Computational investigation of simple memetic approaches for continuous global optimization

Federico Cabassi

Dipartimento di Ingegneria dell'Informazione, Università di Parma
Via G.P. Usberti, 181/A, 43124 Parma, Italy
e-mail: federico.cabassi@studenti.unipr.it

Marco Locatelli

Dipartimento di Ingegneria dell'Informazione, Università di Parma
Via G.P. Usberti, 181/A, 43124 Parma, Italy
e-mail: marco.locatelli@unipr.it

Abstract

Keywords: Global optimization, memetic approaches, funnel landscapes

1 Introduction

The task of globally minimizing a multimodal objective function f over some compact domain $X \subset \mathbb{R}^n$ is a very difficult one. Different approaches exist based on the dimension of the search space and on the objective function and feasible domain's properties. The approaches range from exact (usually branch-and-bound) ones, suitable for highly structured problems (e.g., with a quadratic objective function and a polyhedral feasible domain) when the dimension n is not too large (say, few hundreds of variables), to heuristic ones where only few function evaluations are performed, suitable for problems where a single function evaluation is a rather costly operation. While we refer to [9] for a thorough discussion about all possible different approaches, here we focus our attention on approaches which are suitable for problems where local searches are a relatively cheap task but at the same time the huge number of local minimizers rules out the simplest approach based on multiple local searches, namely Multistart, where local searches are performed from different points randomly generated within the feasible region. In the algorithms we are going to discuss the points observed at each iteration are always local minimizers. To be more precise, the observed points are always the output of local search procedures, which are usually guaranteed to be stationary points but, in fact, are typically also local minimizers. In what follows we will always refer to these points as local minimizers, keeping in mind the clarification we have just made. Of course, observing local minimizers has a cost, since for each observation we need to perform a local search, which requires some function (and

gradient) evaluations. However, this cost is often largely compensated. Indeed, when the landscape of the objective function is rough with high barriers between local minimizers, even points very close to the global minimizer may have large function values. If, as it is often the case, the function value is employed to evaluate the quality of a point and to decide whether to keep or to discard it, such points, though close to the global minimizer, will be discarded from further consideration. Local searches, by driving a point towards a local minimizer, remove the negative effect of high barriers between local minimizers.

The paper is structured as follows. In Section 2 we introduce a general scheme of a global optimization approach based on local searches, and we briefly discuss two existing approaches proposed in the literature together with their strengths and weaknesses. In Section 3 we propose three simple variants of one of the two approaches. In Section 4 we present the set of test problems on which we compare the different variants, and we discuss the results of the computational experiments. We also perform an experimental analysis of the proposed approaches in order to better understand their behavior.

2 Global optimization based on local searches

A general scheme for a global optimization approach based on local searches is displayed in Algorithm 1. In this scheme:

Algorithm 1: Generic model for a global optimization algorithm based on local searches.

Data: objective function $f : \mathbb{R}^n \to \mathbb{R}$; feasible domain X; local solver $\mathcal{L}(f, X, \mathbf{x})$ where $\mathbf{x} \in X$; parameter vector $\mathbf{v}$.
$\mathbf{P} \leftarrow$ *GenerateStartingPoints*$(f, X, \mathcal{L}, \mathbf{v})$;
while *TerminationCriteria*$(\mathbf{P}, f, X, \mathbf{v}) =$ *false* **do**
  for $i \in \{1, \ldots, k\}$ **do**
    $\mathbf{Q}_i \leftarrow$ *Generation*$(f, X, \mathcal{L}, \mathbf{P}, \mathbf{v}, i)$;
    $\mathbf{P} \leftarrow$ *Selection*$(\mathbf{P}, \mathbf{Q}_i, \mathbf{v}, i)$;
    $\mathbf{v} \leftarrow$ *UpdateParameters*$(f, X, \mathbf{P}, \mathbf{v})$;
  end
end

- $\mathbf{P} \in \mathbb{R}^{n \times k} = \{\mathbf{p}_1, \ldots, \mathbf{p}_k\}$ is the *population matrix* containing the k population members as column vectors;
- $\mathbf{Q}_i \in \mathbb{R}^{n \times h} = \{\mathbf{q}_1, \ldots, \mathbf{q}_h\}$ is the *candidate matrix* containing the h candidate points generated by the i-th member of the current population;
- *GenerateStartingPoints*$(f, X, \mathcal{L}, \mathbf{v})$ is the procedure returning the initial population;

- *TerminationCriteria*$(\mathbf{P}, f, X, \mathbf{v})$ checks the termination criteria of the algorithm;
- *Generation*$(f, X, \mathcal{L}, \mathbf{P}, \mathbf{v}, i)$ is the procedure that generates the set of candidate points $\mathbf{Q}_i$;
- *Selection*$(\mathbf{P}, \mathbf{Q}_i, \mathbf{v}, i)$ is the procedure that performs a selection between the candidate points and the current population;
- *UpdateParameters*$(f, X, \mathbf{P}, \mathbf{v})$ is a procedure to update the parameter vector.

Algorithm 1 encompasses many different approaches. These can be classified into two broad categories, depending on the cardinality k of the population. If $k = 1$, then the algorithm is called a *single-track* one, while if $k > 1$ the algorithm is a *population-based* one. In what follows we will fix the two procedures *GenerateStartingPoints* and *TerminationCriteria*. Both are defined in a rather standard way. The former is described in Algorithm 2 and simply returns k local minimizers obtained by running local searches from k points randomly generated over X ($\mathcal{U}(X)$ is a uniform generator over the set X). The latter is

Algorithm 2: The procedure to generate the initial population

Data: f, X and $\mathcal{L}$.
Result: The matrix $\mathbf{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_k\}$ containing randomly generated local minimizers of f over X.
foreach $i \in \{1, \ldots, k\}$ **do**
  $\mathbf{p}_i \leftarrow \mathcal{L}(f, X, \mathcal{U}(X))$;
end

described in Algorithm 3. Different termination criteria come into play. The first termination condition is $NoImp \geq MaxNoImprove$, i.e., we stop if for $MaxNoImprove$ iterations we are unable to improve the best local minimizer. The second termination condition is $(f_{var} \simeq 0 \wedge k > 1)$, i.e., for population-based approaches $(k > 1)$ we stop when we have no variability in the function values of the population members, which, in particular, occurs when the population collapses into a single point. The last termination condition $f(\mathbf{p}_{min}) \leq BestValue$ stops the algorithm when the global minimum value $BestValue$ is reached. Such criterion is, of course, not suitable for real optimization problems where the global minimum value is not known in advance, but is added here to evaluate for each analyzed approach its ability of reaching the global minimizer (which is known for the test problems we employed). With respect to the general scheme, we will also fix $h = 1$, i.e., each population member $\mathbf{p}_i$ will generate a single candidate point $\mathbf{q}_i$.

In what follows we will briefly revise some approaches which can be seen as special cases of the general Algorithm 1, recalling that we fix the procedures *GenerateStartingPoints* and *TerminationCriteria* as described above, and we set $h = 1$. A very simple single-track approach is Multistart, where:

Algorithm 3: The procedure which checks the termination criteria

Data: f, X; the current population $\mathbf{P}$; f_{old} the minimum value of f over the previous population.

Result: $v \in \{\text{true}, \text{false}\}$ that specifies if the termination conditions are fulfilled or not.

$\mathbf{p}_{min} \leftarrow \mathbf{p}^* : f(\mathbf{p}^*) \leq f(\mathbf{p}),\ \forall \mathbf{p} \in \mathbf{P}$;

if $f_{old} \leq f(\mathbf{p}_{min})$ **then**

  $NoImp \leftarrow NoImp + 1$;

else

  $f_{old} \leftarrow f(\mathbf{p}_{min})$;

  $NoImp \leftarrow 0$;

end

$f_{var} \leftarrow \sum_{i,j=1}^{k} |f(\mathbf{p}_i) - f(\mathbf{p}_j)|$;

if $NoImp \geq MaxNoImprove \vee (f_{var} \simeq 0 \wedge k > 1) \vee f(\mathbf{p}_{min}) \leq BestValue$ **then**

  $v = \text{true}$;

else

  $v = \text{false}$;

end

- $k = 1$;
- *Generation* is the uniform random generation over X;
- *Selection* selects the new candidate only if its function value is the best one observed during the execution of the algorithm;
- *UpdateParameters* does nothing (in fact, Multistart has no parameter).

Multistart is doomed to failure for highly multimodal problems. The main reason for its bad behavior is that each new candidate point is generated by ignoring the history of the algorithm, i.e., the outcome of previous local searches. A slight, but quite effective, variant of the Multistart scheme is Monotonic Basin Hopping, MBH in what follows (see [7, 17]). MBH is also a single-track approach (in fact, a population-based version of MBH, called PBH, see [5], has been proposed and we will use its selection mechanism in one of the approaches discussed in the next section). The only difference with respect to Multistart lies in the *Generation* procedure. Indeed, this procedure returns a point generated over a neighborhood $\mathcal{N}$ of the current one $\mathbf{p}_1$. The neighborhood should be small with respect to the feasible region X, otherwise MBH becomes equivalent to Multistart. In the basic MBH approach the neighborhood is defined as follows: generate a uniform random point $\mathbf{z}$ (*perturbed point*) over a box, centered at $\mathbf{p}_1$ and with edge length δ, and set $\mathbf{q}_1 = \mathcal{L}(f, X, \mathbf{z})$. In spite of its simplicity, MBH turned out to be very efficient, especially for some hard global optimization problems, such as molecular conformation ones with Lennard-Jones and Morse potentials, see [7, 17], and packing problems [2]. However, in [11] many weaknesses of

this approach have been outlined and are briefly recalled here. One weakness is intrinsic in the approach, no matter how we choose the neighborhood $\mathcal{N}$. Since MBH always explores the neighborhood of the current local minimizer, it performs a local search over the graph of local minimizers defined by the neighborhood itself. Local minimizers of the objective function over this graph are called funnel bottoms and always include the global minimizer of f over X. MBH usually performs very well when the number of funnel bottoms is low. In particular, it is suitable for single-funnel function landscapes, where the local minimizers define monotone sequences of local minimizers all converging to the unique funnel bottom, which is also the global minimizer. The performance of MBH degrades as the number of funnel bottoms increases. But some difficulties for MBH may arise even over single-funnel landscapes. In particular, the definition of the box over which the perturbed point $\mathbf{z}$ is generated (i.e., the definition of the neighborhood $\mathcal{N}$) may not be a trivial task. In the original MBH approach for molecular conformation problems, a fixed size of the box was identified. However, in other global optimization problems the following difficulties may arise:

- a suitable value δ for the edge length may depend on the subregion of the feasible domain where the current point lies: thus, a fixed value is not the best option but some adaptive rule should be employed;
- the behavior of the objective function may be different with respect to different variables, so that one should use different edge lengths δ_i, $i = 1, \ldots, n$.

In other words, the procedure *UpdateParameters*, which plays no role in the basic MBH approach, should be employed to adaptively update the values δ_i, which are the main parameters of the MBH approach. However, the definition of this procedure is not a trivial task. Thus, some mechanism to overcome these difficulties is needed. A step in this direction has been done with the Memetic Differential Evolution (MDE in what follows) approach (see [10]). This approach is a rather simple Differential Evolution approach (see, e.g., [13, 15]), where a local search is applied to each newly generated point. In MDE the procedure *UpdateParameters* does nothing (MDE has some parameters but these are all fixed in advance). Procedure *Generation* for MDE is reported in Algorithm 4. A new point $\mathbf{y}_i$ is generated with the DE approach, where the parameter CR express the crossover probability, and the parameter F is an amplification coefficient. In fact, we fixed the CR value to 1 (no crossover is performed), and the value of F to 0.5. After the generation of the new point $\mathbf{y}_i$, a local descent is started from it through the local solver $\mathcal{L}$, thus delivering the new candidate $\mathbf{q}_i$. The *Selection* procedure (Algorithm 5) is the same as in DE: the candidate local minimizer $\mathbf{q}_i$ is compared with the current population member $\mathbf{p}_i$ and if its function value is better, $\mathbf{p}_i$ is replaced by $\mathbf{q}_i$. Although this is a simple approach, it proved to be very efficient both over test problems and over packing problems (see [10]). In that paper it is observed that MDE is able to overcome, at least partially, the weaknesses of MBH. Indeed, the generation mechanism can be seen as a perturbation of the population member $\mathbf{p}_{d_1}$ along a search direction which is not randomly generated but is defined by two other members of the population, namely $\mathbf{p}_{d_2}$ and $\mathbf{p}_{d_3}$. This way, the

Algorithm 4: The generation procedure for MDE.

Data: $\mathbf{P}$, the population matrix; i, the index of the evaluated point; $F \in (0,2)$, a real constant; $CR \in (0,1)$, a probability threshold.
Result: $\mathbf{q}_i$, the candidate vector.
Randomly choose $d_1, d_2, d_3 \in \{1,\dots,k\}\backslash\{i\}$ all different;
foreach $j \in \{1,\dots,n\}$ **do**
  if $\mathcal{U}(0,1) \leq CR$ **then** $\mathbf{y}_i^{(j)} \leftarrow \mathbf{p}_{d_1}^{(j)} + F(\mathbf{p}_{d_2}^{(j)} - \mathbf{p}_{d_3}^{(j)})$;
  else $\mathbf{y}_i^{(j)} \leftarrow \mathbf{p}_i^{(j)}$;
end
$\mathbf{q}_i \leftarrow \mathcal{L}(f, X, \mathbf{y}_i)$;

Algorithm 5: The selection procedure for MDE.

Data: $\mathbf{P}$, the population matrix; i, the index of the evaluated point; $\mathbf{q}_i$, the candidate vector.
Result: $\mathbf{P}$ as the new population matrix.
if $f(\mathbf{q}_i) < f(\mathbf{p}_i)$ **then**
  $\mathbf{p}_i \leftarrow \mathbf{q}_i$;
end

previously discussed difficulty of defining a proper way to adaptively update the values δ_i, $i = 1,\dots,n$, is overcome in MDE by the collaboration between members of the population. Besides that, MDE performs better than MBH over multi-funnel landscapes. We refer to [10] for a more thorough discussion about MDE. In the next section we will propose some simple variants of MDE, which will allow us to improve the performance.

3 Some variants of MDE

In this section we discuss three simple variants of MDE, namely the *greedy* variant (G-MDE), the *distance* variant (D-MDE), and the *hybrid* variant (H-MDE). We emphasize that we choose not to investigate any sophisticated variant of the basic MDE scheme. Our aim in this paper is to show that even simple and easy to implement approaches are able to return very good results.

3.1 Greedy MDE

As commented in Section 2, in the classical MBH approach the perturbed point is randomly generated within a box centered at the current local minimizer. In G-MDE we build a discrete distribution function for the perturbed point by exploiting the knowledge of the whole current population and through a mechanism which resembles the MDE generation procedure. Thus, G-MDE can be seen as a variant both of MDE and of MBH. In particular,

the discrete distribution is built as follows. In order to generate a perturbation for the population member $\mathbf{p}_i$, we randomly select another element of the population $\mathbf{p}_j$. Next, we try to predict the function value of the perturbed point by computing the difference between the objective functions at $\mathbf{p}_i$ and $\mathbf{p}_j$:

$$D(\mathbf{p}_i, \mathbf{p}_j) = f(\mathbf{p}_i) - f(\mathbf{p}_j).$$

If $D(\mathbf{p}_i, \mathbf{p}_j) > 0$, then we may assume that a step along the direction $\mathbf{p}_j - \mathbf{p}_i$ will lead to a function value better than $f(\mathbf{p}_i)$. Otherwise, we reverse the direction. Therefore, the perturbed point $\mathbf{y}_i$ is uniformly randomly generated over the discrete set

$$\{\mathbf{p}_i + \phi_j F(\mathbf{p}_j - \mathbf{p}_i)\} \quad \text{where } \forall j \ \phi_j = \text{sgn}(D(\mathbf{p}_i, \mathbf{p}_j)). \tag{1}$$

This generation procedure, which is summarized in Algorithm 6, is called greedy because it favors directions which are (presumably) descent ones (once again, we fixed $F = 0.5$ and $CR = 1$). In the G-MDE generation procedure the perturbations are applied to

Algorithm 6: The generation procedure for G-MDE.

Data: $\mathbf{P}$ is the population matrix; i, the index of the evaluated point; $F \in (0, 2)$, a real constant; $CR \in (0, 1)$, a probability threshold.
Result: $\mathbf{q}_i$, the candidate vector.
Randomly choose $r \in \{1, \ldots, k\} \backslash \{i\}$;
if $f(\mathbf{p}_i) > f(\mathbf{p}_r)$ **then** $\phi = 1$;
else $\phi = -1$;
foreach $j \in \{1, \ldots, n\}$ **do**
  if $\mathcal{U}(0, 1) \leq CR$ **then**
    $\mathbf{y}_i^{(j)} \leftarrow \mathbf{p}_i^{(j)} + \phi F(\mathbf{p}_r^{(j)} - \mathbf{p}_i^{(j)})$;
  else
    $\mathbf{y}_i^{(j)} \leftarrow \mathbf{p}_i^{(j)}$;
  end
end
$\mathbf{q}_i \leftarrow \mathcal{L}(f, X, \mathbf{y}_i)$;

each population member $\mathbf{p}_i$ and the result of the perturbation is compared with $\mathbf{p}_i$ itself as in MBH. However, the shape of the neighborhood and the step sizes are defined as in MDE.

Whereas such greedy procedure can be very effective for single-funnel functions, it may be less effective for multi-funnel functions, where a too fast convergence to a funnel bottom may be a negative feature. For this reason, in the next subsection we introduce a mechanism which counterbalances the greedy moves in G-MDE.

3.2 The Distance MDE

In the field of molecular conformation problems it has been observed that keeping some diversity within the population may be very beneficial (see, e.g., [6, 8, 12]). In [5] a

population-based version of MBH, called PBH, has been introduced. PBH counterbalances the greedy nature of MBH and prevents too fast convergence through a cooperation between the population members when selection is performed. The cooperation aims at maintaining some diversity between the members of the population. The PBH mechanism can be extended to G-MDE. The generation procedure is still Algorithm 6, but the selection procedure is the one described in Algorithm 7. The procedure includes a distance measure between local minimizers. The distance measure $d(\cdot,\cdot)$ we have employed is based on

Algorithm 7: The selection procedure for D-MDE.

Data: $\mathbf{P}$ is the population matrix; i, the index of the evaluated point; $\mathbf{q}_i$ is the candidate point.
Result: The new population matrix $\mathbf{P}$.
$\mathbf{p}_{near} \leftarrow \mathbf{p}^* \in \mathbf{P} : d(\mathbf{q}_i, \mathbf{p}^*) \leq d(\mathbf{q}_i, \mathbf{p}),\ \forall \mathbf{p} \in \mathbf{P}$;
if $f(\mathbf{q}_i) < f(\mathbf{p}_{near})$ **then**
  $\mathbf{p}_{near} \leftarrow \mathbf{q}_i$;
end

function values, i.e.,

$$d(\mathbf{s}, \mathbf{u}) = |f(\mathbf{s}) - f(\mathbf{u})|.$$

However, other distance measures can be defined, including some problem-specific ones (different examples for molecular conformation problems are reported and analyzed in [3]).

Differently from G-MDE, the selection mechanism is not applied any more between the current population element $\mathbf{p}_i$ and the corresponding generated candidate $\mathbf{q}_i$, but between the member of the population nearest to the newly generated candidate $\mathbf{q}_i$ and the candidate itself, i.e., $\mathbf{q}_i$ is compared with

$$\mathbf{p}^* \in \mathbf{P} : d(\mathbf{q}_i, \mathbf{p}^*) \leq d(\mathbf{q}_i, \mathbf{p}),\ \forall \mathbf{p} \in \mathbf{P}.$$

Thus, the population members not only cooperate for the generation of candidate points as in G-MDE, but also in the selection phase. With this selection procedure it is possible to have a population more spread around the feasible region, since, e.g., there cannot be different copies of the same point within the population. Moreover, candidate points with a worse function value with respect to the population members from which they have been generated may enter the population, and population members which generate candidate points with a better function value may survive within the population. Both these events can not occur in G-MDE, and they may reduce the greediness of the approach, thus causing on the one hand a slower convergence of the algorithm but on the other hand an increase of the search area explored by the algorithm.

3.3 Hybrid MDE

In the two previous subsections we have presented two different mechanisms which appear to be effective in different situations. G-MDE converges fast and appears to be suitable

for single-funnel functions, while D-DME converges more slowly but appears to be more effective for multi-funnel functions. The difference between the two approaches only lies in the selection mechanism. If we have no idea about the properties of the function we want to optimize, rather than choosing one of the two previous approaches, it is quite natural to employ a hybrid approach, which mixes the greedy and distance selection procedures. This way we may expect to have reasonably good results (though maybe not the best ones) over all functions, without making use of any prior knowledge about them. Following the principle of using only simple variants of the proposed approaches, we ended up with the hybrid approach described in Algorithm 8. The generation of the candidate point is always the greedy one. If $D(\mathbf{p}_i, \mathbf{p}_j) > 0$, i.e., $\phi = 1$, we also use the G-MDE selection mechanism, thus forcing the greedy behavior. But if $D(\mathbf{p}_i, \mathbf{p}_j) < 0$, i.e., $\phi = -1$, the D-MDE selection mechanism is employed. This way $\mathbf{p}_i$ may survive within the new population even if a better candidate point is generated, since during the selection phase the candidate is not compared with $\mathbf{p}_i$ but with some other member of the population. This should avoid too fast convergence of the population towards regions which do not contain the global minimizer. Of course, other rules to decide whether to employ the greedy or the distance selection procedure could be explored, but the proposed one already delivers satisfying results.

Algorithm 8: Generation and Selection in the Hybrid MDE approach.

Data: f, X; local solver $\mathcal{L}$; current population $\mathbf{P}$.
$\mathbf{q}_i, \phi \leftarrow$ *GMDEGeneration*$(f, \mathcal{L}, \mathbf{P}, i)$;
if $\phi = -1$ **then**
  $\mathbf{P} \leftarrow$ *DMDESelection*$(\mathbf{P}, \mathbf{q}_i, i)$;
else
  $\mathbf{P} \leftarrow$ *MDESelection*$(\mathbf{P}, \mathbf{q}_i, i)$;
end

4 Computational experiments

In this section we make a detailed comparison of the three variants of MDE with MDE itself. We recall that MDE, in spite of its simplicity, already proved to be quite effective (see [10]), outperforming or, at least, being competitive with MBH. We will first introduce the set of test problems over which the comparison will be performed. These are modifications of some well known highly multimodal global optimization test problems (Subsection 4.1). Next, we will present and discuss the tables with all the results (Subsection 4.2) . Finally, we will present some experiments which aim at a better understanding of the algorithms' behavior (Subsection 4.3).

4.1 Test problems

For the computational experiments we used three classical highly multimodal test functions and some of their variants. The functions are (see [1, 14, 16]):

- Rastrigin function:

$$f_1(\mathbf{x}) = 10n + \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i)), \quad \mathbf{x} \in [-5.12, 5.12]^n,$$

  whose global minimizer is $\mathbf{x}^* = \mathbf{0}$ and the global minimum value is 0.

- Ackley function:

$$f_2(\mathbf{x}) = 20 + e - 20\exp\left(-0.2\sqrt{\tfrac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \\ - \exp\left(\tfrac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right), \quad \mathbf{x} \in [-32.768, 32.768]^n,$$

  whose global minimizer is $\mathbf{x}^* = \mathbf{0}$ and the global minimum value is 0.

- Schwefel function:

$$f_3(\mathbf{x}) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{|x_i|}\right), \quad \mathbf{x} \in [-500, 500]^n,$$

  whose global minimizer is $\mathbf{x}^* = (420.9687, \ldots, 420.9687)$ and the global minimum value is $-418.9829n$.

Two more test functions (the Levy and Sinusoidal one) have been discarded because they turned out to be not challenging enough (as already observed in [10]). The Rastrigin function has a huge number (10^n) of local minimizers, but the function is a single-funnel one and the local minimizers are uniformly distributed within the feasible set. In particular, the distance between nearest local minimizers is always approximately equal to one. The Ackley function is also highly multimodal and single-funnel. While the barriers between local minimizers are lower with respect to the Rastrigin function, the nearest distance between local minimizers is not constant throughout the feasible set, being large far from the global minimizer and lower close to the global minimizer. The Schwefel function is multimodal but, for the same dimension n, it has a lower number of local minimizers with respect to the Rastrigin and Ackley functions. However, the Schwefel function is usually much more challenging with respect to the other two functions in view of its highly multi-funnel landscape.

While widely used in the global optimization literature, these functions have some properties which may simplify the detection of the global minimizers. The first simplifying property is *separability*. Both the Rastrigin and the Schwefel function are separable. This fact can be sometimes exploited by global optimization approaches. E.g., all approaches where a single or few variables are perturbed at some iteration or where the value of some

variables is fixed (like, e.g., in the crossover operation) are clearly favored when applied to separable functions. Another simplifying feature is the fact that the global minimizer lies at the center of the feasible set, which is the case both for the Rastrigin and for the Ackley function. Next, a simplifying feature is the fact that the behavior with respect to all variables is the same (any permutation of the variables leads to a solution with the same function value). Finally, the Rastrigin and Ackley function are symmetric with respect to the origin. In the literature, variants of the basic functions are proposed where high multi-modality is maintained, the funnel properties are also maintained (i.e., single-funnel functions are still single-funnel, and the same for multi-funnel ones), but the simplifying features are removed. The variants we considered are the following:

$$f_i(\mathbf{DW}(\mathbf{x} - \bar{\mathbf{x}})), \quad i = 1, \ldots, 3,$$

where

- $\mathbf{D}$ is a diagonal matrix of order n with positive diagonal elements;
- $\mathbf{W}$ is an orthonormal matrix of order n;
- $\bar{\mathbf{x}}$ is a n-dimensional shift vector.

The basic versions of the functions are obtained when

$$\mathbf{D} = \mathbf{W} = \mathbf{I}, \quad \bar{\mathbf{x}} = \mathbf{0},$$

where $\mathbf{I}$ is the identity matrix of order n. The orthonormal transformation eliminates separability, and is applied to all three functions. The diagonal transformation removes the symmetry of the problem with respect to permutation of the variables. This is only applied to the Rastrigin function. The shift vector moves the global minimizer far away from the center of the feasible set. It is not applied to the Schwefel function since the global minimizer of this function is not at the center of the feasible set. For what concerns the Rastrigin function we have also applied the following nonlinear transformation (see [4]) to each component of the argument vector $\mathbf{z} = \mathbf{DW}(\mathbf{x} - \bar{\mathbf{x}})$

$$g(z_i) = \begin{cases} z_i & \text{if } z_i \leq 0 \\ z_i^{\left(1+0.2\frac{i-1}{n-1}\sqrt{z_i}\right)} & \text{otherwise.} \end{cases}$$

This allows to eliminate the symmetry with respect to the global minimizer. Note that when we apply the orthonormal transformation, also the search space is rotated, i.e., the feasible region becomes the polytope

$$X = \{\mathbf{x} \ : \ \mathbf{l}_j \leq \mathbf{W}\mathbf{x} \leq \mathbf{u}_j\}, \quad j = 1, \ldots, 3,$$

where $\mathbf{l}_j, \mathbf{u}_j$ denote the lower and upper bounds for the box constraints of the basic function f_j, $j = 1, \ldots, 3$. The shift vector $\bar{\mathbf{x}} \in X$ is generated by first randomly drawing a vector $\mathbf{z} \in [\mathbf{l}_j, \mathbf{u}_j]$, and then by setting $\bar{\mathbf{x}} = \mathbf{W}^T\mathbf{z}$.

4.2 Results

In this subsection we present all the results we collected. But before describing the experiments we performed and commenting the results we obtained, we make some remarks about the parameters appearing in MDE and all its variants. Most of them have been fixed to a value (the same for all the tests we performed). The $MaxNoImprove$ parameter is used in the termination criterion and throughout the experiments has been fixed to 100. The DE parameters F and CR, as already mentioned before, have been fixed to 0.5 and 1, respectively. The only parameter which has not been fixed is the size k of the population. As typical for population-based approaches, the size of the population is a key parameter. Too small a population may cause a too fast convergence of the algorithm, while too large a population may cause a too large computational effort per iteration. Unfortunately, the best size of the population is strictly problem dependent. In our experiments we observed that for the Ackley function a small population size is enough, for the Rastrigin function a larger size is needed, while the Schwefel function requires the largest size. The detailed values of these sizes will be given later on. Here we only remark that an issue which is worthwhile to investigate in future works is the identification of some adaptive rule for the definition of the population size, allowing both for the removal and for the addition of population members.
While we have now commented all the numerical parameters of MDE and its variants, there is another non-numerical parameter which may have an impact on the performance of the different approaches. Indeed, one can use different local solvers $\mathcal{L}$. The results with different solvers may considerably differ from each other, although often (but not always) the relative performance of the different approaches is similar with the different solvers. For this reason we tested different solvers.

We are now ready to describe the experiments. We used three different local solvers: `SNOPT`, `MINOS`, and the `fmincon` solver through `Matlab`. We performed tests with dimensions $n = 10$ and $n = 50$ with `SNOPT`, `MINOS`, while, due to the very large computing times in `Matlab`, we tested dimensions $n = 10$ and $n = 30$ with `fmincon`. For the Rastrigin function we tested all possible versions (separable, rotated, rotated and shifted, rotated and shifted and scaled). With `SNOPT` we also tested the non symmetric version. For the Ackley function we tested the separable, the rotated, and the rotated and shifted versions. For the Schwefel function we tested the separable and rotated version. As previously commented, we used different population sizes not only for different dimensions but also for different functions. For the Rastrigin function, we set $k = 10$ for $n = 10$, and $k = 40$ for $n = 50$ ($k = 20$ for $n = 30$ in the tests with `fmincon`). For the Ackley function we set $k = 10$ for $n = 10$ and $k = 20$ for $n = 50$ ($k = 20$ also for $n = 30$ in the tests with `fmincon`). For the Schwefel function we set $k = 40$ for $n = 10$ and $k = 100$ for $n = 50$ ($k = 100$ also for $n = 30$ in the tests with `fmincon`).

The following columns appear in each table:

- S_N: the number of successes over N trials;

- LS: the average number of local searches;
- D: the average distance over the instances where a failure occurs, computed as $|f(\mathbf{x}^*) - f(\mathbf{x}^\dagger)|$, between the global minimum value attained at the global minimizer $\mathbf{x}^*$, and the best function value reached by the algorithm, attained at some point $\mathbf{x}^\dagger$ ($D = 0$ if no failure occurs).

We will make a separate comment for each function.

4.2.1 Ackley

Tables 1-3 report the results with the different versions of the 10-dimensional Ackley function with the `SNOPT`, `MINOS` and `fmincon` local solver, respectively. Tables 4-5 report the results with the different versions of the 50-dimensional Ackley function with the `SNOPT` and `MINOS` solvers. Finally, Table 6 reports the results with the different versions of the 30-dimensional Ackley function with the `fmincon` solver.

The results for the Ackley function are quite consistent through the different dimensions and solvers. As expected in view of the single-funnel nature of the function, G-MDE performs quite well with a very high percentage of successes and a low average number of local searches. H-DME performs very similarly to G-MDE, thus showing that in this case the hybrid mechanism is able to preserve the good performance of the greedy approach. Both G-MDE and H-MDE outperform the original MDE approach. D-MDE is a robust approach (no failure occurs) but with the largest number of local searches. That was expected: D-MDE converges more slowly than the other approaches.

Ackley10 - SNOPT									
Alg.	*Separable*			*Rot.*			*Rot + Shift*		
-	S_{100}	LS	D	S_{100}	LS	D	S_{100}	LS	D
MDE	96	196.2	1.6469	100	85.20	0	99	100.4	1.1556
G-MDE	99	147.8	2.0142	100	58.1	0	100	76.5	0
D-MDE	100	459.7	0	100	218.4	0	100	279.4	0
H-MDE	100	149.7	0	100	60.7	0	100	75.7	0

Table 1: Tests with `SNOPT` and a population of 10 points.

Ackley10 - MINOS									
Alg.	*Separable*			*Rot.*			*Rot + Shift*		
-	S_{100}	*LS*	*D*	S_{100}	*LS*	*D*	S_{100}	*LS*	*D*
MDE	99	176.1	12.9061	100	74.3	0	92	83.3	3.4259
G-MDE	95	101.7	5.3551	100	35.8	0	100	46.0	0
D-MDE	100	314.0	0	100	79.3	0	100	135.2	0
H-MDE	95	106.1	6.6433	100	35.3	0	100	46.0	0

Table 2: Tests with `MINOS` and a population of 10 points.

Ackley10 - fmincon									
Alg.	*Separable*			*Rot.*			*Rot + Shift*		
-	S_{50}	*LS*	*D*	S_{50}	*LS*	*D*	S_{50}	*LS*	*D*
MDE	50	59.8	0	50	54.4	0	50	70.36	0
G-MDE	50	36.8	0	50	34.9	0	50	52.32	0
D-MDE	50	114.6	0	50	82	0	50	121.56	0
H-MDE	50	36.0	0	50	33.8	0	50	50.84	0

Table 3: Tests with `fmincon` and a population of 10 points.

Ackley50 - SNOPT									
Alg.	*Separable*			*Rot.*			*Rot + Shift*		
-	S_{50}	*LS*	*D*	S_{50}	*LS*	*D*	S_{50}	*LS*	*D*
MDE	46	1374.4	19.2857	50	503.6	0	50	587.2	0
G-MDE	50	164.8	0	50	86.0	0	50	124.0	0
D-MDE	50	1040.0	0	50	460.1	0	50	706.8	0
H-MDE	50	158.0	0	50	86.4	0	50	126.8	0

Table 4: Tests with `SNOPT` and a population of 20 points.

Ackley50 - MINOS									
Alg.	*Separable*			*Rot.*			*Rot + Shift*		
-	S_{50}	LS	D	S_{50}	LS	D	S_{50}	LS	D
MDE	50	723.2	0	50	213.2	0	50	244.4	0
G-MDE	50	202.0	0	50	82.0	0	50	109.6	0
D-MDE	50	1381.6	0	50	299.2	0	50	425.6	0
H-MDE	50	194.8	0	50	77.6	0	50	110.4	0

Table 5: Tests with `MINOS` and a population of 20 points.

Ackley30 - fmincon									
Alg.	*Separable*			*Rot.*			*Rot + Shift*		
-	S_{10}	LS	D	S_{10}	LS	D	S_{10}	LS	D
MDE	10	189.6	0	10	157.4	0	10	219.2	0
G-MDE	10	81.6	0	10	76.7	0	10	161.7	0
D-MDE	10	396.2	0	10	282.6	0	10	604.1	0
H-MDE	10	75.0	0	10	71.4	0	10	127.5	0

Table 6: Tests with `fmincon` and a population of 20 points.

4.2.2 Rastrigin

Tables 7-10 report the results with the different versions of the 10-dimensional Rastrigin function with the `SNOPT`, `MINOS` and `fmincon` local solver, respectively (only `SNOPT` for the non-symmetric version). Tables 11-13 report the results with the different versions of the 50-dimensional Rastrigin function with the `SNOPT` and `MINOS` solvers (once again, only `SNOPT` for the non-symmetric version). Finally, Table 14 reports the results with the different versions of the 30-dimensional Rastrigin function with the `fmincon` solver.

The Rastrigin function is single-funnel. Thus, we expected a behavior similar to the Ackley function. In particular, we expected a good performance of G-MDE with respect to the other variants, the usual robustness (i.e., ability to reach the global minimizer) for D-MDE, although at the cost of a larger number of local searches, and a behavior of H-MDE close to that of G-MDE. With `SNOPT` this is almost always the case, although the performance over the Rotated+Shifted version and the Rotated+Shifted+Scaled version is not the best one for G-MDE and H-MDE. With `MINOS` G-MDE has not the best performance with the Rotated version (this is true in particular with $n = 10$) but the D value reveals that, when not reaching the global minimizer, G-MDE almost always stops (in fact, always for $n = 10$) at the second best local minimizer, whose function value is 0.9959. With `fmincon` G-MDE usually performs very well, the only execption being the Rotated+Shifted+Scaled version, where, however, G-MDE outperforms MDE. It is remarkable that in this case H-MDE

represents a very good mix of the two strategies. Its performance is always very close to (and sometimes better than) the best of the other approaches.

Rastrigin10 - SNOPT						
Alg.	*Separable*			*Rot.*		
-	S_{100}	LS	D	S_{100}	LS	D
MDE	87	80.4	2.2983	73	94.4	2.2501
G-MDE	100	48.5	0	99	62.1	0.9959
D-MDE	100	182.1	0	100	203.4	0
H-MDE	99	60	0.9959	98	73.3	0.9959
Alg.	*Rot. + Shift*			*Rot. + Shift + Scaled*		
-	S_{100}	LS	D	S_{100}	LS	D
MDE	59	133.8	4.1781	49	233.7	1.8552
G-MDE	80	307.2	1.5935	84	406.2	1.7429
D-MDE	100	369.2	0	95	784.7	0.9959
H-MDE	66	451.1	2.5777	78	422.6	1.3128

Table 7: Tests with `SNOPT` and a population of 10 points.

RastriginNonSym10 - SNOPT									
Alg.	*Separable*			*Rot.*			*Rot + Shift*		
-	S_{100}	LS	D	S_{100}	LS	D	S_{100}	LS	D
MDE	80	173.6	1.4939	82	199.7	2.4345	61	216.9	2.929
G-MDE	98	86.2	0.9959	100	77.9	0	96	136.8	0.9959
D-MDE	95	399.4	0.9959	96	423.6	0.9959	89	641.6	0.9963
H-MDE	100	115.0	0	99	87.5	0.9959	95	141.9	1.1951

Table 8: Tests with `SNOPT` and a population of 10 points.

Rastrigin10 - MINOS						
Alg.	*Separable*			*Rot.*		
-	S_{100}	*LS*	*D*	S_{100}	*LS*	*D*
MDE	98	66.5	0.9959	22	206.4	1.035
G-MDE	100	38.9	0	25	920.0	0.9959
D-MDE	100	133.6	0	89	573.8	0.9959
H-MDE	100	40.9	0	48	442.9	0.9959
Alg.	*Rot. + Shift*			*Rot. + Shift + Scaled*		
-	S_{100}	*LS*	*D*	S_{100}	*LS*	*D*
MDE	62	171.6	5.6481	60	190.5	6.5259
G-MDE	100	85.8	0	97	160.6	0.9959
D-MDE	92	520.2	6.7127	97	593.8	9.8023
H-MDE	97	123.2	3.1918	91	210.4	0.9959

Table 9: Tests with `MINOS` and a population of 10 points.

Rastrigin10 - fmincon						
Alg.	*Separable*			*Rot.*		
-	S_{50}	*LS*	*D*	S_{50}	*LS*	*D*
MDE	45	141	4.9747	37	184	4.1328
G-MDE	50	100.2	0	50	104.5	0
D-MDE	50	361	0	50	409	0
H-MDE	50	94.0	0	50	105.68	0
Alg.	*Rot. + Shift*			*Rot. + Shift + Scaled*		
-	S_{50}	*LS*	*D*	S_{50}	*LS*	*D*
MDE	27	183.5	5.3304	2	322.28	4.8437
G-MDE	39	296.46	5.9318	17	1041.8	1.7484
D-MDE	45	606.58	3.978	22	1503.6	1.2071
H-MDE	41	309.6	5.0288	23	873.8	1.4370

Table 10: Tests with `fmincon` and a population of 10 points.

Rastrigin50 - SNOPT						
Alg.	*Separable*			*Rot.*		
-	S_{50}	*LS*	*D*	S_{50}	*LS*	*D*
MDE	50	584.0	0	50	796.0	0
G-MDE	50	229.6	0	50	233.6	0
D-MDE	50	2265.6	0	50	2555.2	0
H-MDE	50	224.0	0	50	231.2	0
Alg.	*Rot. + Shift*			*Rot. + Shift + Scaled*		
-	S_{50}	*LS*	*D*	S_{50}	*LS*	*D*
MDE	50	1243.2	0	50	2241.6	0
G-MDE	43	1340.0	2.2757	45	1871.2	3.186
D-MDE	50	2601.6	0	50	7931.2	0
H-MDE	39	1805.6	2.8067	46	1993.6	0.9957

Table 11: Tests with `SNOPT` and a population of 40 points.

RastriginNonSym50 - SNOPT									
Alg.	*Separable*			*Rot.*			*Rot + Shift*		
-	S_{50}	*LS*	*D*	S_{50}	*LS*	*D*	S_{50}	*LS*	*D*
MDE	50	1570.4	0	50	1075.2	0	46	1462.4	0.995
G-MDE	50	412.0	0	50	275.2	0	32	2251.2	1.2172
D-MDE	50	5054.4	0	50	3490.4	0	50	5554.4	0
H-MDE	50	417.6	0	50	276.0	0	40	1428.0	1.294

Table 12: Tests with `SNOPT` and a population of 40 points.

Rastrigin50 - MINOS						
Alg.	*Separable*			*Rot.*		
-	S_{50}	*LS*	*D*	S_{50}	*LS*	*D*
MDE	50	469.6	0	12	2551.2	1.127
G-MDE	50	190.4	0	27	4761.6	1.0391
D-MDE	50	1676.8	0	48	5986.4	1.9915
H-MDE	50	196.8	0	49	1876.0	0.995
Alg.	*Rot. + Shift*			*Rot. + Shift + Scaled*		
-	S_{50}	*LS*	*D*	S_{50}	*LS*	*D*
MDE	50	1420.8	0	48	3267.2	6.6675
G-MDE	50	418.4	0	50	933.6	0
D-MDE	47	5308.0	2.3239	43	9740.0	18.665
H-MDE	50	424.0	0	50	893.6	0

Table 13: Tests with `MINOS` and a population of 40 points.

Rastrigin30 - fmincon						
Alg.	*Separable*			*Rot.*		
-	S_{10}	*LS*	*D*	S_{10}	*LS*	*D*
MDE	10	485.5	0	9	933.5	18.9042
G-MDE	10	382.8	0	10	357.8	0
D-MDE	10	2226.8	0	9	2626.5	12.934
H-MDE	10	396.7	0	10	385.5	0
Alg.	*Rot. + Shift*			*Rot. + Shift + Scaled*		
-	S_{10}	*LS*	*D*	S_{10}	*LS*	*D*
MDE	8	779.2	2.62	0	1643.0	8.85
G-MDE	9	709.1	1.95	2	3655.7	1.737
D-MDE	6	3168.8	7.31	9	4995.5	0.994
H-MDE	10	362.1	0	9	1528.7	0.994

Table 14: Tests with `fmincon` and a population of 20 points.

4.2.3 Schwefel

Tables 15-17 report the results with the different versions of the 10-dimensional Schwefel function with the `SNOPT`, `MINOS` and `fmincon` local solver, respectively. Tables 18-20 report the results with the different versions of the 50-dimensional Schwefel function with the `SNOPT` and `MINOS` solvers (note that with `MINOS` we reported the results also with a population of 200 members). Finally, Table 21 reports the results with the different versions of the 30-dimensional Schwefel function with the `fmincon` solver.

The results with the Schwefel function display some variability with respect to the local solver employed. What we expected was a very good performance of the D-MDE approach, which converges more slowly but explores a larger portion of the feasible region, a positive feature when dealing with the multi-funnel landscape of the Schwefel function. The experiments confirm the expectations: D-MDE is quite robust, with a very high percentage of successes. We also expected a bad performance of G-MDE, in view of its fast (too fast, in this case) convergence. That was the case with the solvers `MINOS` and `fmincon`. But with the solver `SNOPT` we had an unexpected very good behavior of G-MDE. A possible explanation is the following. The behavior of any local solver is not necessarily "local" in the sense that the local minimizer reached from a starting point is the closest one to this point. It may happen (and, as we will see in the next subsection, it often happens) that the final local minimizer is far away from the starting point. Thus, any local solver does explore a large portion of the feasible region and the region of attraction of the global minimizer may contain points quite far from it. This seems to be the case for the global minimizer of the Schwefel function when `SNOPT` is used, thus making the convergence to the global minimizer easier for G-MDE. The performance of H-DME lies, as expected, between the one of G-MDE and the one of D-MDE, and is very close to G-MDE with `SNOPT`. Note that with `MINOS`, where we had a small number of successes with the population size $k = 100$, we also tested the population size $k = 200$ (see Table 20). The larger population size allows for a larger number of successes, in particular with D-MDE, although also the average number of local searches increases.

Schwefel10 - SNOPT						
Alg.	*Separable*			*Rot.*		
-	S_{100}	LS	D	S_{100}	LS	D
MDE	34	850.4	198.1560	49	479.2	226.5803
G-MDE	99	930.0	117.609	100	442.4	0
D-MDE	100	949.2	0	100	982.4	0
H-MDE	97	752.8	157.089	100	563.2	0

Table 15: Tests with `SNOPT` and a population of 40 points.

Schwefel10 - MINOS						
Alg.	*Separable*			*Rot.*		
-	S_{100}	LS	D	S_{100}	LS	D
MDE	22	893.2	207.2756	12	932.8	233.3909
G-MDE	3	5681.6	277.9061	58	3045.2	154.2690
D-MDE	93	2963.6	134.5292	91	2985.2	331.6888
H-MDE	19	4689.2	228.8111	32	3984.8	192.5073

Table 16: Tests with `MINOS` and a population of 40 points.

Schwefel10 - fmincon						
Alg.	*Separable*			*Rot.*		
-	S_{50}	LS	D	S_{50}	LS	D
MDE	13	957.0	230.7617	16	1051.5	187.6588
G-MDE	5	7002.0	232.0848	5	5868.2	242.6464
D-MDE	40	1633.0	177.8098	49	2132.2	118,4
H-MDE	33	1529.4	160.4189	21	4468.24	179.8034

Table 17: Tests with `fmincon` and a population of 40 points.

Schwefel50 - SNOPT						
Alg.	*Separable*			*Rot.*		
-	S_{10}	LS	D	S_{10}	LS	D
MDE	0	14090.0	5588.28	7	5290.0	193.2530
G-MDE	10	3740.0	0	10	3610.0	0
D-MDE	10	32570.0	0	10	8360.0	0
H-MDE	10	3730.0	0	10	4620.0	0

Table 18: Tests with `SNOPT` and a population of 100 points.

Schwefel50 (100) - MINOS						
Alg.	*Separable*			*Rot.*		
-	S_{10}	*LS*	*D*	S_{10}	*LS*	*D*
MDE	0	6920.0	979.956	0	9300.0	1085.49
G-MDE	0	14290.0	2771.41	0	30710.0	651.216
D-MDE	5	37590.0	280.108	1	47960.0	245.8922
H-MDE	1	18870.0	1513.6222	0	17570.0	1097.33

Table 19: Tests with `MINOS` and a population of 100 points.

Schwefel50 (200) - MINOS						
Alg.	*Separable*			*Rot.*		
-	S_{10}	*LS*	*D*	S_{10}	*LS*	*D*
MDE	2	15960.0	291.9512	0	27220.0	635.423
G-MDE	0	27440.0	2795.86	1	50940.0	1062.1422
D-MDE	9	76380.0	1180.24	8	64080.0	589.565
H-MDE	0	40400.0	1670.39	2	35520.0	691.6812

Table 20: Tests with `MINOS` and a population of 200 points.

Schwefel30 - fmincon						
Alg.	*Separable*			*Rot.*		
-	S_{10}	*LS*	*D*	S_{10}	*LS*	*D*
MDE	2	5198.0	237.635	2	5941.5.0	207.6375
G-MDE	0	17603.0	1226.526	0	17610.0	1095.39
D-MDE	10	12306.8	0	10	16046.7	0
H-MDE	7	18865.4	476.7886	4	23623.1	417.05

Table 21: Tests with `fmincon` and a population of 100 points.

4.3 Some insight into the results

In this section we present a few experiments which have been made to gain some insight about the proposed approaches.

In [10] it was observed that MDE is able to identify the directions along which to move, and to self-adjust the step size along different directions. To confirm this fact, we analyzed

the covariance matrix of the population, defined as follows

$$\mathbf{C} = \frac{1}{k}\sum_{i=1}^{k}(\mathbf{p}_i - \sigma)(\mathbf{p}_i - \sigma)^T,$$

where σ is the average of the population, i.e.,

$$\sigma = \sum_{i=1}^{k}\frac{\mathbf{p}_i}{k}.$$

Since the covariance matrix $\mathbf{C}$ is a positive definite matrix we can decompose it as follows

$$\mathbf{C} = \mathbf{SHS}^T, \tag{2}$$

where $\mathbf{S}$ is an orthonormal matrix and $\mathbf{H}$ is a diagonal matrix with positive diagonal entries. We can use the covariance matrix to define an hyper-ellipsoid with equation $\mathbf{x}^T\mathbf{Cx} = 1$. Thus in the decomposition in (2) the diagonal entries of matrix $\mathbf{H}$ are the squared length of the principal axes of the hyper-ellipsoid, while the columns of matrix $\mathbf{S}$ are the directions of the axes of the hyper-ellipsoid. MDE and its variants will have the tendency to generate steps along the directions of the axes, with a step length along these directions related to the diagonal entries of $\mathbf{H}$. Figure 1 shows the curves of the (ordered) diagonal entries of $\mathbf{H}$ over an instance with the Rastrigin function. It is worthwhile to remark two facts. The first one is that during the first iterations the diagonal entries may considerably differ from each other, thus allowing for different step sizes along different directions. The second fact is that as the iteration counter increases, the step-size along each direction is decreasing (the algorithm is approaching the unique funnel bottom of the function). With the Schwefel function the situation is different. The curves are not decreasing ones. They sometimes decrease and some other times increase. That happens because the global minimizer and the other low-level local minimizers are not close to each other as in the Rastrigin case. Figure 2a displays a case where the global minimizer could finally be reached, while Figure 2b displays a case where the global minimizer was not reached: after some iterations, the population was frozen and the covariance matrix did not change any more, thus causing the curves to become constant ones. This irregular behavior is a good feature with respect to the Schwefel function. We need to avoid too fast convergence and we need to be able to perform large steps.

The second experiment we performed with the greedy generation procedure is the following. At each iteration we computed the quantity (we refer to Algorithm 6 for the notation)

$$\frac{1}{k}\sum_{i=1}^{k}\|\mathbf{p}_i - \mathbf{y}_i\|.$$

This is the average distance between the population members and the corresponding perturbations (before a local search is applied to them). The corresponding curve is quite

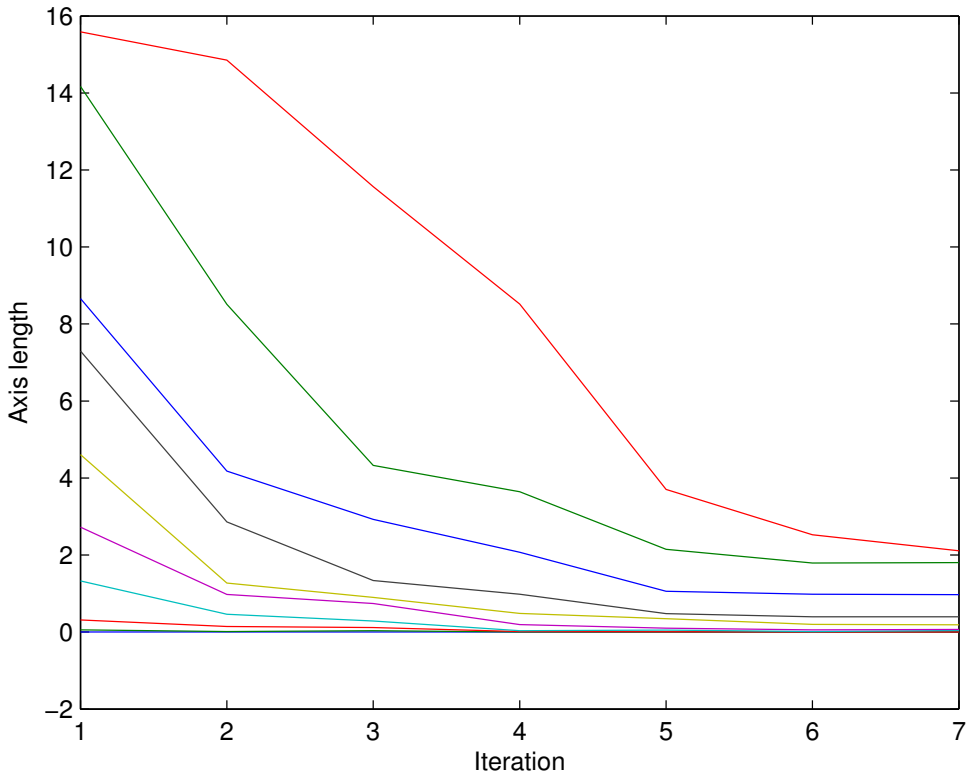
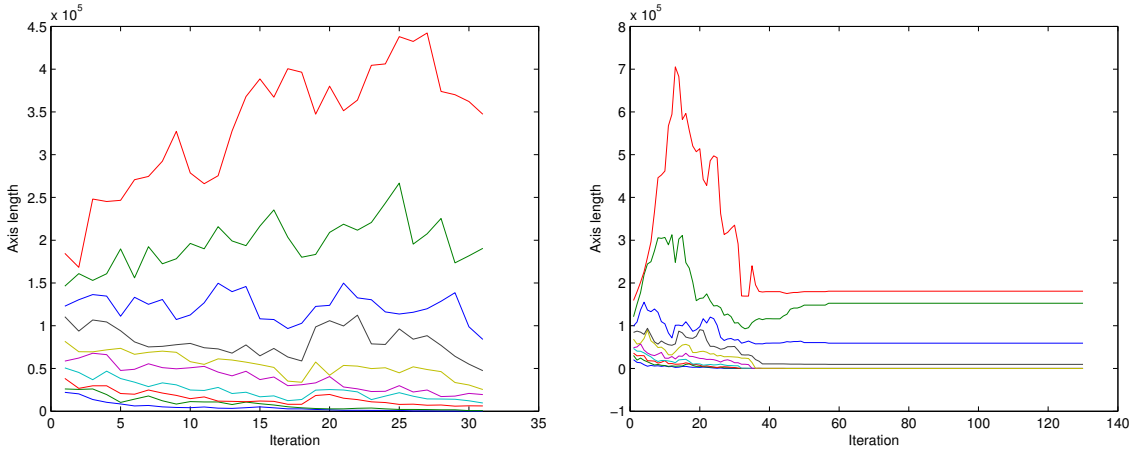


Figure 1: The curves represent the evolution of the ordered diagonal entries of matrix **H** over a test with the Rastrigin function.



(a) (b)

Figure 2: The curves represent the evolution of the ordered diagonal entries of matrix **H** over a test with the Schwefel function in a case where the global minimizer is reached (a) and in a case where a failure occurs (b).

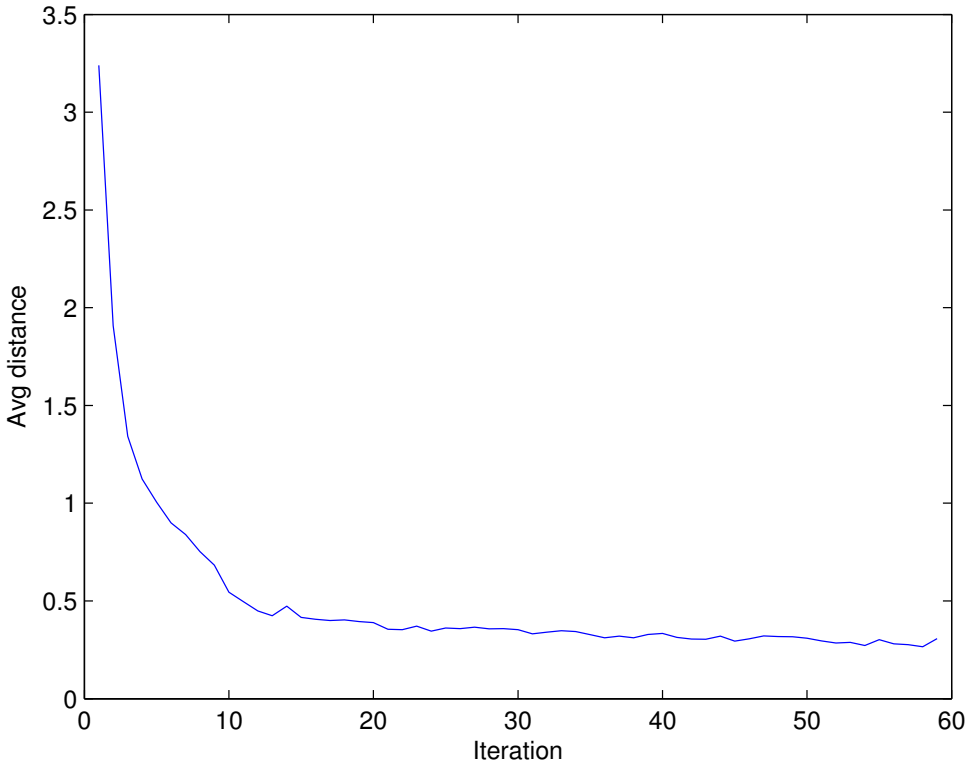


Figure 3: The curve of the average distance between population members and the corresponding perturbed points in the greedy generation procedure with the Rastrigin function with scaling matrix $\mathbf{D} = 4\mathbf{I}$.

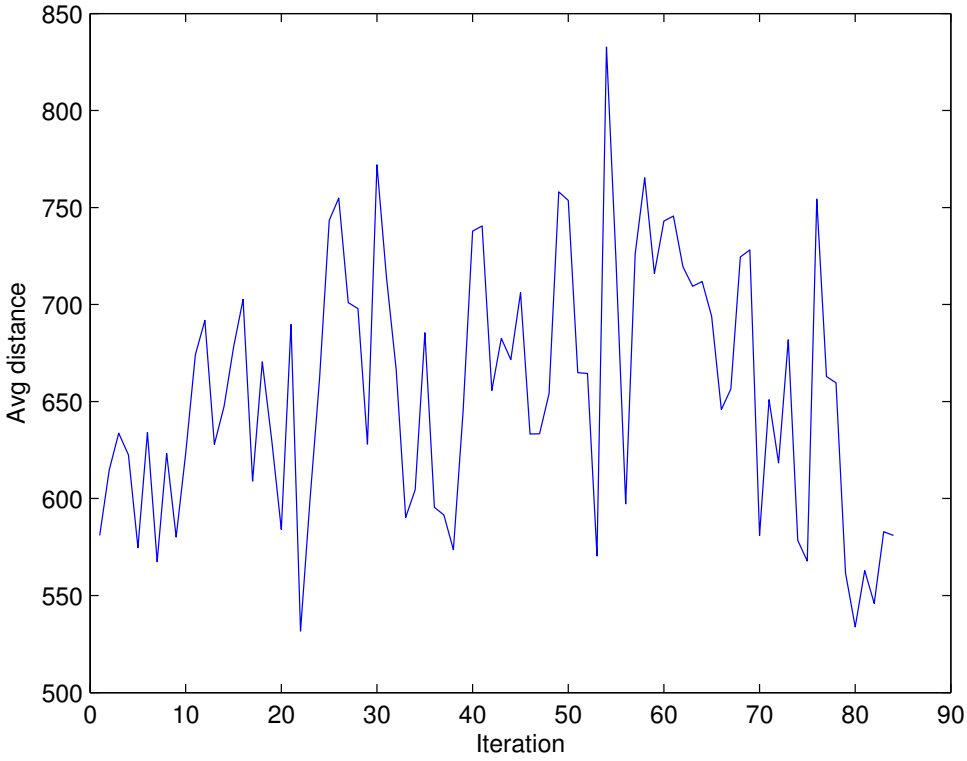


Figure 4: The curve of the average distance between population members and the corresponding perturbed points in the greedy generation procedure with the Schwefel function.

regular and decreasing for the Rastrigin function (see Figure 3) with the convergence towards the distance between the global minimizer and the second best local minimizer. The curve associated to the Schwefel function (see Figure 4) is, once again, much more irregular, which reflects the multi-funnel nature of the function.

In our third experiment we aimed at analyzing the behavior of the hybrid selection procedure. For each population member at each iteration we kept track of the policy (greedy or distance) applied during the selection phase. We did this both with the Rastrigin function and with the Schwefel function. The results are shown in Figures 5a and 5b, respectively. In these figures the colored dots show, for each population member, which policy has been applied: the blue ones are those for which the greedy policy was applied, while the red color represents the distance policy. The filled dots show if the selection mechanism led to a new individual with a best function value. The piecewise linear curve represents the function values of the individual that first reaches the global minimizer. It is clearly seen from the figures that the population members with a low function value have the blue color with the Rastrigin function, and the red color with the Schwefel function. This was the expected behavior: the best approach is the greedy one for the single-funnel Rastrigin function, and the distance one for the Schwefel function.

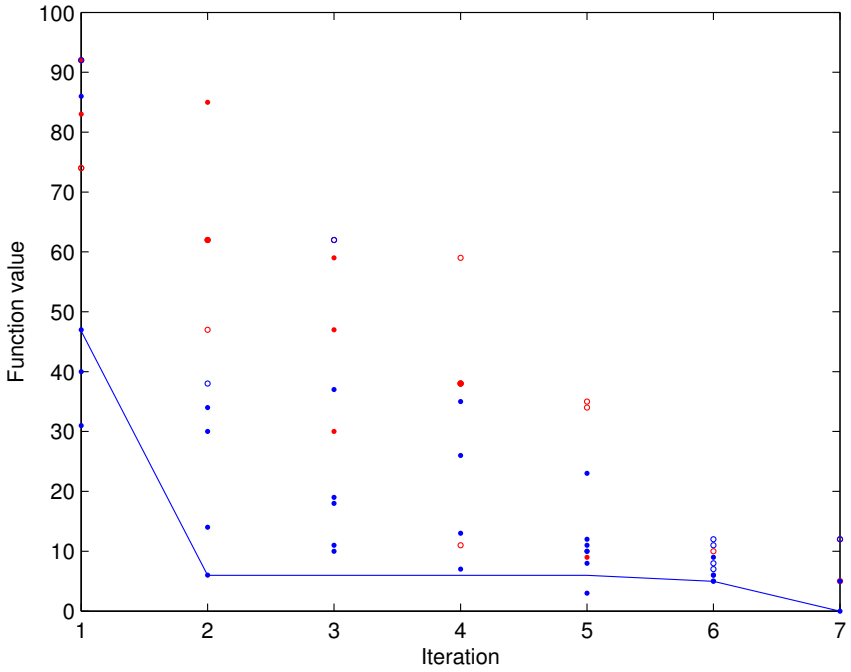
Our final experiment aimed at exploring the global aspects of local solvers. We previously commented that local searches often lead to local minimizers which are quite far away from the starting points. Thus, we performed the following experiment. We considered the Rastrigin function, for which, in the n-dimensional case, the minimal distance between a point and its closest local minimizer is (approximately) bounded from above by $\frac{\sqrt{n}}{2}$. Then, we measured the average Euclidean distance between the starting points of the local searches (points $\mathbf{y}_i$ in the greedy generation procedure Algorithm 6) and the local minimizer reached by the local searches (performed with `fmincon`), i.e., we measured

$$\frac{1}{k}\sum_{i=1}^{k}\|\mathbf{q}_i - \mathbf{y}_i\|.$$

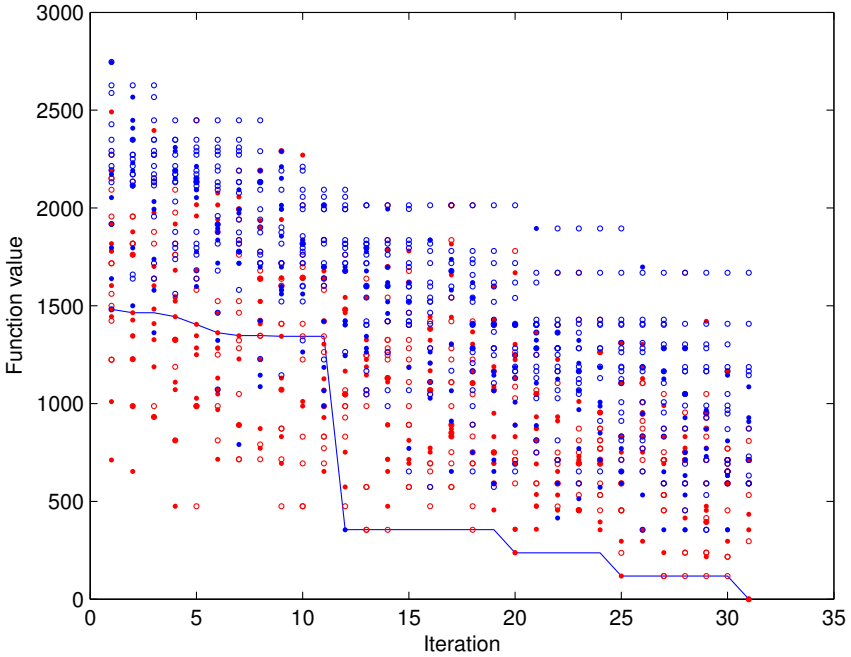
In Figure 6 we tested the case $n = 10$ (for which the upper bound on the minimal distance is ≈ 1.58). It is possible to see that the distance is always greater than the upper bound with very large distances during the first iterations. This shows that, in particular during the first iterations, the local solver itself is performing some sort of global search.

5 Conclusions

In this paper we have computationally investigated some simple variants of MDE, a memetic approach for continuous global optimization, which has been proved to be quite efficient in [10]. The analysis revealed that the best of such variants often outperforms the performance of MDE, but at the same time that the best variant is not always the same



(a) Rastrigin



(b) Schwefel

Figure 5: Function values of the population members to which the greedy policy has been applied (blue) and those to which the distance policy was applied (red). The piecewise linear curve represents the function values of the individual which first reaches the global minimizer.

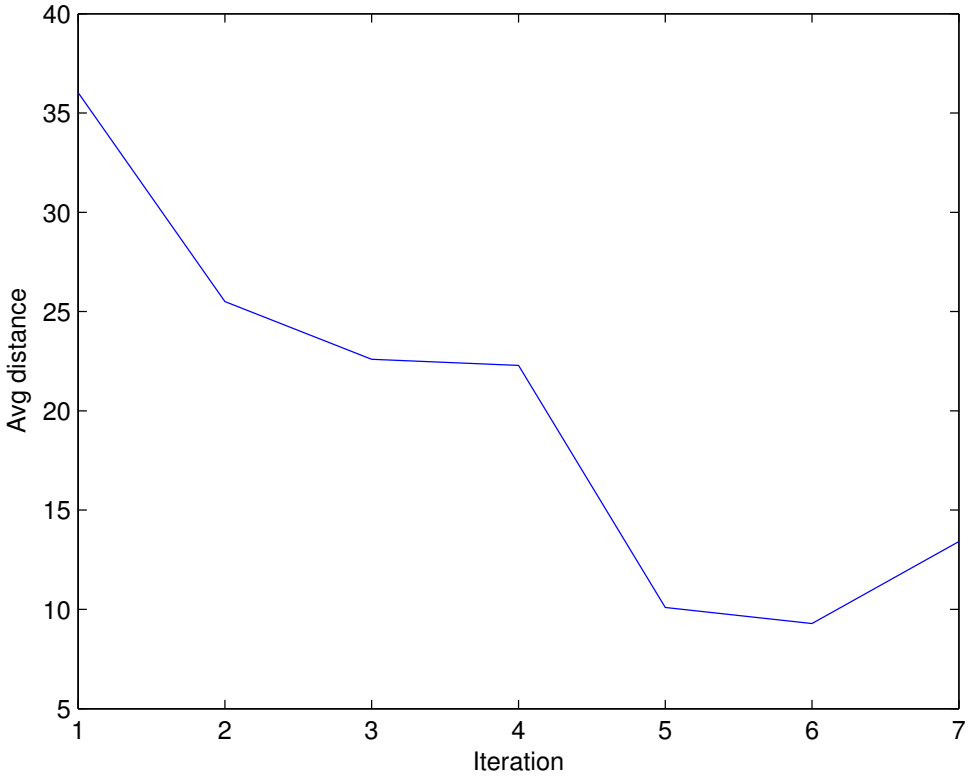


Figure 6: Average distance between the starting points of the local searches and the local minimizers reached. Test performed with the 10-dimensional separable Rastrigin function and the `fmincon` solver.

and is strictly problem dependent. In particular, the quickly convergent greedy variant is the best one for single-funnel functions, while the slowly convergent distance variant is usually the best one with multi-funnel functions. An hybrid approach has also been introduced in order to have a reasonably good (though not necessarily the best) performance over all functions. The computational analysis also reveals that the procedure employed for the local searches may have a considerable impact on the performance of each of the approaches we tested. We point out here that we have deliberately chosen to investigate only very simple variants of the (already simple) MDE approach. Indeed, in this study it was our intention to avoid any complication in order to show that even some basic strategies, applied in a proper way, may lead to very good results. More sophisticated strategies could probably allow for further improvements and could be an interesting topic for future research.

References

[1] D.H. Ackley. *A connectionist machine for genetic hillclimbing*. Kluwer Academic Publishers, Boston, 1987.

[2] Bernardetta Addis, Marco Locatelli, and Fabio Schoen. Disk packing in a square: A new global optimization approach. *INFORMS J. on Computing*, 20(4):516–524, 2008.

[3] Andrea Cassioli, Marco Locatelli, and Fabio Schoen. Dissimilarity measures for population-based global optimization algorithms. *Computational Optimization and Applications*, 45(2):257–281, 2010.

[4] S. Finck, N. Hansen, R. Rosz, and A. Auger. Real-parameter black-box optimization benchmarking 2010: Noiseless functions definitions. Technical Report RR6829, INRIA, 2011.

[5] Andrea Grosso, Marco Locatelli, and Fabio Schoen. A population based approach for hard global optimization problems based on dissimilarity measures. *Mathematical Programming*, 110(2):373–404, 2007.

[6] Bernd Hartke. Global cluster geometry optimization by a phenotype algorithm with niches: Location of elusive minima, and low-order scaling with cluster size. *Journal of Computayional Chemistry*, 20:1752–1759, 1999.

[7] Robert H. Leary. Global optimization on funneling landscapes. *Journal of Global Optimization*, 18:367–383, 2000.

[8] Julian Lee, In-Ho Lee, and Jooyoung Lee. Unbiased global optimization of Lennard-Jones clusters for $N \leq 201$ by conformational space annealing method. *Physical Review Letters*, 91(8):1–4, 2003.

[9] Marco Locateli and Fabio Schoen. *Global Optimization: Theory, Algorithms, and Applications*. MOS SIAM Series on Optimization. SIAM, 2013.

[10] M. Locatelli, M. Maischberger, and F. Schoen. Differential evolution methods based on local searches. *Computers and Operations Research*, 43:169–180, 2014.

[11] M. Locatelli and F. Schoen. Global optimization based on local searches. *4OR-Quarterly Journal Operations Research*, 11:301–321, 2013.

[12] Jorge M. C. Marques, A. A. C. C. Pais, and P. E. Abreu. Generation and characterization of low-energy structures in atomic clusters. *Journal of Computational Chemistry*, 31(7):1495–1503, 2010.

[13] K. Price, R.M. Storn, and J.A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Springer, 2005.

[14] H.P. Schwefel. *Numerical Optimization of Computer Models*. John Wiley & Sons, 1981.

[15] Rainer Storn and Kenneth Price. Differential evolution. A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

[16] A. Törn and A. Zilinskas. *Global Optimization*. Springer-Verlag, Berlin, 1989.

[17] David J. Wales and Jonathan P. K. Doye. Global optimization by Basin-Hopping and the lowest energy structures of Lennard–Jones clusters containing up to 110 atoms. *Journal of Physical Chemistry A*, 101(28):5111–5116, 1997.