

INSTITUTO DE COMPUTAÇÃO  
UNIVERSIDADE ESTADUAL DE CAMPINAS

**Multiscale Parameter Search (MSPS):  
A Deterministic Approach for  
Black-box Global Optimization**

*Giovani Chiachia      Alexandre X. Falcão  
Anderson Rocha*

Technical Report - IC-11-15 - Relatório Técnico

July - 2011 - Julho

The contents of this report are the sole responsibility of the authors.  
O conteúdo do presente relatório é de única responsabilidade dos autores.

# Multiscale Parameter Search (MSPS): A Deterministic Approach for Black-box Global Optimization

Giovani Chiachia      Alexandre X. Falcão      Anderson Rocha \*

## Abstract

Optimization problems from where no structural information can be obtained arise everyday in many fields of knowledge, increasing the importance of well-performing black-box optimizers. In this paper, we introduce a new approach for global optimization of black-box problems based on the synergistic combination of scaled local searches. By looking farther to the behavior of the problem, the idea is to speed up the search while avoiding it to become locally trapped. The method is fairly compared to other well-performing techniques, showing promising results. A testbed of benchmark problems and image registration problems are considered and the proposed approach outperforms the other algorithms in most cases.

## 1 Introduction

*Global Optimization* is aimed at finding the overall optimum set of parameters for a given objective function which is usually nonconvex, containing several local minima [7]. The need to optimize such problems has been emerged from all branches of engineering, applied sciences, and sciences [6]. In many cases, no analytical nor structural knowledge can be obtained from objective functions representing real-world problems. Sometimes, even weak characteristics such as monotonicity and roughness cannot be retrieved [2]. This is the case when the function is the output of a computer program, for example. Normally, there is no other alternative to handle these problems than the direct incorporation of its responses into the optimization formulation, where the parameters being optimized set the problem internally, leading to responses from which the optimization is guided [8]. Such a kind of optimization scenario is usually referred to as *black-box* optimization and methods that cope with these problems are useful due to their generality and ease of use [24, 9].

To address black-box problems, many optimization methods have been proposed. These methods can be roughly classified in two classes: (i) the *deterministic methods* and (ii) the *stochastic methods*. Deterministic approaches include classical *derivative-based* algorithms in which the derivative is estimated by *finite differences* [12], *derivative-free* algorithms [23], and *trust region* methods [5]. On the other hand, stochastic methods rely on random

---

\*The authors are with the Department of Information Systems, Institute of Computing - Unicamp, Av. Albert Einstein 1251, 13083-852, Campinas-SP, Brazil. Phone: +55-19-3521-5881. Fax: +55-19-3521-5847. E-mail: {chiachia,afalcao,anderson.rocha}@ic.unicamp.br.

variables to sample the search space and include bio-inspired and evolutionary algorithms [10, 21, 11].

In general, deterministic algorithms are more efficient for the minimization of convex functions, while stochastic algorithms, with a higher cost, approximate better global minima of nonconvex functions [1]. The reason is that former derivative-based and local deterministic methods tend to become locally entrapped. However, this is not the case of some recent deterministic global search methods [6, 13, 22].

As pointed out by Wolpert and Macready [24], no specific algorithm can achieve the best solution in all optimization problems. Depending on the algorithm, its behavior is more suitable for a given class of problems while it may not perform well for other problems. That is the reason why many optimization algorithms have been proposed in order to solve different problems, both in the deterministic and in the stochastic fashions. Considering that new problems arise everyday, global optimization remains a research area of great importance.

In this work, we propose a new deterministic method for global optimization called *Multiscale Parameter Search* (MSPS). As most of stochastic and few of the deterministic approaches, this method considers the objective function as a *black-box*. MSPS can cope with all global optimization problems belonging to the class of the *general unconstrained* ones. As the term suggests, these problems are defined over reals and have no constraints or may have simple bound constraints [7].

The MSPS algorithm is based on the idea that many objective function evaluations can be avoided when the search is conducted in multiple scales of displacements in the parameter space. In addition, by combining the search in different scales, the method can have a broader look to the behavior of the objective function, which is suitable to deal with the nonconvexity of the problems we intend to address. The scales are set according to the bounding constraints inherent to the problem and three parametric information: the *number of scales*, the *increasing degree* of the scales, and a *sampling-range decreasing factor*.

In [21], the authors point out key characteristics that an optimization method should satisfy: (i) ability to handle black-box problems, (ii) to cope with parallel computing, (iii) the ease on using it, and (iv) to have good convergence properties. In MSPS, black-box abilities are assumed as a conception issue. In Section 2, it becomes clear that MSPS can support high parallelization and it is easy to be used. Furthermore, the formulations allow us to assume MSPS as belonging to the class of the *Pattern Search* methods. Taking into account that the exploratory movements of the *Coordinate Search* can be viewed as a subset of the displacements that guide the MSPS algorithm, our deterministic approach can take advantage of the proofs developed in [23] of global convergence on certain well-behaved problems. However, this is primarily of theoretical interest, as most of real-world problems may not satisfy such mathematical assumptions.

To evaluate the performance of our method, the experiments were twofold. First, a collection of benchmark problems widely used in the literature was considered. These problems generalize to higher dimensions and vary from convex to nonconvex functions and from separable to non-separable problems. The other part of the experiments is concerned with *image registration*. The task of registering one image into the spatial domain of another image amounts to a huge number of real-world problems in *medical imaging*, *image process-*

ing, and *computer vision* [4, 26]. Besides, image registration can be modeled as a black-box problem which needs to be optimized. The ability to perform well in this scenario extends the employment of the optimization method to a large number of applications. As we are going to see, the results are promising. Both for the benchmark problems as well as for the image registration problems, MSPS has provided remarkable results, outperforming the other methods in most cases.

This paper is organized as follows: In Section 2, we introduce the MSPS definitions together with an illustrative analysis and an algorithm for its implementation. In Section 3, we carefully explain the employed methodology for a fair comparison of our method to other well-performing approaches, such as *Particle Swarm Optimization* (PSO), *Differential Evolution* (DE), and *Simulated Annealing* (SA). The results are then presented in Section 4. At last, we draw the conclusions about the work in Section 5, finishing with some remarks for further investigations in future.

## 2 Multiscale Parameter Search

The concept of sampling solutions in multiple scales is based on the observation that a considerable number of samplings may be avoided if the search takes an organized broader look to the problem. In addition, it may also prevent the method to become trapped in local minima. With this in mind, the idea to compose a number of scaled local searches has emerged.

Consider  $\theta$  as a first guess of the solution for the problem to be optimized and  $\theta_i$  as the problem parameters, i.e., its dimensions. From this initial position  $\theta = (\theta_1, \theta_2, \dots, \theta_n)$ , the MSPS idea is to keep finding optimal displacement vectors  $\Delta^*$  so that  $\theta$  is iteratively updated into new positions  $\theta = \theta + \Delta^*$  until an optimality criterion is met or a number of iterations is performed. In order to make more reasonable moves, the method conducts the search in multiple scales, perturbing each parameter  $i = \{1, 2, \dots, n\}$  in  $j = \{1, 2, \dots, m\}$  displacement scales. At each iteration, MSPS evaluates  $F(\theta + \Delta)$  for the displacements  $\Delta$  resulting from the perturbations of each parameter in all scales and also derived from the composition of the best perturbations within the scales and among them.

More formally<sup>1</sup>, let  $\Delta_{i,j}$  be a positive displacement along the parameter axis  $i$  for scale  $j$  according to the the exponential function below:

$$\Delta_{i,j} = \frac{j^d}{2m^d}(u_i - l_i) \quad (1)$$

where  $\mathbf{l} = (l_1, \dots, l_n)$  and  $\mathbf{u} = (u_1, \dots, u_n)$  are lower and upper bounding vectors, and  $d$  is a constant which describes the growing degree of the scales. The method takes into account the following displacements:

- The best perturbation (if any) along each parameter axis  $i$  and scale  $j$

---

<sup>1</sup>Here the definitions are for minimization, but the changes to maximization are trivial.

$$\Delta_{i,j}^* = (0, \dots, \Delta_{i,j}^*, \dots, 0)$$

where  $\Delta_{i,j}^* \in \{\Delta_{i,j}, 0, -\Delta_{i,j}\}$  such that

$$F(\theta + \Delta_{i,j}^*) = \min \left\{ \begin{array}{l} F(\theta + \Delta_{i,j}), \\ F(\theta), \\ F(\theta - \Delta_{i,j}) \end{array} \right\}. \quad (2)$$

- The resulting vectors  $\Delta \mathbf{s}_j = \sum_{i=1}^n \Delta_{i,j}^*$ , for  $j = \{1, 2, \dots, m\}$ .
- The resulting vector  $\Delta \mathbf{p} = \sum_{i=1}^n \Delta \mathbf{p}_i$ , where

$$\Delta \mathbf{p}_i = (0, \dots, \Delta p_i, \dots, 0)$$

refers to the best displacement along the scales of the parameter axis  $i$

$$F(\theta + \Delta \mathbf{p}_i) = \min_{j=1,2,\dots,m} \{F(\theta + \Delta_{i,j}^*)\}. \quad (3)$$

Note that  $\Delta \mathbf{p}_i$  takes into account the individual values of  $\Delta_{i,j}^*$  and is actually presented here for the sake of a cleaner formulation.

With all displacements evaluated, the choice of  $\Delta^*$  is made so that

$$F(\theta + \Delta^*) = \min \left\{ \begin{array}{l} F(\theta + \Delta \mathbf{p}). \\ F(\theta + \Delta \mathbf{p}_i) \text{ for } i = 1, 2, \dots, n. \\ F(\theta + \Delta \mathbf{s}_j) \text{ for } j = 1, 2, \dots, m. \end{array} \right\}. \quad (4)$$

If  $\theta + \Delta^*$  represents an improvement over  $\theta$ , i.e.,  $F(\theta + \Delta^*) < F(\theta)$ , then  $\theta = \theta + \Delta^*$  is taken for the next iteration. Otherwise,

$$\Delta_{i,j} = \frac{\Delta_{i,j}}{2^\alpha} \quad (5)$$

is considered in order to refine the search.

One can see this refinement as an increasing on the *exploitation* ability in detriment of the *exploration* capability. In earlier iterations, the aim is to keep exploring unvisited or relatively unexplored search space regions. From a certain iteration, however, the displacements must cope with the fine tuning of the solution so far. In MSPS, this tradeoff is controlled by  $\alpha$ .

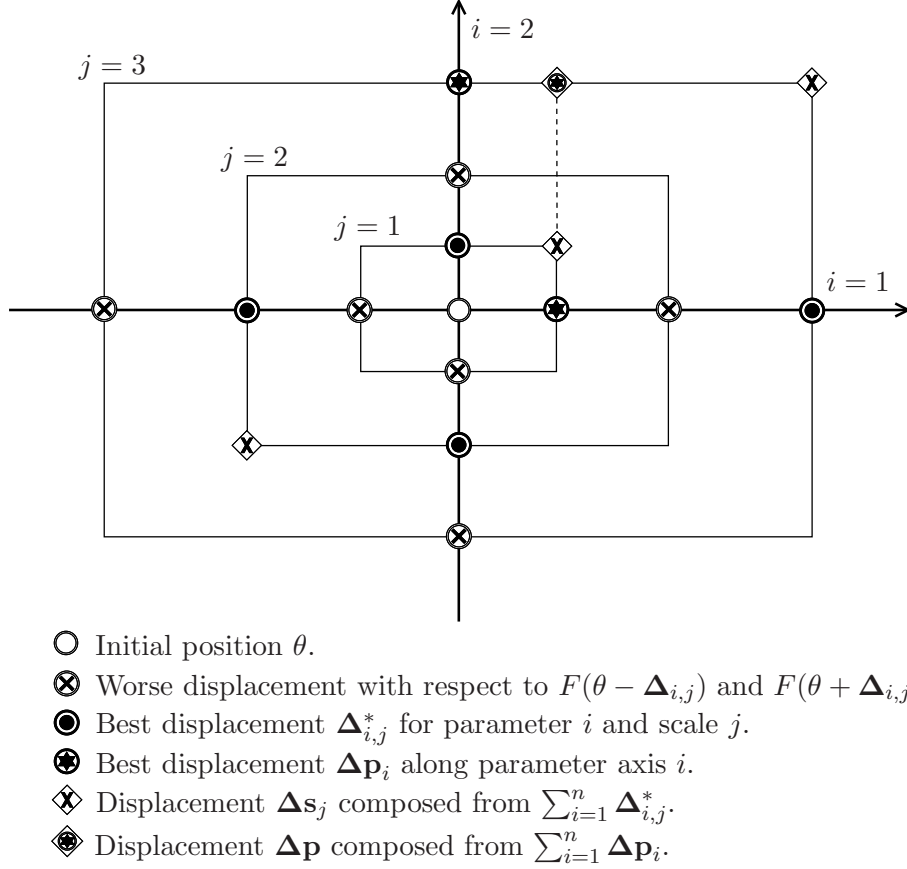


Figure 1: A toy example of an MSPS iteration. In this 2-dimensional problem, the rectangles represent the scales, and the axes the parameters. For each scale and parameter, the solution  $\theta$  at the origin is displaced in both directions where the problem is evaluated. This leads to  $\Delta_{i,j}^*$ , which is synergistically combined in order to compose  $\Delta \mathbf{s}_j$  and  $\Delta \mathbf{p}$ . These displacements are then evaluated before  $\Delta^*$  is established.

In Figure 1, we present an illustration about how the search is done in a given iteration. This toy example considers the optimization of two parameters, a 2-dimensional problem thereof. The rectangles represent the scales, and the axes represent the parameters. For each scale and parameter, the solution at the origin is displaced in both directions where the problem is evaluated. This leads to  $\Delta_{i,j}^*$ , which is synergistically combined in order to compose  $\Delta \mathbf{s}_j$  and  $\Delta \mathbf{p}$ , which in turn are also evaluated before  $\Delta^*$  is established.

With respect to the parallelization issue, one can observe that for all parameters and scales,  $\Delta_{i,j}^*$  can be computed at the same time, given that all movements are independent. Considering that at each iteration the method evaluates the problem  $2mn + m + 1$  times, the time expenditure of the dominant term can be greatly reduced.

Some correspondences between MSPS and swarm-based algorithms can be drawn. In

swarm intelligence, each member executes some particular operations and shares its information with others [19]. Although the operations are simple, their collective effect usually produces very good results. In MSPS, we can consider each evaluated position in a given iteration as a member of the swarm. The information they share are the location and fitness value, and their following movements are also decided based on information from all group members. The difference is that, while swarm-based algorithms solve the problem without using any central controller, in MSPS, we have a conductor orchestrating the scales and directions.

## 2.1 MSPS Algorithm

The implementation of MSPS is straightforward. In the example below, we take the advantage of composing  $\Delta \mathbf{s}_j$  and  $\Delta \mathbf{p}_i$  while  $\Delta_{i,j}^*$  is calculated. This leads to a few control structures and requires an insignificant amount of memory. The notations through the pseudocode are the same as in the definitions before.

### Algorithm 1. – MSPS

INPUT:        Number  $n$  of parameters,  
                  Vectors  $\mathbf{l}$  and  $\mathbf{u}$  with the corresponding lower and upper bounds of each parameter,  
                  The number  $m$  of scales and their increasing degree  $d$ ,  
                  The decreasing factor  $\alpha$  of the sampling-range,  
                  Vector  $\theta$  with the initial parameters,  
                  Objective function  $F$  to be minimized.

OUTPUT:      Vector  $\theta^*$  with optimum parameters,  
                  The value  $V^*$  of the objective function  $F$  evaluated at  $\theta^*$

AUXILIARY:   Search matrix  $\Delta_{i,j}$  as formulated in Equation 1,  
                  Displacement vectors  $\Delta$ ,  $\Delta^*$ ,  $\Delta \mathbf{s}$ , and  $\Delta \mathbf{p}$ ,  
                  Displacement vectors  $\Delta \mathbf{p}_i$ ,  
                  Variable  $V_0$  with the value of  $F$  at the beginning of the iteration,  
                  Variables  $V$ ,  $V^+$ , and  $V^-$  to store temporary values of  $F$ ,  
                  Vector  $Vp_i$  to store the best value of  $F$  along the scales of parameter  $i$ .

1. For all  $(i, j)$ , compute  $\Delta_{i,j}$  by Equation 1.
2.  $V^* \leftarrow F(\theta)$  and  $\theta^* \leftarrow \theta$ .
3. Until the termination criterion is met, do
4.      $V_0 \leftarrow V^*$ ,  $\theta \leftarrow \theta^*$ ,  $\Delta \mathbf{p}_i \leftarrow (0, \dots, 0)$  and  $Vp_i \leftarrow V^*$  for all  $i$ .
5.     For each  $j \leftarrow 1, 2, \dots, m$  do
6.          $\Delta \mathbf{s} \leftarrow (0, \dots, 0)$ .
7.         For each  $i \leftarrow 1, 2, \dots, n$  do
8.              $\Delta \leftarrow (0, \dots, \Delta_{i,j}, \dots, 0)$ ,  $V \leftarrow V_0$ , and  $\Delta^* \leftarrow (0, \dots, 0)$ .
9.              $V^+ \leftarrow F(\theta + \Delta)$  and  $V^- \leftarrow F(\theta - \Delta)$ .
10.            If  $V^+ < V$  then  $V \leftarrow V^+$  and  $\Delta^* \leftarrow \Delta$ .
11.            If  $V^- < V$  then  $V \leftarrow V^-$  and  $\Delta^* \leftarrow -\Delta$ .
12.            If  $V < Vp_i$  then  $\Delta \mathbf{p}_i \leftarrow \Delta^*$  and  $Vp_i \leftarrow V$ .
13.            If  $V < V^*$  then  $\theta^* \leftarrow \theta + \Delta^*$  and  $V^* \leftarrow V$ .
14.             $\Delta \mathbf{s} \leftarrow \Delta \mathbf{s} + \Delta^*$ .
15.             $V \leftarrow F(\theta + \Delta \mathbf{s})$ .
16.            If  $V < V^*$  then  $V^* \leftarrow V$  and  $\theta^* \leftarrow \theta + \Delta \mathbf{s}$ .

17.  $\Delta \mathbf{p} \leftarrow (0, \dots, 0)$ .
18. For each  $i = 1, 2, \dots, n$  do  $\Delta \mathbf{p} \leftarrow \Delta \mathbf{p} + \Delta \mathbf{p}_i$ .
19.  $V \leftarrow F(\theta + \Delta \mathbf{p})$ .
20. If  $V < V^*$  then  $V^* \leftarrow V$  and  $\theta^* \leftarrow \theta + \Delta \mathbf{p}$ .
21. If  $V^* = V_0$  then  $\Delta_{i,j} = \Delta_{i,j}/2^\alpha$  for all  $(i, j)$ .

Recall that the termination criterion may be a maximum number of iterations, the achievement of a desirable value of  $F$ , or even  $V^* = V_0$  after a number of consecutive refinements in  $\Delta_{i,j}$  are considered.

### 3 Evaluation Methodology

In this section we present all subjects of interest for the evaluation of the proposed optimizer. In Sections 3.1 and 3.2, we start with the problems that will be addressed. Next, we consider the methods with which MSPS will be compared (Section 3.3) and some important issues related to the experiments (Section 3.4). After that, an important aspect of the methodology for a fair evaluation is presented in Section 3.5.

#### 3.1 Benchmark Problems

In order to compare MSPS to other well-performing optimization techniques, we evaluate the methods in a testbed of 12 well-known benchmark problems [17]. These problems can be extended to a higher number of dimensions and vary from convex (e.g. Sphere) to nonconvex (e.g. Rastrigin) functions, and from separable (e.g. Sphere) to non-separable (e.g. Rosenbrock) problems. The QuarticNoise problem contains random noise, the Step problem is discontinuous, and the problems Penalized1 and Penalized2 have constraints in the form of penalty functions. These functions represent a very difficult class of problems for many optimization algorithms [25]. On evaluating them, the aim is not to show whether MSPS is better or worse than the other methods, but to find out when and why MSPS is better (or worse). After all, no single search algorithm is the best on average for all problems [24].

The formulations of the benchmark problems are shown in Table 1. These problems all have an optimal fitness value equal to zero, although the QuarticNoise problem has added noise.

#### 3.2 A Real-World Application: Image Registration

Evaluating MSPS on benchmark problems is important because we can have an idea about how the method compares to others in optimization problems widely studied in literature. However, most of black-box optimization problems cannot be explicitly formulated in terms of analytical equations. Real-world problems may be highly nonlinear when, for example, the aim is to find optimal synaptic weights of an *artificial neural network* [13], or to control the walking parameters of a robot [8]. Image registration is another area of study in which black-box optimization techniques play a crucial role. The task of finding correspondences

Table 1: Benchmark problems employed in this work and the considered search intervals [25].

Function	Equation	Search Interval
Ackley	$f(\mathbf{x}) = e + 20 - 20 \exp\left(-0.2\sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2}\right) - \exp\left(\frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i)\right)$	$[-30, 30]^n$
Griewank	$f(\mathbf{x}) = 1 + \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right)$	$[-600, 600]^n$
Penalized1	$f(\mathbf{x}) = \frac{\pi}{n} (10 \sin^2(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 (1 + 10 \sin^2(\pi y_{i+1})) + (y_n - 1)^2) + \sum_{i=1}^n u(x_i, 10, 100, 4)$ $y_i = 1 + (x_i + 1)/4$ $u(x_i, a, k, m) = \begin{cases} k(-x_i - a)^m & , x_i < -a \\ 0 & , -a \leq x_i \leq a \\ k(x_i - a)^m & , x_i > a \end{cases}$	$[-50, 50]^n$
Penalized2	$f(\mathbf{x}) = 0.1 (\sin^2(3\pi x_1) + \sum_{i=1}^{n-1} (x_i - 1)^2 (1 + \sin^2(3\pi x_{i+1})) + (x_n - 1)^2 (1 + \sin^2(2\pi x_n))) + \sum_{i=1}^n u(x_i, 5, 100, 4)$ <p>with <math>u(\cdot)</math> from above</p>	$[-50, 50]^n$
QuarticNoise	$f(\mathbf{x}) = \sum_{i=1}^n (ix_i^4 + \text{random}[0, 1])$	$[-1.28, 1.28]^n$
Rastrigin	$f(\mathbf{x}) = \sum_{i=1}^n (x_i^2 + 10 - 10 \cos(2\pi x_i))$	$[-5.12, 5.12]^n$
Rosenbrock	$f(\mathbf{x}) = \sum_{i=1}^{n-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	$[-100, 100]^n$
Schwefel1-2	$f(\mathbf{x}) = \sum_{i=1}^n \left(\sum_{j=1}^i x_j\right)^2$	$[-100, 100]^n$
Schwefel2-21	$f(\mathbf{x}) = \max\{ x_i  : i \in \{1, \dots, n\}\}$	$[-100, 100]^n$
Schwefel2-22	$f(\mathbf{x}) = \sum_{i=1}^n  x_i  + \prod_{i=1}^n  x_i $	$[-10, 10]^n$
Sphere	$f(\mathbf{x}) = \sum_{i=1}^n x_i^2$	$[-100, 100]^n$
Step	$f(\mathbf{x}) = \sum_{i=1}^n ( x_i + 0.5 )^2$	$[-100, 100]^n$

(e.g., transformations) between two or more images obtained under different conditions is fundamental in image processing and computer vision [26]. This is because images can be captured at different times, using different sensors, and from different viewpoints. In most cases, it is desirable that these factors be alleviated in order to proceed with the interpretation of the images' content.

The problems we present in this section are related to image registration, more specif-

ically with the registration of face pictures taken under different time, viewpoints, and illumination. The prior alignment of the faces is a mandatory step in most of the face recognition algorithms [14], but the way this alignment is carried out, however, may differ. Regularly, the faces are rotated, translated, and scaled with respect to previously detected fiducial points (e.g., nose tip, lip and eye corners) from where correspondences are calculated and the images are registered. Instead of automatically detecting facial landmarks, we are going to follow a different procedure. Based on earlier segmented face images, our registration pipeline can be depicted in major components as in [4]. In the next three sections we shortly describe them.

### 3.2.1 Feature Space

The features employed in registering two images can be as simple as the raw pixel intensities, but other informative characteristics such as edges, contours, surfaces, corners, line intersections, can also be used [4]. Moreover, the feature space may differ based on the role the image plays in the registration. On one side, we have the image whose spatial domain is preserved by the registration, the *model image*, while on the other side, there is the image which is going to be transformed in order to reach the model, the *scene image*.

In our approach, for both the model and the scene images, we obtain their edges by the *Morphological Gradient*. The edges are then thresholded by a certain magnitude  $\lambda$  empirically obtained, leading to a structural description of the faces and their main components. In addition, we apply the *Euclidean Distance Transform* (EDT) on the remaining edges of the model so that we can obtain a surface with valleys and peaks regarding such a structure [15]. The importance of the EDT will become clear in the next section. In Figure 2, we can have an idea about the features on which the registration is based. The initial model and scene images are 2(a) and 2(b), correspondingly. Figures 2(c) and 2(d) represent their thresholded morphological gradient. From 2(c), the EDT is calculated, leading to the surface in 2(e). During the registration, 2(e) is taken as the model image and 2(d) as the scene image representations.

### 3.2.2 Similarity Function

Let us now define our problem. Consider the model image as a pair  $\hat{M} = (D_M, M)$ , where  $D_M \subset Z^2$  is the set of pixels (image domain) and  $M(q)$  is a function that assigns the Euclidean distance of  $q \in D_M$  to the closest edge pixel considered from the original image (Figure 2(e)). Let  $\hat{S} = (D_S, S)$  be the scene image with its own domain  $D_S \subset Z^2$  and with  $S(p)$  mapping each pixel  $p \in D_S$  to a value proportional to its edge membership (Figure 2(d)). By assuming that both images have a great part of the objects in common, the problem is to find the best geometric transformation  $T$  that maps the pixels in  $D_S$  onto the domain  $D_M$ . For sake of efficiency and generalization, we consider only the subset  $S_S \subset D_S$  obtained from the thresholding  $S(p) > \lambda$  mentioned in the previous section. This way, the optimal  $T$  can be found with the minimization of

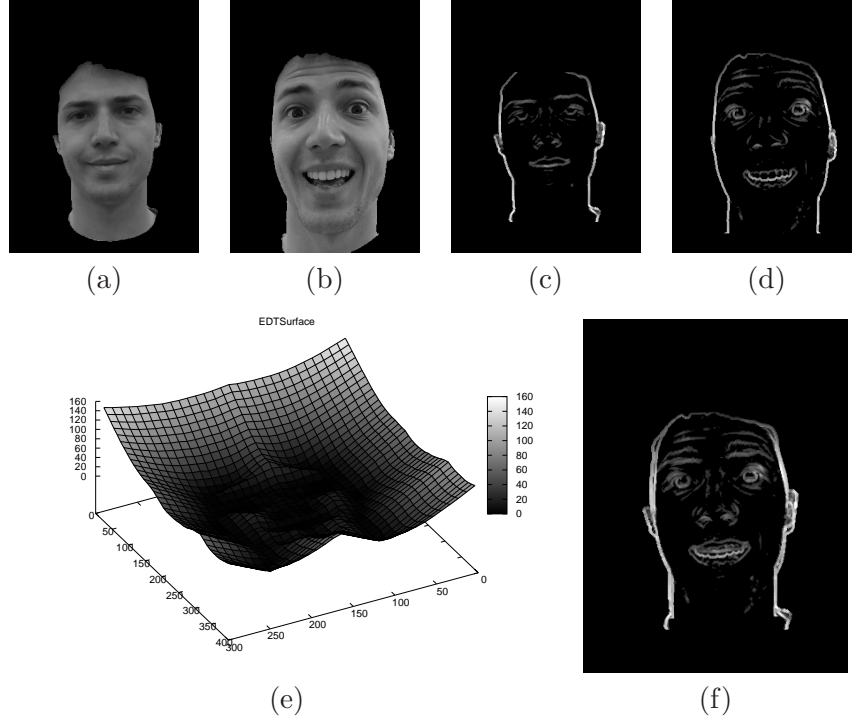


Figure 2: Image registration pipeline considered in this work. The initial model and scene images are (a) and (b), respectively. Figures (c) and (d) are their resulting morphological gradient. The EDT is calculated from (c), leading to (e). The registration is based on the characteristics of (e) and (d) and aims to achieve an optimal solution here illustrated by (f).

$$F(T) = \frac{\sum_{\forall p \in S_S} M(q)S(p)}{|S_S|} \quad (6)$$

such that

$$q = \begin{cases} Tp & , Tp \in D_M \\ \text{nearest}(Tp, D_M) & , \text{otherwise} \end{cases} \quad (7)$$

where  $\text{nearest}(x, D_X)$  returns the Euclidean nearest pixel  $x' \in D_X$ , enabling an estimate for pixels whose mapping is out of the model's domain.

In short, one can see this function as a measurement of how good the edges from the scene image are accommodated along the valleys of the model surface. It also considers the scene edge membership in order to give more importance to the pixels which represent stronger edges. Moreover, the measure is normalized by the size of  $S_S$ , which makes different registration problems to have a fitness value belonging to a common interval. This last property is of great importance to the experiments. As in Figure 2(f), the expectation is that the lower the function response, the better the registration.

Table 2: Boundaries for the image registration search space.

Rotation Degree	Horizontal Displ.	Vertical Displ.	Scaling Factor
$[-20, 20]$	$[-75, 75]$	$[-100, 100]$	$[0.75, 1.25]$

### 3.2.3 Search Space and Strategy

The face database considered in this work has 570 distinct pairs of images. These pairs are from pictures of the same individual taken with a time difference of one week. At total we have 19 subjects, with 30 pictures each from both recording sessions. The images were made  $150 \times 200$  in size and by analyzing the size, the position, and the pose of the faces, we could set up the search space boundaries (Table 2). As it can be noted, we have a 4-dimensional search space, where the first dimension refers to rotation degree, the second and the third are related to the horizontal and vertical displacements in pixels, and the fourth dimension applies to the scaling factor.

Once with well-defined image registration problems, what remains to fulfill the registration pipeline is the strategy (i.e., the optimization method) that is going to be employed to search for their solutions. The different approaches considered in this work are shortly described in next section.

## 3.3 Evaluated Optimization Methods

The optimization methods compared to MSPS in this work are: Particle Swarm Optimization, Differential Evolution, Simulated Annealing, and Pattern Search. The first three were considered due to their popularity and proven effectiveness in solving a wide range of problems (especially PSO and DE), while Pattern Search was evaluated because of the resemblance that MSPS has to such a family of algorithms. In this last case, the aim is to stress the importance of the multiscale property of MSPS. The implementations of these four techniques were from [16].

## 3.4 Experimental Issues

It should be clear that the evaluations in this work are empirical. Although one may argue that theoretical analysis is desirable in order to obtain mathematical certainties about performance, there are many difficulties in providing such an analysis for black-box problems. As a consequence, most of the optimization methods are evaluated empirically by applying them to a collection of problems and comparing the observed solution quality and computational burden [18].

One matter of concern while evaluating responses from optimizations is how they are obtained. Normally, some index of performance on one or more problems is evaluated. In [3], the authors observe that reporting the central tendency of the executions of an optimization algorithm is much more important than its best result, because the latter may be just an over-optimistic measure of performance. This is of particular importance for stochastic

Table 3: Initialization of the agents’ positions for the image registration problems.

Type	Rotation Degree	Horizontal Displ.	Vertical Displ.	Scaling Factor
Single agent	0	0	0	1.0
Population-based	$[-4, 4]$	$[-15, 15]$	$[-20, 20]$	$[0.95, 1.05]$

methods, where randomness is employed within the search. In MSPS we have no random factors, although the solution from where our method starts the search may be random. This way, in accordance to other originally proposed global optimization heuristics [19, 21], the comparisons are done in terms of the mean fitness from a number of independent runs.

With respect to the solutions from where the methods start the search, for the benchmark problems, they were uniformly generated at random within the search intervals (Table 1) for all the optimization methods. For the image registration problems, we have a distinction between the *population-based* methods (PSO and DE) and the *single agent* methods (MSPS, SA, and PS). In our image registration scenario, it is interesting that the optimization starts from the solution (agent) representing the identity transformation. This is because the images are little misaligned. However, this is not possible to consider for PSO and DE, where many guesses should be made a priori. Hence, we defined an initialization interval surrounding the identity solution for these methods, from where the population were generated also uniformly at random. The initialization scheme is presented in Table 3.

For both the benchmark problems and the image registration problems, the results are presented in tables and in performance curves. In the tables, the values are related to the mean of the best found solutions at the end of the allowed number of iterations, while in performance curves, one can obtain relative information about the progress of the optimization over the iterations, i.e., a trade-off between the solution time and solution quality [18]. Each point in the curves is also an average from a number of runs.

MSPS was primarily designed to cope with real-time optimization as desirable in image registration for computer vision applications. To this end, while evaluating the method, we are concerned with a small number of function evaluations. More precisely, we decided to allow 125 evaluations per problem dimension because this makes the image registration to execute in a few milliseconds. Despite the benchmark problems do not impose such a real-time constraint, it would not be reasonable to compare the methods allowing them to perform a great number of function evaluations if this is not our targeted scenario.

### 3.5 Meta-Optimization

As most of the black-box optimization techniques, MSPS has control parameters that need to be tuned in order to achieve its best performance. In our case, these parameters are the number of scales  $m$ , its increasing degree  $d$ , and the sampling-range decreasing factor  $\alpha$ . The tuning of these parameters can be done through manual experimentation. In this case, it is

Table 4: Boundaries for the control parameters search space.

Method	Boundaries
MSPS	$m \in [1, 40]$ , $d \in [0, 7]$ , $\alpha \in [0, 2]$
PSO	$S \in \{1, \dots, 200\}$ , $\omega \in [-2, 2]$ , $\phi_p \in [-4, 4]$ , $\phi_g \in [-4, 4]$
DE	$NP \in \{3, \dots, 200\}$ , $CR \in [0, 1]$ , $F \in [0, 2]$
SA	$r \in [1e - 5, 1]$ , $\alpha \in [1e - 5, 1]$ , $\beta \in [1e - 5, 1]$ , $T \in [15, \#iterations]$
PS	nonparametric

not clear whether a given set of parameter values really reflects the best choice one would have. One way to find optimal values for the control parameters is to conduct an exhaustive search on their combinations. However, both manual experimentation and grid-based search are expensive to execute. While the human effort is costly, the exhaustive search for good combinations of parameters demands an exponentially increasing computation time with the number of parameters.

Another way to find optimum values for the control parameters is to conceive the optimization as a problem to be optimized in its own, i.e., to employ an additional layer of optimization. This concept is here called *Meta-Optimization* and, in our case, is a key issue for the experiments to be fair. The idea may also be found in literature under other names, such as *Meta-Evolution*, and has shown to be comparatively cheap to execute and yet reaching better performance [17, 20].

Similar to the optimization problems of Section 3.1 and Section 3.2, the meta-optimization problems also have boundaries within which the control parameters are searched. These intervals are specific for each optimization method and are presented in Table 4.

The *Local Uniform Sampling* (LUS) technique was chosen to be the meta-optimizer for all evaluated optimization approaches and problems because it can yield good results in a few iterations, which is of great importance in such a scenario. This method is specially designed to quickly optimize simpler problems and is often able to achieve good results on complex problems as well [17].

Considering the fact that LUS may sometimes get locally stuck, all meta-optimizations were executed six times with 300 iterations<sup>2</sup> and the best response from these runs was considered in order to establish the optimal setup of the control parameters. LUS has only one control parameter  $\alpha$  which refers to the sampling decreasing rate. In all meta-optimizations, this parameter was  $\alpha = 1/3$  because it is known that it works well for a wide range of problems [17]. The implementations related to the meta-optimization were also from [16].

The meta-optimization process can take into account a single optimization problem at a time or a group of related problems simultaneously. The measure which guides the meta-optimizer can be called *meta-fitness* and in this paper, depending on the type of the

<sup>2</sup>Six was default in [16] and we decided to keep it, while the aim with 300 meta-iterations was to stress the search and get more stable control parameters.

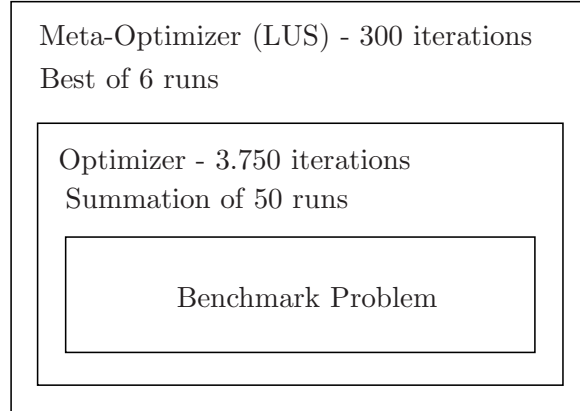


Figure 3: Meta-optimization scheme for the benchmark problems. In a meta-iteration, the summation of 50 executions of the optimizer is taken as the meta-fitness. At the end of the sixth meta-optimization run, the best found parameter setup is taken as the one that reflects the best behavior of the optimizer for the problem being considered.

problems, it can be the summation of fitnesses either of single problems from different runs or from different problems being considered.

### 3.5.1 Benchmark Problems: Specialization

The variety in the shapes and in the behavior of the benchmark problems, despite providing an in depth analysis, does not allow us to consider them altogether in the meta-optimization without biasing the process. This way, the meta-optimizations here take into account only one out of 12 problems at a time. This can be understood as the *specialization ability* of the optimization method in solving each problem [17].

Figure 3 illustrates the scheme applied to the benchmarks. The process can be characterized in three levels. In the most inner level, we have the benchmark problem being considered, which is optimized with a given choice of control parameters. In order to fairly evaluate the performance of the optimization method, this process is repeated 50 times and the summation of the resulting fitness values is reported to the meta-optimizer as the meta-fitness. The meta-optimizer runs six times and the best found control parameters are taken as the most suitable. All problems were made 30-dimensional, which, following the idea of 125 evaluations per problem dimension, leads to 3.750 allowed function evaluations.

### 3.5.2 Image Registration: Generalization

In image registration, each pair of images represents a different problem to the optimizer. Despite the similarity function being the same, the optimization problem may be quite different due to changes in the model and the scene images. However, it is certainly not practical to tune the optimizer parameters for each registration scenario. This would not

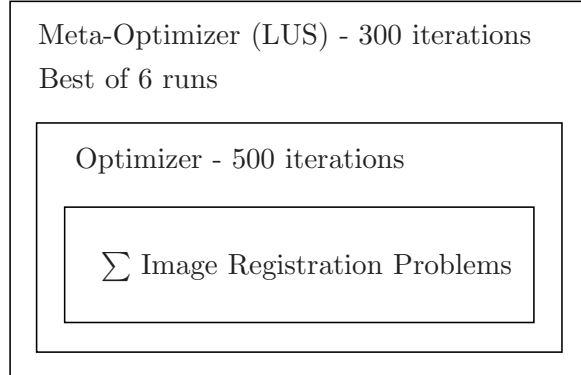


Figure 4: Meta-optimization scheme for image registration. For each meta-iteration, the summation of the 570 best found fitness values from all registration problems is taken as the meta-fitness. At the end of the sixth meta-optimization run, the best found parameter set is taken as the one that reflects the best behavior of the optimizer for these problems.

fit the requirements of most computer vision applications. In fact, only one setup of control parameters is desirable for the whole class of face registration problems.

Considering the fact that our similarity function is a summation normalized by the number of its elements, this measure produces responses in a common interval, independent of the registration problem instance. As highlighted before, this is an important issue because in a given iteration of the meta-optimizer, we can consider the summation of each problem fitness as meta-fitness without penalizing any registration in particular. This way, a common meta-optimization considering all the 570 problems is performed to fairly evaluate the optimization methods. The scheme is shown in Figure 4. Similar to the benchmark problems (Figure 3), the process is divided into three levels. At the problem level we have the 570 registration problems which are optimized and whose fitnesses are summed up so that the optimizer is evaluated in the whole set of problems. The meta-optimizer runs six times and the best found control parameters are taken as the most proper one. The LUS and the evaluated optimizer were allowed to search along 300 and 500 iterations, respectively.

When considering a method being meta-optimized in a number of similar problems, the achieved results reflect what may be called *generalization ability*, i. e., the ability of the method to perform well in problems for which its control parameters were not specifically tuned for [17]. In this case, the found parameters can be used to optimize similar problems in practice.

## 4 Results

In this section, we present the obtained results for the benchmark and the image registration experiments. In order to avoid misunderstandings, let us remark the relation between *iteration* and *function evaluation*. For the sake of a better explanation, in Section 2 we called “iteration” the group of  $2mn + m + 1$  function evaluations that MSPS performs

Table 5: Optimal setups of the control parameters obtained from the meta-optimization of MSPS. The parameters are as described in Section 2.

Problem	Scales	Degree	$\alpha$
Ackley	1	n.a.	1.58
Griewank	1	n.a.	1.52
Penalized1	2	3.67	1.31
Penalized2	2	2.59	1.55
QuarticNoise	12	0.61	1.89
Rastrigin	10	4.58	1.66
Rosenbrock	1	n.a.	0.82
Schwefel1-2	1	n.a.	1.01
Schwefel2-21	1	n.a.	1.29
Schwefel2-22	1	n.a.	1.58
Sphere	1	n.a.	1.58
Step	10	2.97	1.42

before the pivot solution is updated. Henceforth, iteration and function evaluation have the same meaning. The reason is the same as before and this meaning extends to the other optimization methods as well.

#### 4.1 Benchmark Problems

The MSPS optimal parameter setups for each optimization problem obtained from the meta-optimization are shown in Table 5. One can see that for many problems, the best was to optimize them considering just one scale in the search. This kind of behavior is not necessarily surprising and a more detailed analysis is presented with the performance results. Once we have determined these control parameters for all methods, the benchmark problems were one more time optimized with such parameters so that we could obtain the performance curves on every problem. As in the meta-optimization, we carried out 50 runs of each method with the same number of iterations. Each point of the curves presented in Figure 5 and Figure 6 expresses the mean fitness from these runs at each iteration.

Two general aspects can be depicted from the results. The first one is that MSPS outperformed the other methods in six out of the 12 problems. In some cases, the advantage was remarkable (e.g., Ackley). The second aspect is that MSPS has always provided acceptable solutions in comparison to the other methods. Except for the QuarticNoise and the Schwefel1-2 optimizations, the results of our method were always within the best two.

From the 12 problems, five are nonconvex, namely Ackley, Griewank, Penalized1, Penalized2, and Rastrigin. For these, the number of local minima increases exponentially as the dimension of the function increases. The results are then much more important since they reflect the method’s ability to escape from local optima and to locate near-global optimum solutions. When analyzing the performance of MSPS in such problems, we can see that it

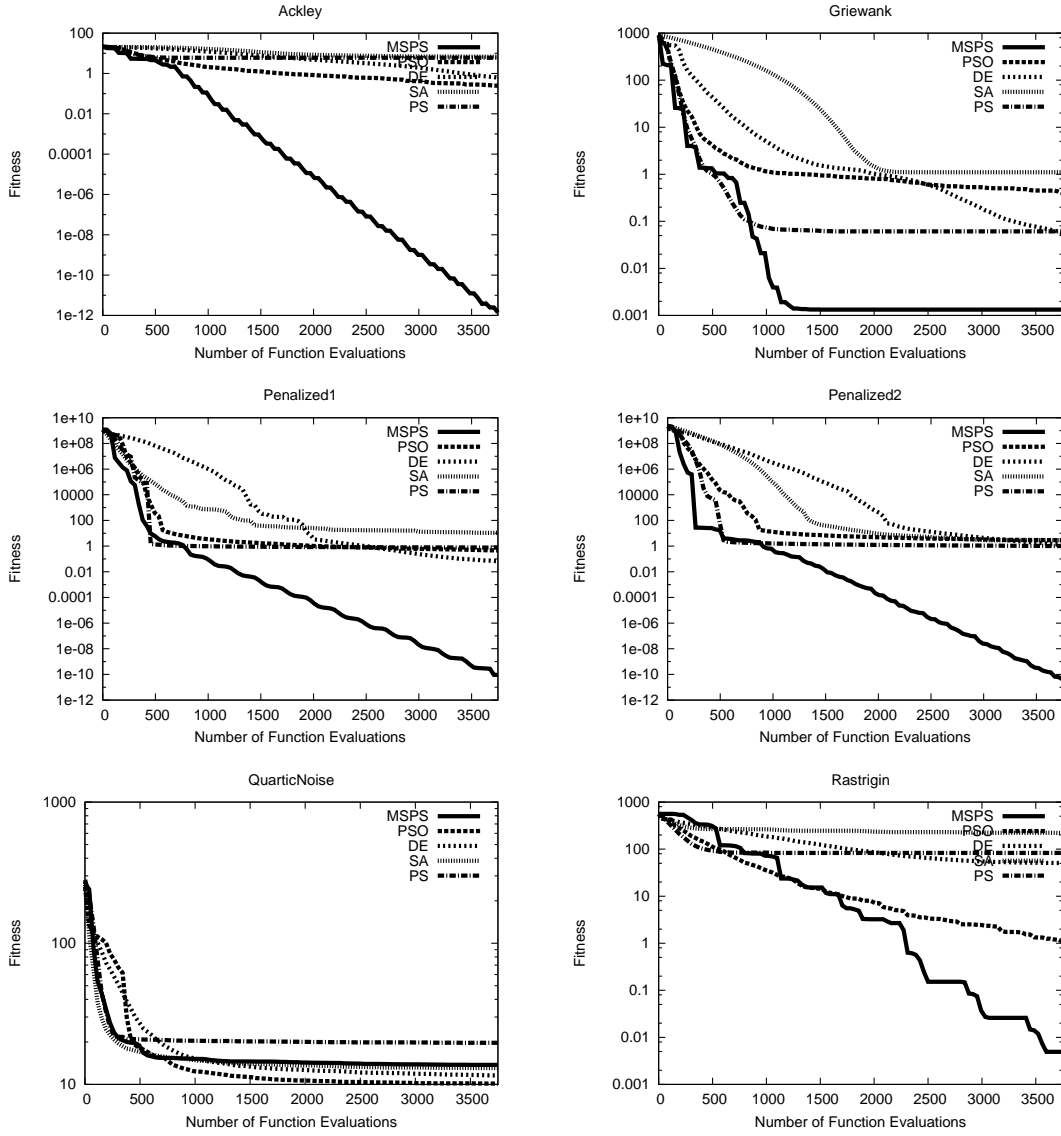


Figure 5: Optimization progress comparison of the employed techniques. The control parameters of MSPS are as in Table 5. For all methods, the parameters were meta-optimized considering one benchmark problem at a time for up to 3750 objective function evaluations. Each observation in the plots is a mean fitness of 50 optimization runs.

outperformed the other methods in all cases. In addition, the plateaus in the performance curves suggest that, in many occasions, some methods become trapped in local minima. This is specially the case of PS, which is the MSPS deterministic contestant. With no doubt, these results show the MSPS’s ability to handle nonconvexity even when the search considers only one scale.

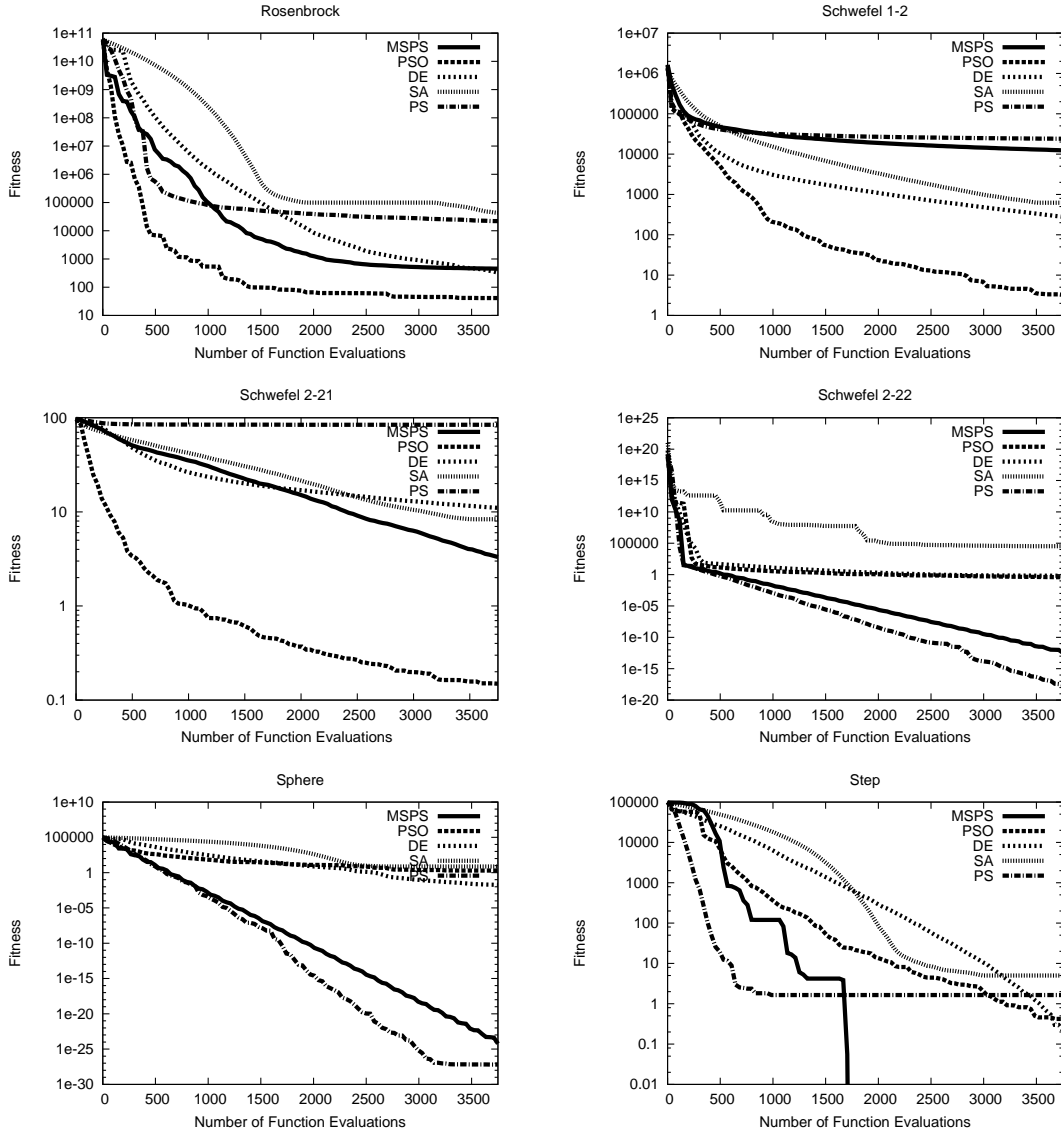


Figure 6: Optimization progress comparison of the employed techniques. The control parameters of MSPS are as in Table 5. For all methods, the parameters were meta-optimized considering one benchmark problem at a time for up to 3750 objective function evaluations. Each observation in the plots is a mean fitness of 50 optimization runs.

The convex problems are Rosenbrock, Schwefel1-2, Schwefel2-21, Schwefel2-22, Sphere and Step. With exception of Step, all others performed better with just one scale, which makes sense considering that the search movements always follows the same direction until the global minimum region is reached. In the case of Step, the multiple scales were fundamental as they allowed the method to overcome the discontinuities of the problem.

Table 6: Mean fitness achieved by each optimization method in 50 optimization runs. The best performances are highlighted in bold.

Problem	MSPS	PSO	DE	SA	PS
Ackley	<b>1.22e-12</b>	0.25	0.62	6.82	5.99
Griewank	<b>1.33e-3</b>	0.40	5.55e-2	1.10	6.10e-2
Penalized1	<b>4.70e-11</b>	0.46	6.76e-2	10.81	0.77
Penalized2	<b>3.95e-11</b>	2.88	1.29	2.88	1.01
QuarticNoise	13.75	<b>10.12</b>	11.56	13.04	19.70
Rastrigin	<b>4.11e-3</b>	1.07	50.53	219.81	83.28
Rosenbrock	453.60	<b>41.36</b>	327.94	4.30e+4	2.20e+4
Schwefel1-2	12.38e+3	<b>3.23</b>	280.55	623.68	2.40e+4
Schwefel2-21	3.31	<b>0.15</b>	11.01	8.36	84.23
Schwefel2-22	3.69e-13	0.38	0.66	3.38e+4	<b>2.95e-18</b>
Sphere	5.93e-25	1.98	1.87e-2	7.35	<b>6.52e-28</b>
Step	<b>0.00</b>	0.42	0.22	5.00	1.64

Different from PS, which became trapped at a certain moment, the multiscale property of MSPS enabled it to keep searching until the global optimum was found.

It is possible to observe that whenever PS had good performance, MSPS performed well too. This is the case with the optimization of the simple Sphere and the Schwefel2-22 problems. As pointed out before, deterministic approaches are good at optimizing convex problems. The MSPS’s performance confirms this idea. For Rosenbrock, its tricky and rapidly increasing shape made PSO to converge faster than MSPS. In the case of Schwefel2-21, the result is acceptable, and for Schwefel1-2, MSPS did not perform well. At last, for the QuarticNoise problem, it seems that the noisy responses did not allow MSPS to fine tune the search.

In Table 6, we can see the mean fitness values at the end of the same optimizations from the plots. One observation that can be derived from these numbers is that, different from MSPS, PS, and even DE, PSO was weak in fine tuning. It seems that PSO presented some difficulty for improving near global optimum solutions.

Up to this point, we have a good idea about the MSPS capabilities in dealing with different types of problems. It is important to emphasize that our analysis is restricted to the given scenario, where the number of allowed function evaluations is small. One more time we could remark that there is no such a technique whose performance is absolutely better [24]. However, it is possible to have some relative idea from the results so far. MSPS has performed better in six problems, PSO in four, and PS in two. Both MSPS and PSO were never the worst technique, while for PS this was not the case. DE had a reasonable performance, while SA performed poorly. Apparently, MSPS has great potential in solving nonconvex problems at the same time it also preserves capabilities common to local search, such as the fine tuning of the solution or the rapidly convergence in simple convex problems. This is exactly the kind of behavior we expect in order to solve real-world problems such

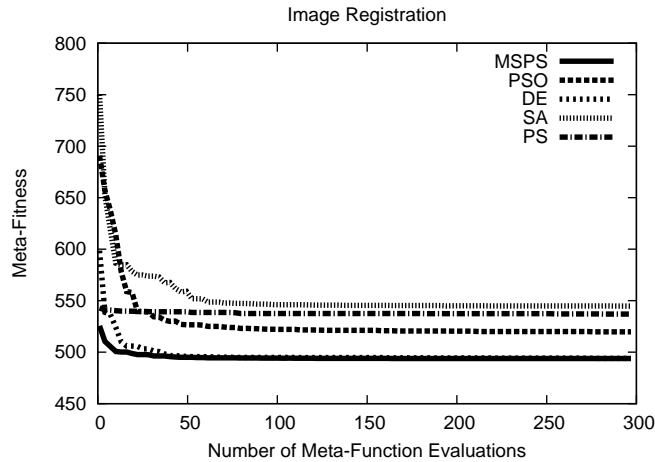


Figure 7: Performance curves of the image registration meta-optimization. In each meta-iteration, the summation of the 570 best found fitness value of each registration problem is taken as the meta-fitness, which in turn is averaged over six meta-optimization runs.

as image registration.

An additional point we noticed while analyzing the results is that MSPS was at least two times faster than PSO, DE, and SA in the benchmark problems, even though the number of iterations was the same for all of them. The reason is that these three stochastic implementations take into account uniformly distributed random values, which makes the sampling a bit more expensive. Considering this a matter of implementation and that in real-world problems the majority of the computation time is spent in the problem evaluation, we discarded the numerical analysis of this data. However, we would like to stress that in computational light problems as the benchmark ones, this results in a considerable time expense difference. In MSPS, the work required to determine the search directions and steps is minimal.

## 4.2 Image Registration

The experimental results from the image registrations are presented in Figure 7 and in Table 7. Different from the curves in Figures 5 and 6, which plot the performance of independent optimizations on each benchmark problem, in Figure 7 we have just one plot for all problems. Each point in the curves is a mean of sums. The sum is for the 570 image registrations, which are averaged over six meta-optimization runs. In Table 7, we have both the best found control parameter setup and the averaged best meta-fitness from the meta-optimization runs. As we said before, these parameters can be considered for further optimizations on the operational scenario.

Considering that we are dealing with real-world problems, we do not know which are the values representing the global minima of the problems, but comparatively, we can do some

Table 7: Optimal control parameters sets and the meta-fitnesses obtained from the meta-optimization of the face registration problems altogether. The values are related to the mean of the six best found solutions from the meta-optimization runs of each method. For MSPS, the parameters are as in Section 2. With respect to PSO and DE parameters, their meaning is of common knowledge. The SA method employed takes  $r$  as the sampling-range factor,  $\alpha$  and  $\beta$  as the starting and ending value of the movement-probability weight, and  $T$  as the number of iterations between resets [16]. Due to the generalization aspect of the meta-optimization 3.5.2, the found parameters would be taken for the optimization of similar problems in an operational scenario. The best performance is in bold.

Method	Optimal Control Parameter Setup	Meta-fitness
MSPS	$m = 3, \quad d = 1.7320, \quad \alpha = 1.0036$	<b>493.79</b>
PSO	$S = 33, \quad \omega = 0.1529, \quad \phi_p = -1.2579, \quad \phi_g = 3.1117$	519.68
DE	$NP = 11, \quad CR = 0.4369, \quad F = 0.9871$	493.92
SA	$r = 0.0435, \quad \alpha = 0.0306, \quad \beta = 0.0980, \quad T = 324$	544.69
PS	nonparametric	536.93

analyses. As we can see in Figure 7, since the first meta-iteration, MSPS outperforms the other optimization methods. Assuming that at each meta-iteration the control parameters are changed, the curve gives us the idea that the method is very stable on this type of problem, giving good results with different setups. Despite being a little more dependent on the control parameters, DE also performed well, followed by PSO, PS, and SA.

From one up to 40, MSPS performed better with three scales. This suggests that the image registration problems are not so well-behaved, otherwise, one search scale would do better. Due to the real-world scenario, we cannot even know if the performance reflects near-global optimum solutions. It may be the case that what MSPS has found are local optima solutions, nevertheless, they are better than the ones provided by PSO, PS, and SA, and slightly better than the solutions provided by DE. The MSPS capabilities found in Section 4.1, that is, the potential for solving nonconvex problems and its efficiency in fine tuning the solutions, may be the reason of the good performance on registering the images.

In the evaluation of the benchmark problems, one point that called our attention was the number of scales found to perform better. For seven problems, the meta-optimizations established that superior results were obtained by searching in one scale. Thanks to the testbed diversity, this did not flaw our assumption that searching in multiple scales is interesting in certain scenarios, which was the case for the other problems.

To further investigate the “number of scales” issue, we conducted an additional experiment. We made little changes in MSPS in order to remove its multiple scale structure, leading to a method with only one parameter, the sampling-range decreasing factor  $\alpha$ . This method was also meta-optimized by LUS in the same way we have done so far. Its best performance was achieved with  $\alpha = 0.74$ , and its meta-optimization performance curve is presented in Figure 8 together with the curve from the original method.

What we conclude from Figure 8 is that, for the image registration problems considered

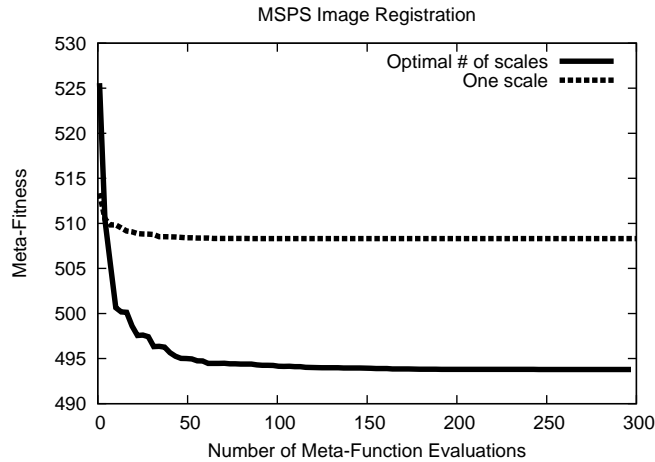


Figure 8: Performance curves from the meta-optimization of MSPS considering multiple scales and with the number of scales fixed in one. In each meta-iteration, the summation of the 570 best found fitness value of each registration problem is averaged over 50 executions of the optimizer, which in turn is averaged over six meta-optimization runs.

here, searching in multiple scales enabled a considerably better performance than for the case where the search was based in just one scale of movements. This endorses the multiscale idea as a good characteristic to consider in black-box optimization. Still about the number of scales, one may argue that this parametric information is very problem-dependent. As for other optimization methods, it may be so. However, we would like to stress that the meta-optimization of the parameters is a small matter of implementation [16]. Depending on the problems in hand, it may be computationally expensive, but the results seem to worth the effort. Finally, we also made experiments replacing LUS by MSPS as the meta-optimizer and the results were comparable, which means that MSPS does not depends of a third-part method in order to fit to the problem requirements.

## 5 Conclusions

In this paper, we proposed a deterministic method for the optimization of black-box problems. Such kind of problems arise everyday in all branches of engineering, applied sciences, and sciences. For this reason, black-box optimization is a research area of great importance. The MSPS idea is to compose a number of scaled local searches in a synergistic manner so that many function evaluations can be avoided. Additionally, the broader look to the behavior of the problem may be suitable to deal with nonconvex real-world problems, leading to solutions closer to the global optimum.

In order to evaluate our proposal, MSPS was compared to other four well-known optimization techniques. The major contestants were PSO and DE, methods whose good performance are of common knowledge in literature. The evaluation was carried out in a

fair way with the employment of the meta-optimization concept. With this framework, we eliminated any possibility of biased results due to arbitrary control parameters tuning.

The experiments were based in a difficult testbed of 12 benchmark problems and in 570 face image registration problems. We obtained promising results both for the benchmark problems as well as for the image registration problems. MSPS has outperformed the other methods in most cases. This allows us to conclude that several other problems can also be well addressed with this technique, placing MSPS in an interesting position for further investigation.

In the near future we intend to apply the method to other real-world problems, mainly in the fields of medical imaging and computer vision. We also plan to evaluate its performance in a scenario where real-time responses are not an issue of primal importance. In addition, many side ideas have emerged while doing this work. For us, the matter of combining the search in multiple scales can be extended in different interesting ways.

## References

- [1] G. Allaire. *Numerical analysis and optimization*. Numerical Mathematics and Scientific Computation. Oxford University Press, Oxford, 2007.
- [2] A. Auger, N. Hansen, J. M. Perez Zerpa, R. Ros, and M. Schoenauer. Experimental comparisons of derivative free optimization algorithms. In Jan Vahrenhold, editor, *8th International Symposium on Experimental Algorithms*, volume 5526 of *LNCS*, pages 3–15. Springer, 2009.
- [3] Mauro Birattari and Marco Dorigo. How to assess and report the performance of a stochastic algorithm on a benchmark problem: Mean or best result on a number of runs? *Optimization Letters*, 1(3):309–311, 2007.
- [4] L. G. Brown. A survey of image registration techniques. *ACM Comput. Surv.*, 24, December 1992.
- [5] Andrew R. Conn, Nicholas I. M. Gould, and Philippe L. Toint. *Trust-Region Methods*. MPS-SIAM Series on Optimization. Society for Industrial Mathematics, January 1987.
- [6] C. A. Floudas and C. E. Gounaris. A review of recent advances in global optimization. *J. of Global Optimization*, 45:3–38, September 2009.
- [7] P. Gray, W. Hart, L. Painton, C. Phillips, M. Trahan, and J. Wagner. A Survey of Global Optimization Methods. Technical report, Sandia National Laboratories, 1997. Consulted December 2010.
- [8] T. Hemker. *Derivative Free Surrogate Optimization for Mixed-Integer Nonlinear Black Box Problems in Engineering*. PhD thesis, Technische Universitat Darmstadt, December 19 2008.

- [9] D. Huang, T. T. Allen, W. I. Notz, and N. Zheng. Global optimization of stochastic black-box systems via sequential kriging meta-models. *Journal of Global Optimization*, 34:441–466, March 2006.
- [10] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948, August 2002.
- [11] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680, 1983.
- [12] Tamara G. Kolda, Robert Michael Lewis, and Virginia Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, 45(3):385–482, 2004.
- [13] J. Lee. A novel three-phase trajectory informed search methodology for global optimization. *J. of Global Optimization*, 38:61–77, May 2007.
- [14] S. Z. Li and A. K. Jain. Introduction. In S. Z. Li and A. K. Jain, editors, *Handbook of Face Recognition*, chapter 1, pages 1–11. Springer, 2004.
- [15] R. A. Lotufo, A. X. Falcao, and F. A. Zampiroli. Fast euclidean distance transform using a graph-search algorithm. In *Proceedings of the 13th Brazilian Symposium on Computer Graphics and Image Processing*, pages 269–275, Washington, DC, USA, 2000. IEEE Computer Society.
- [16] M. E. H. Pedersen. *SwarmOps: Black-Box Optimization in ANSI C*, July 2008.
- [17] M. E. H. Pedersen. *Tuning & Simplifying Heuristical Optimization*. PhD thesis, University of Southampton, December 2010.
- [18] Ronald L. Rardin and Reha Uzsoy. Experimental evaluation of heuristic optimization algorithms: A tutorial. *Journal of Heuristics*, 7(3):261–304, 2001.
- [19] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi. Gsa: A gravitational search algorithm. *Inf. Sci.*, 179(13):2232–2248, 2009.
- [20] S. K. Smit and A. E. Eiben. Comparing parameter tuning methods for evolutionary algorithms. In *Proc. of 11th Conference on Congress on Evolutionary Computation, CEC’09*, pages 399–406. IEEE Press, 2009.
- [21] R. Storn and K. Price. Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.
- [22] W. Sun and Y. Dong. Study of multiscale global optimization based on parameter space partition. *Journal of Global Optimization*, pages 149–172, 2011.

- [23] Virginia Torczon. On the convergence of pattern search algorithms. *SIAM J. on Optimization*, 7:1–25, January 1997.
- [24] David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.
- [25] X. Yao, Y. Liu, and G. Lin. Evolutionary programming made faster. *Evolutionary Computation, IEEE Transactions on*, 3(2):82–102, 1999.
- [26] B. Zitov and J. Flusser. Image registration methods: a survey. *Image and Vision Computing*, 21:977–1000, 2003.