A Multi-Layer Line Search Method to Improve the Initialization of Optimization Algorithms

Benjamin Ivorra[1,†], Bijan Mohammadi[2] and Angel Manuel Ramos[1]

1. Departamento de Matemática Aplicada & Instituto de Matemática Interdisciplinar, Universidad Complutense de Madrid, Plaza de Ciencias, 3, 28040, Madrid, Spain
[†] Tel.: +34-91-394-4415, email: ivorra@mat.ucm.es
2. Institut de Mathématiques et de Modélisation de Montpellier, Université de Montpellier II, Place Eugène Bataillon, 34095, Montpellier, France

---

Abstract

We introduce a novel metaheuristic methodology to improve the initialization of a given deterministic or stochastic optimization algorithm. Our objective is to improve the performance of the considered algorithm, called core optimization algorithm, by reducing its number of cost function evaluations, by increasing its success rate and by boosting the precision of its results. In our approach, the core optimization is considered as a sub-optimization problem for a multi-layer line search method. The approach is presented and implemented for various particular core optimization algorithms: Steepest Descent, Heavy-Ball, Genetic Algorithm, Differential Evolution and Controlled Random Search. We validate our methodology by considering a set of low and high dimensional benchmark problems (i.e., problems of dimension between 2 and 1000). The results are compared to those obtained with the core optimization algorithms alone and with two additional global optimization methods (Direct Tabu Search and Continuous Greedy Randomized Adaptive Search). These latter also aim at improving the initial condition for the core algorithms. The numerical results seem to indicate that our approach improves the performances of the core optimization algorithms and allows to generate algorithms more efficient than the other optimization methods studied here. A Matlab optimization package called "Global Optimization Platform" (GOP), implementing the algorithms presented here, has been developed and can be downloaded at: `http://www.mat.ucm.es/momat/software.htm`

Keywords: Metaheuristics; Global optimization; Multi-layer line search

1. Introduction

We consider a general optimization problem of the form:

$$\min_{x\in\Omega} h_0(x) \tag{1}$$

where $h_0 : \Omega \to \mathbb{R}$ is the cost function, x is the optimization parameter and $\Omega \subset \mathbb{R}^N$, with $N \in \mathbb{N}$, is the admissible space.

When solving (1) by an iterative procedure the choice of the initial condition is essential. For instance, this is the case with the gradient methods such as the Steepest Descent algorithm (**SD**) (Luenberger and Ye, 2008), the Newton algorithm (Polyak, 2007) or with the Heavy-Ball algorithm (**HB**) (Attouch et al., 2000). When h_0 has several local minima these algorithms converge to one of those depending on their initialization. However, these algorithms can still find the global optimum if the initial condition belongs to the attraction basin of the infimum. Another example where the initialization is of prime importance is with Genetics Algorithms (**GA**) (Goldberg, 1989; Gonalves et al., 2002) where a lack of diversity in the individuals of the initial population can result to a premature convergence to a local minimum of h_0 (Rocha and Neves, 1999).

Thus, developing methods that intend to generate suitable initial conditions is interesting in order to improve the efficiency of existing optimization methods. For a given convergence accuracy, a better initialization may lead to a reduction in the number of functional evaluations, which is particularly important when working with expensive functional evaluations as in industrial design problems (Carrasco et al., 2012, 2015; Muyl et al., 2004; Gomez et al., 2011; Ivorra et al., 2006, 2009, 2013, 2014).

The idea of improving optimization algorithms by choosing a suitable initialization is widely present in the literature. For instance, the Direct Tabu Search algorithm (**DTS**) (Hedar and Fukushima, 2006; Lamghari and Dimitrakopoulos, 2012) and the Tunneling algorithm (Gomez and Levy, 1982; Levy and Gomez., 1985), are based on a modification of the functional by the addition of penalty terms to avoid the algorithm to revisit previously explored regions. Other techniques, like the Greedy Randomized Adaptive Search Procedure (**GRASP**) (Hirsch et al., 2010; Mart et al., 2015) or the

Universal Evolutionary Global Optimizer (Redondo et al., 2009)are based on the construction of a greedy solution combined with a local search step.

Another technique consists in coupling line search methods (Luenberger and Ye, 2008; Vieira and Lisboa, 2014) with another optimization algorithm. For instance, in Gardeux et al. (2011) the authors propose an optimization method, called **EM323**, well suited for the solution of high-dimensional continuous non-linear optimization problems. The algorithm **EM323** consists in combining the Enhanced Unidirectional Search method (Gardeux et al., 2009) with the 3-2-3 line search procedure (Glover, 2010). Another example can be found in Grosan and Abraham (2007), in the context of Multi-Objective optimization problems. The authors develop a method combining several line search algorithms: one for determining a first point in the Pareto front and another one for exploring the front.

In this work, we propose a novel metaheuristic technique also based on line search methods to dynamically improve the initialization of a given optimization method. The paper is organized as follows. In Section 2 we reformulate problem (1) as a sub-optimization problem where the initial condition of the considered optimization algorithm is the optimization parameter. This new problem is solved by considering an original multi-layer semi-deterministic line search algorithm. In Section 3, we focus on the implementation of our approach by considering two families of optimization algorithms: descent methods (in particular, **SD** and **HB**) and Evolutionary Algorithms (in particular, **GAs**, Controlled Random Search algorithms (**CRS**) (Price, 1983) and Differential Evolution algorithms (**DE**) (Price et al., 2005)). In Section 4, we validate our approach by considering various test cases in both low (Floudas and Pardalos, 1999) and high (Li et al., 2013) dimensions. The results are then compared with those given by the following optimization algorithms: **SD**, **HB**, **DTS**, Continuous **GRASP** (**CGR**), **CRS**, **DE** and **GA**.

2. General optimization method

We consider an optimization algorithm $A_0 : V \to \Omega$, called core optimization algorithm (**COA**), to solve problem (1). Here, V is the space where we can choose the initial condition for A_0 (various examples are given in Section 3, for simplicity we can consider $V = \Omega$). The other optimization parameters of A_0 (such as the stopping criterion, the number of iterations, etc.) are

fixed by the user. We omit them in the presentation in order to simplify the notations.

We assume the existence of $v \in V$ such that, for a given precision $\epsilon \geq 0$, $h_0(A_0(v)) - \min_{x\in\Omega} h_0(x) < \epsilon$. Thus, solving problem (1) with algorithm **COA** means:

$$\text{Find } v \in V \text{ such that } A_0(v) \in \text{argmin}_{x\in\Omega} h_0(x). \tag{2}$$

In order to solve problem (2), we propose to use a multi-layer semi-deterministic algorithm (called in the sequel the Multi-Layer Algorithm and denoted by **MLA**) based on line search methods (see, for instance, Luenberger and Ye (2008); Vieira and Lisboa (2014); Mohammadi and Saïac (2003)).

More precisely, we introduce $h_1 : V \to \mathbb{R}$ as:

$$h_1(v) = h_0(A_0(v)). \tag{3}$$

Thus, problem (2) can be rewritten as

$$\text{Find } v \in V \text{ such that } v \in \text{argmin}_{w\in V} h_1(w). \tag{4}$$

A geometrical representation of $h_1(.)$ in one dimension is shown in Figure 1 for a situation where the **COA** is the **SD** applied with 10000 iterations, $\Omega = V = [-10, 6]$ and $h_0(x) = \frac{1}{2}\cos(2x) + \sin(\frac{1}{3}x) + 1.57$. We see that $h_1(.)$ is discontinuous with plateaus. Indeed, the same solution is reached by the algorithm starting from any of the points of the same attraction basin. Furthermore, $h_1(.)$ is discontinuous where the functional reaches a local maximum. One way to minimize such kind of functionals in the one dimensional case is to consider line search optimization methods (such as the secant or the dichotomy methods, see Mohammadi and Saïac (2003)).

Thus, in order to solve problem (4), we introduce the algorithm $A_1 : V \to V$ which, for any $v_1 \in V$, returns $A_1(v_1) \in V$ after the following steps:

Step 1- Choose v_2 randomly in V.

Step 2- Find $v \in \text{argmin}_{w\in\mathcal{O}(v_1,v_2)} h_1(w)$, where $\mathcal{O}(v_1, v_2) = \{v_1 + t(v_2 - v_1), t \in \mathbb{R}\} \cap V$, using a line search method.

Step 3- Return v.

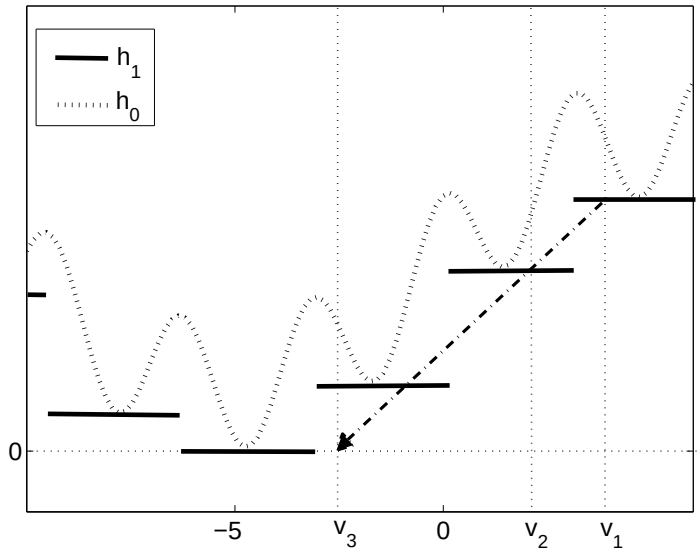

Figure 1: **(Dotted line)** Graphical representation of $h_0(x) = \frac{1}{2}\cos(2x) + \sin(\frac{1}{3}x) + 1.57$, for $x \in \Omega = V = [-10, 6]$. **(Continuous line)** Graphical representation of $h_1(.)$ when the **SD** is used as **COA** with 10000 iterations. **(Slash-dotted line)** Graphical representation of one execution of the algorithm $A_1(v_1)$, described in Section 3.1.1, when v_1 is given and $t_1 = 1$. v_2 is generated randomly in $[-10, 6]$ in the first Step of the algorithm. v_3 is built by the secant method performed in Step 2.2. v_3 (the best initial condition) is returned as the output in Step 3, since $h_1(v_3)$ is lower than $h_1(v_1)$ and $h_1(v_2)$.

The user may choose of the line search minimization algorithm in A_1.

This construction can be pursued looking for an optimal initialization for A_1. This can be done adding an external layer to algorithm A_1 and introducing $h_2 : V \to \mathbb{R}$ defined by

$$h_2(v) = h_1(A_1(v)) \tag{5}$$

and considering the following problem:

$$\text{Find } v \in V \text{ such that } v \in \operatorname{argmin}_{w \in V} h_2(w). \tag{6}$$

To solve problem (6), we use the two-layers algorithm $A_2 : V \to V$ that, for each $v_1 \in V$, returns $A_2(v_1) \in V$ given by

Step 1- Choose v_2 randomly in V.

Step 2- Find $v \in \operatorname{argmin}_{w \in \mathcal{O}(v_1, v_2)} h_2(w)$ using a line search method.

Step 3- Return v.

As previously, the user may choose the line search minimization algorithm in A_2. Due to the fact that the line search direction $\mathcal{O}(v_1, v_2)$ in A_1 is constructed randomly, the algorithm A_2 performs a multi-directional search of the solution of problem (2).

This construction can be pursued recursively defining

$$h_i(v) = h_{i-1}(A_{i-1}(v)), \quad \text{for} \quad i \in \mathbb{N}, \tag{7}$$

and considering the problem

$$\text{Find } v \in V \text{ such that } v \in \operatorname{argmin}_{w \in V} h_i(w). \tag{8}$$

Problem (8) is solved by using the i-layers algorithm $A_i : V \to V$ that, for each $v_1 \in V$, returns $A_i(v_1)$ given by

Step 1- Choose v_2 randomly in V.

Step 2- Find $v \in \operatorname{argmin}_{w \in \mathcal{O}(v_1, v_2)} h_i(w)$ using a line search method.

Step 3- Return v.

In practice, as specified in Section 4, we run A_i with a suitable stopping criterion and with $v_1 \in V$ arbitrary (or $v_1 \in V$ a good initial guess, if available).

The choice of the random technique used to generate v_2 in Step 1 of A_i is important and depends on h_0. For instance, if we know that h_0 has several local minima in Ω with a small attraction basins, it seems appropriate to generate v_2 in a small neighborhood of v_1.

Also, the line search minimization algorithm used in Step 2 of A_i depends on the properties of h_0, as discussed in Section 3.

3. MLA implementation

In this Section, we present various **MLA** implementations, considering different **COAs**, in the particular case where h_0 is a non negative function (or, equivalently, greater than a known real number). This specific situation often occurs in industrial problems (see, for instance, Ivorra et al. (2007, 2009, 2006, 2014)).

In particular, we consider two classes of implementations of the **MLA** associated with two kinds of **COAs**: gradient and evolutionary algorithms.

*3.1. **MLA** implementation with descent methods*

We consider a core algorithm A_0 that comes from the discretization of the following initial value problem (Attouch et al., 2000):

$$\begin{cases} \eta \dfrac{\mathrm{d}^2 x}{\mathrm{d}t^2}(t) + M(x(t),t)\dfrac{\mathrm{d}x}{\mathrm{d}t}(t) = -d(x(t)), \quad t \geq 0 \\ \quad x(0) = x_0, \quad \dfrac{\mathrm{d}x}{\mathrm{d}t}(0) = x_{t,0}, \end{cases} \tag{9}$$

where t is a fictitious time, $\eta \in \mathbb{R}$, $M : \Omega \times \mathbb{R} \to M_{N\times N}$ (with $M_{N\times N}$ denoting the set of matrix $N \times N$) and $d : \Omega \to \mathbb{R}^N$ is a function giving a descent direction. For instance:

- Assuming $h_0 \in C^1(\Omega, \mathbb{R})$, when $\eta = 0$, $d = \nabla h_0$ and $M(x,t) =$Id (the identity operator) for all $(x,t) \in \Omega \times \mathbb{R}$, we recover, considering a suitable discretization, the **SD** method (Luenberger and Ye, 2008).

- Assuming $h_0 \in C^2(\Omega, \mathbb{R})$, when $d = \nabla h_0$ and $M(x,t) =$Id for all $(x,t) \in \Omega \times \mathbb{R}$, we recover, considering an adequate discretization, the **HB** method (Attouch et al., 2000). This algorithm presents an exploratory character in the optimization process, in comparison to the case $\eta = 0$, allowing to escape from attraction basins.

According to the previous notation, we denote by $A_0(x_0, x_{t,0})$ the solution returned by the **COA** starting from $x_0 \in \Omega$ with an initial "velocity" $x_{t,0} \in \mathbb{R}^N$ and the set of remaining algorithm parameters fixed by the user.

We consider two possible choices for the initial condition of the **COA** (either x_0 or $x_{t,0}$) leading to two different formulations of problem (2). In the following, we propose the corresponding **MLAs** for solving each case.

3.1.1. Considering x_0 as the initial condition

We consider the following formulation of problem (2):

$$\text{Find } x_0 \in V = \Omega \text{ such that } A_0(x_0, x_{t,0}) \in \text{argmin}_{w\in\Omega} h_0(w), \tag{10}$$

where $x_{t,0} \in \mathbb{R}^N$ is fixed. We note that any $x_0 \in \text{argmin}_{x\in\Omega} h_0(x)$ is a solution of (10).

To solve problem (10), we consider a particular implementation of the algorithms A_i, $i \in \mathbb{N}$, introduced previously. For $i \in \mathbb{N}$, $A_i(v_1)$ is applied with a secant method in order to perform the line search as following:

Step 1- Choose $v_2 \in \Omega$ randomly.

Step 2- For l from 1 to $t_i \in \mathbb{N}$:

Step 2.1- If $h_i(v_l) = h_i(v_{l+1})$ go to **Step 3**

Step 2.2- Set $v_{l+2} = \text{proj}_\Omega(v_{l+1} - h_i(v_{l+1})\frac{v_{l+1}-v_l}{h_i(v_{l+1})-h_i(v_l)})$, where $\text{proj}_\Omega : \mathbb{R}^N \to \Omega$ is a projection algorithm over Ω defined by the user.

Step 3- Return the output: $\text{argmin}\{h_i(v_m), m = 1, ..., t_i\}$.

In the previous algorithm, the values i and t_i depend on the desired computational complexity. In Section 4, we check the efficiency of those algorithms for various set of values.

A geometrical representation of one execution of algorithm A_1 in one dimension is shown in Figure 1.

From a theoretical point of view, as the secant method is adapted to find the zeros of a function (Mohammadi and Saïac, 2003), those algorithms perform better if the minimum value of h_0, denoted by $h_0^* \in \mathbb{R}$, is known (for instance, in some inverse problems, see Ivorra et al. (2014)). Indeed, we can assume that $h_0^* = 0$ (we minimize $h_0 - h_0^*$ instead of h_0) and thus A_i intends to find the zero of h_i (see Ivorra (2006) for more details). However, in practice (see experiments presented in Section 4), if the only information

available is that h_0 is greater than a known real number $h_l \in \mathbb{R}$, we consider $h_0 - h_l$ instead of h_0 and this algorithm still gives good numerical results. This efficiency can be in part explained by the structure of the secant iteration that gives a quick information about the behavior of h_0. If there is a significant evolution of the cost function value between v_{l+1} and v_l, the secant iteration generates v_{l+2} close to v_{l+1} (because the slope of the straight line passing trough the points (v_{l+1},$h_0(v_{l+1})$) and (v_{l+2},$h_0(v_{l+2})$) is steep) performing a refined search around v_{l+1}. Otherwise, the secant method generates v_{l+2} far from v_{l+1} allowing to explore a new region.

Remark 1. *Although this case is not considered in this paper, if no information about a lower bound value of h_0 is available we could consider other **MLA** implementations. For instance, we can replace the secant method used in **Step 2.2** by the **SD** iteration starting from v_{l+1} and using $-\frac{v_{l+1}-v_l}{h_i(v_{l+1})-h_i(v_l)}$ as the descent direction. This new step reads:*

> ***Step 2.2-*** *Set $v_{l+2} = proj_\Omega(v_{l+1} - \rho \frac{v_{l+1}-v_l}{h_i(v_{l+1})-h_i(v_l)})$, where $proj_\Omega : \mathbb{R}^N \to \Omega$, is a projection algorithm over Ω defined by the user and $\rho \in \mathbb{R}$ is obtained by solving numerically $min_{\rho \in \mathbb{R}^+} h_0(proj_\Omega(\ v_{l+1} - \rho \frac{v_{l+1}-v_l}{h_i(v_{l+1})-h_i(v_l)}))$ by using a dichotomy method.*

3.1.2. Considering $x_{t,0}$ as the initial condition

We consider the following formulation of problem (2):

$$\text{Find } x_{t,0} \in V = \mathbb{R}^N \text{ such that } A_0(x_0, x_{t,0}) \in \text{argmin}_{w \in \Omega} h_0(w), \tag{11}$$

where $x_0 \in \Omega$ is fixed. In this case, under convenient hypotheses, we can prove the existence of $x_{t,0} \in \mathbb{R}^N$ such that the solution of problem (11) can be approximated numerically, as stated in the following theorem:

Theorem 1. *Let $h_0 \in C^2(\mathbb{R}^N, \mathbb{R})$ having a minimum at $x_m \in \mathbb{R}^N$. We assume that its gradient, denoted by $\nabla h_0(.)$, is Lipschitz continuous. Thus, for every $(x_0, \epsilon) \in \mathbb{R}^N \times \mathbb{R}^+$, there exists $(\sigma, \tau_b) \in \mathbb{R}^N \times \mathbb{R}^+$ such that the solution of the following dynamical system*

$$\begin{cases} \eta \dfrac{\mathrm{d}^2 x}{\mathrm{d}t^2}(t) + \dfrac{\mathrm{d}x}{\mathrm{d}t}(t) = -\nabla h_0(x(t)), \quad t \geq 0, \\ x(0) = x_0, \quad \dfrac{\mathrm{d}x}{\mathrm{d}t}(0) = \sigma, \end{cases} \tag{12}$$

with $\eta \in \mathbb{R}$, passes at time τ_b into the ball of center x_m and radius ϵ, denoted by $B_\epsilon(x_m)$.

Proof : We assume $x_0 \neq x_m$ (as $x_0 = x_m$ is a trivial case). Let $\delta \geq 0$, we consider the initial value problem

$$\begin{cases} \eta \dfrac{\mathrm{d}^2 y_\delta}{\mathrm{d}t^2}(t) + \delta \dfrac{\mathrm{d}y_\delta}{\mathrm{d}t}(t)(t) = -\delta^2 \nabla h_0(y_\delta(t)), \quad t \geq 0, \\ y_\delta(0) = x_0, \quad \dfrac{\mathrm{d}y_\delta}{\mathrm{d}t}(t)(0) = \varrho(x_m - x_0), \end{cases} \tag{13}$$

with $\varrho \in \mathbb{R}^+ \backslash \{0\}$. Let us show that y_δ passes at some time into the ball $B_\epsilon(x_m)$:

- If $\delta = 0$, we obtain the following system

$$\begin{cases} \eta \dfrac{\mathrm{d}^2 y_0}{\mathrm{d}t^2}(t) = 0, \quad t \geq 0, \\ y_0(0) = x_0, \quad \dfrac{\mathrm{d}y_0}{\mathrm{d}t}(0) = \varrho(x_m - x_0). \end{cases} \tag{14}$$

  System (14) describes a straight line of origin x_0 and passing at some time $\tau_\varrho \in \mathbb{R}^+$ by the point x_m (i.e., $y_0(\tau_\varrho) = x_m$).

- If $\delta \neq 0$, System (13) can be rewritten as

$$\frac{\mathrm{d}w}{\mathrm{d}t}(t) = \begin{pmatrix} \dfrac{\mathrm{d}y_\delta}{\mathrm{d}t}(t) \\ -\delta \dfrac{\mathrm{d}y_\delta}{\mathrm{d}t}(t) - \delta^2 \nabla h_0(y_\delta(t)) \end{pmatrix} = f(t, w(t), \delta), \tag{15}$$

  with $w(t) = \left(y_\delta(t), \eta \dfrac{\mathrm{d}y_\delta}{\mathrm{d}t}(t) \right)$ and f continuous in t and in δ and Lipschitz continuous in $w(t)$ (Attouch et al., 2000). Then, applying the Cauchy-Lipschitz theorem (see, for instance, Verhulst (1996)) and the Continuity Theorem 3.4 found in Hale (2009), we obtain that

$$\lim_{\delta \to 0} |y_\delta(\tau_\varrho) - y_0(\tau_\varrho)| = 0.$$

Thus, for every $\epsilon \in \mathbb{R}^+ \backslash \{0\}$, there exists $\delta_\epsilon \in \mathbb{R}^+$ such that for every $\delta \leq \delta_\epsilon$

$$|y_\delta(\tau_\varrho) - x_m| < \epsilon. \tag{16}$$

Let $\epsilon \in \mathbb{R}^+ \backslash \{0\}$. We consider the change of variable given by $s = \delta_\epsilon t$ and $x(s) = y_{\delta_\epsilon}(\frac{s}{\delta_\epsilon})$. Then, System (13) becomes

$$\begin{cases} \eta \dfrac{\mathrm{d}^2 x}{\mathrm{d}s^2}(s) + \dfrac{\mathrm{d}x}{\mathrm{d}s}(s) = -\nabla h_0(x(s)), \quad s \geq 0, \\ x(0) = x_0, \quad \dfrac{\mathrm{d}x}{\mathrm{d}s}(0) = \frac{\varrho}{\delta_\epsilon}(x_m - x_0). \end{cases} \tag{17}$$

Let $\tau_b = \delta_\epsilon \tau_\varrho$. Under this assumption, $x(\tau_b) = y_{\delta_\epsilon}(\tau_\varrho)$. Thus, due to (16), $|x(\tau_b) - x_m| < \epsilon$. We have found $\sigma = \frac{\varrho}{\delta_\epsilon}(x_m - x_0) \in \mathbb{R}^N$ and $\tau_b \in \mathbb{R}^+$ such that the solution of System (12) passes at time τ_b into the ball $B_\epsilon(x_m)$.

□

In order to determine a solution of problem (11), we can consider, for example, the same implementation of algorithms A_i, with $i \in \mathbb{N}$, as the one introduced in Section 3.1.1.

*3.2. **MLA** implementation with Evolutionary Algorithms*

3.2.1. General framework

Evolutionary Algorithms (**EA**) are population-based metaheuristic optimization algorithms which try to solve problems similar to (2) (Ashlock, 2010). From a general point of view, they start from a finite set of points in the search space Ω, called initial population, and intend to improve the value of the considered cost function by applying processes based on an analogy with the Darwinian evolution of species. For instance, we can cite some classical **EAs** (considered in the experiments presented in Section 4): the Genetic (Goldberg, 1989), the Controlled Random Search (Price, 1983) and the Differential Evolution (Price et al., 2005) algorithms. The **EAs** have many advantages, as for example: they generally do not require sensitivity computation, they can solve complex optimization problems (e.g., with high dimensional search space or function with various with local minima), and they are intrinsically parallel. However, they also have some important drawbacks as: slower convergence and lower accuracy than other methods (such as gradient algorithms (Ashlock, 2010)).

We denote by $X^0 = \{x_j^0 \in \Omega, j = 1, ..., N_p\}$, with $N_p \in \mathbb{N}$, the set corresponding to the initial population of the considered **EA**. All other parameters of the **EA** are fixed by the user. In this case, problem (2) can be rewritten as:

$$\text{Find } X^0 \in V = \Omega^{N_p} \text{ such that } A_0(X^0) \in \operatorname{argmin}_{w \in \Omega} h_0(w). \tag{18}$$

In the following, we propose a version of the **MLA** used to solve problem (18). This **MLA** is illustrated by considering a particular Genetic Algorithm (**GA**), validated previously in Ivorra (2006); Ivorra et al. (2006, 2014), as the **COA**. We note that the proposed algorithm can be easily extended to any other **EA**, as illustrated in Section 4 with the use of a Controlled Random Search algorithm (**CRS**) and a Differential Evolution algorithm (**DE**) as **COAs**.

As it is out of the scope in this paper, we do not describe the considered **CRS** and **DE** implementations, which are deeply detailed in Hendrix et al. (2001) and Storn and Price (1997), respectively, but we give in Section 3.2.2 the implementation that we use for the **GA**.

3.2.2. ***MLA*** *implementation with GA*

We first describe the **GA** considered here:

- **Step 1- Inputs:** User must define the parameters $N_p \in \mathbb{N}$, $N_g \in \mathbb{N}$, $p_m \in [0,1]$, $p_c \in [0,1]$, $\epsilon \in \mathbb{R}$ and the initial population $X^0 \in \Omega^{N_p}$. The meaning of those parameters is clarified later in the following steps.

- **Step 2- Generating new populations:** Starting from X^0, we recursively create N_g new populations by applying four stochastic processes: 'selection', 'crossover', 'mutation' and 'elitism', which are described in Steps 3.1, 3.2, 3.3 and 3.4, respectively.

  More precisely, let $X^i = \{x_j^i \in \Omega, j = 1, ..., N_p\}$, with $i = 1, ..., N_g - 1$, denotes the population at iteration i. Then, using the (N_p, N)-real valued matrix

$$X^i = \begin{bmatrix} x_1^i \\ \vdots \\ x_{N_p}^i \end{bmatrix} = \begin{bmatrix} x_1^i(1) & \dots & x_1^i(N) \\ \vdots & \ddots & \vdots \\ x_{N_p}^i(1) & \dots & x_{N_p}^i(N) \end{bmatrix},$$

  with $x_j^i = (x_j^i(1), ..., x_j^i(N)) \in \Omega$, X^{i+1} is obtained by considering:

$$X^{i+1} = (I_N - \mathcal{E}^i)(\mathcal{C}^i \mathcal{S}^i X^i + \mathcal{M}^i) + \mathcal{E}^i X^i,$$

  where matrices $\mathcal{S}^i$, $\mathcal{C}^i$, $\mathcal{M}^i$, $\mathcal{E}^i$ and I_N are described below.

  - **Step 2.1- Selection:** We randomly select N_p individuals from X^i with eventual repetitions. Each individual $x_j^i \in X^i$, with $j =$

$1, ..., N_p$, has a probability to be selected in this process which is given by $J^{-1}(x_j^i)/\sum_{k=1}^{N_p} J^{-1}(x_k^i)$. This step can be summarized as

$$X^{i+1,1} = \mathcal{S}^i X^i,$$

where $\mathcal{S}^i$ is a (N_p, N_p)-matrix with $\mathcal{S}^i_{j,k} = 1$ if the k-th individual of X^i is the j-th selected individual and $\mathcal{S}^i_{j,k} = 0$ otherwise.

– **Step 2.2- Crossover:** For each pair of consecutive individuals (rows) $2j-1$ and $2j$ in $X^{i+1,1}$, with $1 \leq j \leq \text{floor}(N_p/2)$ (where $\text{floor}(X)$ is the nearest integer lower or equal than X), we determine, with a probability p_c, if those rows exchange data or if they are directly copied into an intermediate population denoted by $X^{i+1,2}$. Mathematically, this step can be written as

$$X^{i+1,2} = \mathcal{C}^i X^{i+1,1},$$

where $\mathcal{C}^i$ is a real-valued (N_p, N_p)-matrix. The coefficients of the $(2j-1)$-th and $2j$-th rows of $\mathcal{C}^i$, with $1 \leq j \leq \text{floor}(N_p/2)$, are given by

$$\mathcal{C}^i_{2j-1,2j-1} = \lambda_1, \;\; \mathcal{C}^i_{2j-1,2j} = 1-\lambda_1, \;\; \mathcal{C}^i_{2j,2j} = \lambda_2, \;\; \mathcal{C}^i_{2j,2j-1} = 1-\lambda_2$$

where $\lambda_1 = \lambda_2 = 1$, with a probability $1 - p_c$, or λ_1 and λ_2 are randomly chosen in $]0, 1[$, considering a uniform distribution, in other case. Other coefficients of $\mathcal{C}^i$ are set to 0. If N_p is odd then we also set $\mathcal{C}^i_{N_p,N_p} = 1$ and then the N_p-th row of $X^{i+1,1}$ is directly copied in $X^{i+1,2}$.

– **Step 2.3- Mutation:** We decide, with a probability p_m, if each row of $X^{i+1,2}$ is randomly perturbed or not. This step is defined by

$$X^{i+1,3} = X^{i+1,2} + \mathcal{M}^i,$$

where $\mathcal{M}^i$ is a real-valued (N_p, N)-matrix where the j-th row, $j = 1, ..., N_p$, is equal to 0, with a probability 1-p_m, or a random vector $m_j \in \mathbb{R}^N$, generated considering a uniform distribution in the subset of $\mathbb{R}^N$ such that $x_j^{i+1,2} + m_j \in \Omega$, otherwise.

– **Step 2.4- Elitism:** Let x_b^i, where $b \in 1, ..., N_p$, be the individual in X^i with the lowest value of h_0 (or, if there exists various, one of

those individuals selected randomly with a uniform distribution). If x_b^i has a lower h_0 value than all the individuals in $X^{i+1,3}$, it is directly copied at the b-th row of X^{i+1}. This step can be formalized as

$$X^{i+1} = (I_N - \mathcal{E}^i)(X^{i+1,3}) + \mathcal{E}^i X^i,$$

where I_N is the identity matrix of size N and $\mathcal{E}^i$ is a real-valued (N_p, N_p)-matrix such that $\mathcal{E}^i(b,b) = 1$ if x_b^i has a lower h_0 value than all the individuals in $X^{i+1,3}$ and 0 otherwise, $\mathcal{E}^i = 0$ elsewhere.

- **Step 3- Output:** After N_g iterations or if the stopping criterion associated to $\epsilon \in \mathbb{R}$ is satisfied for at least one individual in X^{i+1}, the **GA** stops and returns an output solution denoted by

$$A_0(X^0) = \text{argmin}\Big\{h_0(x_j^i)/ \quad x_j^i \text{ is the } j\text{-th row of } X^i, \\ i = 1, ..., N_g, j = 1, ..., N_p\Big\}.$$

Remark 2. *As a fine convergence is generally difficult to achieve with **GAs**, it is recommended when it is possible, to complete the **GA** iterations with a descent method (Muyl et al., 2004).*

The solution of problem (18) may be determined, for instance, by using the algorithms A_i (with $i = 1, 2, ...$) presented in Section 3.1.1. However, previous studies (see Ivorra (2006); Ivorra et al. (2006)) show that the following variation of A_i (with $i = 0, 1, 2, 3, ...$), denoted by B_i, is better suited to the **GA** case. Let $X_1^0 = \{x_{1,j}^0 \in \Omega, j = 1, ..., N_p\}$ and $B_0(X^0)$=$A_0(X^0)$, then, for $i > 0$, $B_i(X_1^0)$ reads:

Step 1- For l from 1 to $t_i \in \mathbb{N}$:

Step 1.1- Set $o_l = B_{i-1}(X_l^0)$.

Step 1.2- We construct $X_{l+1}^0 = \{x_{l+1,j}^0 \in \Omega, j = 1, ..., N_p\}$ as following: $\forall j \in \{1, ..., N_p\}$, if $h_0(o_l) = h_0(x_{l,j}^0)$ set $x_{l+1,j}^0 = x_{l,j}^0$ else set $x_{l+1,j}^0 = \text{proj}_\Omega(x_{l,j}^0 - h_0(o_l)\frac{o_l - x_{l,j}^0}{h_0(o_l) - h_0(x_{l,j}^0)})$ where $\text{proj}_\Omega : \mathbb{R}^N \to \Omega$ is a projection operator over Ω defined by the user.

Step 2- Return the output: $\text{argmin}\{h_i(o_m), m = 1, ..., t_i\}$

As previously, the values of i and t_i depend on the desired computational complexity.

This version of the algorithm intends to improve, individual by individual, the initial population of B_{i-1}. More precisely, for each individual in the initial population:

- If there is a significant evolution of the cost function between this individual and the best element found by B_{i-1}, the secant method used in **Step 1.2** generates, in the optimized initial population X_{l+1}^0, a new individual close to o_l that performs a refined search near the actual solution.

- Otherwise, the secant method creates a new individual in X_{l+1}^0 far from o_l, to expand the exploration of the admissible space.

Numerical experiments in Section 4 seem to indicate that considering algorithms B_i, with $i > 0$, reduces the computational complexity of **GAs**. In particular, this allows to reduce both parameters N_p and N_g in **GAs**. We will also analyze the application of algorithms B_i with **CRS** and **DE** as the **COA**.

4. Numerical experiments

In order to check the efficiency of the **MLAs** presented in Section 3, we consider two sets of benchmark problems. The first set, described in Section 4.1, consists in low dimensional (i.e., dimension lower than 10) problems (Floudas and Pardalos, 1999). The second set, detailed in Section 4.2, focuses on high dimensional problems (with dimension from 50 to 1000) (Li et al., 2013). Our objective is to see how **MLAs** improve the efficiency of several **COAs** and to compare them with other metaheuristic methods.

4.1. Low dimensional benchmark problems

4.1.1. Considered benchmark problems

We consider the following set of box-constrained benchmark optimization problems: Branin (denoted by **Bra**), Eason (**Eas**), Goldstein-Price (**G-P**), Shubert (**Shu**), Hartmann with 3 (**Hm3**) and 6 (**Hm6**) variables, Rosenbrock with 2 (**Rb2**), 5 (**Rb5**) and 10 (**Rb10**) variables, Shekel with 4 variables and m =5 (**Sk5**), 7 (**Sk7**)and 10 (**Sk10**), and Zakharov with 5 (**Za5**) and 10 (**Za10**) variables. A complete description of those problems with

the considered values of the box restrictions can be found in Floudas and Pardalos (1999). These problems well feature the difficulties of optimization problems and are frequently used to validate optimization algorithms (Hedar and Fukushima, 2006; Hirsch et al., 2010).

4.1.2. Considered algorithms and parameters

To solve numerically the problems introduced in Section 4.1.1, we consider the following **MLAs**:

- When the **SD** is the **COA**, we use the **MLA** implementation presented in Section 3.1.1. We apply this method with different number of layers i: $i = 1$ (the algorithm is then denoted by **SMA1**, as **SD** Multi-Layer Algorithm 1-Layer), $i = 2$ (**SMA2**) and $i = 3$ (**SMA3**). We set $t_0 = 10$ and $t_1 = 1000$ for **SMA1**; $t_0 = t_1 = 10$ and $t_2 = 1000$ for **SMA2**; and $t_0 = t_1 = t_2 = 10$ and $t_3 = 1000$ for **SMA3**. The descent step size ρ is determined using 10 iterations of a dichotomy method starting from $\rho_0 = 1$ (Mohammadi and Saïac, 2003). Those parameters have been determined in Debiane et al. (2006); Ivorra (2006); Ivorra et al. (2006).

- When the **HB** is used as the **COA**, we use the two-layers algorithm A_2, described in Sections 3.1.1 and 3.1.2, with $\eta = 0.1$, $t_0 = t_1 = 10$ and $t_2 = 1000$. The velocity $x_{t,0}$ is the initial condition to be optimized. x_0 is chosen randomly in Ω. This algorithm is denoted by **HMA** (**HB** Multi-Layer Algorithm). Those specific parameters have been proposed in Ivorra (2006).

- With **GA** as the **COA**, we use the algorithm B_2 introduced in Section 3.2 with $t_1 = 10$ and $t_2 = 1000$. This algorithm is denoted by **GMA** (Genetic Multi-Layer Algorithm). The **GA** parameters are set to: $N_p = 10$, $N_g = 10$, $p_c = 0.55$, $p_m = 0.5$. Those parameters have been considered in Gomez et al. (2011); Ivorra (2006); Ivorra et al. (2013).

- When the **DE** described in Storn and Price (1997) is considered as the **COA**, we use the algorithm B_2 introduced in Section 3.2 with $t_1 = 10$ and $t_2 = 1000$. This algorithm is denoted by **DMA** (**DE** Multi-Layer Algorithm). The **DE** parameters used here are the following: the population size is set to 10, the crossover operator is rand/1/exp, the crossover constant is 0.95, the mutation parameter is 0.9 and the

maximum number of iterations is 100. Those parameters have been determined experimentally. The Matlab implementation of the **DE** can be found here:

`http://www1.icsi.berkeley.edu/~storn/code.html`

- When the **CRS** detailed in Hendrix et al. (2001) is considered as the **COA**, we use the algorithm B_2 introduced in Section 3.2 with $t_1 = 10$ and $t_2 = 1000$. This algorithm is denoted by **CMA** (**CRS** Multi-Layer Algorithm). The **CRS** parameters used here are the following: the population size is set to 60, the number of trial points is set to the size of the problem, the maximum number of iterations is set to 300 and the rate of success test is 0.55. These parameters have been determined experimentally.

Furthermore, at the end of the **GMA**, **CMA** and **DMA**, 10 iterations of the **SD** used in **SMA1** are performed to improve the accuracy of the results.

Also, the performances of **MLAs** are compared with those of other metaheuristic methods available in the literature. We consider algorithms **CGR** and **DTS** whose implementation, parameters and results are presented in Hedar and Fukushima (2006) and Hirsch et al. (2010), respectively. Furthermore, to see if applying the **MLAs** allows to improve the performances of the different **COAs**, we also solve the considered benchmark problems by considering the **COAs** alone with the following parameters:

- The **SD** and the **HB** are run starting from a random point in Ω, with the same parameters as in **SMA1** and **HMA** except $t_0 = 3000$.

- The **GA** is run with the same stochastic processes as **GMA** but with $N_g = 1000$, $N_p = 180$, $p_c = 0.45$ and $p_m = 0.15$. The stopping criterion considered here is explained below. These parameters have been identified in previous works (Ivorra et al., 2006, 2014).

- The **DE** is run with the crossover operator set to rand/1/exp, the crossover constant set to 0.9, the mutation parameters set to 0.5, the population size set to 5 times the dimension of the benchmark problem and the maximum number of iterations set to 5000. These parameters were suggested in the literature for low dimensional problems (Storn and Price, 1997).

- The **CRS** is applied with a population size of 200, the number of trial points set to the size of the problem, the maximum number of iterations set to 3000 and the rate of success test set to 0.55. These choices have been suggested in Hendrix et al. (2001).

Again, at the end of the **GA**, **CRS** and **DR**, 10 iterations of the **SD** used in **SMA1** are carried out.

Following Hirsch et al. (2010), as the global minimum denoted by h_0^*, of the different benchmark problems is known, the stopping criterion is defined as

$$|h_0^* - \tilde{h_0}| \leq \epsilon_1 |h_0^*| + \epsilon_2, \tag{19}$$

where $\tilde{h_0}$ is the current solution of the algorithm, $\epsilon_1 = 10^{-4}$ and $\epsilon_2 = 10^{-6}$. The **DE**, **DMA**, **CRS** , **CMA**, **GA** and **GMA** are run with $\epsilon_1 = 10^{-2}$ and $\epsilon_2 = 10^{-3}$ and the **SD** performed at the end of those algorithms is executed with $\epsilon_1 = 10^{-4}$ and $\epsilon_2 = 10^{-6}$.

In addition to this stopping criterion we have limited the maximum number of functional evaluations to 50000 for each run (which can be considered as a high number, see Gomez et al. (2011); Ivorra et al. (2013, 2014); Muyl et al. (2004)). If at the end of the algorithm, (19) is not satisfied, we consider that the algorithm has failed to solve numerically the considered problem.

As specified in Section 3, we recall that the **MLAs** presented previously are adapted to non-negative functions (or with a known lower bound value). So, the benchmark functions h_0 with negative values have been modified adding a real number C_{h_0} large enough to obtain a non-negative function. Here, in order to obtain a stopping criterion (19) comparable to the one used in Hedar and Fukushima (2006); Hirsch et al. (2010), we have considered $C_{h_0} = 2|h_0^*|$.

Due to the stochastic aspect of the algorithms, each benchmark problem has been solved 100 times with each of the optimization algorithm. We define the success rate of an optimization algorithm by the percentage of runs satisfying the stopping criterion (19).

In order to check the improvement of the **MLA** with respect to the **COA** alone, we have also computed the following improvement threshold (in %), denoted by Imp and given by:

$$\text{Imp}(\textbf{MLA}) = 100 \times \frac{\text{Tev}(\textbf{COA})\text{-Tev}(\textbf{MLA})}{\text{Tev}(\textbf{COA})}, \tag{20}$$

where Tev(**A**) is the total number of evaluations required by the algorithm **A** to solve all the benchmark problems (including runs that have failed to satisfy the stopping criterion (19)) and **COA** is the core optimization algorithm of **MLA**. Imp represents the computational effort reduction obtained when using the **MLA** instead of the **COA**.

All experiments have been performed on a Pentium I7 Quad-Core with 3.6 Ghz and 32 Gb of RAM and the algorithms have been implemented in a Matlab 2014 script.

Remark 3. *The **MLAs** performs better if the minimum value $h_0^* = 0$ (see Section 3.1.1). So, when h_0^* is known (e.g., as said previously, in some inverse problems (Ivorra et al., 2014)), we can minimize $h_0 - h_0^*$ instead of h_0 (Ivorra, 2006; Ivorra et al., 2014). However, in industrial applications, this information is generally not available (Carrasco et al., 2015; Gomez et al., 2011; Debiane et al., 2006; Ivorra et al., 2007, 2009; Muyl et al., 2004). Thus, we have decided not to use it. Furthermore, we deduce from the results presented in Section 4.1.3 that this hypothesis is not mandatory as our methodology is efficient also in cases when h_0^* is unknown.*

Remark 4. *The multi-layers linear search method was also applied alone to solve the considered benchmark problems. To do so, we considered $A_0(x_0) = x_0$, and the 3-layers structure A_3 with $t_0 = t_1 = t_2 = 10$ and $t_3 = 10000$. However, this algorithm exhibited a success rate of 0% in all benchmark problems. Thus, the precision of this algorithm seems to be extremely low and the obtained results are not reported in the next Section 4.1.3. This indicates that the use of a **COA** is necessary to generate an efficient **MLA**.*

4.1.3. Results

The average number of functional evaluations (considering only successful runs satisfying the stopping criterion (19)) needed by the optimization algorithms to solve the benchmark problems are shown in Table 1. The algorithms success rates are reported in Table 2. The value of Imp for each **MLA** is shown in Table 3.

Table 2 shows that **SD** and **HD** have the worst success rates for various benchmark functions. However, we also observe on this table that considering these algorithms as **COAs** used together with **MLAs** always improve their success rates. The mean number of evaluations required to satisfy the stopping criterion, presented in Table 1, is generally lower for a **MLA** than

Func.	DTS	CGR	SD	HB	GA	CRS	DE
Bra	212	10090	251	307	1304	2953	2347
Eas	223	5093	-	3600	40125	2877	3851
G-P	230	53	295	660	465	2429	1937
Shu	274	18608	120	1255	7748	9947	3049
Hm3	438	1719	466	956	1119	1493	447
Hm6	1787	29894	217	460	4418	2907	8456
Rb2	254	23544	2275	1919	3918	6177	7952
Rb5	1684	182520	3465	18287	43604	7927	41939
Rb10	9037	725281	5096	25361	44557	43822	44156
Sk5	819	9274	229	337	37328	5702	40032
Sk7	812	11766	208	318	36046	3618	3479
Sk10	6828	17612	-	401	40217	3540	2386
Za5	1003	12467	268	600	24988	5384	40026
Za10	4032	2297937	540	1770	40489	9004	40031

Func.	SMA1	SMA2	SMA3	HMA	GMA	CMA	DMA
Bra	215	140	130	489	252	2936	1821
Eas	2996	7085	7427	16504	3488	9801	1829
G-P	463	312	425	1292	439	3846	1054
Shu	416	4274	4125	12687	1270	9669	1557
Hm3	564	347	415	1052	425	959	3613
Hm6	766	369	591	5681	1054	5466	2972
Rb2	1542	945	1210	2568	1675	4319	1353
Rb5	4420	1777	1866	11513	43972	17351	10307
Rb10	5612	2657	2545	21151	44828	42691	42930
Sk5	3220	1123	1944	9540	6991	7453	6902
Sk7	2648	936	1483	14780	4619	8343	4961
Sk10	3336	3159	2416	6299	1637	8869	3653
Za5	291	152	145	648	2674	7087	4106
Za10	593	309	292	2110	20719	9860	17780

Table 1: Average number (considering only the runs satisfying the stopping criterion (19)) of functional evaluations needed by the optimization algorithms described in Section 4.1.2 to solve the benchmark problems (**Func.**) presented in Section 4.1.1.

Func.	DTS	CGR	SD	HB	GA	CRS	DE
Bra	100	100	100	100	100	100	100
Eas	82	100	0	13	100	100	100
G-P	100	100	53	100	100	100	100
Shu	92	100	25	20	100	100	100
Hm3	100	100	51	58	100	100	100
Hm6	83	100	48	73	100	90	51
Rb2	100	100	80	100	100	100	100
Rb5	85	100	74	87	96	87	97
Rb10	85	100	71	15	95	68	93
Sk5	57	100	16	53	97	17	74
Sk7	65	100	7	48	96	23	88
Sk10	52	100	0	26	96	10	93
Za5	100	100	100	100	100	100	100
Za10	100	100	100	100	100	100	100

Func.	SMA1	SMA2	SMA3	HMA	GMA	CMA	DMA
Bra	100	100	100	100	100	100	100
Eas	12	100	100	48	100	100	100
G-P	100	100	100	100	100	100	100
Shu	47	100	100	91	100	100	100
Hm3	100	100	100	100	100	100	100
Hm6	100	100	100	95	100	100	49
Rb2	95	100	100	100	100	100	100
Rb5	100	100	100	97	92	100	95
Rb10	100	100	100	64	81	100	92
Sk5	66	100	100	66	96	100	97
Sk7	37	100	100	57	98	100	98
Sk10	30	100	100	55	97	94	95
Za5	100	100	100	100	100	100	100
Za10	100	100	100	100	100	100	100

Table 2: Success rate (%) of the optimization algorithms described in Section 4.1.2 when solving the benchmark problems (**Func.**) presented in Section 4.1.1.

MLA	**SMA1**	**SMA2**	**SMA3**	**HMA**	**GMA**	**CMA**	**DMA**
Imp	50	85	93	33	58	41	50

Table 3: Value of Imp (%), defined by Equation (20), obtained when solving the benchmark problems presented in Section 4.2.1 with the following **MLAs**: **SMA1**, **SMA2**, **SMA3**, **HMA**, **GMA**, **CMA** and **DMA**.

for its **COA** alone. In cases where the number of evaluations is greater for **MLA** than for its **COA** alone, we have to take into account that we only consider here the runs for which the algorithms satisfy the stopping criterion. This increases the computational effort of the **MLA** as its success rate is improved in comparison to its **COA** alone, and thus, the additional successful runs require a high number of functional evaluations. The advantage of using **MLA** can be also observed through the Imp values presented in Table 3. We can see that the improvement with **MLA** regarding the total number of evaluations required to solve all benchmark runs is increased from 33% up to 95%. Those results indicate that the **MLA** method improves the efficiency of the **COA** for all these low dimensional problems.

Also, when analyzing the global efficiency of all proposed algorithms, the results indicate that **SMA2** and **SMA3** give the best performances. Indeed, both algorithms have a success percentage of 100 (as **CGR**) and are more efficient with respect to the number of functional evaluations, compared to all other methods. These results can be explained by the secant method chosen to perform the line search: as specified in Section 3.1.1, this method is well adapted for quick search and allows to have a rapid idea of the functional behavior (for instance, the monotonicity of the function in a given direction). Once this is detected the **SD** performs a local optimization. The Shubert (**Shu**) and Easom (**Eas**) benchmark cases are the two functions which require the largest numbers of evaluations for SMA2 and SMA3. This is because both functions have several local minima with small attraction basins and none is coercive (a function $f(x)$ is coercive if $\|x\| \to +\infty$ implies $f(x) \to +\infty$). This makes the global minimum attraction basin difficult to find using the secant method. As a conclusion, at this point **SMA2** and **SMA3** seem to represent the best choices.

When focusing on each **MLA**, **SMA1** presents the lowest success rates of all **MLAs**. This is due to fact that in **SMA1** the search for a suitable initial condition is only a 1D search while **SMA2** and **SMA3** introduce multi-

directional explorations. With **HMA** the number of functional evaluations is higher and the success percentage is lower than with **SMA2** and **SMA3**. This is because **HB** permits for better explorations of the admissible space due to its inertial features but too much inertia prevent from convergence. Finally **GMA**, **CMA** and **DMA** present interesting characteristics. Their success rates are generally equivalent to the **GA** and better than **DTS**, **CRS** and **DE**. Furthermore, depending on the cases, they generally require a lower number of functional evaluations than **GA**, **CRS** or **DE** alone. Therefore, these algorithms are good substitutes to classical evolutionary algorithms in cases where the gradient of the functional is difficult to access.

Remark 5. *In addition to the low dimensional benchmark results presented above,* ***SMA2*** *and* ***GMA*** *have been applied to the solution of several low dimensional design problems involving computationally expensive cost functions with often several local minima. The application domains concern: structural design (Carrasco et al., 2015, 2012), oil skimmer trajectory optimization (Gomez et al., 2011), synthesis of optical fiber (Ivorra et al., 2014), optimization of microfluidic mixers (Ivorra et al., 2013, 2006), temperature and pollution control in flames (Debiane et al., 2006), credit portfolio risk management (Ivorra et al., 2009), control of the solution for a PDE (Ivorra et al., 2007), optimization of the shape of coastal structures (Isebe et al., 2008).*

4.2. High dimensional benchmark problems

4.2.1. Considered benchmark problems

We now consider the following benchmark box-constrained optimization problems in dimension D (Floudas and Pardalos, 1999): Griewank (**GrD**), Rosenbrock (**RbD**) and Zakharov (**ZaD**). Following Li et al. (2013) D is set to 50, 100, 500 and 1000.

4.2.2. Considered algorithms and parameters

To solve the benchmark problems introduced in Section 4.2.1, we only consider those algorithms presented in Section 4.1.2 which have well performed in Section 4.1.3. More precisely, we only consider **SD**, **GA**, **DE**, **CRS**, **SMA3**, **GMA**, **CMA** and **DMA**. **SMA1**, **HB** and **HMA** are omitted due to their poor performances there. Also **SMA2** is not reported as it has similar behavior than **SMA3**.

The differences when comparing the parameters with those used in Section 4.1.3 are:

- **DE**: the population size is set to 60 and the maximum number of iteration is set to 3000, as proposed in Gardeux et al. (2011).
- **CRS**: the population size is set to 400 (see, Hendrix et al. (2001)).
- **SD**: $t_0 = 20000$.
- **GA**: the population size is set to 250.
- **SMA3**: $t_0 = 300$.
- **GMA**: The population size is set to 100 and the maximum number of iterations to 20.
- **CMA**: The population size is set to 50 and the maximum number of iterations is set to 1000.
- **DMA**: The population size is set to 60 and the maximum number of iterations is set to 100.

The parameters for **SD**, **SMA3**, **GMA**, **CMA** and **DMA** are determined experimentally. The same stopping criterion (19) used in Section 4.1.2 is considered here. The maximum number of functional evaluations is set to 150000. Each experiment is repeated 100 times.

4.2.3. Results

The average number of functional evaluations (considering only successful runs satisfying the stopping criterion (19)) needed by the optimization algorithms to solve the benchmark problems are shown in Table 4. The success rate of the algorithms are reported in Table 5. Furthermore, as suggested in Li et al. (2013); Gardeux et al. (2011), we also report in Table 6 the mean final value of the cost function returned by each optimization algorithm for each benchmark problem over the 100 runs. Finally, the Imp values (20) of **SMA3**, **GMA**, **CMA** and **DMA** are shown in Table 7.

From Table 5, we see that the **MLAs** exhibit better success rates than their respective **COAs**. Furthermore, as observed in Table 4, if the stopping criterion (19) is satisfied, the average number of cost function evaluations is generally lower for **MLAs** than for their associated **COAs**. Those results indicate that, as in the low dimensional cases, **MLAs** seem to improve the efficiency of the considered **COA**. However, for these high dimensional cases

the improvement is not as important as for the low dimensional cases. Indeed, we can see in Table 7 that the Imp values of considered **MLAs** are between 2% and 30% (instead of 33% and 95% as in the low dimensional cases). Furthermore, we see that when the cost function is difficult to minimize (such as, **Rb-1000** or **Za50-Za1000**), **MLAs** do not improve the success rate, which stays at 0%. Thus, a **COA** well adapted to the considered optimization problem should be selected in order to create an efficient **MLA**.

Focusing on the global performances of the algorithms, results in Table 6 seem to indicate that the **SMA3** gives the best results. In particular, for the **Za50-Za1000** cases, **SMA3** performs much better. For benchmark functions **Gr50-Gr1000** and **Rb50-Rb1000** all algorithms give similar solutions. When the gradient evaluation is not possible (as in some industrial problems where the cost function is computed by using a black-blox software, see Ivorra et al. (2006, 2013)) **GMA**, **CMA** and **DMA** present good alternatives to **SMA3**.

From a general point of view, the results reported in Table 6 are similar to other studies in the literature including different algorithms solving high dimensional cases (see, for instance, Gardeux et al. (2011)): the solution of the Griewank case can be approximated with a good precision whereas the minimum of the Rosenbrock function is difficult to evaluate.

5. Conclusions

A new multi-layer line search method, denoted by **MLA**, has been developed. This metaheuristic algorithm solves a sub-optimization problem in order to improve the initialization of existing optimization procedures considered as core optimization algorithm (**COA**). A particular implementation of the approach, well suited for minimizing non-negative functions, has been presented and coupled with various **COAs**: Steepest Descent, Heavy-Ball, Genetic, Differential Evolution and Controlled Random Search algorithms. The **MLAs** have been validated on various benchmark problems from low (with dimension less than 10) to high (with dimension up to 1000) dimensional cases. The numerical results seem to indicate that our methodology improves the performances of the **COAs**. The general conclusion is that **MLAs** with **COAs** given by gradient descent algorithms (**SMA2** and **SMA3**) and evolutionary algorithms (**GMA**, **CMA** and **DMA**) should be preferred. These latter are good alternatives to other metaheuristic algorithms such as **DTS**, **DE**, **CRS** and **CGR**.

Func.	SD	GA	CRS	DE
Gr50	7333	54629	96406	3597
Gr100	18540	80117	113500	90766
Gr500	-	125964	125500	120117
Gr1000	-	146934	143137	147395
Rb50	108768	125781	119018	130804
Rb100	-	135409	125409	137854
Rb500	-	149635	135184	-
Rb1000	-	-	-	-
Za50	3540	-	-	-
Za100	-	-	-	-
Za500	-	-	-	-
Za1000	-	-	-	-
Func.	**SMA3**	**GMA**	**CMA**	**DMA**
Gr50	3409	40792	89347	2882
Gr100	9749	70562	100137	5459
Gr500	98353	109766	123830	73413
Gr1000	141378	135028	142552	105185
Rb50	7999	119781	117781	127612
Rb100	62415	122796	122796	134453
Rb500	78749	146002	125409	-
Rb1000	-	-	-	-
Za50	4858	-	-	56578
Za100	-	-	-	-
Za500	-	-	-	-
Za1000	-	-	-	-

Table 4: Average number (considering only the runs satisfying the stopping criterion (19)) of functional evaluations needed by the optimization algorithms described in Section 4.2.2 to solve the benchmark problems (**Func.**) presented in Section 4.2.1.

Func.	SD	GA	CRS	DE	SMA3	GMA	CMA	DMA
Gr50	100	100	100	100	100	100	100	100
Gr100	100	100	100	100	100	100	100	100
Gr500	0	100	100	100	100	100	100	100
Gr1000	0	100	100	100	100	100	100	100
Rb50	15	95	83	81	100	97	89	85
Rb100	0	79	73	68	86	82	76	75
Rb500	0	69	27	0	47	72	39	0
Rb1000	0	0	0	0	0	0	0	0
Za50	17	0	0	0	100	0	0	73
Za100	0	0	0	0	0	0	0	0
Za500	0	0	0	0	0	0	0	0
Za1000	0	0	0	0	0	0	0	0

Table 5: Success rate (%) of the optimization algorithms described in Section 4.2.2 when solving the benchmark problems (**Func.**) presented in Section 4.2.1.

Our current effort is on parallel **MLAs** (Gomez et al., 2011).

A Matlab version of some of the algorithms presented in this paper has been implemented in the free optimization package "Global Optimization Platform", which can be downloaded at

`http://www.mat.ucm.es/momat/software.htm`

Acknowledgements

This work has been supported by the Spanish Ministry of Economy and Competitiveness under projects MTM2011-22658; the Junta de Andalucía; the European Regional Development Fund (ERDF) through projectsP12-TIC301; the research group MOMAT (Ref. 910480) supported by "Banco Santander" and the "Universidad Complutense de Madrid". We would like to thank Dr. Juana Lopez Redondo and Prof. Pilar M. Ortigosa, from the "Universidad de Almería", for providing us with a Matlab implementation of the Controlled Random Search algorithm.

Ashlock, D. (2010). *Evolutionary Computation for Modeling and Optimization*. Springer Publishing Company, Incorporated, 1st edition.

Func.	**SD**	**GA**	**CRS**	**DE**
Gr50	10^{-6}	10^{-6}	10^{-6}	10^{-6}
Gr100	10^{-6}	10^{-6}	10^{-6}	10^{-6}
Gr500	10^{-4}	10^{-6}	10^{-6}	10^{-6}
Gr1000	10^{-2}	10^{-6}	10^{-6}	10^{-6}
Rb50	3×10^{-2}	$5\times\ 10^{-3}$	9×10^{-3}	7×10^{-3}
Rb100	59	5.8	7.5	8.9
Rb500	96	20	47	68
Rb1000	219	139	147	168
Za50	633	37	1800	1228
Za100	2664	38845	28448	83252
Za500	98740	40021	79441	97333
Za1000	3.5×10^{5}	83642	1.2×10^{5}	1.6×10^{5}
Func.	**SMA3**	**GMA**	**CMA**	**DMA**
Gr50	10^{-6}	10^{-6}	10^{-6}	10^{-6}
Gr100	10^{-6}	10^{-6}	10^{-6}	10^{-6}
Gr500	10^{-6}	10^{-6}	10^{-6}	10^{-6}
Gr1000	10^{-6}	10^{-6}	10^{-6}	10^{-6}
Rb50	10^{-6}	3×10^{-3}	7×10^{-3}	6×10^{-3}
Rb100	3.9	5.0	3.1	4.2
Rb500	162	30	39	63
Rb1000	195	125	136	145
Za50	10^{-6}	1	1512	0.5
Za100	24	26159	13052	22784
Za500	403	30897	73362	83252
Za1000	7812	67423	1.1×10^{5}	1.3×10^{5}

Table 6: Mean value of the cost function obtained for the benchmark problems (**Func.**) presented in Section 4.2.1 with the optimization algorithms described in Section 4.2.2.

MLA	**SMA3**	**GMA**	**CMA**	**DMA**
Imp	30	5	2	16

Table 7: Value of Imp (%), defined by Equation (20), obtained when solving the benchmark problems presented in Section 4.2.1 with the following **MLA**: **SMA3**, **GMA**, **CMA** and **DMA**.

Attouch, H., Goudou, X., and Redont, P. (2000). The heavy ball with friction method, i. the continuous dynamical system: Global exploration of the local minima of a real-valued function by asymptotic analysis of a dissipative dynamical system. *Communications in Contemporary Mathematics*, 02(01):1–34.

Carrasco, M., Ivorra, B., and Ramos, A. M. (2012). A variance-expected compliance model for structural optimization. *Journal of Optimization Theory and Applications*, 152(1):136–151.

Carrasco, M., Ivorra, B., and Ramos, A. M. (2015). Stochastic topology design optimization for continuous elastic materials. *Computer Methods in Applied Mechanics and Engineering*.

Debiane, L., Ivorra, B., Mohammadi, B., Nicoud, F., Poinsot, T., Ern, A., and Pitsch, H. (2006). A low-complexity global optimization algorithm for temperature and pollution control in flames with complex chemistry. *International Journal of Computational Fluid Dynamics*, 20(2):93–98.

Floudas, C. and Pardalos, P. (1999). *Handbook of test problems in local and global optimization*. Nonconvex optimization and its applications. Kluwer Academic Publishers.

Gardeux, V., Chelouah, R., Siarry, P., and Glover, F. (2009). Unidimensional search for solving continuous high-dimensional optimization problems. In *Intelligent Systems Design and Applications, 2009. ISDA '09. Ninth International Conference on*, pages 1096–1101.

Gardeux, V., Chelouah, R., Siarry, P., and Glover, F. (2011). Em323: a line search based algorithm for solving high-dimensional continuous non-linear optimization problems. *Soft Computing*, 15(11):2275–2285.

Glover, F. (2010). The 3-2-3, stratified split and nested interval line search algorithms. *Research report, OptTek Systems, Boulder, CO*.

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

Gomez, S., Ivorra, B., and Ramos, A. M. (2011). Optimization of a pumping ship trajectory to clean oil contamination in the open sea. *Mathematical and Computer Modelling*, 54(12):477 – 489.

Gomez, S. and Levy, A. (1982). The tunnelling method for solving the constrained global optimization problem with several non-connected feasible regions. In Hennart, J., editor, *Numerical Analysis*, volume 909 of *Lecture Notes in Mathematics*, pages 34–47. Springer Berlin Heidelberg.

Gonalves, J. F., de Magalhes Mendes, J. J., and Resende, M. G. C. (2002). A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research*, 167:2005.

Grosan, C. and Abraham, A. (2007). Hybrid line search for multiobjective optimization. In Perrott, R., Chapman, B., Subhlok, J., de Mello, R., and Yang, L., editors, *High Performance Computing and Communications*, volume 4782 of *Lecture Notes in Computer Science*, pages 62–73. Springer Berlin Heidelberg.

Hale, J. (2009). *Ordinary Differential Equations*. Dover Books on Mathematics Series. Dover Publications.

Hedar, A.-R. and Fukushima, M. (2006). Tabu search directed by direct search methods for nonlinear global optimization. *European Journal of Operational Research*, 170(2):329 – 349.

Hendrix, E., Ortigosa, P., and Garca, I. (2001). On success rates for controlled random search. *Journal of Global Optimization*, 21(3):239–263.

Hirsch, M., Pardalos, P., and Resende, M. (2010). Speeding up continuous {GRASP}. *European Journal of Operational Research*, 205(3):507 – 521.

Isebe, D., Azerad, P., Bouchette, F., Ivorra, B., and Mohammadi, B. (2008). Shape optimization of geotextile tubes for sandy beach protection. *International Journal for Numerical Methods in Engineering*, 74(8):1262–1277.

Ivorra, B. (2006). *Optimisation globale semi-deterministe et applications industrielles*. ANRT-Grenoble, Reference: 06/MON2/0061.

Ivorra, B., Hertzog, D. E., Mohammadi, B., and Santiago, J. G. (2006). Semi-deterministic and genetic algorithms for global optimization of microfluidic

protein-folding devices. *International Journal for Numerical Methods in Engineering*, 66(2):319–333.

Ivorra, B., Mohammadi, B., and Ramos, A. M. (2009). Optimization strategies in credit portfolio management. *Journal of Global Optimization*, 43(2-3):415–427.

Ivorra, B., Mohammadi, B., and Ramos, A. M. (2014). Design of code division multiple access filters based on sampled fiber bragg grating by using global optimization algorithms. *Optimization and Engineering*, 15(3):677–695.

Ivorra, B., Ramos, A. M., and Mohammadi, B. (2007). Semideterministic global optimization method: Application to a control problem of the burgers equation. *Journal of Optimization Theory and Applications*, 135(3):549–561.

Ivorra, B., Redondo, J. L., Santiago, J. G., Ortigosa, P. M., and Ramos, A. M. (2013). Two- and three-dimensional modeling and optimization applied to the design of a fast hydrodynamic focusing microfluidic mixer for protein folding. *Physics of Fluids (1994-present)*, 25(3):–.

Lamghari, A. and Dimitrakopoulos, R. (2012). A diversified tabu search approach for the open-pit mine production scheduling problem with metal uncertainty. *European Journal of Operational Research*, 222(3):642 – 652.

Levy, A. and Gomez., S. (1985). *The tunneling method applied to global optimization. In Numerical Optimization 1984: Proceedings of the SIAM Conference on Numerical Optimization, Boulder, Colorado, June 12-14, 1984*. Proceedings in Applied Mathematics Series. SIAM.

Li, X., Engelbrecht, A., and Epitropakis, M. G. (2013). Benchmark functions for cec'2013 special session and competition on niching methods for multimodal function optimization. *2013 IEEE Congress on Evolutionary Computation*.

Luenberger, D. and Ye, Y. (2008). *Linear and Nonlinear Programming*. International Series in Operations Research & Management Science. Springer.

Mart, R., Campos, V., Resende, M. G., and Duarte, A. (2015). Multiobjective grasp with path relinking. *European Journal of Operational Research*, 240(1):54 – 71.

Mohammadi, B. and Saïac, J. (2003). *Pratique de la simulation numérique*. Conception: Industrie et Technologies. Dunod.

Muyl, F., Dumas, L., and Herbert, V. (2004). Hybrid method for aerodynamic shape optimization in automotive industry. *Computers and Fluids*, 33(5).

Polyak, B. (2007). Newtons method and its use in optimization. *European Journal of Operational Research*, 181(3):1086 – 1096.

Price, K., Storn, R. M., and Lampinen, J. A. (2005). *Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Price, W. (1983). Global optimization by controlled random search. *Journal of Optimization Theory and Applications*, 40(3):333–348.

Redondo, J. L., Fernández, J., García, I., and Ortigosa, P. M. (2009). Solving the multiple competitive facilities location and design problem on the plane. *Evol. Comput.*, 17(1):21–53.

Rocha, M. and Neves, J. (1999). Preventing premature convergence to local optima in genetic algorithms via random offspring generation. In Imam, I., Kodratoff, Y., El-Dessouki, A., and Ali, M., editors, *Multiple Approaches to Intelligent Systems*, volume 1611 of *Lecture Notes in Computer Science*, pages 127–136. Springer Berlin Heidelberg.

Storn, R. and Price, K. (1997). Differential evolution a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359.

Verhulst, F. (1996). *Nonlinear Differential Equations and Dynamical Systems*. Hochschultext / Universitext. Springer Berlin Heidelberg.

Vieira, D. A. G. and Lisboa, A. C. (2014). Line search methods with guaranteed asymptotical convergence to an improving local optimum of multimodal functions. *European Journal of Operational Research*, 235(1):38 – 46.