Combining the Cross Entropy and the MADS methods for inequality constrained optimization

Charles Audet Romain Couderc Jean Bigeon [*†]

July 7, 2020

Abstract:
Keywords: Cross Entropy; MADS; Derivative-free optimization; Blackbox optimization; Constrained optimization.

[*]GERAD and Département de mathématiques et génie industriel, École Polytechnique de Montréal, C.P. 6079, Succ. Centre-ville, Montréal, Québec, Canada H3C 3A7.

[†]GSCOP 46 avenue FÍix Viallet, 38000, Grenoble

Nomenclature

The next list describes symbols used within the body of the document.In what follows, if the symbol is bold then it is a vector otherwise it is a scalar.

$\boldsymbol{\ell}$	The lower bound of a decision variable
Δ^k	The frame size parameter at iteration k
δ^k	The mesh size parameter at iteration k
ϵ	The stopping criterion
γ	A parameter to estimate in an associated stochastic problem
$\mathbf{u}$	The upper bound of a decision variable
$\mathcal{E}$	The expectation
$\mathcal{N}$	The normal distribution
$\mathcal{V}$	Set of parameters of a probability density function
$\mathcal{X}$	The bounded constraints set of type $\ell \leq x \leq u$
μ	The mean
Ω	The feasible set
ρ	A percentage of quantile
σ	The standard deviation
τ	The mesh size adjustment parameter
$\mathbf{X}$	A random vector
c_j	The j^{th} constraint
D	A positive spanning set
E^k	The set of indices of elite points
F^k	The frame at iteration k
$g(\cdot;\cdot)$	A probability density function
h	The measure of constraints violation
I_x	Indicator function of x

k	The iteration counter
M^k	The mesh at iteration k
n	The dimension of a problem
N_s	Number of sampled data at each iteration
N_e	Number of "elite" population
V	The cache
v	A parameter of a probability density function

1 Introduction

This work studies optimization problems of the form:

$$\min_{\mathbf{x}\in\Omega\subseteq\mathbb{R}^n} f(\mathbf{x}), \quad f : \mathbf{x} \in \mathbb{R}^n \to \mathbb{R} \tag{1}$$

with :

$$\Omega = \{\mathbf{x} \in \mathcal{X} : c_j(\mathbf{x}) \leq 0, j = 1, 2, ..., m\}.$$

The set $\mathcal{X}$ represents bound constraints of type $\boldsymbol{\ell} \leq \mathbf{x} \leq \mathbf{u}$ with $\boldsymbol{\ell}, \mathbf{u} \in \bar{\mathbb{R}}^n$. The specificity of this work is due to the form of the objective function f and of the constraints c_j. They can be the result of a simulation of complex physical phenomena. However, these simulations can take an important amount of time or present some irregularities and therefore classical optimization methods are difficult to apply. Especially, when the gradient of the objective function and/or of the constraints are not explicitly known, hard to compute or its estimation is time consuming. This field is called derivative-free optimization (DFO). In the worst case, the gradient does not even exist, which is called blackbox optimization (BBO). Thus, specialized algorithms, BBO or DFO algorithm have been developed in order to solve this kind of problem.

There are two main categories: model based algorithm [13] and direct search algorithm [6]. This work deals with direct search algorithms which benefit from theoretical convergence results and adding some modifications may improve their performances. In particular, the Mesh Adaptive Direct Search (Mads) algorithm [3] ensures convergence to a point satisfying necessary condition based on the Clarke calculus [11]. This theoretical guarantee, which needs only a locally Lipschitz objective function, is a solid basis for blackbox optimization. However, blackbox optimization algorithms must take into account two other types of difficulties. First, algorithms must be efficient in terms of simulation evaluations (constraints and objective function). Indeed, the simulation in engineering context is often time consuming. Second, blackbox simulations may involve multi-extrema functions, so algorithms must be able to escape from local minimum. Mads may be trapped in a local minimum.

To address the second difficulty, method were developed to escape from local minimum. Several methods had already been combined with Mads for this purpose, including treed gaussian process [19], Variable Neighborhood Search (VNS) [2] or Latin Hypercube Sampling (LHS) [38]. Furthermore, many other methods of global optimization with no convergence guarantees exist, including Simulated Annealing [1], Genetic Algorithm [17] or Tabu search [34]. However, such methods often require a large number of function evaluations which is inconsistent with the first difficulty. In contrast, some methods have been developed to address the first problem, by reducing the overall improve the number of simulation evaluation such that: the use of ensembles of surrogate [7], the use of quadratic models [12] or the integration of Nelder-Mead (NM) algorithm [9]. These different methods improve the efficiency of the Mads algorithm, nevertheless, they do not address the difficulty of local optima.

The objective of the present research is to propose an alternative SEARCH strategy: the Cross Entropy (CE) method. This method is a trade-off between a more global search and a limited number of blackbox evaluations. This method was introduced in 1997 [36], first in a context of rare event in discrete optimization [37] and was adapted to continuous optimization [14].The main benefit of using CE is that it often converges rapidly to a promising region in the space of variables. However, the main drawback is that it does not benefit from theoretical guarantees, and once that it has found a promising region, it requires a large number of simulation evaluations to improve the local accuracy. The two aims of this work are: global exploration with limited number of iterations and keep the convergence guarantee. In this purpose, CE is used as a SEARCH step. The combinations we propose respect the local theoretical convergence guarantee and may hope to escape from local minima with few evaluations. This combination will benefit from CE's ability to identify a promising region, with Mads's ability to perform an efficient local descent.

This paper is divided as follow, Section 2 proposes an overview of Mads and Cross Entropy method. Section 3 presents an algorithm combining CE and Mads. Finally, Section 4 shows the main numerical results comparing the proposed method with Mads-LHS, in which the global exploration is made by a Latin Hypercube Sampling [38]. Section 5 allow to conclude on the work that remains and the contribution of this paper.

2 Description of Mads and Cross Entropy algorithms

In this Section, the Mads algorithm and the CE method are described.

2.1 The Mads constrained optimization algorithm

We consider the Mads algorithm with the progressive barrier (PB) [4] to handle inequality constraints and with dynamic scaling [8] to handle the varying magnitudes of the variables. The target class of optimization algorithm is (1). The PB uses the constraint violation function [15].

$$h(\mathbf{x}) := \begin{cases} \sum_{j=1}^{m} (\max\{c_j(\mathbf{x}), 0\})^2 & \text{if } \mathbf{x} \in \mathcal{X} \subset \mathbb{R}^n \\ \infty & \text{otherwise} \end{cases}$$

The constraints violation function value $h(\mathbf{x})$ is equal to 0 if and only if the point $\mathbf{x}$ belongs to Ω and is nonnegative otherwise.

Mads is a direct search algorithm. Each iteration includes two steps. The SEARCH step where various strategies can be used to explore the space of variables and the POLL step where the space of variable is locally explored by following strict rules guaranteeing convergence. In practice, the search accelerates the convergence to an optimum and it may attempt to

escape from local minimum. The POLL is confined to a region delimited by the so called poll size vector $\boldsymbol{\Delta}^k \in \mathbb{R}^n_+$. All the points generated by the POLL and SEARCH are rounded on a discretization of the space of the variables called the mesh whose the fineness is controlled by the mesh size vector $\boldsymbol{\delta}^k \in \mathbb{R}^n_+$. Formally, the mesh [8] is defined as follows:

$$M^k = V^k + \{diag(\boldsymbol{\delta}^k)\mathbf{z} : \mathbf{z} \in \mathbb{Z}^n\}$$

where the cache V^k contains all the points visited at the iteration k. The mesh and the poll size vectors are updated at the end of each iteration. The coordinates of both vectors are reduced when an iteration fails to improve the current solution and they are increased or remain at the same value otherwise. Algorithm 1 provides a description of Mads, the reader may consult [8] for more details.

Algorithm 1: The Mesh Adaptive Direct Search algorithm (Mads)

Given a user-defined set of starting point: $V^0 \subset \mathbb{R}^n$,
and initial mesh and poll size vectors: typically $\delta_i^0 = \Delta_i^0 = 1$ for $i = 1, 2, \ldots, n$.
Set the iteration counter: $k \leftarrow 0$.

1. Search step (optional):
   Launch the simulation on a finite set S^k of mesh points.
   If successful, go to 3.
2. Poll step:
   Launch the simulation on the set P^k of poll points.
3. Updates:
   Update the cache V^{k+1}, the incumbent $\mathbf{x}^{k+1}$
   and the mesh and poll size vectors $\boldsymbol{\delta}^{k+1}$ and $\boldsymbol{\Delta}^{k+1}$.
   Increase the iteration counter $k \leftarrow k + 1$ and go to 1.

The fundamental convergence result [4] of the Mads algorithm states that if the entire sequence of trial points belongs to a bounded set, and if the set of so-called refining directions is sufficiently rich, then there exists an accumulation point $\mathbf{x}^*$ such that the generalized directional derivative $f^\circ(\mathbf{x}^*; \mathbf{d})$ of Clarke [11] is nonnegative in every hypertangent [22] direction $\mathbf{d}$ to the domain Ω at $\mathbf{x}^*$ provided that $\mathbf{x}^*$ is feasible. A similar result holds for h over X in situation where the iterates never approach the feasible region. In the present work, we will see in Section 3 how to keep this convergence guarantee.

2.2 The Cross Entropy method for continuous optimization

The Cross Entropy method was introduced by Rubinstein in 1997 in the context of a minimization algorithm for estimating probabilities of rare events [36]. Later, it was modified to solve combinatorial optimization problems [37] and then in 2006 to solve continuous problems [14]. The main idea of this method is as follows. First, each optimization problem must be transformed in a rare event estimation problem called associated stochastic problem

(ASP): the deterministic problem (1) is transformed into the related stochastic estimation problem:

$$l(\gamma) = P(f(\mathbf{X}) \geq \gamma) = E(I_{f(\mathbf{X})\geq\gamma}) \tag{2}$$

where $\mathbf{X}$ is a random vector. Then, this ASP must be tackled efficiently by an adaptive algorithm. This algorithm constructs a sequence of solutions which converges to the optimal solution of the ASP. Therefore, there are two iterative steps:

- Generation a sample of random data according to a density of probability.
- Updating the parameters of the density thanks to the data sampled to create a "better" sample in the next iteration.

The main advantage of this method is that it allows to escape from local minima.

2.2.1 An introductory example

For clarity, consider the example from [14] of maximizing the function:

$$f(x) = e^{-(x-2)^2} + 0.8e^{(x+2)^2}, x \in \mathbb{R}. \tag{3}$$

The function f has two local maxima and a single global maximum at $x = 2$.

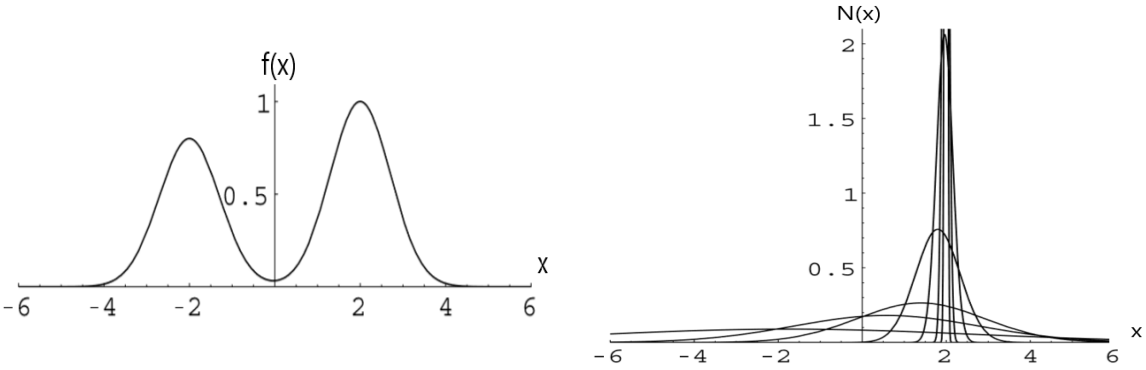

Figure 1: (Image taken from [14]) Graph of objective function f (left) and evolution of the normal law during the seven first iterations with $N_e = 10$ and $N_s = 100$ (right)

Using a normal distribution the CE procedure is the following:

- Initialization : at the first iteration $k = 0$, a mean $\boldsymbol{\mu}^0 \in \mathbb{R}^n$ and a standard deviation $\boldsymbol{\sigma}^0 \in \mathbb{R}^n$ (with n the dimension of the problem) are arbitrarily chosen. A large value of $\boldsymbol{\sigma}^0$ is taken in order to escape from local solutions.
- Iterative part: at each iteration $k \geq 1$:

- First, a sample $\mathbf{X}_1, ..., \mathbf{X}_{N_s}$ of points in $\mathbb{R}^n$ is generated from a normal law $\mathcal{V}(\boldsymbol{\mu}^{k-1}, \boldsymbol{\sigma}^{k-1})$ where N_s is the number of samples.
- Then, f is evaluated at each sampled points and a number of "elite" points N_e, with the highest value of f. $\boldsymbol{\mu}^k$ and $\boldsymbol{\sigma}^k$ are the mean and standard deviation of these N_e points.
- Termination: once the standard deviation becomes sufficiently small, the procedure is stopped.

If $\mathcal{N}$ denotes the normal distribution then $\mathcal{N}$ will evolve as in Figure 2. This example shows how the CE procedure escapes from the local maximum and converges in seven iterations to the neighborhood of 2.

2.2.2 The general CE method

Before presenting the algorithm, we consider the ASP and precise the two iterative steps of the algorithm. Problem (1) is transformed into an ASP. With this purpose, a family of probability distribution function (pdf) $\{g(\cdot; \mathbf{v}) : \mathbf{v} \in \mathcal{V}\}$ is defined. g is the law chosen to sample the different points at each iteration. $\mathcal{V}$ is the set of vector parameters of the pdf g which are calculated at each iteration. In the previous example g is the normal law and $\mathcal{V}$ is the set of $\mathbf{v}^k = (\boldsymbol{\mu}^k, \boldsymbol{\sigma}^k)$. Therefore, the ASP related to (1) as follows:

$$l(\gamma) = P_{\mathbf{v}}(f(\mathbf{X}) \geq \gamma) = E_{\mathbf{v}}(I_{\{f(\mathbf{X}) \geq \gamma\}}) \tag{4}$$

where $\mathbf{v} \in \mathcal{V}$ is a vector of parameter and $\mathbf{X}$ is a random vector with a pdf $g(\cdot; \mathbf{v})$. γ is a variable. At this stage, for a given value of γ, the parameter $\mathbf{v}$ may be estimated. Conversely, given a vector of parameters $\mathbf{v}$, γ may be also estimated. The CE method is based on these two estimations, at each iterations, the algorithm tries to estimate one then the other one. In Example (3), in the iterative part, the first item corresponds to the estimation of γ and the second one to the estimation of $\mathbf{v}$. More precisely, we denote $\gamma^* \in \mathbb{R}$ as the supremum of the objective function, $\mathbf{v}^*$ the parameters and $g(\cdot; \mathbf{v}^*)$ the pdf associated to this supremum. Then our goal is to generate a sequence of $(\gamma^k, \mathbf{v}^k)$ which converge to $(\gamma^*, \mathbf{v}^*)$. To achieve this goal, a sequence of pdf $g(\cdot; \mathbf{v}^0), g(\cdot; \mathbf{v}^1), ...$ which converges to $g(\cdot; \mathbf{v}^*)$ is created. To assure the convergence, one must have a "measure" of the difference between the iterate pdf $g(\cdot; \mathbf{v}^k)$ and the objective one $g(\cdot; \mathbf{v}^*)$. The Kullback-Leibler (KL) divergence [25] is used:

$$D(g(\cdot; \mathbf{v}^*)||g(\cdot; \mathbf{v}^k)) = \int_{-\infty}^{\infty} g(\mathbf{x}; \mathbf{v}^*) \ln \left(\frac{g(\mathbf{x}; \mathbf{v}^*)}{g(\mathbf{x}; \mathbf{v}^k)} \right) d\mathbf{x}. \tag{5}$$

The iterative steps may now be described. ρ is defined as a very small quantity, corresponding to the proportion of "elite" points which are kept from an iteration to another. The procedure is:

- **Adaptive update of γ^k.** With a fixed parameter of pdf $\mathbf{v}^{k-1}$, γ^k is defined such that it is the $(1-\rho)$-quantile of $f(\mathbf{X})$ under v_{k-1}. Then :

$$P_{\mathbf{v}^{k-1}}(f(\mathbf{X}) \geq \gamma^k) \geq \rho, \tag{6}$$

$$P_{\mathbf{v}^{k-1}}(f(\mathbf{X}) \leq \gamma^k) \geq 1-\rho \tag{7}$$

where $X \sim g(\cdot; \mathbf{v}^{k-1})$. The γ^k is denoted $\hat{\gamma}^k$. To obtain this estimator, a sample $\mathbf{X}_1, ..., \mathbf{X}_{N_s}$ is drawn from $g(\cdot; \mathbf{v}^{k-1})$ and evaluated. Then, the $(1-\rho)$ quantile is:

$$\hat{\gamma}^k = f_{\lceil (1-\rho)n \rceil}. \tag{8}$$

- **Adaptive update of $\mathbf{v}^k$.** With a fixed γ^k and knowing $\mathbf{v}^{k-1}$, $\mathbf{v}^k$ is a solution of:

$$\begin{aligned} \max_{\mathbf{v}} D(\mathbf{v}) &= \max_{\mathbf{v}} E_{\mathbf{v}^{k-1}} I_{\{f(\mathbf{X}) \geq \gamma^k\}} \ln(g(\mathbf{X}; \mathbf{v})) \\ &= \min_{\mathbf{v}} E_{\mathbf{v}^{k-1}} I_{\{f(\mathbf{X}) \geq \gamma^k\}} \ln\left(\frac{I_{\{f(\mathbf{X}) \geq \gamma^k\}}}{g(\mathbf{X}; \mathbf{v})}\right) \end{aligned} \tag{9}$$

which is the minimization of the KL divergence at iteration k (taking the convention $0 \ln(0) = 0$). Nevertheless, in practice, the real expectation and the real γ^k are not known, estimators must be used and the following equation is solved rather:

$$\tilde{\mathbf{v}}^k \in \max_{\mathbf{v}} \widehat{D}(\mathbf{v}) = \max_{\mathbf{v}} \frac{1}{N} \sum_{i=1}^{N} I_{\{f(\mathbf{X}_i) \geq \hat{\gamma}^k\}} \ln(g(\mathbf{X}_i; v). \tag{10}$$

Last but not least, $\hat{\mathbf{v}}^k$ is not set to $\tilde{\mathbf{v}}^k$. Indeed, some component of $\tilde{\mathbf{v}}^k$ could be set to 0 or 1 at the first few iterations and the transition between $\tilde{\mathbf{v}}^{k-1}$ and $\tilde{\mathbf{v}}^k$ could be discontinuous. To avoid these problems, we use rather the following convex combination:

$$\hat{\mathbf{v}}^k = \alpha \tilde{\mathbf{v}}^k + (1-\alpha)\hat{\mathbf{v}}^{k-1} \tag{11}$$

with $0 < \alpha \leq 1$. Theoretically, we can use any distribution which converges in the neighborhood where the global maximum is attained as normal, exponential or beta distribution. Nevertheless, in practice, the updating step is quite simple with the normal distribution that one chooses often this one. Therefore, the detailed algorithm is the following:

Algorithm 2: The Cross Entropy (CE) algorithm with a normal law pdf

Choose $\hat{\boldsymbol{\mu}}^0$ and $\hat{\boldsymbol{\sigma}}^0$
Set the iteration counter: $k \leftarrow 0$.
N_s number of sampled data at each iteration
N_e number of elite population
α the parameter of convex combination

1. Estimation of γ^k:
   Generate a random sample $\mathbf{X}_1, .., \mathbf{X}_{N_s}$ from $N(\hat{\boldsymbol{\mu}}^{k-1}, \hat{\boldsymbol{\sigma}}^{k-1})$ distribution.
   Evaluation of the N_s points by the simulation and then go to 2.
2. CE step
   Let E^k be the indices of the N_e best perfoming samples.
   Set $\tilde{\boldsymbol{\mu}}^k = \frac{1}{N_e} \sum_{i \in E^k} \mathbf{X}_i$
   and $(\tilde{\boldsymbol{\sigma}}^k)^2 = \frac{1}{N_e - 1} \sum_{i \in E^k} (\mathbf{X}_i - \boldsymbol{\mu}^k)^2$
3. Updates:
   Apply the convex combinations:
   $\hat{\boldsymbol{\mu}}^k = \alpha \tilde{\boldsymbol{\mu}}^k + (1 - \alpha) \hat{\boldsymbol{\mu}}^{k-1}$
   $\hat{\boldsymbol{\sigma}}^k = \alpha \tilde{\boldsymbol{\sigma}}^k + (1 - \alpha) \hat{\boldsymbol{\sigma}}^{k-1}$
   Increase the iteration counter $k \leftarrow k + 1$ and go to 1.

3 The CE-MADS constrained optimization algorithm

We now present our CE-inspired MadsSEARCH step. Section 3.1 presents the constraints handling, the update of $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ and the condition to enter in the CE search step. Section 3.2 presents the algorithm of the CE SEARCH step.

3.1 The CE-search step

3.1.1 Handling the constraints

Section 2 presents the CE method for unconstrained optimization. Nevertheless in [14], they can treat the constrained case: bound constraints are treated using a truncated normal law and a penalty approach is used for the general inequality constraints. In our work, we use also the truncated normal law to treat the bound constraints. For the general inequality constraints, we keep also the penalty approach instead of a method based on progressive barrier [4]. That allows to vary the method to handle the constraints and produce a good result finally. When the algorithm must choose the elite sample, it uses the following function Best (defined in [9] and recalled here). The definition relies on both the objective and the constraint violation functions f and h.

Definition 3.1. *The function* Best : $\mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}^n$

$$\mathsf{Best}(\boldsymbol{x},\boldsymbol{y}) = \begin{cases} \boldsymbol{x} & \textit{if } \boldsymbol{x} \textit{ dominates } \boldsymbol{y} \textit{ or if } h(\boldsymbol{x}) < h(\boldsymbol{y}), \\ \boldsymbol{y} & \textit{if } \boldsymbol{y} \textit{ dominates } \boldsymbol{x} \textit{ or if } h(\boldsymbol{y}) < h(\boldsymbol{x}), \\ Older(\boldsymbol{x},\boldsymbol{y}) & \textit{Otherwise} \end{cases}$$

returns the best *of two points.*

The function *Older* gives the point which was generated before the former one. Thanks to this definition, CE may treat the general inequality constraints with the terminology used in Mads.

3.1.2 Update of the mean and standard deviation

Three elements differ compared to classical CE method concerning the mean and the standard deviation. First, the elite points taken to compute the mean and the standard deviation are not only the N_s points generated by the normal law. The elite points are chosen from the cache at the iteration k, denotes $V^k \subset \mathbb{R}^n$, so any points generated by the Mads algorithm may be selected. This set is ordered with the function Best, in order to select the N_e elite points, it is sufficient to take the N_e first of points of V^k.

Second, the mean and the standard deviation initialization procedure differs from the CE method proceeds. Indeed, Mads always begins with a starting point, thus there is at least one point in the cache (the set of evaluated points). Moreover, to avoid to field of exploration, we always bound the problem as follows (using $\bar{\mathbf{x}}^k$ the poll center at iteration k):

$$\forall i \in [1,n]\ (\ell_i^k, u_i^k) = \begin{cases} (\ell_i, u_i) & \text{if } u_i \neq \infty \text{ and } \ell_i \neq -\infty, \\ (\bar{x}_i^k - 10 \times \Delta_i^k, u_i) & \text{if } u_i \neq \infty \text{ and } \ell_i = -\infty, \\ (\ell_i, \bar{x}_i^k + 10 \times \Delta_i^k) & \text{if } u_i = \infty \text{ and } \ell_i \neq -\infty. \end{cases} \tag{12}$$

Once the problem has finite bound constraints, there are two cases to calculate the mean and the standard deviation:

- In case where the number of points in the cache is too small to be relevant, i.e. less point that the number of problem dimension, then the mean and the standard deviation are determined such that:

$$\boldsymbol{\mu}^k = \bar{\mathbf{x}}^k \tag{13}$$

$$\boldsymbol{\sigma}^k = \mathbf{u}^k - \boldsymbol{\ell}^k \tag{14}$$

- In the others cases, we make the same calculation that in the original CE process:

$$\boldsymbol{\mu}^k = \frac{1}{N_e} \sum_{j \in E^k} \mathbf{X}_j$$

$$\boldsymbol{\sigma}^k = \sqrt{\frac{1}{N_e - 1} \sum_{j \in E^k} (\mathbf{X}_j - \boldsymbol{\mu}^k)^2}$$

Third, to generate the point during the CE search, we always used the truncated normal law with the bounds created in (12). Moreover, the "elite" points come not only from the previous normal sampling but also of the other kind of search step. That gives a vector of standard deviation which tends to zero very quickly, the other methods doing generally a local search. That is why, to avoid that the standard deviation is calculated as in 11 with a coefficient $\alpha = 0.95$.

3.1.3 The condition to pass in the CE-SEARCH step

The goal of the CE method is to explore in few evaluations the space to determine the promising region. The number of evaluations used by the CE SEARCH step must be quite small. For this purpose, the algorithm does not perform at each iteration the CE SEARCH step. The standard deviation can be seen as a measure of the incertitude on the data, we use it to define the condition to pass through the CE SEARCH step. First, a new variable called $\boldsymbol{\sigma}^p$ is introduced, it represents the incertitude measured the last time the algorithm passes through the CE SEARCH step. This variable is initialized to ∞. Then, the condition to pass in the CE SEARCH is the following:

$$\frac{1}{n}\sum_{i=0}^{n-1}(\sigma_i^k)^2 < \frac{1}{n}\sum_{i=0}^{n-1}(\sigma_i^p)^2 \tag{15}$$

This conditions means that the current incertitude is smaller than the previous one, the interesting area is refined. Each time these conditions are respected, $\boldsymbol{\sigma}^p$ is updated with the standard deviation obtained after the CE step.

3.2 The complete algorithm

The CE SEARCH step of Mads algorithm is presented here:

Algorithm 3: The CE SEARCH step

1. Calculation of $\boldsymbol{\mu}^k$ and $\boldsymbol{\sigma}^k$:
   - if $card(E^k) < n$:
     - $\boldsymbol{\mu}^k = \bar{\mathbf{x}}_0$
     - $\boldsymbol{\sigma}^k$ update with 14
   - else:
     - $\boldsymbol{\mu}^k = \frac{1}{N_e} \sum_{j \in E^k} \mathbf{X}_j$
     - $(\boldsymbol{\sigma}^k)^2 = \frac{1}{N_e - 1} \sum_{j \in E^k} (\mathbf{X}_j - \boldsymbol{\mu}^k)^2$
2. CE SEARCH
   - If $\frac{1}{n} \sum_{i=0}^{n-1} (\sigma_i^k)^2 < \frac{1}{n} \sum_{i=0}^{n-1} (\sigma_i^p)^2$:
     - Generate a random sample $\mathbf{X}_1, .., \mathbf{X}_{N_s}$ from $\mathcal{N}(\boldsymbol{\mu}^k, \boldsymbol{\sigma}^k)$ distribution.
     - Evaluation of the N_s points by the simulation.
     - Update:
     - $\boldsymbol{\mu}^{k+1} = \frac{1}{N_e} \sum_{j \in E^k} \mathbf{X}_j$
     - $(\boldsymbol{\sigma}^{k+1})^2 = \frac{1}{N_e - 1} \sum_{j \in E^k} (\mathbf{X}_j - \boldsymbol{\mu}^{k+1})^2$
     - $(\boldsymbol{\sigma}^p)^2 = (\boldsymbol{\sigma}^{k+1})^2$

4 Computational experiments

In the present work, we use data profiles to compare the different algorithm. Data profiles [32] are presented below to assess if algorithms are successful in generating solution values close to the best objective function values. To identify a successful run, a convergence test is required. Let denote $\mathbf{x}_e$ the best iterates obtained by one algorithm on one problem after e evaluations, $\mathbf{x}_0$ the first feasible iterates (if it exists otherwise the test is failed) and f^* the best solution obtained by all tested algorithms on all run instances of that problem. Then, the problem is said to be solved within the convergence tolerance τ when:

$$f(\mathbf{x}_0) - f(\mathbf{x}_e) \geq (1 - \tau)(f(\mathbf{x}_0) - f^*)$$

In the present work, we consider that different initial points constitute different problem. Moreover, an instance of a problem corresponds to a particular pseudo-random generator seeds. The horizontal axis of a data profile represents the number of evaluations for problems of fixed dimension, and represents group of $n + 1$ evaluations when problems of different dimension are involved. The vertical axis corresponds to the proportion of problems solved within a given tolerance τ. Each algorithm has its curve to allow comparison of algorithms capability to converge to the best objective function value.
In this section, we present the different numerical results we obtained. It is divided in two

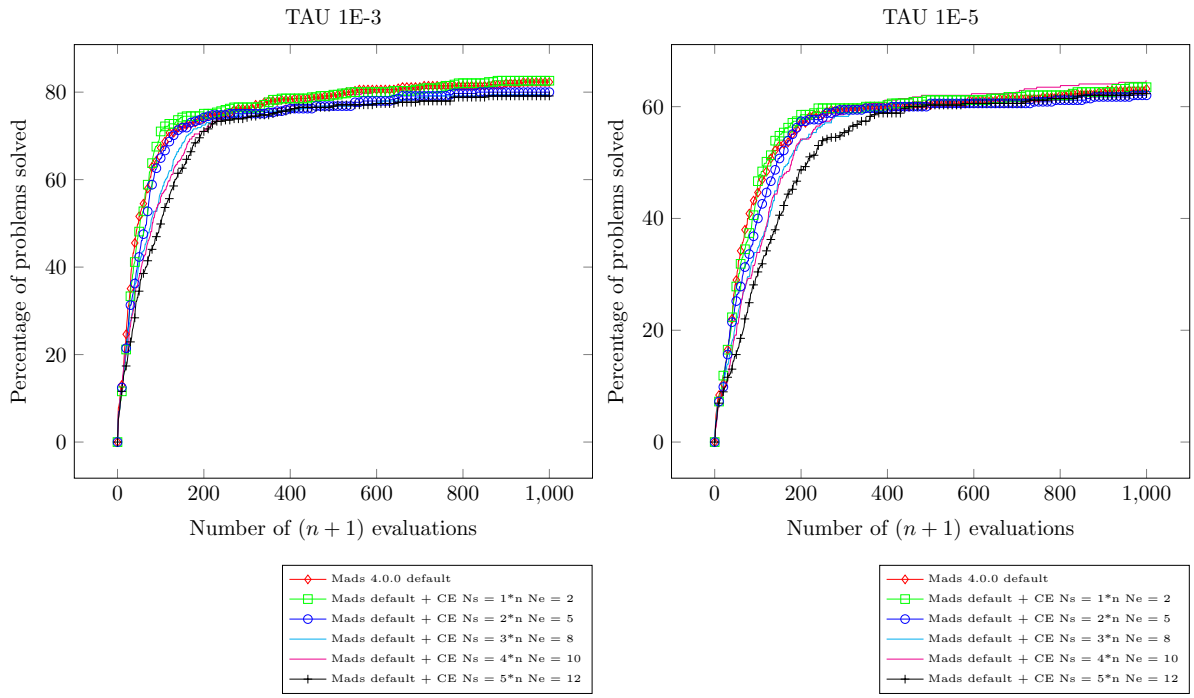

Figure 2: Result of calibration of the hyper-parameter of CE-MADS on the 69 unconstrained test problems

subsections: Section 4.1 the results obtained using a CE SEARCH step to calibrate different parameters and compare different results on analytical problems and Section 4.2 the results obtained on three real engineering problems.

4.1 Preliminary experiments to calibrate parameters

In this section, computational experiments are conducted using the beta version 4.0.0 and the version 3.9.1 of NOMAD [26] software package. All tests use the Mads strategy with $n+1$ poll directions, with the use of the NM search [9] and without the use of quadratic models. In addition, the CE search is used as the first SEARCH step.

Numerical experiments on analytical test problems are conducted to set default values to the two algorithmic parameters: the number of sampled data at each iteration N_s and the number of "elite" population N_e. Mads-CE is tested on 100 analytical problems from the optimization literature. The characteristics and sources of these problems are summarized in Table 1 in appendix. The number of variables ranges from 2 to 60; 26 problems have constraints other than bound constraints. In order to have a more precise idea of the difference between the hyper-parameter, we have made three series of tests:

- A series of tests on the 69 unconstrained test problems with a number maximum of function evaluations of 10 000 and each problem is run on 5 different seeds.

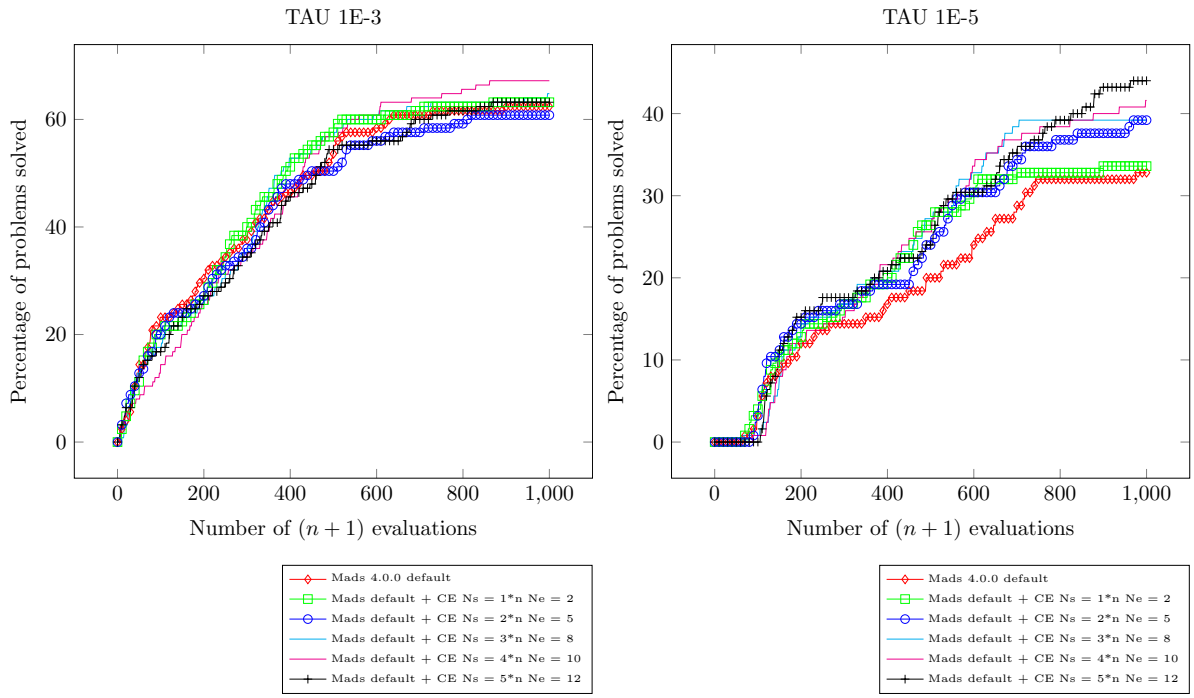

Figure 3: Result of calibration of the hyper-parameter of CE-MADS on the 25 constrained test problems

- A series of tests on the 25 constrained test problems with a number maximum of function evaluations of 10 000 and each problem is run on 5 different seeds.
- A series of tests on the 6 bigger problem in term of dimension, three are constrained and the three others are not. We put the maximum number of evaluations of 50 000 and each problem is run on 5 different seeds.

For each series of tests, we compare the five following CE-MADS setup of hyper-parameters: $(N_e, N_s) = \{(2, n), (5, 2n), (8, 3n), (10, 4n), (12, 5n)\}$ with n the dimension of the test problem. We also add a run of NOMAD default in each series of test to calibrate our result with the current NOMAD software. We present the result on Figure 2, 3 and 4 with different values of τ.

We analyse these results by series of problems:

- On the unconstrained problems, there are few differences between the difference algorithms regardless of the value of τ, it is difficult to choose one hyper-parameter rather than another one even if the couple $N_e = 2$ and $N_s = n$ seems to be more efficient.
- On the constrained problems, it is a bit different according to the value of τ. For $\tau = 1e-3$, none algorithms seems to be dominant. In the contrary for $\tau = 1e-5$, we can see that greater is the value of N_s, higher is the percentage of problems solved finally. That can be explained because a great N_s allows a better exploration of the space, and so a more precise result at the end.

- Finally, on the big problems, once again we must discuss according to the value of τ. For $\tau = 1e-2$, none hyper parameters may be distinguished and the CE-Mads algorithm seem to be more efficient than the Mads-default. This latter result is still more noticeable with the value of $\tau = 1e-3$. However, again none of sets of hyper parameter is really dominant compared to the other ones.

To conclude on the calibration of the hyper parameter of CE-Mads, we can say two things:

- The values of the hyper parameters is not very important in the performance of the CE-Mads algorithm. It is rather a good thing because that allows to avoid some calibration experiments before to apply the algorithm on a new test problem.

- In case where you have a large budget of function evaluations, you can try to increase the N_s parameters, because it seems to improve the solution quality. Otherwise used a small number of N_s may be more efficient.

In the following of the paper, since we do not accord a big amount of budget evaluations for the test problems, we choose a trade-off between exploration and efficiency $N_e = 4$ and $N_s = 2n$.

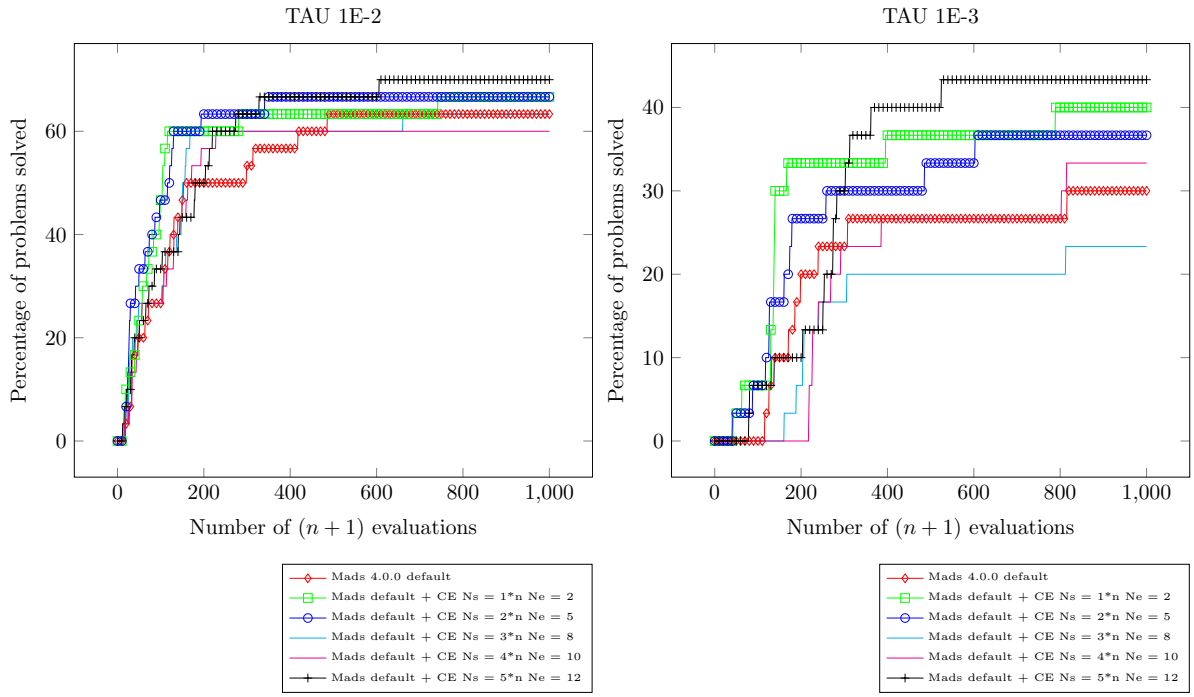

Figure 4: Result of calibration of the hyper-parameter of CE-MADS on the 6 "big" test problems

4.2 Test on engineering problems

In this section, we test the CE-Mads algorithm on three different engineering problems. We compare its results with two algorithms, the Mads-default (without quadratic models) and the LH-Mads which is a default Mads with in addition a LHS search. The comparison with the last algorithm is crucial because it is the LH search is a method aiming to explore the space of design variables.

4.2.1 The MDO problems

Mads-default, CE-Mads and LH-Mads are tested to solve a simple multidisciplinary wing design optimization problem [16]. Each initial point defines a MDO problem. Solving the problem consists in maximizing the range of an aircraft subject to 10 general constraints. The problem has 10 scaled design variables bounded in $[0; 100]$. Figure 5 shows the result on a data profile when solving 20 MDO problems on different initial points using 3000 function evaluations or less. The initial points are real randomly selected within the bounds. Each run is done with three different seeds in order to minimize the impact of the seed.

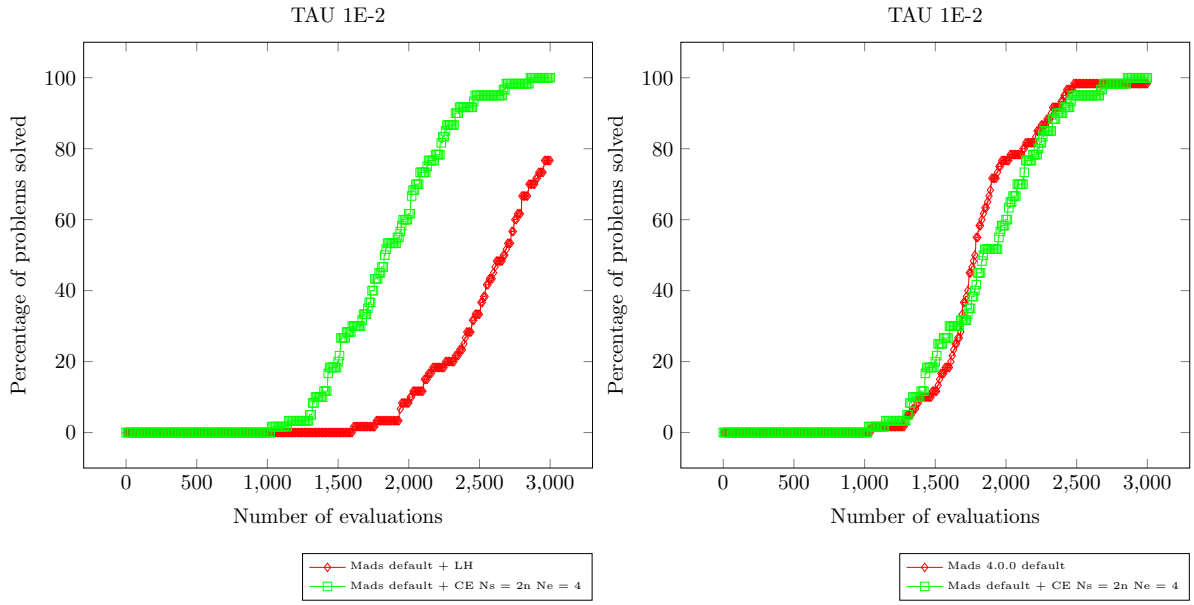

Figure 5: Result on the 20 MDO test problems: between CE-MADS and LH-MADS (left) and between CE-MADS and MADS default (right).

On the plot to compare the LH-Mads with the CE-Mads, we can observe that CE-Mads outperformed the mads with LH. The comparison between CE-Mads and Mads-default is much tinier. The behavior of the both algorithms are very similar. There is non dominant algorithm in this case.

4.2.2 The STYRENE problems

The Mads-default, CE-Mads and LH-Mads algorithms are tested to optimize a styrene production process [2]. This problem is a simulation of a chemical process. This process relies on a series of interdependent calculation of blocks using common numerical tools as Runge-Kutta, Newton, fixed point and also chemical related solver. The particularity of this problem is the presence of "hidden" constraints, i.e. sometimes the process does not finish and just return an error. In the case where the chemical process ends, the constraints (not hidden) and the objective functions may be evaluated during a post-processing. The objective is to maximize the net value of the styrene production process with 9 industrial and environmental regulations constraints.

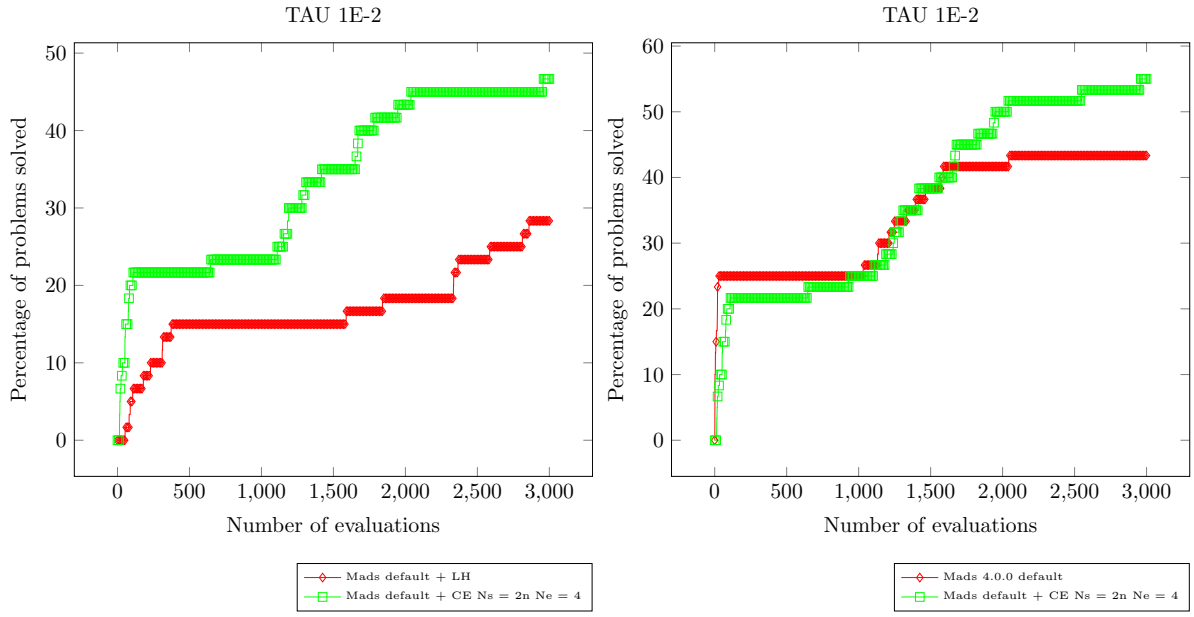

Figure 6: Result on the 20 STYRENE test problems: between CE-MADS and LH-MADS (left) and between CE-MADS and MADS default (right).

In this work, a STYRENE problem possesses eight independent variables influencing the styrene production process. The variables considered during the optimization process are all scaled and bounded in $X = [0, 100]^8$. As we do for the MDO test problems, we test the three algorithms with 20 different starting points taken in X. We use a maximal number of evaluations of 3000 and we run each problem with three different seeds. The results are provided on the Figure 6. In both comparison, we can see that the CE-Mads algorithm outperformed the other algorithms.

4.2.3 The LOCKWOOD problems

5 Discussion

A Appendix

#	Name	Source	n	m	Bnds
1	ARWHEAD10	[18]	10	0	no
2	ARWHEAD20	[18]	20	0	no
3	BARD	[31]	3	0	no
4	BDQRTIC10	[18]	10	0	no
5	BDQRTIC20	[18]	20	0	no
6	BEALE	[31]	2	0	no
7	BIGGS	[18]	6	0	no
8	BOX	[31]	3	0	no
9	BRANIN	[20]	2	0	yes
10	BROWNAL5	[18]	5	0	no
11	BROWNAL7	[18]	7	0	no
12	BROWNAL10	[18]	10	0	no
13	BROWNAL20	[18]	20	0	no
14	BROWNDENNIS	[31]	4	0	no
15	BROWN_BS	[31]	2	0	no
16	B250		60	1	yes
17	B500		60	1	yes
18	CHENWANG_F2_X0	[10]	8	6	yes
19	CHENWANG_F2_X1	[10]	8	6	yes
20	CHENWANG_F3_X0	[10]	10	8	yes
21	CHENWANG_F3_X1	[10]	10	8	yes
22	CRESCENT	[4]	10	2	no
23	DISK	[4]	10	1	no
24	DIFFICULT2	[4]	10	0	no
25	ELATTAR	[27]	6	0	no
26	EVD61	[27]	6	0	no
27	FILTER	[27]	9	0	no
28	FREUDENSTEINROTH	[31]	2	0	no
29	GAUSSIAN	[31]	3	0	no
30	G2_10	[5]	10	2	yes
31	G2_20	[5]	20	2	yes
32	G2_50	[5]	50	2	yes
33	GOFFIN		50	0	no
34	GRIEWANK	[20]	10	0	yes
35	GULFRD		3	0	no
36	HELICALVALLEY	[31]	3	0	no
37	HS19	[21]	2	2	yes
38	HS78	[27]	5	0	no
39	HS83_X0	[21]	5	6	yes
40	HS83_X1	[21]	5	6	yes
41	HS114_X0	[27]	9	6	yes
42	HS114_X1	[27]	9	6	yes
43	JENNRICHSAMPSON	[31]	2	0	no
44	KOWALIKOSBORNE	[31]	4	0	no
45	L1HILB		50	0	no
46	MAD6_X0	[27]	5	7	no
47	MAD6_X1	[27]	5	7	no
48	MCKINNON	[28]	2	0	no
49	MEYER	[31]	3	0	no
50	MEZMONTES	[29]	2	2	yes

#	Name	Source	n	m	Bnds
51	MXHILB		50	0	no
52	OPTENG_RBF	[24]	3	4	yes
53	OSBORNE1	[31]	5	0	no
54	OSBORNE2	[27]	11	0	no
55	PBC1	[27]	5	0	no
56	PENALTY1_4	[18]	4	0	no
57	PENALTY1_10	[18]	10	0	no
58	PENALTY1_20	[18]	20	0	no
59	PENALTY2_4	[18]	4	0	no
60	PENALTY2_10	[18]	10	0	no
61	PENALTY2_20	[18]	20	0	no
62	PENTAGON	[27]	6	15	no
63	PIGACHE_X00	[33]	4	11	yes
64	PIGACHE_X01	[33]	4	11	yes
65	POLAK2	[27]	10	0	no
66	POWELL_BS	[31]	2	0	no
67	POWELLSG4	[18]	4	0	no
68	POWELLSG8	[18]	8	0	no
69	POWELLSG12	[18]	12	0	no
70	POWELLSG20	[18]	20	0	no
71	RADAR	[30]	7	0	yes
72	RANA	[23]	2	0	yes
73	RASTRIGIN	[20]	2	0	yes
74	RHEOLOGY	[6]	3	0	no
75	ROSENBROCK	[31]	2	0	yes
76	SHOR	[27]	5	0	no
77	SNAKE	[4]	2	2	no
78	SPRING_X00	[35]	3	4	yes
79	SPRING_X01	[35]	3	4	yes
80	SROSENBR6	[18]	6	0	no
81	SROSENBR8	[18]	8	0	no
82	SROSENBR10	[18]	10	0	no
83	SROSENBR20	[18]	20	0	no
84	TAOWANG_F2_X00	[39]	7	4	yes
85	TAOWANG_F2_X01	[39]	7	4	yes
86	TREFETHEN	[23]	2	0	yes
87	TRIDIA10	[18]	10	0	no
88	TRIDIA20	[18]	20	0	no
89	TRIGONOMETRIC	[31]	10	0	no
90	VARDIM8	[18]	8	0	no
91	VARDIM10	[18]	10	0	no
92	VARDIM20	[18]	20	0	no
93	WANGWANG_F3	[40]	2	0	yes
94	WATSON9	[31]	9	0	no
95	WATSON12	[31]	12	0	yes
96	WONG1	[27]	7	0	no
97	WONG2	[27]	10	0	no
98	WOODS4	[18]	4	0	no
99	WOODS12	[18]	12	0	no
100	WOODS20	[18]	20	0	no

Table 1: Description of the set of 100 analytical problems.

References

[1] C.Martini A. Corana, M. Marchesi and S.Ridella. Minimizing multimodel functions of continuous variables with "Simulated Annealing" algorithm. *Transactions on Mathematical Software*, 13(3):262–280, 1987.

[2] C. Audet, V. Béchard, and S. Le Digabel. Nonsmooth optimization through Mesh Adaptive Direct Search and Variable Neighborhood Search. *Journal of Global Optimization*, 41(2):299–318, 2008.

[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, 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.

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

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

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

[9] C. Audet and C. Tribes. Mesh-based Nelder-Mead algorithm for inequality constrained optimization. *Computational Optimization and Applications*, 71(2):331–352, 2018.

[10] X. Chen and N. Wang. Optimization of short-time gasoline blending scheduling problem with a DNA based hybrid genetic algorithm. *Chemical Engineering and Processing: Process Intensification*, 49(10):1076–1083, 2010.

[11] 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.

[12] A.R. Conn and S. Le Digabel. Use of quadratic models with mesh-adaptive direct search for constrained black box optimization. *Optimization Methods and Software*, 28(1):139–158, 2013.

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

[14] R. Y. Rubinstein D. P. Kroese, S.Porotsky. The Cross-Entropy method for continuous and multi-extremal optimization. *Metodol Comput Appl Probab*, 8:383–407, 2006.

[15] R. Fletcher and S. Leyffer. Nonlinear programming without a penalty function. *Mathematical Programming*, Series A, 91:239–269, 2002.

[16] A.A. Giunta. *Aircraft Multidisciplinary Optimization using Design of Experiments Theory and Response Surface Modeling Methods*. PhD thesis, Virginia Tech, Houston, Texas, 1997; available as Tech. Rep. MAD 97-05-01, May 1997, Department of Aerospace and Ocean Engineering, Virginia Tech, 215 Randolph Hall, Blacksburg, Virginia 24061.

[17] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[18] N.I.M. Gould, D. Orban, and Ph.L. Toint. CUTEr (and SifDec): A constrained and unconstrained testing environment, revisited. *ACM Transactions on Mathematical Software*, 29(4):373–394, 2003.

[19] R.B. Gramacy and S. Le Digabel. The mesh adaptive direct search algorithm with treed Gaussian process surrogates. *Pacific Journal of Optimization*, 11(3):419–447, 2015.

[20] A.-R. Hedar. Global Optimization Test Problems. http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO.htm. (last accessed on 2017-10-20).

[21] W. Hock and K. Schittkowski. *Test Examples for Nonlinear Programming Codes*, volume 187 of *Lecture Notes in Economics and Mathematical Systems*. Springer, Berlin, Germany, 1981.

[22] J. Jahn. *Introduction to the Theory of Nonlinear Optimization*. Springer, Berlin, 1994.

[23] M. Jamil and X.-S. Yang. A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 4(2):150–194, 2013.

[24] S. Kitayama, M. Arakawa, and K. Yamazaki. Sequential approximate optimization using radial basis function network for engineering optimization. *Optimization and Engineering*, 12(4):535–557, 2011.

[25] S. Kullback and R.Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.

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

[27] L. Lukšan and J. Vlček. Test problems for nonsmooth unconstrained and linearly constrained optimization. Technical Report V-798, ICS AS CR, 2000.

[28] K.I.M. McKinnon. Convergence of the Nelder-Mead simplex method to a nonstationary point. *SIAM Journal on Optimization*, 9(1):148–158, 1998.

[29] E. Mezura-Montes and C.A. Coello. Useful Infeasible Solutions in Engineering Optimization with Evolutionary Algorithms. In *Proceedings of the 4th Mexican International Conference on Advances in Artificial Intelligence*, MICAI'05, pages 652–662, Berlin, Heidelberg, 2005. Springer-Verlag.

[30] N. Mladenović, J. Petrović, V. Kovačević-Vujčić, and M. Čangalović. Solving spread spectrum radar polyphase code design problem by tabu search and variable neighbourhood search. *European Journal of Operational Research*, 151(2):389–399, 2003.

[31] J.J. Moré, B.S. Garbow, and Kenneth E. Hillstrom. Testing unconstrained optimization software. *ACM Transactions on Mathematical Software*, 7(1):17–41, 1981.

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

[33] F. Pigache, F. Messine, and B. Nogarede. Optimal Design of Piezoelectric Transformers: A Rational Approach Based on an Analytical Model and a Deterministic Global Optimization. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 54(7):1293–1302, 2007.

[34] P. Siarry R. Chelouah. A hybrid method combining continuous tabu search and Nelder-Mead simplex algorithm for the global optimization of multiminima functions. *European Journal of Operational Research*, 161:636–654, 2005.

[35] J.F. Rodríguez, J.E. Renaud, and L.T. Watson. Trust Region Augmented Lagrangian Methods for Sequential Response Surface Approximation and Optimization. *Journal of Mechanical Design*, 120(1):58–66, 1998.

[36] R. Y. Rubinstein. Optimization of computer simulation models with rare events . *European Journal of Operational Research*, 99:89–112, 1997.

[37] R. Y. Rubinstein and D. P. Kroese. *The Cross-Entropy Method: A unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning*. Springer: Berlin Heidelberg, New York, USA, 2004.

[38] M. Stein. Large sample properties of simulations using latin hypercube sampling. *Technometrics*, 29(2):143–151, 1987.

[39] J. Tao and N. Wang. DNA Double Helix Based Hybrid GA for the Gasoline Blending Recipe Optimization Problem. *Chemical Engineering and Technology*, 31(3):440–451, 2008.

[40] K. Wang and N. Wang. A novel RNA genetic algorithm for parameter estimation of dynamic systems. *Chemical Engineering Research and Design*, 88(11):1485–1493, 2010.