2 > f^k(x_*)$$

which is equivalent to the result claimed by the Lemma:

$$\forall k \geq K,\ x_* \in \arg\min_{x \in E} f^k(x).$$

□

Lemma 3.2 can be extended if the images by the deterministic function f are all different:

Corollary 3.3 (Partial orders defined by the estimates $f^k(\cdot)$ are consistent on finite sets)**.** *With the notations from Lemma 3.2, assume the images $f(x)$ are all different (in the sense that $\forall (x, y) \in E^2 \mid y \neq x,\ f(x) \neq f(y)$). The estimates f^k eventually defines a coherent partial-order relation on E:*

$$\mathbb{P}\left(\exists K : \forall k \geq K,\ \forall x, y \in E^2,\ f(x) < f(y) \iff f^k(x) < f^k(y)\right) = 1.$$

Proof. Let $E = \{x_1, \dots, x_{e+1}\}$ with the elements x_i ordered as $f(x_1) \leq f(x_2) \leq \dots \leq f(x_{e+1})$. One can recursively apply Lemma 3.2 to E, then $E \setminus \{x_1\}$, then $E \setminus \{x_1, x_2\}$, ..., until $E \setminus \{x_1, x_2, \dots, x_e\}$. One obtains some constants denoted $K_0, K_1, \dots, K_e$. Then, $K = \sum_{i=0}^{e} K_i$ satisfies the desired definition (because K_0 iterations makes x_1 to be the minimiser over f^k, then K_1 makes x_2 to minimise the remaining f^k, and so on). □

Notice that neither Lemma 3.2 nor Corollary 3.3 ensures anything for points x and y such that $f(x) = f(y)$. The reason is when two points have the exact same image by f, the estimates $f^k(x)$ and $f^k(y)$ could satisfy either $f^k(x) < f^k(y)$ or $f^k(x) > f^k(y)$ for any iteration k. Estimates comparisons are therefore irrelevant when $f(x) = f(y)$.

It follows from Lemma 3.2 that if two points x and y are evaluated infinitely often, the p-value of hypothesis "$f(x) < f(y)$" knowing $\mathcal{V}^k$ will converge to 0 (if $f(x) < f(y)$) or 1 (if $f(x) > f(y)$).

Lemma 3.4 (Behaviour of the p-value used to compare points)**.** *Let x and y be two points satisfying $f(x) \neq f(y)$. Let $f^k(\cdot)$ be the maximum likelihood estimate of $f(\cdot)$ constructed with k observations: $\#\mathcal{V}^k(\cdot) = k$. Let p^k be the p-value of the hypothesis "$f(x) < f(y)$", knowing $\mathcal{V}^k$. This p-value satisfies:*

$$\begin{cases} f(x) < f(y) & \Rightarrow & p^k & \underset{k \to +\infty}{\longrightarrow} 1, \\ f(x) > f(y) & \Rightarrow & p^k & \underset{k \to +\infty}{\longrightarrow} 0. \end{cases}$$

Proof. Assume, without loss of generality, that $f(y) > f(x)$. let $\delta = (f(y) - f(x))/2$. Lemma 3.2 ensures that $\exists K \mid \forall k \geq K,\ f^k(y) - f^k(x) > \delta$. Therefore, denoting Φ the cumulative distribution function of the law $\mathcal{N}(0, 1)$,

$$\forall k \geq K,\ p^k = \Phi\left(\frac{f^k(x) - f^k(y)}{\sqrt{\sigma^k(x)^2 + \sigma^k(y)^2}}\right) < \Phi\left(\frac{-\delta}{\sqrt{\sigma^k(x)^2 + \sigma^k(y)^2}}\right).$$

Also, denote $(\sigma_1, \ldots, \sigma_k)$ the k-standard deviations of the k estimates of $f(x)$. As any of those σ_i is at most $\sigma_{max} = \lim_{r \to -\infty} \rho(r)$, one have:

$$\sigma^k(x) = \sqrt{\frac{1}{\sum_{i=1}^{k} 1/\sigma_i^2}} \leq \sqrt{\frac{\sigma_{max}^2}{k}}.$$

With a similar argument, $\sigma^k(y) \leq \sqrt{\frac{\sigma_{max}^2}{k}}$. Therefore,

$$p^k \leq \Phi\left(\frac{-\delta\sqrt{k}}{\sqrt{2}\sigma_{max}}\right) \underset{k \to +\infty}{\longrightarrow} 0,$$

which concludes the proof. □

In the specific case of two different points x and y such that $f(x) = f(y)$, the behaviour of p^k is different. Lemma 3.5 describes it:

Lemma 3.5 (Behaviour of the p-value for points with identical image)**.** *With notations from Lemma 3.4 and assumption $f(x) = f(y)$, p^k statistically satisfies:*

$$\forall \varepsilon \in [0, 0.5],\ \mathbb{P}\left(p^k > 0.5 + \varepsilon\right) = \mathbb{P}\left(p^k < 0.5 - \varepsilon\right) = 0.5 - \varepsilon.$$

Proof. The estimates $f^k(\cdot)$ statistically follow normal laws $\mathcal{N}\left(f(\cdot), \sigma^k(\cdot)^2\right)$. As $f^k(x)$ and $f^k(y)$ are independent, their reduced difference follows $\dfrac{f^k(x) - f^k(y)}{\sqrt{\sigma^k(x)^2 + \sigma^k(y)^2}} \sim \mathcal{N}(0, 1)$.

Also, recall that $p^k = \Phi\left(\dfrac{f^k(x) - f^k(y)}{\sqrt{\sigma^k(x)^2 + \sigma^k(y)^2}}\right)$. As such, at any iteration k its expected value is 0.5 but it can reach any value in $[0, 1]$. This ensures the result, because:

$$\forall \varepsilon \in [0, 0.5],\ \begin{cases} \mathbb{P}\left(p^k > 0.5 + \varepsilon\right) &= \mathbb{P}\left(\mathcal{N}(0,1) > \Phi^{-1}(0.5 + \varepsilon)\right) &= 0.5 - \varepsilon, \\ \mathbb{P}\left(p^k < 0.5 - \varepsilon\right) &= \mathbb{P}\left(\mathcal{N}(0,1) < \Phi^{-1}(0.5 - \varepsilon)\right) &= 0.5 - \varepsilon. \end{cases}$$

□

A crucial requirement of the MADS algorithm is that all generated trial points lie on the mesh $\mathcal{M}^k$. This assumption leads to the following lemma, which ensures that at any iteration k, the generated optimum x_*^k and any point evaluated by the algorithms lie on a given mesh.

Lemma 3.6 (All points evaluated up to iteration k lie on a given mesh)**.** *Denote, for iteration k, the smallest mesh parameter encountered up to iteration k by $\delta_{min}^k = \min_{i \leq k} \left\{\delta_p^i\right\}$. Then, the incumbent x_*^k, and any point evaluated ($\{x \mid \mathcal{V}^k(x) \neq \emptyset\}$) lie on a given mesh:*

$$\forall x \in \mathbb{R}^n \mid \mathcal{V}^k(x) \neq \emptyset,\ x \in \mathcal{M}_{\delta_{min}^k}\left(x_*^0\right).$$

Proof. The fact that $x_*^0 \in \mathcal{M}_{\delta_{min}^0}(x_*^0)$ holds trivially. Recursively, assume x_*^k and all the points generated up to iteration k are on the mesh. Remind also that δ_p^{k+1} is, by construction, on the form $2^{(s^k)}\delta_{min}^k$ with $s^k \in \mathbb{N}$. One can observe that iteration $k+1$ evaluates only x_*^k (which is assumed to be on the mesh), the candidates coming from $\mathcal{P}^{k+1}$, and potentially other points generated by the *Search* step (which all are, by construction, elements of the larger mesh $\mathcal{M}^{k+1} = \mathcal{M}_{\delta_m^{k+1}}(x_*^k) \subseteq \mathcal{M}_{\delta_{min}^{k+1}}\left(x_*^k\right)$). □

3.2 Convergence theorems

The next result shows that the sequence of incumbents remains bounded if all the level sets of f are bounded.

Theorem 3.7 (A bounded level set contains every visited points)**.** *Assume that all level sets of the objective function f are bounded. Let x_*^0 be the feasible starting point of an execution of one of the algorithms, and $\left(x_*^k\right)_k$ be the sequence of estimates generated during the optimization process. There exists, with probability one, a ball centred on x_*^0 which contains the entire sequence $\left(x_*^k\right)_k$:*

$$\mathbb{P}\left(\exists R > 0 : \forall k \in \mathbb{N},\ x_*^k \in B_R\left(x_*^0\right)\right) = 1.$$

Proof. The proof exploits the Borel-Cantelli Lemma, which provides:

> [Borel-Cantelli Lemma] Let $(A^i)_{i\in\mathbb{N}}$ be a sequence of events. There is a condition to determine if all but a finite number of these events are realised:
> $$\sum_{i=0}^{\infty} \mathbb{P}\left(A^i\right) < +\infty \implies \mathbb{P}\left(A^i \text{ infinitely often}\right) = 0.$$

Since f has bounded level sets $L_i = \{x \in \mathbb{R}^n \mid f(x) \leq f(x_*^0) + i\}$, one can define bounds d_i so that $L_i \subseteq B_{d_i}(x_*^0), \forall i \in \mathbb{N}$. Applying the Borel-Cantelli lemma to the following sequence of events:

$$\begin{aligned} \forall i \in \mathbb{N},\ A^i &= \{(x_*^k)_k \text{ has an element outside of } L_i\} \\ &= \{\kappa(i) < +\infty\},\ \text{with } \kappa(i) \text{ the lowest index } k \text{ such that } x_*^k \notin L_i, \end{aligned}$$

it is possible to prove that all but a finite number of these events happens, almost surely. Then, denoting ℓ the index of the last A^i which is realised, the ball $B_{d_\ell}(x_*^0)$ contains the whole sequence of incumbents. To show that $\sum_{i=0}^{\infty} \mathbb{P}(A^i)$ converges, one may compute the probability that the k^{th} incumbent is outside of L_i, knowing the cache $\mathcal{V}^k$:

$$\begin{aligned} \mathbb{P}\left(x_*^k \notin L_i \mid \mathcal{V}^k\right) &= \mathbb{E}_{\mathcal{V}^k}\left(\mathbb{1}_{L_i^c}\left(x_*^k\right)\right) \\ &= \sum_{x\in\mathcal{V}^k} \mathbb{1}_{L_i^c}(x)\, \mathbb{P}\left(x_*^k = x \mid \mathcal{V}^k\right) \\ &= \sum_{x\in\mathcal{V}^k} \mathbb{1}_{L_i^c}(x)\, \mathbb{P}\left(\forall y \in \mathcal{V}^k,\ f^k(y) \geq f^k(x) \mid \mathcal{V}^k\right) \\ &= \sum_{x\in\mathcal{V}^k} \mathbb{1}_{L_i^c}(x) \prod_{y\in\{x,x_*^0\}} \mathbb{P}\left(f^k(y) \geq f^k(x) \mid \mathcal{V}^k\right) \prod_{y\in\mathcal{V}^k\setminus\{x,x_*^0\}} \mathbb{P}\left(f^k(y) \geq f^k(x) \mid \mathcal{V}^k\right) \\ &= \sum_{x\in\mathcal{V}^k} \mathbb{1}_{L_i^c}(x)\, \Phi\left(\frac{f(x_*^0) - f(x)}{\sqrt{\sigma^k(x_*^0)^2 + \sigma^k(x)^2}}\right) \prod_{y\in\mathcal{V}^k\setminus\{x,x_*^0\}} \Phi\left(\frac{f(y) - f(x)}{\sqrt{\sigma^k(y)^2 + \sigma^k(x)^2}}\right). \end{aligned}$$

Any point x for which $\mathbb{1}_{L_i^c}$ is non-zero satisfies $f(x) \geq f(x_*^0) + i$, and all the evaluated points satisfies $\sigma^k(\cdot) \leq \sigma_{max}$. Then, denoting $\mathcal{V}_r^k = \mathcal{V}^k \setminus \{x_*^0\}$:

$$\mathbb{P}\left(x_*^k \notin L_i \mid \mathcal{V}^k\right) \quad \leq \quad \sum_{x \in \mathcal{V}_r^k} \mathbb{1}_{L_i^c}(x)\, \Phi\left(\frac{-i}{\sqrt{2\sigma_{max}^2}}\right) \prod_{y \in \mathcal{V}_r^k \setminus \{x\}} \Phi\left(\frac{f(y) - f(x)}{\sqrt{\sigma^k(y)^2 + \sigma^k(x)^2}}\right).$$

If all the points in $\mathcal{V}_r^k$ are in L_i, this expression is zero. Otherwise, let v^k be the number of points belonging to the cache which are outside of L_i and denote them by $\{x_1, \ldots, x_{v^k}\} = \mathcal{V}_r^k \cap L_i^c$.

Denote $\Phi_{i,j} = \Phi\left(\frac{f(x_i) - f(x_j)}{\sqrt{\sigma^k(x_i)^2 + \sigma^k(x_j)^2}}\right)$. The previous expression can be written as:

$$\mathbb{P}\left(x_*^k \notin L_i \mid \mathcal{V}^k\right) \quad \leq \quad \Phi\left(\frac{-i}{\sqrt{2}\sigma_{max}}\right) \sum_{i=1}^{v^k} \prod_{j \neq i} \Phi_{i,j}.$$

As $\Phi_{i,j} = 1 - \Phi_{j,i}$, this last expression is of the form studied in Lemma 3.1. Thanks to this Lemma, one can propose the following upper bound:

$$\mathbb{P}\left(x_*^k \notin L_i \mid \mathcal{V}^k\right) \leq \Phi\left(\frac{-i}{\sqrt{2}\sigma_{max}}\right) \times 1 \underset{i \text{ large enough}}{\leq} \exp\left(-i/\sqrt{2}\sigma_{max}\right).$$

Remind that $\kappa(i)$ denotes the index of the first iteration k for which $x_*^k \notin L_i$. Recall also that (A^i infinitely often) $\Leftrightarrow \forall i,\ \kappa(i) < +\infty$. Then, ($A^i$ infinitely often) if and only if there is a sequence $(\kappa(i))_i \in \mathbb{N}^{\mathbb{N}}$ of iteration indexes satisfying $\forall i,\ x_*^{\kappa(i)} \notin L_i$, knowing the set of evaluated points $\mathcal{V}^{\kappa(i)}$.

However, the sum $\sum_{i=0}^{+\infty} \mathbb{P}\left(x_*^{\kappa(i)} \notin L_i \mid \mathcal{V}^{\kappa(i)}\right) \leq \sum_{i=0}^{+\infty} \exp\left(-i/\sqrt{2}\sigma_{max}\right)$ converge. Then, the Borel-Cantelli Lemma ensures that each (A^i) happens infinitely often with probability zero, because there is probability zero that the sequence of $(\kappa(i))_i$ which generates (A^i) exists. □

Theorem 3.7 ensures that any execution of the algorithm generates a bounded sequence of incumbents. Therefore, the sequence of mesh size parameters necessarily approaches zero, in the following sense:

Theorem 3.8 (The mesh becomes refined infinitely often)**.** *Under the assumption that f has all its level sets bounded, the inferior limit of the sequence $(\delta_m^k)_k$ is almost surely zero:*

$$\mathbb{P}\left(\liminf_{k \to +\infty} \delta_m^k = 0\right) = 1.$$

Proof. Recall the connection between δ_m^k and δ_p^k: $\delta_m^k = \min(\delta_p^k, (\delta_p^k)^2)$. Recall also, from Theorem 3.7, that there almost surely exists a constant R such that $\forall k \in \mathbb{N},\ x_*^k \in B_R(x_*^0)$. If δ_m^k becomes too large, iteration k necessarily generates candidates outside of $B_R(x_*^0)$:

$$\delta_m^k > 2R \implies \mathcal{M}^k \cap B_R(x_*^0) = \{x_*^k\} \implies \mathcal{P}^k \cap B_R(x_*^0) = \emptyset.$$

Therefore, iteration k is not a success (otherwise, it would have generated a new incumbent x_*^{k+1} outside of $B_R(x_*^0)$). However, $\delta_m^{k+1} > \delta_m^k$ is impossible if iteration k is not a success. This ensures that the sequence $(\delta_m^k)_k$ has an upper bound δ_m^{sup}.

With this argument, one can deduce there is almost surely a ball ($B_{3R}(x_*^0)$) which contains all the points evaluated by the algorithm during the entire optimization process: $\forall k,\ \forall x,\ \mathcal{V}^k(x) \neq \emptyset \Rightarrow x \in B_{3R}(x_*^0)$.

Now assume, for the sake of contradiction, that there is a strictly positive minimal mesh size parameter, of index k_{min}: $\exists k_{min} \mid \forall k \in \mathbb{N},\ \delta_m^k \geq \delta_m^{k_{min}} = \delta_m^{min}$. Lemma 3.6 gives $\forall k,\ \forall x,\ \mathcal{V}^k(x) \neq \emptyset \Rightarrow x \in \mathcal{M}_{\delta_m^{min}}(x_*^0)$: the thinnest mesh contains all the evaluated points. One also have δ_p^{min} such that $\delta_m^{min} = \min(\delta_p^{min}, (\delta_m^{min})^2)$.

One can deduce that the algorithm evaluate a finite number of points, because the set of all points which could be visited is $B_{3R}(x_*^0) \cap \mathcal{M}_{\delta_m^{min}}(x_*^0)$, which is finite. Among this set, there is a subset E of points visited infinitely often.

Assume also the minimiser of f over E is unique. As E is finite, Lemma 3.2 is applicable, so there is almost surely an index κ for which $\forall k \geq \kappa$, the estimates $f^k(\cdot)$ and truth objective function $f(\cdot)$ share the same minimiser over E. Any iteration $k \geq \kappa$ is necessarily not a success, therefore two situations might happen. If $p^k \in [0, 0.5]$ becomes close enough to 0 ($p^k < \beta_\ell$), then $\delta_p^{k+1} = \delta_p^k/2$. Otherwise, precision index r^{k+1} becomes higher than r^k. As the sequence $(\delta_p^k)_k$ is assumed to be bounded by $\delta_p^{min} > 0$, the precision index r^k grows arbitrarily high (and then, standard deviation $\sigma^k(\cdot)$ of x_*^k and other elements from E becomes arbitrarily close to 0). However, due to Lemma 3.4, this implies $p^k \underset{k\to+\infty}{\longrightarrow} 0$. So there is an index ℓ such that $p^\ell < \beta_\ell$. This implies a contradiction: $\delta_p^{\ell+1} < \delta_p^\ell = \delta_p^{min}$.

If the minimiser of f over E is not unique, there is at least two points x_1 and x_2 for which $f(x_1) = f(x_2)$. The situation is analgous if there is more than two minimisers. Denote κ an index for which $\forall k \geq \kappa,\ \forall x \neq x_1, x_2,\ f^k(x) > f(x_1) = f(x_2)$. Denote also δ_m^{max} the largest mesh size such that $\{x_1, x_2\} \subset \mathcal{M}_{\delta_m^{max}}(x_1)$, which is necessarily smaller than δ_m^{sup}. At iteration $k \geq \kappa$, x_*^k can either be x_1 or x_2, depending on the values of $f^k(x_1)$ and $f^k(x_2)$. If δ_m^k becomes larger than δ_m^{max}, iteration k is necessarily not a success because the minimiser over $\mathcal{M}^k$ becomes unique. For $\delta_m^k \in \{\delta_m^{min}, 2\delta_m^{min}, \ldots, \delta_m^{max}\}$, the p-value p^k is driven by a behaviour described in Lemma 3.5. Then, $\delta_p^{k+1} = 2\delta_p^k$ with probability $1-\beta_u$, $\delta_p^{k+1} = \delta_p^k$ with probability $\beta_u - \beta_\ell$, and $\delta_p^{k+1} = \delta_p^k/2$ with probability β_ℓ. The sequence $(\delta_p^k)_k$ can therefore be seen as a stochastic process with the following behaviour:

$$\begin{cases} \delta_m^k = \delta_m^{max} \quad \Rightarrow \quad \delta_p^{k+1} = \begin{cases} \delta_p^k & \textit{with probability } 1-\beta_\ell, \\ \delta_p^k/2 & \textit{with probability } \beta_\ell, \end{cases} \\ \\ \delta_m^{min} \leq \delta_m^k < \delta_m^{max} \quad \Rightarrow \quad \delta_p^{k+1} = \begin{cases} 2\delta_p^k & \textit{with probability } 1-\beta_u, \\ \delta_p^k & \textit{with probability } \beta_u - \beta_\ell, \\ \delta_p^k/2 & \textit{with probability } \beta_\ell. \end{cases} \end{cases}$$

As such, results about stochastic processes ensures that $(\delta_p^k)_k$ reaches $\delta_p^{min}/2$ at least once, with probability one. This contradicts the definition of δ_p^{min}. □

Theorem 3.8 ensures that there exists an infinite number of iteration indexes $K \subseteq \mathbb{N}$ such that $\lim_{k\in K} \delta_m^k = 0$. Considering that the sequence $(x_*^k)_{k\in K}$ have all its elements included in a compact (the ball defined by Theorem 3.7), there exists another infinite set $L \subseteq K$ such that $(x_*^k)_{k\in L}$ converges. Then, following the definition of a *refining sequence* given in [3], $(x_*^k)_{k\in L}$ is a refining sequence, and its limit x_* is a refined point: the sequence $(x_*^k)_{k\in L}$ converges to x_* and $(\delta_m^k)_{k\in L}$ converges to 0.

Also, for any unitary direction d there is a subsequence of indexes such that d is a refining direction (as defined in [4]) for that subsequence: $\exists (d^k)_{k\in L}$ a sequence of directions such that $\frac{d^k}{\|d^k\|_2} \to d$ and $\forall k,\ x_*^k + d^k \in \mathcal{M}^k$ and $\|d^k\|_\infty \leq \delta_p^k$.

The following property is satisfied by x_*:

Theorem 3.9 (Limit point given by the optimization process satisfies optimality conditions)**.** *Recall it is assumed in this section that $D_f = \mathbb{R}^n$ and f has its level sets bounded. The optimization process almost surely generates at least one refined point x_* which satisfies:*

$$\mathbb{P}\left(\forall d \text{ unitary},\ f^\circ(x_*; d) \geq 0\right) = 1$$

where $f^\circ(x;d)$ is the Clarke-derivative of f at point x in the direction d, defined in [14].

Proof. Recall the definition of f° and Result 3.9 from [3]:

$$\begin{aligned} f^\circ(x,d) &= \limsup_{y\to x,\ t\searrow 0} \frac{f(y+td)-f(y)}{t} \\ &= \limsup_{y\to x,\ t\searrow 0,\ v\to d} \frac{f(y+tv)-f(y)}{t}. \end{aligned}$$

Denoting $d_u^k = d^k/\|d^k\|_2$ and $\delta^k = \|d^k\|_\infty$, one can deduce

$$\begin{aligned} f^\circ(x_*,d) &\geq \limsup_{k\to+\infty} \frac{f(x_*^k+\delta^k d_u^k)-f(x_*^k)}{\delta^k} \\ &= \lim_{k\to+\infty} \sup_{\ell\geq k} \frac{f(x_*^\ell+\delta^\ell d_u^\ell)-f(x_*^\ell)}{\delta^\ell}. \end{aligned}$$

Then, the proof is complete if the (random) refining sequence $(x_*^k)_{k\in L}$ satisfies

$$\mathbb{P}\left(\lim_{k\to+\infty} \sup_{\ell\geq k} \frac{f(x_*^\ell+\delta^\ell d_u^\ell)-f(x_*^\ell)}{\delta^\ell} \geq 0\right) = 1$$

or, equivalently

$$\mathbb{P}\left(\lim_{k\to+\infty} \sup_{\ell\geq k} f(x_*^\ell+\delta^\ell d_u^\ell)-f(x_*^\ell) < 0\right) = 0.$$

With an equivalent formulation of the lim sup, one can rewrite this as

$$\mathbb{P}\left(\exists \mu>0, \exists k\in\mathbb{N} \text{ such that } \forall \ell\geq k,\ f(x_*^\ell+\delta^\ell d_u^\ell)-f(x_*^\ell)\leq -\mu\right) = 0.$$

Recall that $\forall \ell,\ x_*^\ell + \delta^\ell d_u^\ell \in \mathcal{V}^\ell$ and $x_*^\ell \in \arg\min\{f^\ell(x),\ x\in\mathcal{V}^\ell\}$. As a consequence, x_*^ℓ satisfies $f(x_*^\ell+\delta^\ell d_u^\ell)-f(x_*^\ell)\leq -\mu$ if the following event is realised:

$$A_\mu^\ell : f^\ell(x_*^\ell) < f^\ell(x_*^\ell+\delta^\ell d_u^\ell),\ \text{knowing } f(x_*^\ell+\delta^\ell d_u^\ell)\leq f(x_*^\ell)-\mu.$$

However, recall also that $f^\ell(\cdot)\sim\mathcal{N}\left(f(\cdot),\sigma^\ell(\cdot)^2\right)$ and the $f^\ell(\cdot)$ are independent. So:

$$\mathbb{P}(A_\mu^\ell) \leq \mathbb{P}\left(X<0 \mid X\sim\mathcal{N}\left(\mu, 2\sigma_{max}^2\right)\right) = \Phi\left(\frac{-\mu}{\sqrt{2}\sigma_{max}}\right) < 1.$$

Then, $\forall \mu>0$, there is almost surely all but a finite number of A_μ^ℓ which are realised. So the result holds because there is a probability zero that a threshold $\mu>0$ satisfies the condition. □

Theorem 3.9 concludes the convergence analysis. As a summary, assuming that:

- f is defined on $\mathbb{R}^n$ entirely (although this could be restricted to x_* relying in the interior of D_f), and all the level sets of f are bounded,
- the lower bound σ_{min} of the precision function ρ is zero,
- the algorithmic parameters β_ℓ and σ_{max} satisfies $0 < \beta_\ell \leq 0.5$ and $\sigma_{max} < +\infty$,
- the evolution of the precision index r^k satisfies the C_{DP} condition in (5),

any algorithm following the structure introduced in Algorithm 2 with the minimal requirement (C_{DP}) on the *UpdateR* function generates a refined point x_* satisfying the Clarke necessary optimality conditions.

4 Computational study

This section compares an implementation of the algorithms DpMads and MpMads with the NOMAD [20] implementation of the Robust-Mads algorithm. A first set of comparisons are presented on two analytical problems. Then, Section 4.3 compares the algorithms on a real and computationally expensive Monte-Carlo stochastic problem. DpMads and MpMads are implemented in Python 3.6, while Robust-Mads implementation is taken from NOMAD 3.9.1. The first two tests, performed on analytical functions, are executed on an Intel® Core™ i5-8250U CPU @ 1.60GHz with 8 cores and 8GB of RAM.

4.1 Comparison of stochastic algorithms

Comparisons of algorithms are commonly performed using some tools like the *performance profiles* [16], *data profiles* [22] and *accuracy profiles* [10]. However, these profiles are inappropriate in an adaptive precision context. Usually, their discriminating criteria is the number of blackbox calls. This metric is irrelevant in adaptive precision context, as a few calls with great precision can be more expensive than many calls with low precision. One needs to adapt these profiles to a relevant metric: the computational effort. This is considered through th Monte-Carlo draws consumption. Two situations can arise in any adaptive precision problems. If the noise magnitude is chosen directly by a number of Monte-Carlo draws, then the metric is trivially set to that number. Otherwise, one may create a fictive Monte-Carlo simulation which gives equivalent results for a given number of draws. This exploits a well-known approximation of Monte-Carlo estimates : denoting $\widetilde{\mathcal{A}_N}$ an estimate of $\mathcal{A}$ coming from N Monte-Carlo draws, there exists a constant C such that $\widetilde{\mathcal{A}_N} \sim \mathcal{N}(\mathcal{A}, C/N)$. Then, considering $C = 1$ for simplicity, an estimate obtained with a standard deviation σ can be interpreted as the result of a Monte-Carlo simulation with $N = 1/\sigma^2$ draws. Thus, $1/\sigma^2$ is a metric which can be interpreted as a Monte-Carlo draws consumption. Former profiles are modified with this new metric. The fundamental object they all use is the accuracy of a given algorithm within a given budget :

$$f_{acc}^N = \frac{f\left(x^N\right) - f\left(x_*^0\right)}{f\left(x_*\right) - f\left(x_*^0\right)},$$

where x_*^0 is the initial point, x^N the incumbent found with a budget of N Monte-Carlo draws, x_* is the optimum of f, and $f(\cdot)$ is the true value of points (if known) or its estimated

value $f^k(\cdot)$ otherwise. It is therefore possible to determine the minimal budget required by an algorithm a to solve a problem p with a given tolerance τ. The following formula gives this budget: $N_{a,p} \in \underset{N\in\mathbb{R}^+}{\arg\min} \left\{ f_{acc}^N \mid f_{acc}^N \geq 1-\tau \right\}$ if such N exists, and $+\infty$ otherwise. Although it is not used in the following graphs, an alternative formula giving the decimal logarithm of this budget could be considered: $N_{a,p}^{log} \in \underset{N\in\mathbb{R}^+}{\arg\min} \left\{ f_{acc}^{10^N} \mid f_{acc}^{10^N} \geq 1-\tau \right\}$.

With these quantities, *performance* and *accuracy profiles* can be constructed in a way similar to their deterministic equivalent. However, *data profiles* have to be more deeply modified. With the original profiles, the abscissa represents a number of calls divided the number of variables ($\frac{k}{n+1}$). As a positive basis of $\mathbb{R}^n$ requires at least $(n+1)$ vectors, $\frac{k}{n+1}$ represents the number of positive bases that could have been created within a budget of k blackbox calls. As this is no longer relevant, the profiles are modified. Remind that to guarantees a given standard deviation σ to the output of a blackbox call, $N = 1/\sigma^2$ draws are required. The modified data profiles, defined for a reference standard deviation, represents in abscissa the quantity $\frac{k}{N}$, the number of estimates at guaranteed standard deviation σ which could have been computed within a budget of k draws.

4.2 Analytical problems

The first analytical problem discussed here, named Norm2, is easy to solve in the deterministic context. It is used to compare algorithms during the intensification close to an optimum.

$$\min_{(x,y)\in\mathbb{R}^2} \|(x,y)\|_2 \text{ with } \left(x_*^0, y_*^0\right) = \left(\pi^2, e^2\right).$$

Its noisy equivalent applies a noise $\mathcal{N}\left(0,\sigma^2\right)$ at any computation of $\|(x,y)\|_2$, with σ decided by algorithms. The equivalent number of Monte-Carlo draws is $N = 1/\sigma^2$. The stopping criteria is related to the frame parameter: $\delta_p < 10^{-10}$. The noisy problem is:

$$\min_{(x,y)\in\mathbb{R}^2} \left(\lim_{\#\mathcal{V}^k(x,y)\to+\infty} f^k(x,y) \right) \text{ with } \begin{cases} \left(x_*^0, y_*^0, \delta_p^{min}\right) = \left(\pi^2, e^2, 10^{-10}\right), \\ f^k \text{ and } \sigma^k \text{ defined as in Section 2.1} \\ \text{via } f_\sigma(\cdot) = \|\cdot\|_2 + \mathcal{N}\left(0,\sigma^2\right). \end{cases} \tag{6}$$

Figure 4 shows convergence versus consumption. Robust-Mads is used with the standard deviation fixed to $\sigma = 10^{-10}$. Preliminary tests shows that the algorithm struggle to reach an objective value lower than $\sigma/10$, which is lower than the chosen stopping criteria. One can observe that all Robust-Mads runs are close: it always reaches an objective function value of 10^{-10} after approximately 3.8×10^{22} draws and cannot intensify more because σ becomes high compared to the small variations of f around the optimum. Meanwhile, DpMads and MpMads successfully goes closer to the optimum. However, DpMads seems more reliable than MpMads: at a given budget it reachs a lower objective. Also, all its runs converges within an equivalent budget (10^{21} to 10^{23} draws) while some of MpMads runs requires up to 10^{28} draws.

This is also shown by the profiles in Figure 5. The accuracy profiles show that 10^{23} draws is the minimal budget required to make all the algorithms to converge at good precision (while a lower budget makes Robust-Mads to fail and MpMads to be dominated by DpMads). With the performance and data profiles, it appears that for any tolerance greater

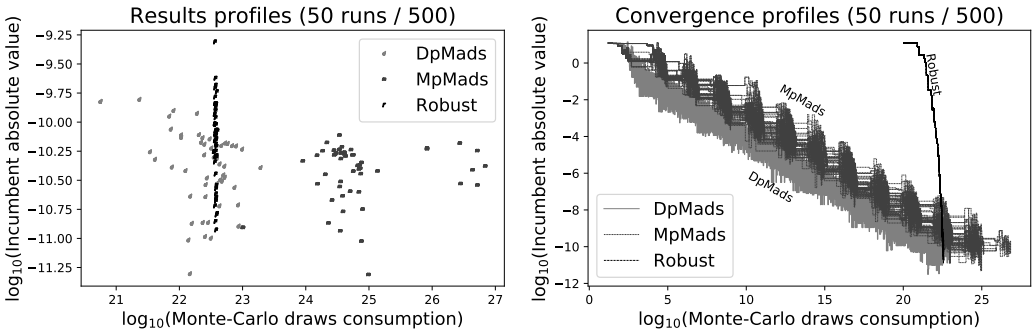

Figure 4: "Norm2" - (a) Results, (b) Monte-Carlo convergence profiles.

than 10^{-13}, DpMads outperforms the other two algorithms, notably around $\tau = 10^{-10}$ or 10^{-11}. The precision reference for the data profiles is 10^6 draws ($\sigma = 10^{-3}$).

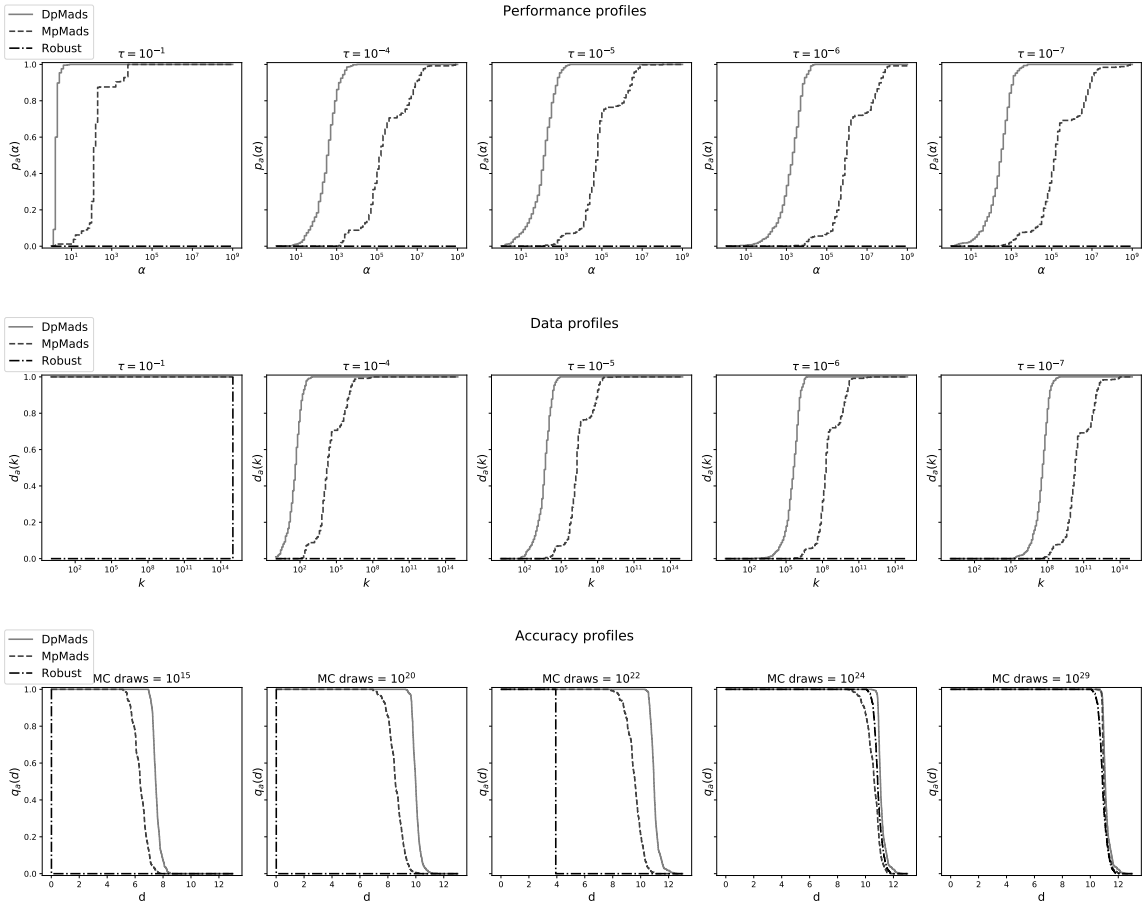

Figure 5: "Norm2" - (a) Performance profiles, (b) Data profiles, (c) Accuracy profiles.

The second analytical problem, denoted "Moustache", aims to compare the algorithms during their exploration process, in a restrictive space of variables. The domain is the thin

region illustrated in Figure 6.

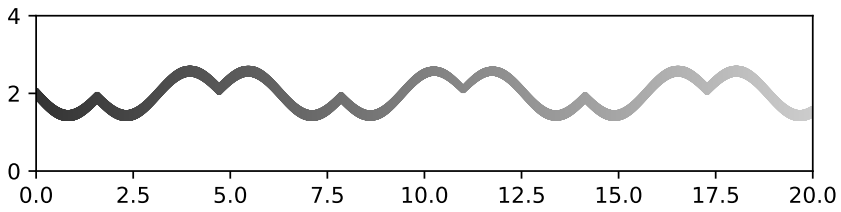

Figure 6: The domain of the "Moustache" function.

Starting from the feasible point $(0, 2)$, the objective is to maximise x (thus minimise $f(x, y) = -x$), with f not defined outside of the tight ribbon. At a given x, the interval of the admissible y values is denoted $I(x)$, defined as:

$$\begin{aligned} I(x) &= [g(x) - \varepsilon(x), g(x) + \varepsilon(x)], \\ g(x) &= -(|\cos(x)| + 0.1)\sin(x) + 2, \\ \varepsilon(x) &= l_{min} + (l_{max} - l_{min})\left(1 - \frac{1}{1 + |x - x_m|}\right), \\ (l_{min}, l_{max}, x_m) &= (0.05, 0.1, 11). \end{aligned}$$

The noise appears at the computation of $-x$ in the objective. This does not mean that the variable x itself is noisy, the noise appears when the value $-x$ is returned by the objective function. The equivalent Monte-Carlo consumption is $N = 1/\sigma^2$. The problem is:

$$\min_{(x,y)\in\mathbb{R}^2} \left(\lim_{\#\mathcal{V}^k(x,y)\to+\infty} f^k(x, y)\right) \quad \text{with} \quad \begin{cases} \left(x_*^0, y_*^0, \delta_p^{min}\right) = (0, 2, 10^{-5}), \\ f^k \text{ and } \sigma^k \text{ defined as in Section 2.1 via} \\ f_\sigma(x, y) = -x + \mathcal{N}(0, \sigma^2). \end{cases} \quad (7)$$
$$\text{s.t.} \quad (x, y) \in [0, 20] \times I(x)$$

Robust-Mads is run with $\sigma = 0.001$, a trade-off between quality of results (higher σ fails more often) and consumption ($\sigma \leq 0.0005$ consumes at least 10^{10} Monte-Carlo draws but overall the results are equivalent). The problem is solved by DpMads every time. As Figure 7 shows, the optimal value -20 is always reached, within a budget of up to 10^{11} Monte-Carlo draws. DpMads appears to be robust to random effects: a budget of 10^7 draws is actually always sufficient to solve the problem, the remaining is used trying to intensify around the frontier $x = 20$. MpMads consumes more (from 10 to 10^3 times more computational efforts) and has fewer guarantees of quality: it fails once to reach -20.

This analysis can be recovered from the profiles in Figure 8. On the performance profiles, one can observe that DpMads and MpMads solves the problem every time with a tolerance $\tau \geq 10^{-6}$ and starts to fail at $\tau = 10^{-7}$. However, MpMads struggles to reach the optimum as fast as DpMads. The data profiles with reference precision of 10^6 draws ($\sigma = 10^{-3}$) show that DpMads requires less effort to reach the optimum at a given tolerance than MpMads. All Robust-Mads runs require an equivalent computational effort regardless of the tolerance. When it reachs the optimum, it performs well on accuracy profiles and attains the optimum at very low tolerance. Because of the fact that the optimum is at the frontier of D_f, the algorithms generates numerous points at x close to 20. Then, the smoothing effect helps Robust-Mads to improve its incumbent. The other two algorithms do not have any smoothing effect, then they struggle to intensify as much as Robust-Mads.

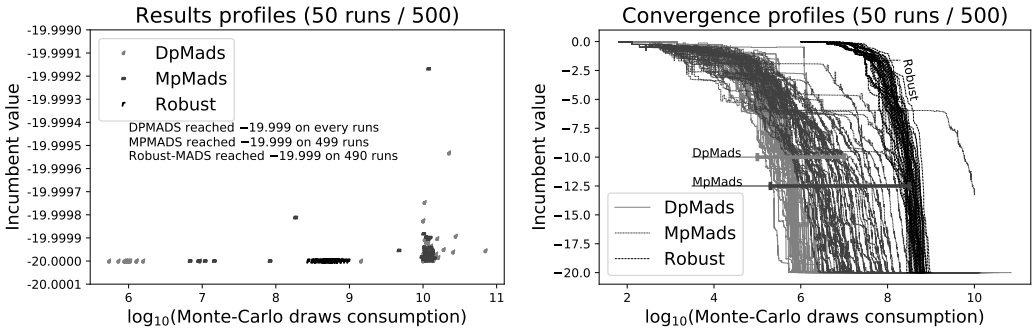

Figure 7: "Moustache" - (a) Results, (b) Monte-Carlo convergence profiles.

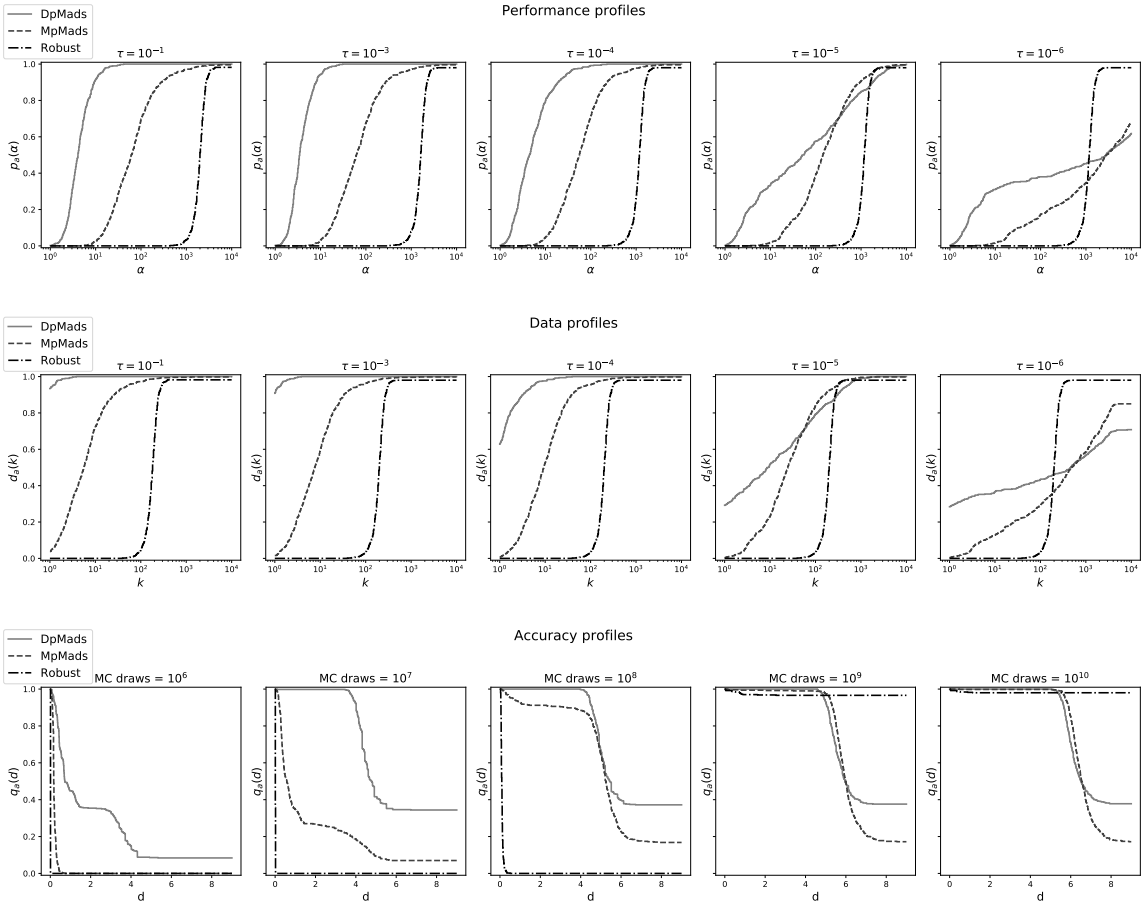

Figure 8: "Moustache" - (a) Performance profiles, (b) Data profiles, (c) Accuracy profiles.

Eventually, the DpMads and MpMads accuracy profiles decreases at $d = 5$, because reaching higher values of d means reaching 20 at a distance smaller than 10^{-5} (the stopping criteria on δ_p). This is made difficult because at such low distance between two points, the values of their images are close, so it becomes hard to make estimates sufficiently accurate.

4.3 Asset Management Problem

The two algorithms proposed in the present work are now compared with Robust-Mads on an asset management problem from [11]. The instance considered has four assets relying on the same spare stock in case of failure. The replacement date needs to be determined for each asset, and the acquisition date needs to be identified to replenish the stock. The five dates must be from a 350-month horizon, which corresponds to a little more than 29 years, to maximise the expected Net Present Value (NPV). Given a set of fixed dates, the NPV is computed using a tool called VME, developed by EDF R&D to evaluate asset investments [21]. VME is a discrete-event simulator where an asset management strategy is tested against asset failures that are randomly generated by Monte-Carlo methods.

The problem stated above is much more difficult than the one in [11] where the variables were annual instead of monthly, and covered a 10-year horizon.

Let $x = (a_1, a_2, a_3, a_4, s)$ be a solution of the above problem where a_i is the replacement date for the asset $i \in \{1, 2, 3, 4\}$ and s the date at which a new spare is added to the stock. Each of the five variables are integer and take any value from 1 to 350.

Computational experiments are performed on a HP Z420 Workstation, Intel Dual-Core Xeon E5-1620 @ 3.60GHz, RAM 32.0 Go, 64 bits, Windows 7 Pro SP1. The initial solution for all experiments is $x_*^0 = (240, 240, 132, 240, 120)$.

4.3.1 Preconditioning

In VME, the source of stochasticity lies a complex Monte-Carlo simulation over a so-called "tree of scenarios". Since the true value of the objective funciton is defined as the expected negative cashflow induced by a given choice of variables, knowing any potential scenario and its probability, it is difficult to recover the exact law followed by the noise. Therefore, some proactive choices have been made while analysing the problem.

Intense tests showed that the following law makes an acceptable approximation of the noise law. It overestimates its magnitude on some points x but never underestimates it.

The blackbox receives a value σ from the algorithms and translates it to a number N of Monte-Carlo draws. The tests showed that $N = 2^{10}$ leads to a standard deviation $\sigma = 1800$, and σ is divided by 2 when N is multiplied by 4. Denoting $f_N(\cdot)$ the Monte-Carlo approximation run by the blackbox with N draws, the law of the noise is approximated by:

$$f_\sigma(x) \sim \mathcal{N}\left(f(x), \sigma^2\right) = f_N(x) \quad \text{with} \quad N = 2^{10-2\log_2(\sigma/1800)} = \frac{2^{10} \times 1800^2}{\sigma^2}.$$

Then, when the optimization algorithms runs with a given value of σ, the blackbox computes the corresponding number N of draws, then performs a Monte-Carlo simulation with these N draws and returns the output to the algorithm. This value can be interpreted by the algorithm as an observation of $\mathcal{N}\left(f(x), \sigma^2\right)$, as required. The least value of N is set to 1024 and therefore $\sigma_{max} = 1800$.

During these tests, an hidden constraint may be triggered. The program restrict the number of Monte-Carlo draws to be at most $2850812 \simeq 2^{21.4429}$, and does not run any attempt to use more than this number. As a consequence, the standard deviation of the noise cannot fall arbitrarily close to zero: the smallest possible value of σ is $\sigma_{min} = 34.1144$. Thus, using monotonic strategies such ad MpMads, the user needs to ensure that the precision is unlikely to grow too high, otherwise the algorithm may eventually ask for a

σ which cannot be computed by the blackbox. This restriction contributes to make the dynamic strategy interesting, as DpMads naturally avoids to increase the precision as much as possible. To avoid problems, the value $\sigma_{min} = 50$ is chosen on the tests.

The following tests compare DpMads, MpMads and Robust-Mads, using this preconditioned formula, and an implementation of the deterministic Mads algorithm. For Robust-Mads and Mads, the fact that the variables are all integer is integrated through the *Granular Mesh* proposed in [5]. For DpMads and MpMads, a lower bound on the δ_m mesh size is implemented and set to 1, such that $(\delta_m^k)_k$ remains above this bound: $\delta_m^{k+1} \leftarrow \delta_m^k/2$ is not computed if $\delta_m^k = 1$.

4.3.2 Results on the asset management problem

For the deterministic algorithm, a fixed number of draws per evaluation is fixed a priori and the algorithm runs as if the computation of f_N were exact. Table 1 gives results using the NOMAD [20] implementation of Mads, version 3.9.1, with neither models nor anisotropic mesh, and the speculative search turned off. For comparisons, replications of runs were made on the instances that required less than two weeks to complete. One can observe the influence of the noise: with few draws per evaluation, objective function values $f^k(x_*)$ are high (around 8700) while the runs with more draws propose reach objective values near 5400. As this is a maximisation problem, optima on the first runs are overestimated by approximately 3000 units.

draws / eval	x_*					$f^k(x_*)$	$\sigma^k(x_*)$	Evals	Time (s)	Time
1024	261	288	120	250	107	8668.33	1800	1889	1891	32 m
1024	262	292	129	240	112	9005.12	1800	2050	2055	34 m
1024	247	345	144	289	122	8371.24	1800	1891	1910	32 m
10000	272	336	121	248	111	6140.15	576	2594	17636	5.9 h
10000	247	332	84	209	66	5893.69	576	2247	15724	4.4 h
10000	257	286	117	301	97	6209.33	576	3319	22419	6.2 h
100000	267	281	119	229	108	5623.74	182	3268	214310	2.5 d
100000	259	297	121	245	108	5630.17	182	2316	148752	1.7 d
100000	260	304	125	248	112	5622.03	182	3582	229966	2.7 d
500000	251	296	120	224	105	5431.04	81.5	2985	961232	11.1 d
500000	268	243	133	213	119	5470.23	81.5	2324	736527	8.5 d
1000000	257	292	130	226	118	5437.37	57.6	3403	2187009	25.3 d

Table 1: Mads (NOMAD 3.9.1).

Robust-Mads also requires a fixed number of draws per evaluations. Table 2 reports results obtained with the NOMAD implementation of Robust-Mads without the anisotropic mesh and with no speculative search. The column labelled $f_\sigma(x;\omega)$ represents the value computed by the blackbox on x, and $f^k(x)$ the estimated smoothed value proposed by Robust-Mads. Due to the Robust-Mads mechanics, the cache $\mathcal{V}^k(x)$ is either empty or contains a single observation. However, the precision $\sigma^k(x_*)$ is intractable because of the smoothing. One can observe that the smoothed values are coherent, because regardless of the number of draws, the proposed smoothed values are all close to 5500.

draws / eval	x_*					$f_\sigma(x_*;\omega)$	$f^k(x_*)$	Evals	Time (s)	Time
100	268	306	145	191	132	11313.60	5601.01	36526	14557	4.0 h
100	237	229	115	266	103	15456.60	5576.94	29707	12978	3.6 h
100	300	321	145	245	134	11508.50	5803.96	36678	16728	4.7 h
1024	257	273	132	208	119	7850.24	5434.91	21390	21507	5.9 h
1024	267	247	124	295	112	9024.06	5276.66	22056	22338	6.2 h
10000	261	283	133	221	119	6138.82	5367.90	22168	152875	1.8 d
100000	275	261	134	230	121	5589.82	5352.53	21847	1408037	16.3 d

Table 2: Robust-MADS (NOMAD 3.9.1).

These results shows that if the number of draws per evaluation is fixed a priori, it needs to be high. With the accuracy gain provided by Robust-Mads' smoothing effect, this number can remain close to 10^5 but then, the computation time is important (16.3 days). These observations, combined with the estimated optimal solution, $x_* = (275, 261, 134, 230, 121)$ with $f(x_*) \simeq 5352$, are used for comparisons with DpMads and MpMads.

For these two algorithms, preliminary tests showed that the *UpdateR* function has a considerable influence. Since the noise magnitude is initially very high, the function need to allow a fast improvement of the precision index. Otherwise (as it appeared on the runs with such a situation), numerous iterations are performed at very low precision and then, the cache $\mathcal{V}^k$ becomes large and full of imprecise estimates which all are re-estimated by the *Search*. This process consumes a noticeable amount of draws. The following results are generated by DpMads with default parameters except that the threshold τ in the *Search* is increased to 40%, and by MpMads with an extended *UpdateR* function allowing the precision increase to exceed one unit. Figure 9 illustrates one run for the three non-deterministic algorithms. The shaded area around the curves depicts the estimated standard deviation of the incumbent values. All the other runs have similar results.

The figure shows that DpMads reaches a nearly constant incumbent function value after 10^6 draws, while Robust-Mads stabilises only after 10^8 draws. Also, the standard deviation of DpMads' incumbent is close to 100 while DpMads stabilises around it. All the computational efforts performed after are dedicated to the reduction of the standard deviation, with no change of incumbent. After 10^7 draws, because of the search and the precision index going high, the incumbent and other quasi-optimal solutions have a standard deviation $\sigma^k < 10$. In summary, to reach the optimal solution, Robust-Mads requires 100 times more draws than DpMads. DpMads uses the remaining budget to intensively analyse the most promising solutions. A detailed analysis of the precision index r shows that it starts to improves strongly at that point of the optimization. Therefore, the *Poll* step (Algorithm 1) does not contributes anymore to the optimization, because all the candidates already satisfies $\sigma^k(x) < \sigma^k = \rho(r^k) \simeq \sigma_{min} = 50$. Then, behind 10^7 draws consummated, estimates improvements are performed by the *Search* step only.

The MpMads curve shows an analogous behaviour to DpMads. However, one can note the drop in the incumbent value from 10^5 to 10^7 draws. A more detailed analysis reveals that MpMads encountered a nearly optimal basin of solutions and spent many draws exploring it. MpMads is affected by this effect on almost all of its runs, while DpMads also visited this basin but did not waste as many draws in exploring it. A

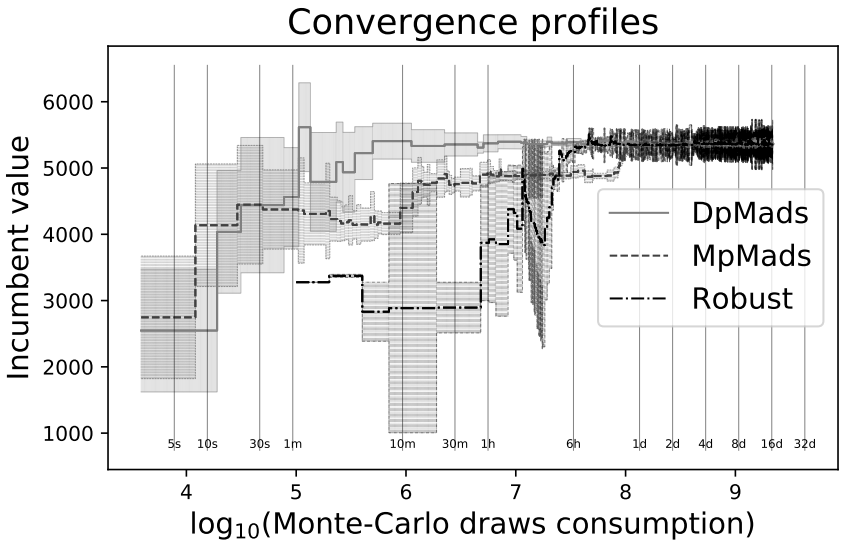

Figure 9: Convergence graph, with estimated value of the incumbent in ordinate. The length of the vertical range around a curve is twice the estimated standard deviation of the incumbent value.

possible explanation, recovered from the detailed logs, is that the low precision used by DpMads actually helps it to avoid this basin, while MpMads has a precision high enough to detect that it is interesting.

Figure 9 may also be used to compare the quality of the solutions in terms to time elapsed. After 10 minutes of computing, DpMads has found a nearly optimal solution while MpMads is still exploring near the 4000 mark, and Robust-Mads is way below at 3000. It takes an entire day of computing for the three methods to reach a comparable solution.

5 Discussion

This work compares two generic strategies to optimize noisy problems. The monotonic MpMads strategy avoid uncertainty as much as possible, leading to highly plausible data anytime in the optimization process (such as the localisation of the incumbent) with the drawback of an important computational effort. Following a different paradigm, the dynamic precision DpMads algorithm reduces the computational effort per iteration but faces more uncertainty as a consequence. A noticeable point is the theoretical equivalence of these two strategies, as they share the same convergence analysis.

However, these two strategies behave differently in practical contexts. Both algorithms outperform strategies which do not exploit the adaptive precision. The DpMads algorithm outperforms MpMads on the test problems considered in this work, in the sense that it reaches an higher quality solution within a lower computational budget. Also, the two algorithms should not be considered for the same usage. Within a prescribed computational budget, the dynamic strategy tends to explore more solutions than the monotonic one. However, with the monotonic paradigm, the smaller set of evaluated points is well-known,

in the sense that the estimated objective function value is more precise for all these points.

It should be noted that some improvements can be developed. Parameter values and implementation choices could be challenged. Notably, one could define some specific parameter values for given families of problems. In addition, precision growing to infinity leads to extremely long computational time in practical contexts, therefore the monotonic strategy needs to be used with precaution. Meanwhile, the dynamic strategy struggles on problems with a flat objective function because it avoid as much as possible to improve the precision. Also, usage of the precision index could be made more flexible: for example, its value could be modified at every evaluation rather than every iteration.

For future research, an important conceptual step would be the possibility to solve constrained problems, with constraints affected by an adaptive precision noise. Generalisation of the law followed by the noise, from centred normal to generic, could also be considered. The theory could also be enhanced with the addition of models to predict the behaviour of the objective.

Overall, the dynamic strategy could be chosen when one desires numerous solutions, while the monotonic strategy should be considered if one prefers fewer solutions but with an high precision on each. This comes from the fact that the dynamic strategy is designed to limit as much as possible the consumption of the computational budget per solution, while the monotonic allows more efforts per solution in order to rapidly identify the interesting ones.

Acknowledgements

Thanks to the research and development team at EDF for sharing the asset management problem, and NSERC for its support through the CRD grant (#RDCPJ 490744-15) with Hydro-Québec and Rio Tinto.

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] S. Amaran, N.V. Sahinidis, B. Sharda, and S.J. Bury. Simulation optimization: A review of algorithms and applications, 2017.

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

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

[5] C. Audet, S. Le Digabel, and C. Tribes. The mesh adaptive direct search algorithm for granular and discrete variables. Technical Report G-2018-16, Les cahiers du GERAD, 2018.

[6] C. Audet, K.J. Dzahini, S. Le Digabel, and M. Kokkolaras. Constrained stochastic blackbox optimization using probabilistic estimates. Technical Report G-2019-30, Les cahiers du GERAD, 2019.

[7] C. Audet and W. Hare. *Derivative-Free and Blackbox Optimization*. Springer Series in Operations Research and Financial Engineering. Springer International Publishing, Berlin, 2017.

[8] C. Audet, A. Ihaddadene, S. Le Digabel, and C. Tribes. Robust optimization of noisy blackbox problems using the Mesh Adaptive Direct Search algorithm. *Optimization Letters*, 12(4):675–689, 2018.

[9] F. Augustin and Y.M. Marzouk. A trust-region method for derivative-free nonlinear constrained stochastic optimization. *arXiv preprint arXiv:1703.04156*, 2017.

[10] V. Beiranvand, W. Hare, and Y. Lucet. Best practices for comparing optimization algorithms. *Optimization and Engineering*, 06 2017.

[11] T. Browne, B. Iooss, L. Le Gratiet, J. Lonchampt, and E. Remy. Stochastic simulators based optimization by Gaussian process metamodels - Application to maintenance investments planning issues. *Quality and Reliability Engineering International*, 32(6):2067–2080, 2016.

[12] K.H. Chang. Stochastic nelder-mead simplex method - a new globally convergent direct search method for simulation optimization. *European Journal of Operational Research*, 220(3):684–694, 2012.

[13] X. Chen and C.T. Kelley. Optimization with hidden constraints and embedded Monte Carlo computations. *Optimization and Engineering*, 17(1):157–175, 2016.

[14] F.H. Clarke. *Optimization and Nonsmooth Analysis*. John Wiley & Sons, New York, 1983. Reissued in 1990 by SIAM Publications, Philadelphia, as Vol. 5 in the series Classics in Applied Mathematics.

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

[16] E.D. Dolan and J.J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002.

[17] E. Frandi and A. Papini. Improving direct search algorithms by multilevel optimization techniques. *Optimization Methods and Software*, 30:1–18, 04 2015.

[18] F. Häse, L.M. Roch, C. Kreisbeck, and A. Aspuru-Guzik. Phoenics: A universal deep bayesian optimizer. *arXiv preprint arXiv:1801.01469v1*, 01 2018.

[19] D.R. Jones, M. Schonlau, and W.J. Welch. A data analytic approach to Bayesian global optimization. In *Proceedings of the ASA*, 1997.

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

[21] J. Lonchampt. VME a tool for probabilistic models valuation in engineering asset management. In *Risk, Reliability and Safety: Innovating Theory and Practice – Walls, Revie & Bedford (Eds)*, pages 1158–1164, London, 2017. CRC Press, Taylor & Francis Group. Proceedings of ESREL 2016 (Glasgow, Scotland, 25-29 September 2016).

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

[23] J.A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965.

[24] T. Bewley P. Beyhaghi, R. Alimo. A derivative-free optimization algorithm for the efficient minimization of functions obtained via statistical averaging. *arXiv preprint arXiv:1910.12393*, 2019.

[25] C. Paquette and K. Scheinberg. A Stochastic Line Search Method with Convergence Rate Analysis. *arXiv e-prints*, page arXiv:1807.07994, Jul 2018.

[26] V. Picheny, D. Ginsbourger, Y. Richet, and G. Caplin. Quantile-based optimization of Noisy Computer Experiments with Tunable Precision. working paper or preprint, March 2012.

[27] E. Polak and M. Wetter. Precision control for generalized pattern search algorithms with adaptive precision function evaluations. *SIAM Journal on Optimization*, 16(3):650–669, 2006.

[28] M. Rivier and P. M. Congedo. Surrogate-assisted bounding-box approach for optimization problems with tunable objectives fidelity. *Journal of Global Optimization*, Aug 2019.

[29] M.J. Sasena. *Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations*. PhD thesis, University of Michigan, 2002.

[30] S. Shashaani, F. Hashemi, and R. Pasupathy. ASTRO-DF: A Class of Adaptive Sampling Trust-Region Algorithms for Derivative-Free Stochastic Optimization. *arXiv e-prints*, page arXiv:1610.06506, Oct 2016.

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

[32] C. Xiaojun, C.T. Kelley, F. Xu, and Z. Zhang. A smoothing direct search method for monte carlo-based bound constrained composite nonsmooth optimization. *SIAM Journal on Scientific Computing*, 40:A2174–A2199, 01 2018.