Selection of variables in parallel space decomposition for the mesh adaptive direct search algorithm *

Stéphane Alarie† Nadir Amaioua‡ Charles Audet,‡
Sébastien Le Digabel‡ Louis-Alexandre Leclaire†

June 11, 2018

Abstract. The parallel space decomposition of the Mesh Adaptive Direct Search algorithm (PSD-MADS proposed in 2008) is an asynchronous parallel method for constrained derivative-free optimization with large number of variables. It uses a simple generic strategy to decompose a problem into smaller dimension subproblems. The present work explores new strategies for selecting subset of variables defining subproblems to be explored in parallel. These strategies are based on ranking the variables using statistical tools to measure their influence on each output. Determining the most influential variables is performed using the k-mean algorithm. In addition, an hybrid approach using both the k-mean method and the random selection of variables to build subproblems is presented. These new methods improve the decomposition of the problem into smaller more relevant subproblems. Computational tests are conducted on a set of constrained instances with up to 4,000 variables. The results show an important improvement over the original version of PSD-MADS.
Keywords. Parallelism; Derivative-free optimization; k-mean algorithm; PSD-MADS; Mesh adaptive direct search.

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

†Expertise Réseaux électriques et Mathématique, Institut de recherche d'Hydro-Québec (IREQ), 1800, boul. Lionel-Boulet, Varennes, Québec, J3X 1S1, Canada, Alarie.Stephane@ireq.ca, Leclaire.Louis-Alexandre@ireq.ca.

‡GERAD and Département de mathématiques et génie industriel, École Polytechnique de Montréal, C.P. 6079, Succ. Centre-ville, Montréal, Québec, Canada H3C 3A7, Nadir.Amaioua@gerad.ca, www.gerad.ca/Charles.Audet, www.gerad.ca/Sebastien.Le.Digabel.

1 Introduction

We focus on the following constrained optimization problem:

$$\begin{array}{ll} \min\limits_{x \in \mathcal{X}} & c_0(x) \\ \text{subject to} & c_j(x) \leq 0, \quad j \in \{1,2,\ldots,m\} \end{array} \tag{1}$$

where $\mathcal{X}$ is a subset of $\mathbb{R}^n$, $(c_j)_{j\in\{0,1,\ldots,m\}}$ are real-valued functions considered as blackboxes and m and n are positive integers. We study problems that possess the following pair of properties. First, the objective function and all constraints are given by computer simulations which may be expensive to evaluate and/or for which the derivatives do not exist or are difficult to compute or to estimate. Under these circumstances, these computer simulations are considered as blackboxes and treated by appropriate derivative-free optimization (DFO, [8, 15]) techniques. The second property is that the dimension of the problem n is large for the DFO context. This paper studies problems with up to 4,000 variables.

DFO methods are typically designed to handle small-sized blackboxes. For instance, the NOMAD [43] package, which relies on the mesh adaptive direct search algorithm (MADS, [5]), is recommended for problems up to 20 variables. When handling larger problems, other techniques such as parallelism are needed.

With the rapid growth of technology, everyone has access to parallel hardware. Softwares and algorithms are turning to parallel programming as it tends to save time and allows treatment of more complex and larger problems.

The present work aims at improving the parallel space decomposition of the MADS algorithm, (PSD-MADS introduced 10 years ago in [7]). PSD-MADS decomposes the optimization Problem (1) into a series of smaller subproblems, treated by parallel processes with the MADS algorithm. Building these subproblems relies on randomly selecting a small number of variables that would vary in the subspace while keeping the rest of the variables fixed. The main objective is to find a more effective strategy to select the variables. While the original version of PSD-MADS handled problems up to 500 variables, our new approach presents computational results on problems with up to 4,000 variables.

The concluding remarks of [7] suggests that future work could improve upon the random selection of variables. We propose an improved version of PSD-MADS that relies on statistical tools to determine the influence of each variable on the outputs of the problem and construct a matrix called the sensitivity matrix. Different strategies are introduced to exploit the sensitivity matrix efficiently and classification methods are used to group the variables with similar influence into clusters. Furthermore, clustering algorithms also allow to automatically choose the size of subproblems.

Section 2 starts by a literature review of parallelism within DFO, briefly describes the MADS and PSD-MADS algorithms and concludes by an exposition of variable selection techniques. Section 3 presents the improved variable selection methods used in the revised PSD-MADS algorithms. These algorithms depend on a total of five parameters, which will be denoted by $\mathcal{P}_{ns}$, $\mathcal{P}_{bbe}$, $\mathcal{P}_{vr}$, $\mathcal{P}_K$ and $\mathcal{P}_S$. The computational experiments reported in Section 4 allow to determine default values and validate the proposed methods. Finally, Section 5

summarizes the paper and concludes with recommendations for future work.

2 Literature Review

This section describes briefly the MADS and PSD-MADS algorithms and presents a literature review on parallel space decomposition techniques and parallelism in DFO.

2.1 The mesh adaptive direct search algorithm (MADS)

The MADS algorithm [5, 6] is an iterative DFO method for blackbox optimization that deploys a spatial discretization on the solution space called the mesh. The mesh is characterized by a coefficient called the mesh size parameter which scales the space discretization. MADS uses this mesh to select trial points in order to find a better solution than the current iterate. Every iteration of MADS consists of three main steps.

First, the search step is flexible since it is not essential for the convergence analysis of MADS. The search can either be skipped, defined specifically by the user or may rely on surrogate-based [9, 12] techniques. In the current default version of MADS, the search uses quadratic models to find promising trial points to evaluate when dealing with problems that have 50 or less variables. If the search step succeeds, meaning that it found a new best iterate, then the poll step is skipped and the iteration goes straight to the update step. If this step fails, then the poll step is invoked.

The poll step uses a small set of directions, which positively spans the solution space, to select several trial points on the mesh within a region called the poll frame. The frame is centered around the current iterate and is regulated by a poll size parameter dictating the maximum distance allowed between the poll center and the trial points. The set of the normalized directions used during a run of MADS must be dense in the unit sphere to guarantee the strongest results of the convergence analysis [5]. MADS converges to a Clarke stationary point if the objective function is locally Lipschitz.

If the poll fails to find a better iterate, then both the mesh and poll size parameters are reduced in the update step, so that the next iteration explores closer to the current best known solution. Alternately, the update step increases both the mesh and the poll size parameters when the poll finds a better solution. By making the poll size parameter decrease at a slower rate than the mesh size parameter, the number of potential polling directions increases.

This sequence of steps iterates until a stopping criteria is met; typically a maximum number of evaluations or a minimum mesh size parameter value. Figure 1 proposes a schematic illustration of the MADS algorithm. MADS uses the extreme [5] or progressive barrier [6] to handle the constraints. The extreme barrier simply rejects infeasible points, and the progressive barrier uses a constraint violation aggregation function to allow steps in the infeasible region in order to reach an optimal solution.

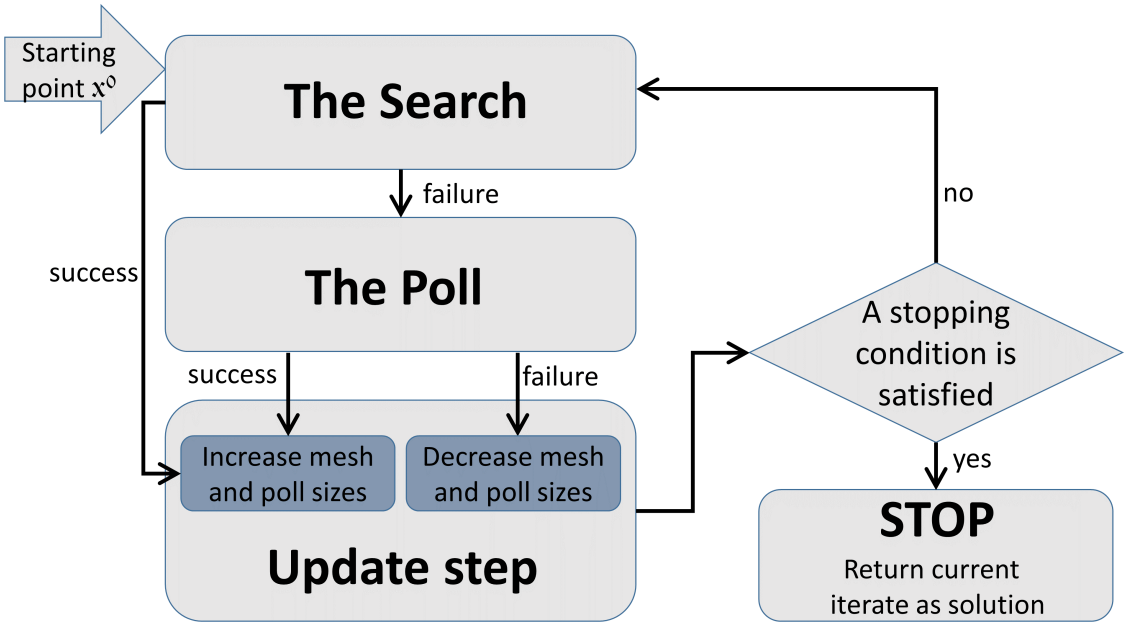


Figure 1: Overview of the MADS algorithm (Figure adapted from [2]).

2.2 Parallel space decomposition methods for nonlinear optimization

The Block-Jacobi algorithm [10] is used to decompose a problem into smaller subproblems, treated in parallel using several processes. The parallel variable distribution method (PVD) [20] is used for unconstrained problems or instances with convex block-separable constraints. It relies on the same steps as Block-Jacobi except that it can slightly modify the fixed variables by adding some so-called "forget-me-not" terms to the subproblems. If I is the set of variable indices used in Problem (1), then each process p can only modify a subset $I_p \subset I$ of variables while the remaining inputs vary restrictively. This algorithm possesses two main steps: a parallelization phase where the subproblems are treated and a synchronization phase which improves over the solutions obtained at the parallelization phase. The "forget-me-not" terms make the algorithm more robust and faster to converge. Since the PVD algorithm can fail when the constraints are not bloc separable, a dual method of the exact penalty is recommended.

The parallel gradient distribution (PGD) [47] for unconstrained optimization communicates different parts of the gradient to the available processes and each process uses its own search direction independently of the other processes. A synchronization step follows.

A generalization of PVD replacing the exact subproblems solutions by sufficient descent conditions is introduced by Solodov [64] in order to improve the work load of the different processes. In addition, this method can handle problems with general convex constraints by modifying the way secondary variables are handled [65]. In [21] a parallel space decomposition is described as combining the PVD method with Schwarz algebraic techniques.

An algorithmic framework of parallel variable transformation (PVT) for unconstrained optimization is described by Fukushima in [22] where he shows that PVT is a generalization of PVD, PGD as well as the method introduced by Han [30]. In addition, in [68] Fukushima proposes practical strategies for implementing PVT like methods and tests the Block-Jacobi method (which is conform to the PVT framework) on the Fujitsu VPP500 supercomputer.

Three other algorithms which are based on the PGD and the PVT methods are proposed in [44]. Two versions of PVD for constrained optimization are described in [59]. The first one treats problems with bloc-separable general constraints using Sequential Quadratic Programming (SQP). The second approach works on general convex constrained problems by using approximations of the projected gradient directions. The first method of [59] is improved in [29] by using a new line search approach in SQP. In [28], a PVD-based method is described to find an approximate solution for total least square problems.

2.3 Parallelism in DFO

Parallel programming appeared in the DFO context at the end of the eighties. One of the first method to be parallelized was the Nelder-Mead algorithm [54]. The computational results of this parallel version showed that Nelder-Mead converged to a point that is not a local optimum [66]. It also leads to conceive the multidirectional search algorithm [17] for unconstrained problems which is supported by a convergence analysis.

In [24], a parallel algorithm for unconstrained optimization is described. This approach relies on a nonsmooth necessary condition to find a point with a sufficiently lower objective value and lead to a new DFO approach for bound constrained problems [23]. The Asynchronous Parallel Pattern Search algorithm (APPS) is introduced in [34] and is implemented in the APPSPACK package for blackbox optimization [25] with improvements such as a dynamic affectation of evaluations to processes [40]. In [27] a parallel version of the generating set search algorithm (GSS) [41] is described and included in APPSPACK. The APPS algorithm is integrated in the Hybrid Optimization Parallel Search Package (HOPSPACK) [55]. This framework simplifies the hybridization of several optimization methods and heuristics into one algorithm that can use parallel function evaluations in order to exploit the strength of different optimization routines. A hybrid DFO approach combining DIRECT [38] and GSS [41] is introduced in [26].

Two parallel implementations of the Hooke and Jeeves method [33] and the simulated annealing heuristic [39] are presented in [57] and aim to fully exploit the processing power without degrading the quality of the solution. A divide-and-conquer method is designed in [49] for large size unconstrained blackboxes. This method can identify independent subproblems which are treated by the covariance matrix adaptation evolutionary strategy (CMA-ES) [31].

Several parallel versions of the coordinate descent method exist such as the parallel distributed block version introduced in [48]. Also a new coordinate descent method is described in [58] relying on an arbitrary probability law to choose a random set of variables to update at each iteration. A framework for convergence analysis of different classes of parallel coordinate descent is defined in [53].

Two variations of the coordinate search derivative-free nonsmooth method (CS-DFN [19])

are proposed in [45] and used in two hybrid parallel approaches involving the NOMAD package [43]. The study in [45] concludes that combining these stand-alone algorithms gives better results than using them individually.

By using NOMAD in [3], a comparison on how to exploit available resources is conducted between parallelizing the solver or parallelizing the blackbox in OPAL [4]. In [42], a new multistart DFO method is defined for finding multiple local minimums for bound constrained problems.

2.4 The PSD-MADS algorithm

The PSD-MADS algorithm [7] is based on the Block-Jacobi technique. It is an asynchronous parallel method which decomposes the problem into several smaller subproblems that are treated by the MADS algorithm. The number of available processes is denoted by p. PSD-MADS allocates $p-3$ regular slave processes for treating the subproblems, a special slave process. The remaining 3 processes are the pollster, for polling over the entire space of variables, another special process called the cache server, which stores past evaluations and finally, a master process that regulates all the other processes (see Figure 2).

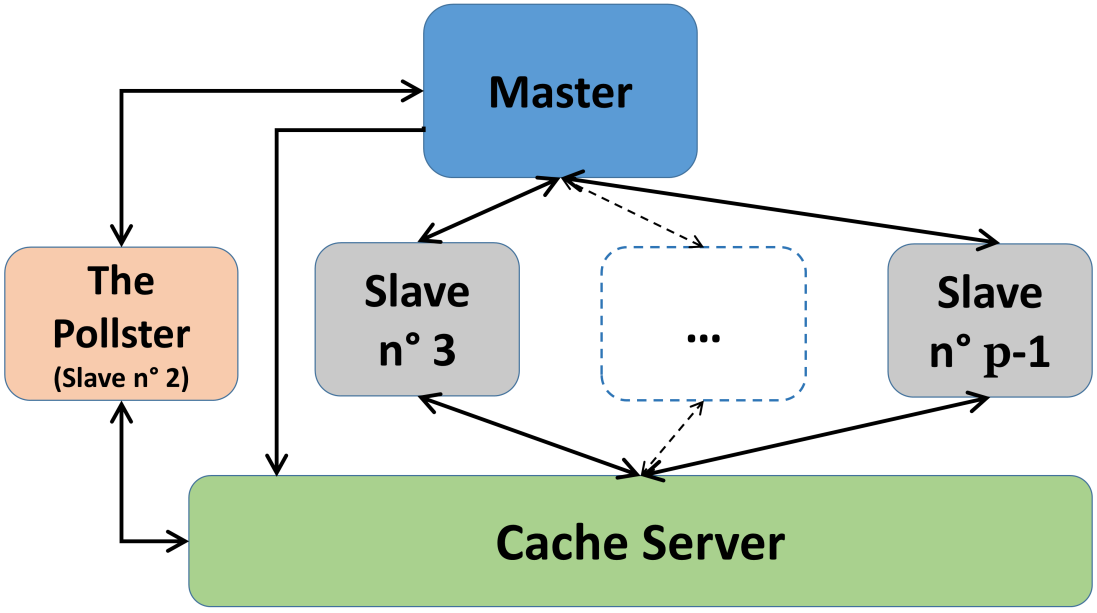


Figure 2: The interactions between the p different processes in PSD-MADS.

The pollster is a special slave process which launches a single poll step of the MADS algorithm on the original problem with all the variables. While this poll set consists of a single direction, the set of normalized directions used in the pollster through the entire execution of the algorithm grows dense in the unit sphere. This process is important since it allows to

benefit from the convergence analysis of MADS [5]. The other slave processes perform a limited number of evaluations in a subspace defined by randomly selecting a fixed number of decision variables to be optimized. This optimization is performed by the MADS algorithm with the progressive barrier to deal with the constraints and without any search strategy. The cache server records evaluated points to avoid revisiting the same points. This is especially useful when dealing with expensive blackboxes. The master process defines the subproblems, hands them to the slave processes and waits for their results to update the data and assign new subproblems to the slaves.

In PSD-MADS, the user can define two parameters which apply for all slave processes. The first parameter $\mathcal{P}_{ns}$ is the number of variables that are randomly selected to build a subproblem and the second parameter $\mathcal{P}_{bbe}$ is the maximum number of blackbox evaluations allowed for solving each subproblem by MADS. In [7], setting the parameters to $\mathcal{P}_{bbe} = 10$ and $\mathcal{P}_{ns} = 2$ gave the best results. Several values of these two parameters are tested in Section 4.

2.5 Screening methods

Screening techniques aim to select the most important variables of a given problem in order to reduce its dimension. It is possible to define two types of screening methods depending of whether it treats single-response or multi-response problems.

Selecting variables for single-response problems is performed by techniques such as the analysis of variance (ANOVA), principal component analysis (PCA) [37] or sensitivity analysis [14, 52, 62].

If $Y = [y_1\ y_2\ \ldots\ y_N]$ is the matrix formed by N centered observations of dimension n (for example the inputs of the PSD-MADS cache for a problem with scaled variables), PCA [37] starts by computing the covariance matrix $cov(Y)$ and performing an eigen analysis:

$$cov(Y) = \frac{1}{N-1} YY^\top = QDQ^\top \tag{2}$$

where D is the diagonal matrix containing the eigenvalues of $cov(Y)$ in a descending order and $Q = [v_1\ v_2\ \ldots\ v_n]$ is formed of the corresponding orthogonal eigenvectors. By selecting the $p < n$ first vectors (corresponding to the highest k eigenvalues), the matrix $Q_p = [v_1\ v_2\ \ldots\ v_p]$ can be used to compute the p principal components $P = Q_p^\top Y$ which reduces the data of size n to a set of data of size $p < n$. The principal feature analysis (PFA) [46] uses the PCA to compute the previous matrix $Q_p^\top$ which can be expressed in terms of its rows. Then PFA uses the k-mean algorithm to cluster the rows into several groups. For each cluster, the closest row to its center is found and the corresponding variable is selected as an important input. Using k clusters would then choose k important variables of the problem. This method is applied in [46] for selecting facial motion feature points and in [67] for analyzing functional magnetic resonance imaging (fMRI) for brain decoding.

Sensitivity analysis can be grouped into two categories: local and global approaches. Local methods, on the one hand, verify the effect of a variable near a given point. This category includes finite-difference analysis, using tornado diagrams [35] and differentiation

based methods [32]. Global methods, on the other hand, take into account the full space of the function variation. Some of these methods are variance based such as the Sobol [63] and FAST (Fourier Amplitude Sensitivity Test) [16] methods, some are surrogate-based [56], some are based on multiple linear regression [60], and others are density based [13].

When treating multi-response problems, several methods can be defined to screen variables such as selecting the most important variables for each response, identifying the variables regarding their average effect on all outputs or selecting the unimportant inputs by eliminating the most influential variables for each response. For more details about screening multi-response problems, see [14].

STATS-MADS [1] is an instantiation of MADS for large unconstrained optimization problems. It uses statistical methods to identify the most influential variables for the objective function and organizes them into subspaces. The method alternates optimizations on the whole space (intended to gather information on the variables influence) and optimizations on the subspaces (intensification step to explore further the promising areas). The implementation of STATS-MADS uses sensitivity analysis tools for screening variables. STATS-MADS is tested in [1] on problems with up to 500 variables.

In [61], screening methods and space decomposition are cited among the five main strategies used to treat high-dimensional problems alongside mapping, space reduction and visualization techniques. Although [61] identifies also space decomposition as the most promising method since it allows the use of parallel computing on smaller-sized subproblems that can be treated with different algorithms, the discussion raised a question on how to decompose a problem relying on the correlation between variables and outputs. The following section is going to detail a screening strategy adapted from [1] and augmented with a classification algorithm which guides the parallel space decomposition in [7] to help build more meaningful subproblems.

3 An improved variable selection strategy for PSD-MADS

The main goal of this work is to propose a strategy to select the most influential variables of a problem. Building subproblems with these influential variables is expected to be more efficient than randomly choosing the variables as suggested in the conclusion of the original PSD-MADS [7].

3.1 Building the sensitivity matrix

For the purpose of selecting influential variables, we use the work of [1] to estimate the sensitivity coefficients of the variables regarding the objective function. These estimators can be generalized to the influence of the variables over the constraints as well.

For each variable x_i ($i \in \{1,2,\ldots,n\}$) and for each output c_j ($j \in \{0,1,\ldots,m\}$), the sen-

sitivity coefficient is estimated using the following formula:

$$\tilde{s}_{ij} = \frac{\sum_{\ell=1}^{r_i} \left|A_i^\ell\right| \left((\bar{c}_j)_i^\ell - \bar{c}_j\right)^2}{\sum_{x \in V} (c_j(x) - \bar{c}_j)^2} \tag{3}$$

where :

$$\bar{c}_j = \frac{1}{|V|} \sum_{x \in V} c_j(x) \qquad \text{and} \qquad (\bar{c}_j)_i^\ell = \frac{1}{\left|A_i^\ell\right|} \sum_{x \in A_i^\ell} c_j(x), \tag{4}$$

with V is the set of points in the cache, r_i is the number of different values that the variable x_i takes in the cache and A_i^ℓ is the set of points for which the variable x_i is set to its ℓ^{th} value.

For the implementation point of view, it is difficult to track all the values that a variable takes in the cache due to the numerical precision of decimal comparison. When comparing decimal values, an error term $\varepsilon > 0$ is introduced. Two decimal number a and b are considered numerically equal if:

$$|b - a| < \varepsilon. \tag{5}$$

If ε is chosen too small, it is possible to end up with each element of the cache having a different value of x_i which gives an estimator of the sensitivity coefficient equal to 1. An option is to define a parameter $\mathcal{P}_{vr}$ where $10^{\mathcal{P}_{vr}}$ represents the number of variable ranges. This parameter divides the initial range of a variable into $10^{\mathcal{P}_{vr}}$ ranges of the same size. A formula similar to (3) is used to estimate the sensitivity coefficients except that $r_i = 10^{\mathcal{P}_{vr}}$ is the number of variable ranges and A_i^ℓ is the set of points for which the variable x_i is included in the ℓ^{th} range:

$$\tilde{s}_{ij} = \frac{\sum_{\ell=1}^{10^{\mathcal{P}_{vr}}} \left|A_i^\ell\right| \left((\bar{c}_j)_i^\ell - \bar{c}_j\right)^2}{\sum_{x \in V} (c_j(x) - \bar{c}_j)^2}. \tag{6}$$

When $\mathcal{P}_{vr}$ tends to infinity, the variable ranges are reduced to singletons which means that this new formula of computing the sensitivity coefficients reverts back to formula (3). These sensitivity coefficients take values between 0 and 1: the closer $\tilde{s}_{ij}$ is to 1, the more influence the variable x_i has over the output c_j. Once all the estimators $\tilde{s}_{ij}$ are computed, a $n \times m$ matrix called the *sensitivity matrix* is formed:

$$\tilde{S} = \begin{bmatrix} \tilde{s}_{10} & \tilde{s}_{11} & \cdots & \tilde{s}_{1m} \\ \tilde{s}_{20} & \tilde{s}_{21} & \cdots & \tilde{s}_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{s}_{n0} & \tilde{s}_{n1} & \cdots & \tilde{s}_{nm} \end{bmatrix} \in \mathbb{R}^{n \times m}. \tag{7}$$

This matrix is used to create clusters of variables which are appended to a waiting queue. Each time a slave process requests a new subproblem, the master selects a group of variables

from the waiting list and forwards it to the slave process. When the waiting list is empty, the sensitivity matrix is updated and new clusters are appended to the queue.

Many strategies can be defined to exploit the sensitivity matrix more efficiently. A training data set $\{\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_n\}$ is built by considering the entry $\tilde{s}_i = [\tilde{s}_{i0}\ \tilde{s}_{i1}\ \ldots\ \tilde{s}_{im}]^\top$ for each variable x_i ($i \in \{1,2,\ldots,n\}$) consisting of the sensitivity coefficients of x_i on the different outputs. A classification method can sort the training data into several clusters. Sorting the clusters by computing the distance of their centroids from the point $[1\ 1\ \ldots\ 1]^\top$ reveals the closest clusters as, potentially, the most influential ones. This strategy can be helpful when running the algorithm on systems with a limited number of processes. Nevertheless, our tests are conducted on a large supercomputer and in order to populate a large amount of processes, all the clusters given by the k-mean algorithm are appended to the waiting list.

Another type of strategy would be to consider only a submatrix of $\tilde{S}$ by selecting a subset of the outputs. For example, on the one hand, if the subset of violated constraints (at the current iterate) is selected, choosing the most influential clusters would potentially help satisfying some of the constraint. On the other hand, for the set of satisfied constraints, the less influential clusters would have the best chance of improving the objective without altering the feasibility of the selected outputs. For all the selected problems in the computational tests section, the starting points are feasible, which means that a decomposition into satisfied and violated constraint can not be used.

Instead, the following four strategies S1, S2, S3 and S4 are defined and designated by the parameter $\mathcal{P}_S$:

- Strategy S1 applies a clustering technique on the training data set $\{\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_n\}$ with:

$$\tilde{s}_i = [\tilde{s}_{i0}\ \tilde{s}_{i1}\ \ldots\ \tilde{s}_{im}]^\top \text{ for } i \in \{1,2,\ldots,n\}. \tag{8}$$

- Strategy S2 uses two independent executions of the classification algorithm to populate the waiting list: the first execution is applied on the set related to the objective function $\{\tilde{s}_{10}, \tilde{s}_{20}, \ldots, \tilde{s}_{n0}\}$ and the second run is used on the set related to the constraints $\{\tilde{s}_{1[1,m]}, \tilde{s}_{2[1,m]}, \ldots, \tilde{s}_{n[1,m]}\}$ with:

$$\tilde{s}_{i[1,m]} = [\tilde{s}_{i1}\ \tilde{s}_{i2}\ \ldots\ \tilde{s}_{im}]^\top \text{ for } i \in \{1,2,\ldots,n\}. \tag{9}$$

- Strategy S3 is identical to S2 except that the second run of the clustering algorithm is applied to the training data from strategy S1.

- Strategy S4 applies the clustering on the set $\{\tilde{s}_{1\{j_1,j_2\}}, \tilde{s}_{2\{j_1,j_2\}}, \ldots, \tilde{s}_{n\{j_1,j_2\}}\}$ where:

$$\tilde{s}_{i\{j_1,j_2\}} = [\tilde{s}_{ij_1}\ \tilde{s}_{ij_2}]^\top \text{ for } i \in \{1,2,\ldots,n\} \tag{10}$$

  with j_1 and j_2 two output indices randomly selected.

3.2 Exploiting the sensitivity matrix with k-mean

Unsupervised learning is a branch of machine learning that attempts to identify structure in a set of unlabeled training data. The k-mean algorithm [36] is one of the most popular unsupervised classification method used to automatically partition data into k clusters.

Algorithm 1: k-mean algorithm [36]

1: Initialize k centroids (center of clusters) using Latin Hypercube Sampling.
2: Assign each input from the training data to the cluster with the closest centroid.
3: Update the centroids positions.
4: Repeat 2 and 3 until the centroids do not move anymore.

Each cluster contains variables that share similar characteristics. In this case, a cluster contains the variables that have a similar influence on the problem. These clusters are appended to a waiting list, and each time a slave process asks for a new subproblem, one of the clusters is dispatched by the master to this slave process.

Since the k-mean method is local and the k parameter cannot be chosen efficiently before the computation, the method is enclosed in a loop to determine the best parameter k^* on a predefined domain while initializing the centroids with a Latin Hypercube Sampling at each iteration:

$$k^* \in \underset{k \in dom}{\arg\min} \sum_{i=1}^{n} \|\tilde{s}_i - centroid_k(\tilde{s}_i)\| \tag{11}$$

where dom represents the range of values for k, and $centroid_k(\tilde{s}_i)$ is the function that returns the centroid closest to $\tilde{s}_i$ for the run of the k-mean algorithm with k clusters. We introduce the parameter $\mathcal{P}_K \in \{\text{Q,H}\}$ to designate the two different domains that are compared in the computational results (Q stands for *quarter* and H for *half*):

$$k \in \begin{cases} \{\lceil \frac{3}{4}\sqrt{n} \rceil, \ldots, \lfloor \sqrt{n} \rfloor\} & \text{if} \quad \mathcal{P}_K = \text{Q} \\ \{\lceil \frac{\sqrt{n}}{2} \rceil, \ldots, \lfloor \sqrt{n} \rfloor\} & \text{if} \quad \mathcal{P}_K = \text{H}. \end{cases} \tag{12}$$

3.3 Redefining the original PSD-MADS parameters

The two parameters $\mathcal{P}_{ns}$ and $\mathcal{P}_{bbe}$ defined in the original version of PSD-MADS extend to this improved version with a small difference. While $\mathcal{P}_{ns}$ represents the number of variables selected randomly to build the subproblems in the original PSD-MADS, in the improved k-mean-based PSD-MADS, $\mathcal{P}_{ns}$ defines an upper bound on the number of selected variables. This means that the size of the subproblems is now dynamic. $\mathcal{P}_{bbe}$ still represents the maximum number of function evaluations allowed to solve a subproblem by MADS.

The chart in Figure 3 summarizes where all the parameters intervene in the variable selection process.

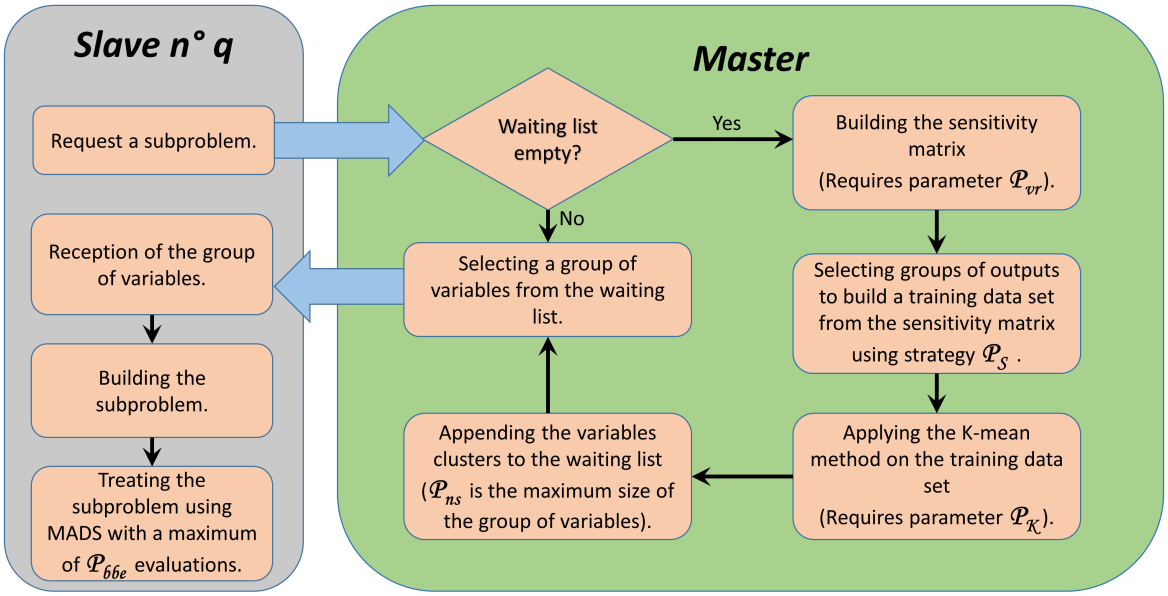


Figure 3: The process of selecting a group of variables for a regular slave.

3.4 Hybrid approach

A limitation of the previous method is that after several blackbox evaluations, the sensitivity matrix tends to stabilize and the k-mean algorithm proposes similar clusters at each iteration. This generates points that have already been evaluated while new points become scarce. Preliminary tests confirm this phenomenon, which becomes more important as the number of blackbox evaluations increases: the k-mean algorithm populates the waiting list several times before finding a new point which makes reaching the maximum number of evaluations slower.

To remedy to this situation, we propose a hybrid approach using the same k-mean-based method as previously, except that, if the k-mean algorithm fails to find a new iterate for a given number of times $\mathcal{P}_H^{in}$ then we switch to a random selection of variables as described in the original PSD-MADS algorithm. After a predetermined number of evaluations $\mathcal{P}_H^{out}$ are executed, the algorithm reverts back to the k-mean-based selection method. The random selection of variables serves as a diversification step to shake up the sensitivity matrix and discover new clusters of efficient variables.

The parameters used for the transition between the two variable selection methods are chosen to limit wasting time on the k-mean algorithm when it produces the same clusters and to evaluate enough new points with the random selection to disrupt the sensitivity matrix and discover new groups of variables. If the k-mean routine does not produce a subproblem that leads to a new better point after filling the clusters waiting list for three consecutive times ($\mathcal{P}_H^{in} = 3$), the hybrid approach switches to a random selection of variables. This random

selection is adopted for building $\mathcal{P}_H^{out} = 5n$ subproblems before shifting back to the k-mean routine again.

The remaining parameters of the hybrid approach are exactly the same as the ones defined for the improved method, except for $\mathcal{P}_{ns}$ which represents the upper bound on the number of selected variables in the k-mean phase and the size of the subproblems in the random variable selection phase.

4 Computational results

In this section, different versions of PSD-MADS are compared on a set of constrained problems. After a description of the test set and the algorithmic variants, the first part of the computational results focuses on the selection of suitable default values for the different parameters $\mathcal{P}_{ns}$, $\mathcal{P}_{bbe}$, $\mathcal{P}_{vr}$, $\mathcal{P}_K$ and $\mathcal{P}_S$, introduced earlier. In the second part, the improved PSD-MADS with these default parameter values is compared with the hybrid method using a problem of 4,000 variables.

4.1 Test problems and algorithmic variants

This first part lists the parameters that tune the improved PSD-MADS algorithm and presents its different versions. In addition, the test problems are presented, as well as a description of the comparison tools used in this study. Table 1 presents the five parameters with sets of values to test.

Param	Description	Values
$\mathcal{P}_{ns}$	Upper bound of the number of variables used to build subproblems.	2: maximum size of subproblems is 2 3: maximum size of subproblems is 3 5: maximum size of subproblems is 5 10: maximum size of subproblems is 10
$\mathcal{P}_{bbe}$	Number of blackbox evaluations allowed in each subproblem.	10: 10 evaluations 20: 20 evaluations 30: 30 evaluations
$\mathcal{P}_{vr}$	Number of variable ranges used to build the sensitivity matrix.	2: 10^2 variable ranges 3: 10^3 variable ranges 4: 10^4 variable ranges
$\mathcal{P}_K$	The range of values used to select the best number of clusters for the k-mean algorithm.	Q: $\lceil \frac{3}{4}\sqrt{n} \rceil \leq k \leq \lfloor \sqrt{n} \rfloor$ H: $\lceil \frac{\sqrt{n}}{2} \rceil \leq k \leq \lfloor \sqrt{n} \rfloor$
$\mathcal{P}_S$	Different strategies of regrouping outputs to apply k-mean on a submatrix of the sensitivity matrix.	S1: The entire sensitivity matrix S2: Objective alone and then constraints S3: Objective alone and then entire matrix S4: Random selection of outputs

Table 1: Parameters of the improved version of PSD-MADS.

The original version of PSD-MADS uses a random selection of $\mathcal{P}_{ns}$ variables to build a subproblem with a budget of $\mathcal{P}_{bbe}$ evaluations. In this section, the following algorithm is

referred to as Original-$\mathcal{P}_{ns}$-$\mathcal{P}_{bbe}$. For our improved method relying on the k-mean algorithm, the other three parameters are defined as well. These improved variants are noted: Improved-$\mathcal{P}_{ns}$-$\mathcal{P}_{bbe}$-$\mathcal{P}_{vr}$ $\mathcal{P}_K$ $\mathcal{P}_S$. Table 2 lists all versions of PSD-MADS tested to select default values of the parameters.

Algorithm	$\mathcal{P}_{ns}$	$\mathcal{P}_{bbe}$	$\mathcal{P}_{vr}$	$\mathcal{P}_K$	$\mathcal{P}_S$	Algorithm	$\mathcal{P}_{ns}$	$\mathcal{P}_{bbe}$	$\mathcal{P}_{vr}$	$\mathcal{P}_K$	$\mathcal{P}_S$	Algorithm	$\mathcal{P}_{ns}$	$\mathcal{P}_{bbe}$
Improved-5-20-3QS1	5	20	3	Q	S1	Improved-5-20-3HS1	5	20	3	H	S1	Original-5-10	5	10
Improved-5-20-3QS2	5	20	3	Q	S2	Improved-5-10-3QS1	5	10	3	Q	S1	Original-5-20	5	20
Improved-5-20-3QS3	5	20	3	Q	S3	Improved-5-30-3QS1	5	30	3	Q	S1	Original-5-30	5	30
Improved-5-20-3QS4	5	20	3	Q	S4	Improved-2-20-3QS1	2	20	3	Q	S1	Original-2-20	2	20
Improved-5-20-2QS1	5	20	2	Q	S1	Improved-3-20-3QS1	3	20	3	Q	S1	Original-3-20	3	20
Improved-5-20-4QS1	5	20	4	Q	S1	Improved-10-20-3QS1	10	20	3	Q	S1	Original-10-20	10	20

Table 2: Algorithms used in the computational tests.

Table 3 presents the 12 constrained optimization problems to test the algorithmic variants with a number of variables ranging from 60 to 4,000.

name	n	m	source	name	n	m	source	name	n	m	source
B250	60	1	[11]	CRESCENT-2000	2,000	2	[6]	G2-500	500	2	[50]
B500	60	1	[11]	DISK-500	500	1	[6]	G2-1000	1,000	2	[50]
CRESCENT-500	500	2	[6]	DISK-1000	1,000	1	[6]	G2-2000	2,000	2	[50]
CRESCENT-1000	1,000	2	[6]	DISK-2000	2,000	1	[6]	G2-4000	4,000	2	[50]

Table 3: List of test problems.

A budget of $100n$ blackbox evaluations is allowed for each run of a variant of PSD-MADS on a problem with n variables, except for B250 and B500 which have a maximum of 60,000 evaluations. Since there are some aspects of the improved algorithms that rely on randomness (selection of variables, initialization of the k-mean centroids, output selection, etc.), each algorithm from Table 3 is executed 30 times on each problem (except G2-4000 which is used later in the second part of this section). Each instance is run in parallel using 500 distributed processes of CASIR, the supercomputer based at Hydro-Québec with 4,300 cores.

Data profiles [18, 51] are used to analyze the runs. If $\mathcal{S}$ is the set of algorithms to compare and $\mathcal{P}$ the set of instances, the performance measure $t_{p,s}$ $(p \in \mathcal{P}, s \in \mathcal{S})$ is the minimum number of evaluations needed to comply with the following convergence test:

$$f(x_0) - f(x) \geq (1-\tau)(f(x_0) - f_L), \tag{13}$$

where x_0 is the starting point of p, τ is the chosen tolerance and f_L is the minimum objective function value found by all solvers $s \in \mathcal{S}$ for a predefined number of evaluations and for each problem p. For a solver $s \in \mathcal{S}$, the data profile is

$$d_s(\kappa) = \frac{1}{card(\mathcal{P})} card\left\{p \in \mathcal{P} : \frac{t_{p,s}}{n_p + 1} \leq \kappa\right\}, \tag{14}$$

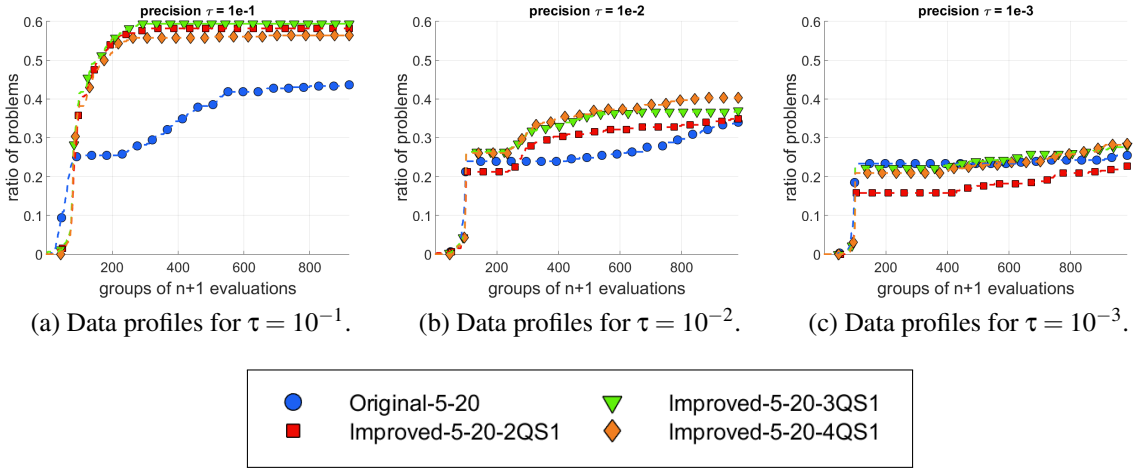
with n_p the number of variables of p. The data profile represents the ratio of instances solved by s given κ groups of $n_p + 1$ evaluations and a tolerance τ.

4.2 Selection of default values for the improved PSD-MADS algorithm

The first objective of this computational comparison is to determine a set of default values for the improved PSD-MADS coefficients. We choose the following starting values for the five parameters: $\mathcal{P}_{ns} = 5$, $\mathcal{P}_{bbe} = 20$, $\mathcal{P}_{vr} = 3$, $\mathcal{P}_K =$Q and $\mathcal{P}_S =$S1. In the next five subsections, all but one parameter are fixed. The parameter that varies takes values shown in Table 1. The best value of each parameter is chosen as the default value for the improved version of PSD-MADS.

Parameter $\mathcal{P}_{vr}$

To choose a default value for this parameter, the other coefficients are fixed as follows: $\mathcal{P}_{ns} = 5$, $\mathcal{P}_{bbe} = 20$, $\mathcal{P}_K =$Q and $\mathcal{P}_S =$S1. Figure 4 compares between Original-5-20, Improved-5-20-2QS1, Improved-5-20-3QS1 and Improved-5-20-4QS1.



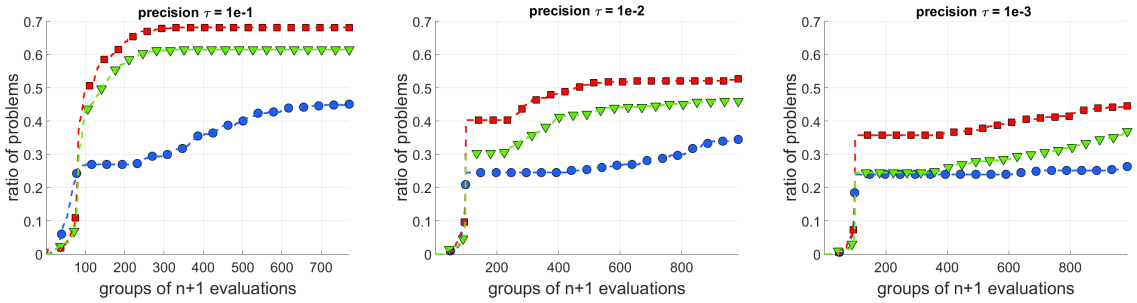
(a) Data profiles for $\tau = 10^{-1}$. (b) Data profiles for $\tau = 10^{-2}$. (c) Data profiles for $\tau = 10^{-3}$.

Figure 4: Comparison of the original and improved methods with different number of variable ranges $10^{\mathcal{P}_{vr}}$ for building the sensitivity matrix.

From the data profiles in Figure 4, it is difficult to designate a clear winner among the 3 versions of the improved algorithm. Nevertheless, for precisions $\tau = 10^{-2}$ and $\tau = 10^{-3}$, Improved-5-20-4QS1 slightly outperforms the remaining methods after $900(n+1)$ evaluations. This concurs with the fact that the larger $\mathcal{P}_{vr}$ the more accurate the sensitivity matrix is.

Parameter $\mathcal{P}_K$

The remaining coefficients are set as follows: $\mathcal{P}_{ns} = 5$, $\mathcal{P}_{bbe} = 20$, $\mathcal{P}_{vr} = 3$ and $\mathcal{P}_S$ =S1. Figure 5 compares Original-5-20, Improved-5-20-3QS1 and Improved-5-20-3HS1.



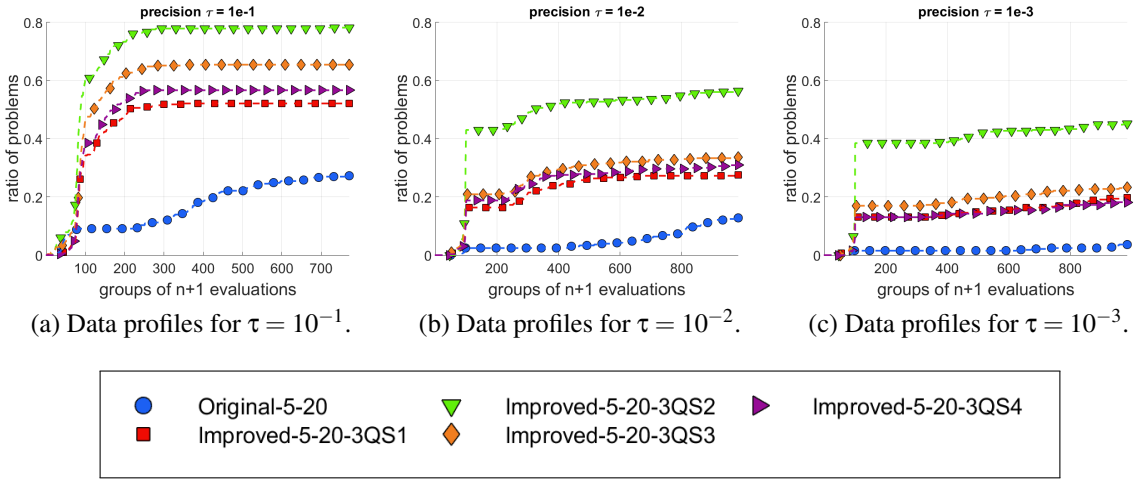
(a) Data profiles for $\tau = 10^{-1}$. (b) Data profiles for $\tau = 10^{-2}$. (c) Data profiles for $\tau = 10^{-3}$.

Figure 5: Comparison of the improved methods with different ranges of the parameter k for the k-mean algorithm ($\mathcal{P}_K$).

Both the improved algorithms produce better solutions that the original one. Although $\mathcal{P}_K$ =H covers a larger range of coefficients k (for the k-mean method) than $\mathcal{P}_K$ =Q, the computational results suggest that using the latter value gives better results. Figure 5 shows that Improved-5-20-3QS1 is preferable to the other variants.

Parameter $\mathcal{P}_S$

The parameters are fixed to $\mathcal{P}_{ns} = 5$, $\mathcal{P}_{bbe} = 20$, $\mathcal{P}_{vr} = 3$ and $\mathcal{P}_K$ =Q, and we compare between Original-5-20, Improved-5-20-3QS1, Improved-5-20-3QS2, Improved-5-20-3QS3 and Improved-5-20-3QS4.



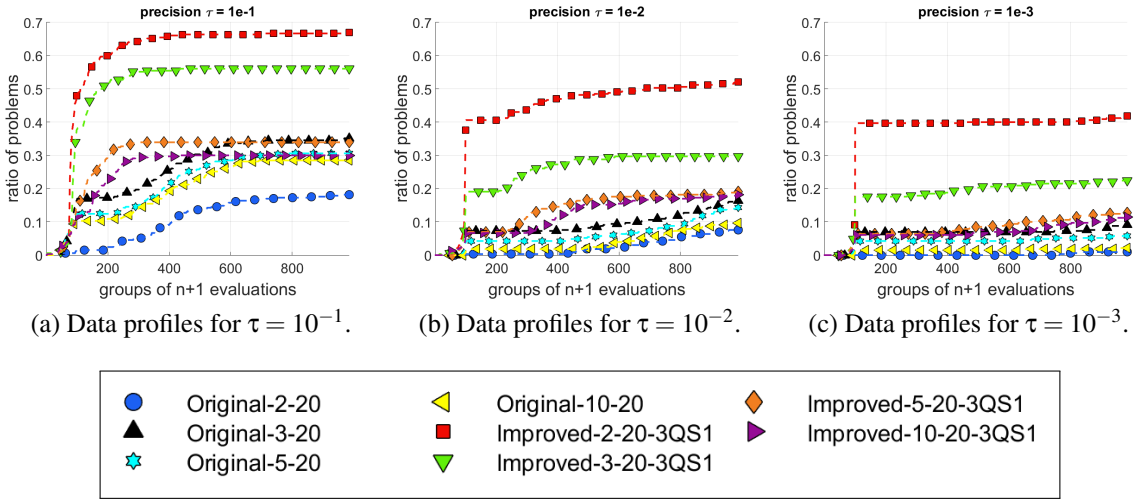
(a) Data profiles for $\tau = 10^{-1}$. (b) Data profiles for $\tau = 10^{-2}$. (c) Data profiles for $\tau = 10^{-3}$.

Figure 6: Comparison of methods with different strategies $\mathcal{P}_{vr}$ for regrouping outputs.

The Improved-5-20-3QS2 is the clear winner in this comparison. In Figure 6, Strategy S2 outperforms the other methods with a large margin. Strategies S2 and S3 run two clustering routines instead of one which can explain their superior results compared to the remaining approaches. There is still a large gap between Strategies S2 and S3. It seems that, while the first clustering in Strategy S2 is based on the objective function, which allows to improve the current solution, the second clustering, using only the constraints, builds subproblems that generate even more feasible trial points which helps explore the solution space more efficiently. Also, all the versions of the improved method outperform the original version of PSD-MADS again.

Parameter $\mathcal{P}_{ns}$

For this parameter, we set $\mathcal{P}_{bbe} = 20$, $\mathcal{P}_{vr} = 3$, $\mathcal{P}_K =$Q, $\mathcal{P}_S =$S1 and vary $\mathcal{P}_{ns} \in \{2,3,5,10\}$. The data profiles in Figure 7 show the comparison between Original-2-20, Original-3-20, Original-5-20, Original-10-20, Improved-2-20-3QS1, Improved-3-20-3QS1, Improved-5-20-3QS1 and Improved-10-20-3QS1.



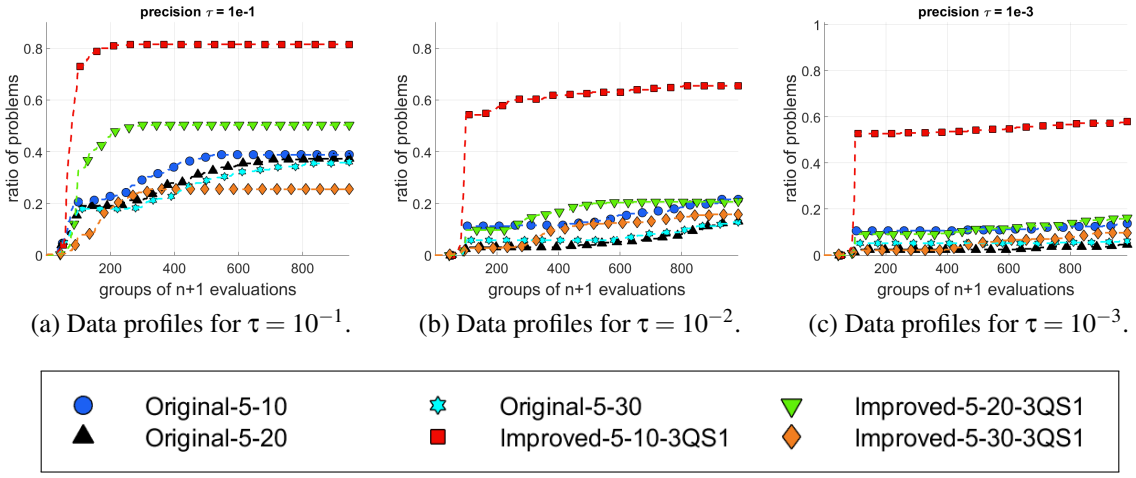
(a) Data profiles for $\tau = 10^{-1}$. (b) Data profiles for $\tau = 10^{-2}$. (c) Data profiles for $\tau = 10^{-3}$.

Figure 7: Selecting a default value of the upper bound on the subproblem size $\mathcal{P}_{ns}$ for the improved version of PSD-MADS.

The improved method of selecting variables relying on the k-mean algorithm outperforms the random selection used in the original version of PSD-MADS. This is confirmed in the data profiles from Figure 7 which show that using the improved approach with an upper bound $\mathcal{P}_{ns} = 2$ yields the best results. Solving smaller subproblems seems to make PSD-MADS explore the solution space better. This concurs with the fact that the smaller the subproblem is, the more efficient MADS can solve it. The strategy Improved-2-20-3QS1 dominates the others.

Parameter $\mathcal{P}_{bbe}$

This time, we set $\mathcal{P}_{ns} = 5$, $\mathcal{P}_{vr} = 3$, $\mathcal{P}_K =$Q, $\mathcal{P}_S =$S1 and let $\mathcal{P}_{bbe}$ vary among 10, 20 and 30 blackbox evaluations. Data profiles in Figure 8 sum up the comparison between Original-5-10, Original-5-20, Original-5-30, Improved-5-10-3QS1, Improved-5-20-3QS1 and Improved-5-30-3QS1.



(a) Data profiles for $\tau = 10^{-1}$. (b) Data profiles for $\tau = 10^{-2}$. (c) Data profiles for $\tau = 10^{-3}$.

Figure 8: Selecting a default value of the maximum blackbox evaluations $\mathcal{P}_{bbe}$ for the subproblems in PSD-MADS.

Allowing $\mathcal{P}_{bbe} = 10$ blackbox evaluations for each subproblems with the improved variable selection method gives the best results with a large gap from the other methods. In Figure 8(b), while Improved-5-10-3QS1 solves almost 65% of the test instances, the remaining methods solve less than 20% of the problems after $900(n+1)$ evaluations. When the number of evaluations allocated for each subproblem is small, PSD-MADS builds more subproblems which means a better coverage of the solution space. The strategy Improved-5-10-3QS1 dominates the others.

In summary, the above extensive computational experiments allow us to propose the following default values for the five algorithmic parameters: $\mathcal{P}_{ns} = 2$, $\mathcal{P}_{bbe} = 10$, $\mathcal{P}_{vr} = 4$, $\mathcal{P}_K =$Q and $\mathcal{P}_S =$S2. This strategy will be refereed to as Improved-2-10-4QS2.

4.3 The hybrid approach

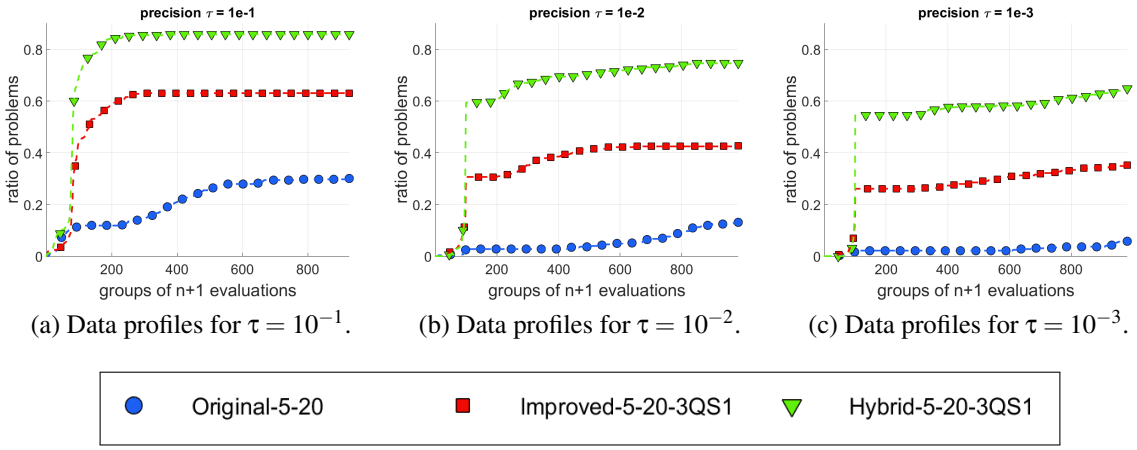
The hybrid method, as defined in Section 3.4, combines the random selection of variables used in the original PSD-MADS and the new k-mean-based approach used in the improved PSD-MADS.

To test the hybrid approach, the parameters are set as follows: $\mathcal{P}_{ns} = 5$, $\mathcal{P}_{bbe} = 20$, $\mathcal{P}_{vr} = 3$, $\mathcal{P}_K =$Q and $\mathcal{P}_S =$S1. The hybrid method is designated by Hybrid-5-20-3QS1 and it is compared with Original-5-20 and Improved-5-20-3QS1[1].

The results of this comparison are summarized in Figure 9 and in Table 4. For each problem and for each algorithm, the table gives the best (Z_{best}) and the worst (Z_{worst}) values

[1]The computational tests presented in this subsection were conducted before we completed the analysis from the previous subsection. Therefore, we had not identified yet the values of the default parameters. This is why we have tested the hybrid approach with other parameter values. Access to the supercomputer is limited, and we could not use additional access time to perform additional runs.

of the objective function from the 30 instances, in addition to the average value (Z_{avg}) of all the runs. The best values are reported in boldface characters.



(a) Data profiles for $\tau = 10^{-1}$. (b) Data profiles for $\tau = 10^{-2}$. (c) Data profiles for $\tau = 10^{-3}$.

Figure 9: Comparing the hybrid method to the original and improved PSD-MADS.

Algo	Prob.	Z_{best}	Z_{worst}	Z_{avg}	Prob.	Z_{best}	Z_{worst}	Z_{avg}
Original-5-20	G2-500	-0.15432	-0.058469	-0.11804	CRESCENT-500	-40.52	-12.61	-19.453
Improved-5-20-3QS1		-0.26114	-0.22541	-0.23881		-55.54	**-17.01**	**-27.401**
Hybrid-5-20-3QS1		**-0.26274**	**-0.22974**	**-0.2555**		**-103.98**	-16.19	-25.436
Original-5-20	G2-1000	-0.13701	-0.062465	-0.096506	CRESCENT-1000	-66.823	-17.37	-38.851
Improved-5-20-3QS1		-0.2581	-0.2331	-0.24885		**-171.971**	-44.524	**-88.469**
Hybrid-5-20-3QS1		**-0.2606**	**-0.2483**	**-0.25318**		-120.333	**-50.616**	-79.514
Original-5-20	G2-2000	-0.10528	-0.066553	-0.086113	CRESCENT-2000	-120.354	-31.78	-64.303
Improved-5-20-3QS1		**-0.24533**	-0.22492	**-0.23985**		**-199.131**	**-84.175**	**-137.266**
Hybrid-5-20-3QS1		-0.24344	**-0.23622**	-0.23975		-169.894	-82.831	-129.555
Original-5-20	DISK-500	-34.73	-29.1	-31.415	B250	9.754	120.996	31.894
Improved-5-20-3QS1		-28.482	-15.02	-22.027		**8.6219**	53.388	**19.429**
Hybrid-5-20-3QS1		**-54.7**	**-44.16**	**-49.422**		8.9291	**53.242**	21.065
Original-5-20	DISK-1000	-36.13	-29.85	-33.797	B500	130.334	342.251	258.683
Improved-5-20-3QS1		-52.44	-16.298	-26.921		119.211	399.753	234.291
Hybrid-5-20-3QS1		**-66.77**	**-53.99**	**-61.922**		**80.495**	**322.79**	**194.048**
Original-5-20	DISK-2000	-31.96	-30	-30.95				
Improved-5-20-3QS1		-30.783	-14.956	-21.954				
Hybrid-5-20-3QS1		**-37.185**	**-30.398**	**-32.45**				

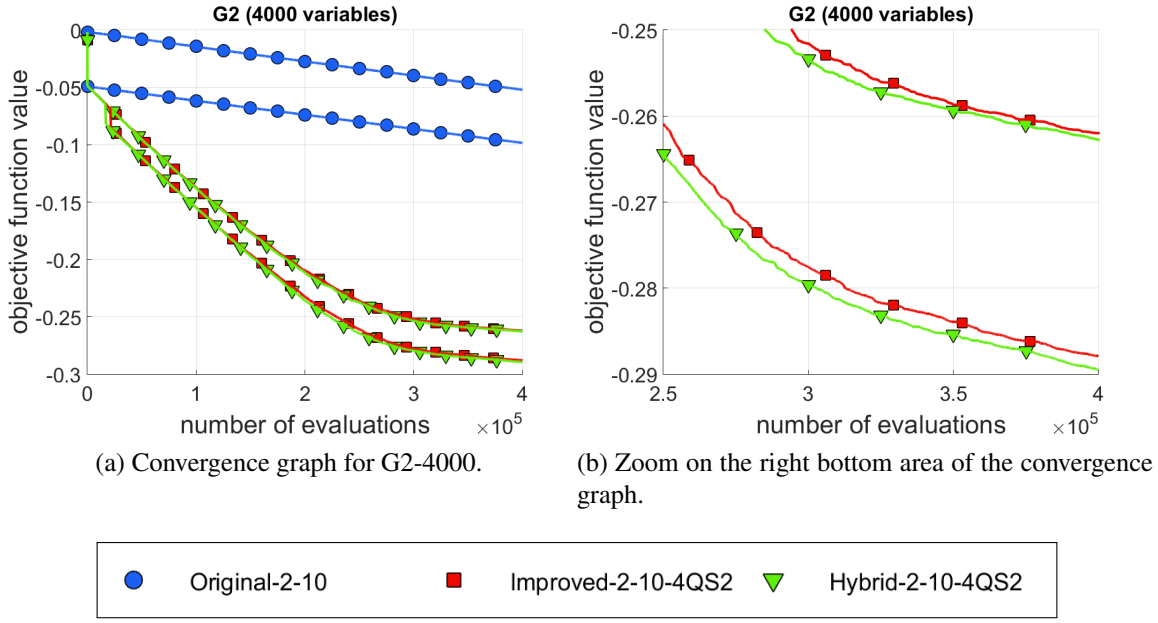
Table 4: Results summary for the original, improved and hybrid methods.

Figure 9 shows that using the k-mean method to exploit the sensitivity matrix alongside the random selection as a strategy to pick variables for building subproblems gives the best results by a large margin: for $\tau = 10^{-3}$, the hybrid approach solves almost twice as much problems than when using the k-mean algorithm alone. In Table 4, the hybrid method has the best Z_{worst} value for 9 problems out of 11. It shows that bringing back the random selection in the improved method helps exploring the solution space better: for problems DISK-500

and DISK-1000, Z_{worst} for the hybrid method has a better value than the Z_{best} for the other two methods.

4.4 Tests on problem with 4,000 variables

Finally, let us set the parameters to their default values found in the previous test results: $\mathcal{P}_{ns} = 2$, $\mathcal{P}_{bbe} = 10$, $\mathcal{P}_{vr} = 4$, $\mathcal{P}_K$ =Q and $\mathcal{P}_S$ =S2. We compare the original, improved and hybrid versions of PSD-MADS over the problem G2 with 4,000 variables. Each algorithm is executed a total of 30 times with the same starting points and only the best and the worst runs for each method are illustrated in the convergence plot in Figure 10 and reported in Table 5.



(a) Convergence graph for G2-4000.

(b) Zoom on the right bottom area of the convergence graph.

Figure 10: Convergence graph of the best and the worst runs of each algorithm with problem G2-4000.

Algo	Prob.	Z_{best}	Z_{worst}	Z_{avg}
Original-2-10	G2-4000	-0.098349	-0.052057	-0.079154
Improved-2-10-4QS2		-0.28788	-0.26198	-0.27218
Hybrid-2-10-4QS2		**-0.28949**	**-0.26274**	**-0.27681**

Table 5: Results summary of the comparison between the original, improved and hybrid versions of PSD-MADS on the G2-4000 problem.

In Figure 10(a), both improved and hybrid methods outperform the original PSD-MADS. The difference between the hybrid and improved approaches is not visible on Figure 10(a).

Nevertheless, Table 5 shows that the hybrid method is slightly better than the improved PSD-MADS method. A zoom on the area between 250,000 and 400,000 evaluations in the convergence graph 10(b) shows the slight superiority of the hybrid method.

5 Discussion

This work proposes a more efficient method to build the subproblems than the random selection of variables used in the original PSD-MADS algorithm. This improved method relies on a statistical approach to build a sensitivity matrix which quantifies the influence of the different variables on the various outputs of a problem. The k-mean algorithm is used to extract information from the sensitivity matrix and provide clusters of variables used to build the subproblems.

The improved PSD-MADS algorithm relies on five parameters which regulate the construction of the sensitivity matrix, the strategies of outputs grouping to extract data from this matrix, the choice of the parameter k for the k-mean algorithm, the upper bound on the subproblems size, and the maximum blackbox evaluations allowed for all subproblems. Several values of these parameters are tested to select a default configuration for the new variable selection strategy. The computational results on constrained problems with up to $4,000$ variables show a significant improvement over the original PSD-MADS, especially when using the hybrid method that switch back to the random selection of variables whenever the k-mean algorithm fails to define subspaces that lead to an improvement of the problem solution.

For future work, one of the extensions that seem natural is to add a search step to the MADS approach used to work on the subproblems. Since the subproblem size is generally small, it seems that using surrogates in a search strategy should improve the results returned by slave processes. Another path to explore is using the PFA method mentioned in Section 2.5. The problem with this approach is that it relies on multiplication and eigenvalues analysis of dense and large matrices which are time expensive. The idea would be to dedicate a slave process to execute the PFA algorithm on the set of history points and populate a list of variable clusters. Another extension involves the strategies discussed in this paper which are designed for using PSD-MADS in large systems and focus on populating all the available processes. Nevertheless, on smaller systems, these strategies have to be redefined to fully exploit the limited number of available processes. One of these strategies, mentioned in Section 3.1, relies on a decomposition between satisfied and violated constraints. This strategy is useful only when PSD-MADS starts with a non-feasible point. Finally, some preliminary tests showed that the sensitivity matrix can detect correlations between outputs, which opens up the way to define a new strategy grouping correlated outputs.

References

[1] 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*, 8(5):1599–1610, 2014.

[2] N. Amaioua, C. Audet, A.R. Conn, and S. Le Digabel. Efficient solution of quadratically constrained quadratic subproblems within a direct-search algorithm. *European Journal of Operational Research*, 268(1):13–24, 2018.

[3] C. Audet, C.-K. Dang, and D. Orban. Efficient use of parallelism in algorithmic parameter optimization applications. *Optimization Letters*, 7(3):421–433, 2013.

[4] C. Audet, C.-K. Dang, and D. Orban. Optimization of algorithms with OPAL. *Mathematical Programming Computation*, 6(3):233–254, 2014.

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

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

[7] 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.

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

[9] C. Audet, M. Kokkolaras, S. Le Digabel, and B. Talgorn. Order-based error for managing ensembles of surrogates in derivative-free optimization. *Journal of Global Optimization*, 70(3):645–675, 2018.

[10] D.P. Bertsekas and J.N. Tsitsiklis. *Parallel and distributed computation: numerical methods*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[11] A.J. Booker, E.J. Cramer, P.D. Frank, J.M. Gablonsky, and J.E. Dennis, Jr. Movars: Multidisciplinary optimization via adaptive response surfaces. AIAA Paper 2007–1927, 2007.

[12] A.J. Booker, J.E. Dennis, Jr., P.D. Frank, D.B. Serafini, V. Torczon, and M.W. Trosset. A Rigorous Framework for Optimization of Expensive Functions by Surrogates. *Structural and Multidisciplinary Optimization*, 17(1):1–13, 1999.

[13] E. Borgonovo. A new uncertainty importance measure. *Reliability Engineering & System Safety*, 92(6):771–784, 2007.

[14] E. Borgonovo and E. Plischke. Sensitivity analysis: a review of recent advances. *European Journal of Operational Research*, 248(3):869–887, 2016.

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

[16] R.I. Cukier, H.B. Levine, and K.E. Shuler. Nonlinear sensitivity analysis of multiparameter model systems. *Journal of computational physics*, 26(1):1–42, 1978.

[17] J.E. Dennis, Jr. and V. Torczon. Direct search methods on parallel machines. *SIAM Journal on Optimization*, 1(4):448–474, 1991.

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

[19] G. Fasano, G. Liuzzi, S. Lucidi, and F. Rinaldi. A linesearch-based derivative-free approach for nonsmooth constrained optimization. *SIAM Journal on Optimization*, 24(3):959–992, 2014.

[20] M.C. Ferris and O.L. Mangasarian. Parallel variable distribution. *SIAM Journal on Optimization*, 4(4):815–832, 1994.

[21] A. Frommer and R.A. Renaut. A unified approach to parallel space decomposition methods. *Journal of Computational and Applied Mathematics*, 110(1):205–233, 1999.

[22] M. Fukushima. Parallel variable transformation in unconstrained optimization. *SIAM Journal on Optimization*, 8(3):658–672, 1998.

[23] U.M. García-Palomares, I.J. García-Urrea, and P.S. Rodríguez-Hernández. On sequential and parallel non-monotone derivative-free algorithms for box constrained optimization. *Optimization Methods and Software*, 28(6):1233–1261, 2013.

[24] U.M. García-Palomares and J.F. Rodríguez. New sequential and parallel derivative-free algorithms for unconstrained optimization. *SIAM Journal on Optimization*, 13(1):79–96, 2002.

[25] G.A. Gray and T.G. Kolda. Algorithm 856: APPSPACK 4.0: Asynchronous parallel pattern search for derivative-free optimization. *ACM Transactions on Mathematical Software*, 32(3):485–507, 2006.

[26] J.D. Griffin and T.G. Kolda. Asynchronous parallel hybrid optimization combining direct and gss. *Optimization Methods & Software*, 25(5):797–817, 2010.

[27] J.D. Griffin, T.G. Kolda, and R.M. Lewis. Asynchronous parallel generating set search for linearly-constrained optimization. *SIAM Journal on Scientific Computing*, 30(4):1892–1924, 2008.

[28] H. Guo and R.A. Renaut. Parallel variable distribution for total least squares. *Numerical linear algebra with applications*, 12(9):859–876, 2005.

[29] C. Han, T. Feng, G. He, and T. Guo. Parallel variable distribution algorithm for constrained optimization with nonmonotone technique. *Journal of Applied Mathematics*, 2013(1):401–420, 2013.

[30] S.-P Han. Optimization by updated conjugate subspaces. In *Numerical analysis (Dundee, 1985)*, volume 140 of *Pitman Res. Notes Math. Ser.*, pages 82–97. Longman Sci. Tech., Harlow, 1986.

[31] N. Hansen. The CMA Evolution Strategy: A Comparing Review. In J. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, editors, *Towards a New Evolutionary Computation*, volume 192 of *Studies in Fuzziness and Soft Computing*, pages 75–102. Springer Berlin Heidelberg, 2006.

[32] J.C. Helton. Uncertainty and sensitivity analysis techniques for use in performance assessment for radioactive waste disposal. *Reliability Engineering & System Safety*, 42(2):327–367, 1993.

[33] R. Hooke and T.A. Jeeves. "Direct Search" Solution of Numerical and Statistical Problems. *Journal of the Association for Computing Machinery*, 8(2):212–229, 1961.

[34] P.D. Hough, T.G. Kolda, and V. Torczon. Asynchronous parallel pattern search for nonlinear optimization. *SIAM Journal on Scientific Computing*, 23(1):134–156, 2001.

[35] R.A. Howard. Uncertainty about probability: A decision analysis perspective. *Risk Analysis*, 8(1):91–98, 1988.

[36] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[37] I.T. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986.

[38] D.R. Jones, C.D. Perttunen, and B.E. Stuckman. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Application*, 79(1):157–181, 1993.

[39] S. Kirkpatrick, C.D. Gelatt Jr., and M.P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, 1983.

[40] T.G. Kolda. Revisiting asynchronous parallel pattern search for nonlinear optimization. *SIAM Journal on Optimization*, 16(2):563–586, 2005.

[41] T.G. Kolda, R.M. Lewis, and V. Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, 45(3):385–482, 2003.

[42] J. Larson and S.M. Wild. A batch, derivative-free algorithm for finding multiple local minima. *Optimization and Engineering*, 17(1):205–228, 2016.

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

[44] C.-S. Liu and C.-H. Tseng. Parallel synchronous and asynchronous space-decomposition algorithms for large-scale minimization problems. *Computational Optimization and Applications*, 17(1):85–107, 2000.

[45] G. Liuzzi and K. Truemper. Parallelized hybrid optimization methods for nonsmooth problems using nomad and linesearch. *Computational and Applied Mathematics*, pages 1–36, 2017.

[46] Y. Lu, I. Cohen, X.S. Zhou, and Q. Tian. Feature selection using principal feature analysis. In *Proceedings of the 15th ACM international conference on Multimedia*, pages 301–304. ACM, 2007.

[47] O.L. Mangasarian. Parallel gradient distribution in unconstrained optimization. *SIAM Journal on Control and Optimization*, 33(6):1916–1925, 1995.

[48] K. Matsui, W. Kumagai, and T. Kanamori. Parallel distributed block coordinate descent methods based on pairwise comparison oracle. *Journal of Global Optimization*, 69(1):1–21, 2017.

[49] Y. Mei, M.N. Omidvar, X. Li, and X. Yao. A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization. *ACM Transactions on Mathematical Software*, 42(2):13, 2016.

[50] Z. Michalewicz and M. Schoenauer. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary computation*, 4(1):1–32, 1996.

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

[52] M.D. Morris. Factorial sampling plans for preliminary computational experiments. *Technometrics*, 33(2):161–174, 1991.

[53] I. Necoara and D. Clipici. Parallel random coordinate descent method for composite minimization: Convergence analysis and error bounds. *SIAM Journal on Optimization*, 26(1):197–226, 2016.

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

[55] T.D. Plantenga. HOPSPACK 2.0 User Manual. Technical Report SAND2009-6265, Sandia National Laboratories, Livermore, CA, October 2009.

[56] N. V Queipo, R. T Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P.K. Tucker. Surrogate-based analysis and optimization. *Progress in aerospace sciences*, 41(1):1–28, 2005.

[57] R.G. Regis. Constrained optimization by radial basis function interpolation for high-dimensional expensive black-box problems with infeasible initial points. *Engineering Optimization*, 46(2):218–243, 2014.

[58] P. Richtárik and M. Takáč. On optimal probabilities in stochastic coordinate descent methods. *Optimization Letters*, 10(6):1233–1243, 2016.

[59] C.A. Sagastizábal and M.V. Solodov. Parallel variable distribution for constrained optimization. *Computational Optimization and Applications*, 22(1):111–131, 2002.

[60] A. Saltelli and J. Marivoet. Non-parametric statistics in sensitivity analysis for model output: a comparison of selected techniques. *Reliability Engineering & System Safety*, 28(2):229–253, 1990.

[61] S. Shan and G.G. Wang. Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. *Structural and Multidisciplinary Optimization*, 41(2):219–241, 2010.

[62] I.M. Sobol. Sensitivity estimates for nonlinear mathematical models. *Mathematical modelling and computational experiments*, 1(4):407–414, 1993.

[63] I.M. Sobol. Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates. *Mathematics and computers in simulation*, 55(1):271–280, 2001.

[64] M.V. Solodov. New inexact parallel variable distribution algorithms. *Computational Optimization and Applications*, 7(2):165–182, 1997.

[65] M.V. Solodov. On the convergence of constrained parallel variable distribution algorithms. *SIAM Journal on Optimization*, 8(1):187–196, 1998.

[66] V. Torczon. *Multi-Directional Search: A Direct Search Algorithm for Parallel Machines*. PhD thesis, Department of Mathematical Sciences, Rice University, Houston, Texas, 1989; available as Tech. Rep. 90-07, Department of Computational and Applied Mathematics, Rice University, Houston, Texas 77005-1892.

[67] L. Wang, Y. Lei, Y. Zeng, L. Tong, and B. Yan. Principal feature analysis: A multivariate feature selection method for fmri data. *Computational and mathematical methods in medicine*, 2013, 2013.

[68] E. Yamakawa and M. Fukushima. Testing parallel variable transformation. *Computational Optimization and Applications*, 13(1–3):253–274, 1999.