UNIVERSITÀ
DEGLI STUDI
FIRENZE

Università degli Studi di Firenze
Dipartimento di Ingegneria dell'Informazione (DINFO)
Corso di Dottorato in Ingegneria dell'Informazione
Curriculum: Automatica, Ottimizzazione e Sistemi Complessi

SAVING LOCAL SEARCHES IN GLOBAL OPTIMIZATION

Candidate
Luca Tigli

Supervisors
Prof. Fabio Schoen

Prof. Marco Sciandrone

PhD Coordinator
Prof. Fabio Schoen

ciclo XXXII, 2016-2019

Università degli Studi di Firenze, Dipartimento di Ingegneria dell'Informazione (DINFO).

Ai miei genitori.

Acknowledgments

I would like to thank my supervisor Prof. Fabio Schoen for his patience and precious input, Prof. Marco Sciandrone and all of my colleagues at GOL who were of great help during my research.

Abstract

This thesis concerns the use of local searches within global optimization algorithms. In particular, we focus our attention on the strategies to decide whether to start or not a local search from a starting point. More specifically, our aim is to avoid the waste of computational effort due to local searches which lead to already detected local minima or to local minimizers with a poor function value. This is done by a re-discovery of clustering methods as good candidates to use within GO algorithms to locate the regions of attraction of local minimizers. In the first contribution, we show how to improve a GO algorithm with information from its past runs and making a smart use of the latter. Moreover, the idea of extracting compact geometrical descriptors from GO solutions is used for both stopping redundant lines of search of a perturbation-based GO method and to adapt clustering algorithms to work in a reduced feature space, in which configurations are compared and grouped by means of their overall characteristics rather than the value of their variables. In the second part of this thesis, we show how the cost of "CPU heavy" local searches in memetic complex GO schemes may be reduced with clustering methods by avoiding multiple rediscoveries of the local optima. Though we restrict our attention on a quite efficient memetic differential evolution, our strategy can be easily used to enhance the performance of any evolutionary method in general. Finally, we show how to successfully apply our clustering-based memetic approach in large dimensions.

Contents

Chapter 1

Introduction

1.1 Global Optimization

Given a compact set $\Omega \subset \mathbb{R}^n$ and a continuous function $f : \Omega \to \mathbb{R}$, consider the Global Optimization (GO) problem

$$\min_{x \in \Omega} f(x) \tag{1.1}$$

that consists in the minimization of f over Ω.

Extending the class of objective functions to include multimodal functions, i.e., functions may have several local minima in the region of interest, makes the optimization problem hard to solve in general. In particular, such problems may differ considerably with respect to the computational effort needed to perform a function (or gradient) evaluation. Here we restrict our attention on methods which are suitable for problems where function evaluation and local optimization are relatively cheap tasks. Of course, such local optimization has a computational cost which is higher rather than a simple function evaluation, but such additional cost is often rewarded by the possibility of defining much more efficient algorithms. Particularly, some parts of the feasible space may be deemed more interesting than others and more accurate solutions in these regions may be available. Because both global reliability and local refinement are important features of a GO method, most good heuristic methods consist of a clever mixture of both a global and a local phase, and no serious method can lack the global strategy. Concerning the global phase, its aim is usually that of exploring the search domain, as

opposed to that of the local phase which is more concerned with refinement of the current solution. Examples of local phases are standard local searches performed by means of local optimization methods such as gradient descent method or the quasi-Newton method. Unfortunately, severe nonlinearities commonly impose the problem of the amount of CPU time needed to compute f, which means that the affordable number of function (and gradient) evaluations should be as small as possible. To overcome these difficulties, good starting points for local searches are commonly required, however an appropriate sample of points is not always available in the formulations of many practical problems.

Thus, some approaches have been developed to overcome this difficulty. As an example, in [26] the author investigated the impact of the local search frequency for the optimization of common test functions; local searches are started according to a specific probability and Multistart is obtained by setting this probability equal to 1. More recently, in [8], the authors proposed to use machine learning, and, in particular, Support Vector Machines (SVM), in order to decide whether to start or not a local search from a candidate point. In particular, between the early seventies and nineties, a global optimization paradigm based on the so-called *clustering method* was studied by some researchers. According to Törn and Zilinskas [67], the motivation for exploring clustering methods was based on principles of efficient local search as well as elimination of redundant cost function evaluations within a single cluster. Here we focus on the latter class of methods.

1.2 Clustering methods for GO

When the first papers on GO appeared (see, e.g., [13, 14]), the most relevant and practical computational tool for solving generic GO problems was Multistart. This is the most basic GO algorithm, consisting in the following:

1. (Randomly Uniformly) sample a set of points in the feasible region
2. Start a local descent from each sampled point
3. Decide whether to stop or to go back to (1)

This basic computational scheme has, evidently, several limitations: it is not always easy to sample in a feasible region defined by constraints; local optimization is computationally demanding; stopping is not an easy decision

as early stopping might cause the global optimum to be missed, while late stopping implies a CPU time waste. At the time when clustering methods were proposed, one of the main concerns in Multistart was the fact that, when running Multistart, the same local optimum might have been observed several times during a run, thus wasting computational time. Some researchers started to experiment with methods whose ideal aim was to be able to run a local search at most once from the region of attraction of each local optimum. The idea of clustering was that of using some statistical technique based on similarity in order to recognize the points grouped around different local minima (the clusters). It shall not require that the procedure yield an optimal division according to some criterion function because this requirement normally implies extensive calculations; one of the earliest successful ideas was proposed in [5, 66, 68] and further developed in [52, 53].

To illustrate clustering methods, let us consider the GO problem (1.1) where we also assume that the feasible set $\Omega \subset \mathbb{R}^n$ is the unit hypercube $[0,1]^n$. In what follows the symbol $\mathscr{S}$ will be used to denote the set of all points randomly generated by the algorithm, while with $\mathscr{C}$ we will denote the set of points obtained by means of a "concentration step", i.e., points obtained from the original sample but concentrated in the regions of attraction of local minima, as we will see later. Finally, $\mathscr{O} \subseteq \mathscr{C}$ denotes those points in the concentrated sample $\mathscr{C}$ starting from which the algorithm chooses to perform a complete local optimization run.

The basic scheme of clustering methods is the following:

Let $\mathscr{S} = \emptyset,\ k = 0$

Generation: a batch of $n_b \geq 1$ sample points are drawn in $\Omega = [\ 0,1\ ]^n$, usually by random uniform sampling; these n_b points are added to $\mathscr{S}$. Let $k := k + 1$;

Concentration: newly generated points are concentrated either by means of a few steps of a local descent algorithm or by temporarily discarding a fraction $\gamma \in (0,1)$ of points in $\mathscr{S}$ with the highest function value. The whole transformed sample is denoted by $\mathscr{C}$;

Clustering and Selection: a point $x \in \mathscr{C}$ is selected if it had never been selected in the previous phases of the algorithm and if there is no other

point $y \in \mathscr{C}$ such that

$$\|x - y\| \leq \tau(kn_b) \tag{1.2}$$

$$f(y) \leq f(x) \tag{1.3}$$

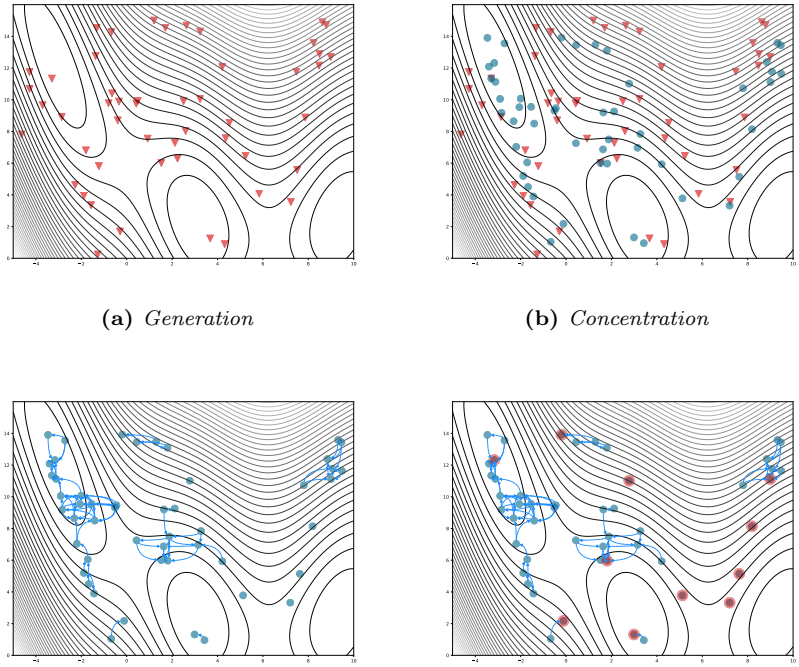
where $\tau(\cdot)$ is a suitably defined threshold. Points whose relative distance is no more than $\tau(kn_b)$ are assigned to the same cluster. For each cluster a single point with the smallest function value is selected and added to the set $\mathscr{O} \subseteq \mathscr{C}$;

Full optimization: from each point in $\mathscr{O}$ a complete local optimization is performed by running a specific local optimization (descent) algorithm $\mathcal{L}$ which, when initialized at a feasible point x, is capable of delivering a point $\bar{x}$ such that $f(\bar{x}) \leq f(x)$. If this local optimization ends with a new local optimum $\bar{x}$, then this point is added to the set $\mathscr{C}$;

Stopping: if a stopping rule is not satisfied the whole procedure is restarted.

In Figure 1.1 a pictorial representation of a phase of this algorithm is depicted.

This is not the most general scheme of GO clustering methods, but it is by far the one which received much attention. Some comments are in order. The concentration step is meant to transform the sample from a purely uniform random one to another in which the density of points is no more uniform everywhere, but tends to be higher in the regions of attraction of different local optima. Two main tools are usually devoted to sample concentration: one is the selection of a fraction of points with the lowest function value. This procedure is very easy to apply and it lends itself to some theoretical analysis. Another tool, which, although slightly more costly from a computational point of view, seems to be the most efficient one, is that of starting from each newly sampled point a few descent steps with a suitable local optimization algorithm. Whichever the tool used to concentrate the original uniform sample, the resulting set of concentrated points is no more uniform. Thus it is reasonable to look for statistical clustering techniques capable of identifying those regions characterized by the highest point density. The main idea of clustering for GO methods is that this statistical procedure should try to associate different clusters to regions of attraction of different minima. Thus, the following step prescribes to start a full local optimization from a single point from each cluster, provided that such a full optimization had not been already performed in previous phases of the algorithm.



(a) *Generation* **(b)** *Concentration*

(c) *Clustering* **(d)** *Selection*

Figure 1.1: A graphical illustration of MSL. A uniform sample (triangles in (1.1a)) is first drawn; then the sample is concentrated by running a few steps of a local descent algorithm (disks in (1.1b)); the resulting transformed sample is no longer uniform and sub-samples with higher density, or clusters, can be identified through a clustering technique (in (1.1c), clustered points are connected by arcs). From each cluster, a single point with minimum function value is selected (larger disks in (1.1d)); a full local optimization will then be run from these selected points only.

1.2.1 Multilevel Single Linkage

Of course, the success of clustering methods is strongly dependent on the assumption that clusters are associated to regions of attraction. Many attempts have been made towards this aim, trying to minimize the likelihood of two possible errors: clustering points belonging to more than one region of attraction, or missing the region of attraction of a local optimum. Differently from classical statistical clustering, in our setting there is additional information which should be exploited: first, to each point a function value is associated, so clusters will not only take into account the distance between points, but also this information; moreover, regions of attraction are associated to a local descent path, so that when building clusters this information might also be taken into account.

Among several clustering approaches proposed in the literature, we cite here the most popular ones (for more details see [52, 53]):

Density clustering. Clusters are built starting from *seed points* which, usually, consist of already observed local optima. Level sets of the objective function are assumed to be ellipsoids characterized by the hessian matrix at each local optimum. Based on this assumption, clusters are defined as

$$\{x \in \Omega : (x - \bar{x})^T \nabla^2 f(\bar{x})(x - \bar{x}) \leq r^2\}$$

where $\bar{x}$ is a seed (local optimum) and r is a threshold. Let $K = kn_b$ and

$$r_i = \pi^{-1/2} \left\{ i\Gamma\left(1 + \frac{n}{2}\right) \sqrt{\det(\nabla^2 f(\bar{x}))}\, \frac{\sigma \log K}{K} \right\}^{1/n}$$

where Γ is the gamma special function and $\sigma > 0$ is a parameter. In order to chose r, a cluster is grown around $\bar{x}$ by letting first $i = 1$ and increasing its value until the enlargement of the ellipsoid does not include any new point.

Single Linkage. A defect of Density clustering is that an ellipsoidal shape is assumed for the region of attraction of a minimum; this is very often a false assumption. Thus, in Single Linkage, starting from seed points, new points in the reduced sample are added to a cluster if their distance from a point in the already grown cluster is below a threshold r, defined in a similar way as above, without any assumption on the shape of the resulting cluster.

Multilevel Single Linkage (MSL). These methods are built on top of single linkage ones, but they restrict the association of a reduced sample point to a cluster to the event that from this point there should be a descent path leading to a close enough point in the cluster. The main difference with Single Linkage is the fact that the additional information given by function values is taken into account, so that it is possible to link points by means of directed arcs pointing towards low level sets. Although the first methods within this class assumed that clusters were started from seed points, MSL does not assume any seed point at all. The main idea of MSL is to apply a full local search to a point x in the reduced sample $\mathscr{C}$, i.e., to add x to $\mathscr{O}$, if and only if:

1. x is at least ε far from already observed local optima
2. x is at least ε far from the boundary of the feasible region
3. there exist no point $y \in \mathscr{C}$:

$$\begin{aligned} \|x - y\| &\leq \tau(K) \\ f(y) &\leq f(x) \end{aligned}$$

where $\varepsilon > 0$ is a constant, $K = kn_b$ is the total current sample size and

$$\tau(K) = \pi^{-1/2} \left(\Gamma\left(1 + \frac{n}{2}\right) \mu(\Omega) \frac{\sigma \log K}{K} \right)^{1/n} \tag{1.4}$$

where $\mu(\Omega)$ is the Lebesgue measure of the feasible set ($\mu(\Omega) = 1$ with our assumptions) and $\sigma > 0$ is a parameter.

The choice of the threshold (1.4) is justified in [53]. Here we outline a very brief explanation. It is well known that the volume of an n–dimensional hypersphere of radius r is

$$\pi^{n/2} r^n \;/\; \Gamma(1 + n/2)$$

so that the volume of an hyphersphere whose radius is given by (1.4) turns out to be

$$\frac{\sigma \log K}{K}$$

The probability that, in a random uniform sample of cardinality K, no sampled point is found in a sphere of radius $\tau(K)$ around a specific point is thus

$$\left(1 - \frac{\sigma \log K}{K}\right)^K \tag{1.5}$$

and this quantity, which is $O(k^{-\sigma})$, is decreasing to zero at a speed which guarantees that for the overall MSL procedure the following holds:

Theorem 1.2.1 ([53]). *For a clustering method based on MSL equipped with the threshold (1.4) it is guaranteed that:*

1. *the probability that a point in the reduced sample $\mathscr{C}$ is selected as a starting point for a full local optimization tends to 0 as $k \to \infty$*

2. *if $\sigma > 2$ the probability that a new full local search is started during phase k of the algorithm tends to 0 as $k \to \infty$*

3. *if the algorithm never stops, the total number of full local optimization started remains finite with probability 1*

4. *every local optimum will be found with probability 1 within a finite number of iterations*

The above properties look very attractive from the GO point of view and, in fact, MSL has been the most common choice for solving GO problems (at least box–constrained ones) for a long period of time. Nowadays, however, this method is no more used for many reasons, some of which we will explain here. First, at the time when MSL appeared, local optimization was a computationally demanding task, given the available algorithms and computational power. Saving even a few full local optimization runs was considered as a top priority. Secondly, and, in our opinion, more importantly: the status of GO in those years was such that no attempt was ever made to solve problems whose dimension was larger than, say, ten. Later, in the optimization history, great advances were made in developing very efficient local optimization tools, and meanwhile the computational power of available computers grew very fast. These two facts have led applied researchers to develop methods capable of approximately solving very large dimensional GO problems. Many methods exist, some of which are based on the exploitation of local optimization – see, e.g., [43] for a recent survey.

1.3 Contributions

When considering large dimensional GO problems, MSL fails as clustering is no more possible in high dimensional spaces, unless the sample size grows exponentially fast. In this research, our aim is to propose a re-discovery of clustering methods as good candidates to use within GO algorithms. We investigated such approaches in order to avoid the waste of computational effort due to local searches which lead to already detected local optima. In other words, we will examine the relevant issue related to the decision whether to start or not a local search from a starting point. We aimed at showing that with proper modifications, clustering methods are capable to obtain the same quality in terms of returned solutions, but at a fraction of the computational cost, with respect to more recent GO approaches.

In Chapter 2 we show how to improve a GO algorithm with information from its past runs and making a smart use of the latter. Moreover, the idea of extracting compact geometrical descriptors from GO solutions is used for both stopping redundant lines of search of a perturbation-based GO method and to adapt clustering algorithms to work in a reduced feature space, in which configurations are compared and grouped by means of their overall characteristics rather than the value of their variables. This way clustering methods are rediscovered and employed in a modern and innovative fashion. Though we applied it to the scenario of atomic structure prediction (an active field in computational chemistry), our technique is straightforward and high-level, and can be easily used to enhance complex GO schemes and solve problems from different areas. In Chapter 3 we show how the cost of "CPU heavy" local searches in memetic complex GO schemes may be reduced with clustering methods by avoiding multiple rediscoveries of the local optima. Though we restrict our attention on a quite efficient memetic DE schema, our strategy can be easily used to enhance the performance of any evolutionary method such as PSO, GAs, etc. Finally, since the results obtained in previous chapters were encouraging enough to motivate the use of the same ideas, in Chapter 4 we show how to successfully apply our clustering-based memetic DE to standard large scale GO problems.

Chapter 2

Clustering Methods for Geometrical GO

In this chapter we show that for some specific problem classes, it is possible to design Global Optimization algorithms which mimic the behavior of existing procedures and obtain the same quality but at a fraction of their computational cost. The way we achieved this is through the application of clustering methods designed to identify the regions of attraction of good local minima. Ideally, if we were able to identify the shape of these regions of attraction, a single local search starting from each of them would lead to the global minimum. This idea had been a winning one in the 80's, and later abandoned when large dimensional global optimization problems were used to test global optimization algorithms. In this chapter we show that by using the idea of clustering in a feature space of much smaller dimension than one of the original variables, very significant speedups can be obtained. We present the general idea and apply it to two of the most widely studied families of hard, large scale, global optimization problems: the optimization of the potential energy of atomic clusters, and the problem of packing a set of identical spheres of largest radius in the unit hypercube. In this latter problem we could even improve some existing putative optima, thus proving that the proposed method is not only very efficient but also effective in exploring the feasible space.

2.1 Introduction

As mentioned in the introduction, the main drawback of MSL, and, in our opinion, the most relevant reason for the early abandonment of MSL in the recent GO literature, is the *curse of dimensionality*. In particular, given a critical distance r, the probability that a uniformly sampled point in $[0,1]^n$ will be at distance not greater than r from at least one out of K other uniformly sampled ones, exponentially decreases with the dimension n, so that, in order not to start too many local searches, nor to risk missing the relevant ones, an exponentially increasing number of points needs to be sampled. As a simple observation, in order to place one point in each box obtained by dividing each edge of the unit hypercube into two halves, 2^n points are needed. So MSL, in high dimension, is hopeless and, in fact, most of its historical applications had been for problems with no more than 10 variables.

However, the idea of MSL was a bright one and, in this chapter, we aim at showing that with a slight modification it is possible to successfully apply MSL in large dimension, provided that the problem at hand lends itself to a compact representation. In what follows, we call a problem geometrical when its variables, constraints and objective have a clear physical meaning.

2.2 MSL in a feature space

Drawing from ideas in Machine Learning, we assume that to each feasible solution of the GO problem a small set of problem-specific descriptors can be defined. The number of descriptors characterizes the dimensionality of a feature space that, in general, is smaller than that of the original problem. This way clustering may be performed in a low-dimensionality space, where the decision whether to start a complete local search is also taken. Let $\phi : \Omega \subset \mathbb{R}^n \to \mathbb{R}^d$ be a mapping from the solution space to a d-dimensional *feature* space, where we assume that d is chosen such that $d \ll n$. In this section, in order to successfully apply MSL in large dimension, a slight modification of the original scheme is proposed; we will call this new method FS-MSL (MSL in a Feature Space). The pseudocode of the overall procedure is described in Algorithm 1, where the *threshold* distance $\tau_d(\cdot)$ is defined as follows

$$\tau_d(K) = \pi^{-1/2} \left(\Gamma\Big(1 + \frac{d}{2}\Big) \frac{\sigma \log K}{K} \right)^{1/d} \tag{2.1}$$

Notice that this is the same threshold as in (1.4), the only difference being that the dimension of the original space is replaced by the dimension of the feature space. Also, we assume without loss of generality, that the Lebesgue measure of the d–dimensional feasible space for features is equal to one. The algorithm requires a few input parameters. In particular, it is required that the user chooses a real value $\sigma > 0$ for the parameter in (2.1) as well as the specification of a *batch* size (the number of new sample points to be generated at each iteration). Moreover, a suitable stopping rule should be defined, which, in general, is either a maximum number of iterations performed without an improvement of the best observed feasible solution or a maximum number of performed iterations. Another parameter, n_c, is also necessary, which represents the number of descent iterations to be performed starting from each newly sampled point, in order to concentrate the sample.

Algorithm 1 MSL in a Feature Space (FS-MSL)

Require: σ, n_b, n_c
$\mathscr{C} \leftarrow \emptyset$
$K \leftarrow 0$
while a stopping condition is not satisfied **do**
  $X_b \leftarrow$ (randomly uniformly) generate n_b feasible solutions
  $K = K + n_b$
  $X_c \leftarrow$ perform n_c local optimization steps from each element in X_b
  $\mathscr{C} \leftarrow \mathscr{C} \cup X_c$
  $\mathscr{O} \leftarrow \emptyset$
  for $x \in \mathscr{C}$ **do**
    if $\nexists\ y \in \mathscr{C} : f(y) \leq f(x)$ and $\|\phi(y) - \phi(x)\| \leq \tau_d(K)$ **then**
      $\mathscr{O} \leftarrow \mathscr{O} \cup \{x\}$
    end if
  end for
  $X^* \leftarrow$ perform a full optimization starting from each selected point in $\mathscr{O}$
  $\mathscr{C} \leftarrow \mathscr{C} \cup X^*$
end while
return the best solution

In the above algorithm, it is assumed that a "local" optimization procedure is available and that this procedure consists of a finite number of "steps". In the concentration phase, only a few of these, n_c, are performed, while in the full optimization phase the procedure is allowed to run until convergence. In the original MSL algorithm, local optimization was indeed a gradient descent method like, e.g., L-BFGS. In our modern

version of the method, this procedure might be, and indeed usually is, a complex global optimization procedure like, e.g., Monotonic Basin Hopping (see, e.g., [42, 43, 72]). By this we mean that our "local" procedure is indeed a finite sequence of repeated calls to a local descent algorithm. In Monotonic Basin Hopping (MBH), each descent is started from a point which is randomly generated in a neighborhood of the current one; the local optimum obtained from such a point replaces the current one if the objective function improves. In this framework, the concentration phase corresponds to performing a few, namely n_c, runs of the descent algorithm, while the full optimization phase corresponds to performing the optimization run until its natural stopping.

In order to be able to define an effective computational procedure it is necessary to design the algorithm taking into account specific problem-dependent characteristics. We summarize them here:

Feature space. This is of fundamental importance and it is the basis for the correct behaviour of this method. The selected features should be as few as possible, taking into account the difficulties of clustering in high dimension, but sufficiently discriminant so that it is unlikely that two sufficiently different solutions have very similar features. Moreover the features should characterize not just an optimal solution, but the whole "funnel" associated to the region of attraction of the optimal solution. It is in fact important to be able to predict the features of the local optimum already from those associated to the concentrated solution.

"Funnel" optimizer. By this we mean a procedure which, when started from a randomly generated solution, is capable of generating a sequence of improving local optima. Intuitively, a "funnel" is a global trend on a search space that consists of a basin of clustered local minima. A search method with a reasonable probability of moving between adjacent local optima should be able to locate the bottom of a single-funnel function (see [38] for a complete discussion on the multilevel structure of global optimization problems). So, the procedure should be good enough to produce solutions with low function value. However, it should not be "too global": by this we mean that during the optimization procedure, it is expected that the generated solutions do improve, but that the "structure" of these solutions does not change too much, in the sense that there should be no large modification of the set of features during

the descent. As an example, a Monotonic Basin Hopping method might be a good local optimization, provided that the perturbation radius chosen at each step is large enough to guarantee the possibility of descent, but small enough so that the solution does not "jump" outside the current funnel.

These tools are *problem-dependent*. In the following we will exemplify their practical implementation for two classes of optimization problems: atomic cluster structure prediction, and optimal packing of equal, non overlapping spheres in the unit cube; these are two radically different problems, the first one being unconstrained, while the second one has a large number of non convex constraints, with some common structure.

2.3 Geometrical GO problems

Global optimization is an active field with rapid growth for many chemical and physical problems, such as structural optimization, chemical engineering design, and molecular biology. In general, the GO of an arbitrary function requires a search through the whole conformational space. The problem is generally NP-hard due to the fact that the conformational space grows exponentially with the problem size. Moreover, when dealing with *geometrical* problems, GO algorithms' job is made harder by the existence of symmetries and multiple (often infinite) equivalent solutions. In this regard, the idea of extracting geometrical descriptors may help algorithms both in reducing the dimension of the search space to be explored, and to adapt clustering algorithms to work in a reduced space, where configurations are compared by means of their characteristics rather than the value of their variables. Driven by our personal experience, in this work we restrict our attention to a couple of well known GO problems, namely molecular conformation and sphere packing problems; however, we point out that many other geometrical GO problems arising from applications can be found in the literature.

2.3.1 Atomic cluster structure prediction

In recent years there have been growing interest and significant research developments in the experimental and theoretical studies of the atomic clusters, owing to their peculiar physical and chemical properties compared with bulk matters. Atomic clusters can generally be described as aggregates of atoms

or molecules containing few to several thousands of elements. To understand the unique electronic, optical, and magnetic properties of an atomic cluster, the fundamental research problem is to locate their stable structures in the three-dimensional space. These conformations are associated to the minimum possible energy for the system. Experiments alone often provide an incomplete picture of the atomic cluster structure and only by combining them with independent theoretical investigations, a complete description of the geometric arrangement and the corresponding properties can be established. Empirical potential played an important role in computer simulation and are largely used to fit the interactions among particles – the results are generally acceptable within a certain precision.

Here we describe the simplest possible model for atomic clusters, where groups of atoms interacting via empirical *pairwise* potential, with no chemical bond. A conformation problem for atomic clusters can be defined as the following unconstrained GO problem

$$\min_{x_1,x_2,\ldots,x_N} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} V\left(\|x_i - x_j\|_2\right) \tag{2.2}$$

where N is the number of atoms in the cluster, $x_i \in \mathbb{R}^3,\ i = 1, \ldots, N$, represent the coordinates of the center of the i-th atom, and V is a pair potential whose analytic expression might depend on each atom's type. Hence, the task is to perform a global optimization of the potential energy as a function of all atom coordinates.

The Lennard-Jones and Morse potentials are the two most famous pair potentials, which often act as the benchmark systems to evaluate the newly developed atomic cluster geometry optimization methods. The Lennard-Jones (LJ) potential is defined as

$$V(r) = \epsilon \left(\frac{r_0}{r}\right)^6 \left(\left(\frac{r_0}{r}\right)^6 - 2\right) \tag{2.3}$$

where r is the inter-atomic distance, r_0 is the equilibrium pair distance, and ϵ is the pair well depth. Here, we set $\epsilon = r_0 = 1$ for simplification and then the LJ function is written as

$$V(r) = r^{-12} - 2r^{-6} \tag{2.4}$$

Alternatively, the Morse potential can be used to estimate both long- and

short-range interactions. The Morse function is defined as

$$V(r) = e^{\rho(1-r)} \left(e^{\rho(1-r)} - 2 \right) \tag{2.5}$$

where ρ is the parameter of the potential. Generally speaking, low values of ρ give a long-ranged potential and high values a short-ranged potential. At $\rho = 6$, the Morse function is very similar to the LJ one and the two potential functions can be unified by a modified Morse function.

The potential range is the most important factor for a pair potential determining the favorite cluster structure: at $\rho = 14$, Morse clusters are notoriously difficult to be optimized by an unbiased GO method, where the favorite conformations (motifs) are decahedral, tetrahedral, and close packed.

Both pair potentials are nonconvex functions of the relative distance between two atoms; when these pairwise contributions are summed up over all pairs of atoms in a cluster, the resulting Potential Energy Surface (PES) is a highly multimodal, nonconvex function of the coordinates of the N atoms. Because of the enormous number of local minimum structures of atomic and molecular clusters, it is difficult to locate the lowest energy conformation.

2.3.2 Sphere packing

Sphere packings is one of the most fascinating and challenging subjects in discrete and computational geometry. Almost four centuries ago, Kepler studied the densities of sphere packings and made his *Kepler's conjecture*. Several decades later, Gregory and Newton discussed the kissing numbers of spheres, i.e., how many spheres can be arranged so that they all just touch another sphere of the same size, and proposed the *thirteen spheres problem*. Since then, these problems and related ones have attracted the attention of many prominent mathematicians such as Dirichlet, Gauss, Lagrange, Minkowski, and Voronoi. Beyond the theoretically challenging character of the problem, there are several ways in which the solution methods can be applied to practical situations. Of course there are connections with chemistry and physics, since crystallographers have studied three-dimensional lattices since the beginning of the subject. Furthermore the sphere packings turn out to have connections, sometimes totally unexpected, with other branches of mathematics. There are direct applications of lattice packings to number theory, for example in solving Diophantine equations, or in digital communications from the design of signals for data transmission and storage. For example, it

is known that the optimal way of sending digital signals over noisy channels corresponds to the densest sphere packing in a high dimensional space [12].

In what follows, we restrict our attention to a rather classical packing problem, namely that of placing identical non-overlapping spheres into a cube, but we point out that other packing problems involving different objects and/or containers have been discussed in the literature. As we have already mentioned, the sphere packing problem is a geometrical problem that can also be viewed as a continuous GO problem. A general formulation of the latter in $\mathbb{R}^n$ can be defined as

$$\begin{array}{lll} \max\limits_{x_1,\ldots,x_N\in\mathbb{R}^n,\, r\in\mathbb{R}} & r & \\ \text{subject to} & \|x_i - x_j\|_2 \geq 2r & \forall (i,j) \in I \\ & x_i^k \geq r & k = 1,\ldots,n \quad i = 1,\ldots,N \\ & x_i^k \leq 1 - r & k = 1,\ldots,n \quad i = 1,\ldots,N \end{array} \tag{2.6}$$

which corresponds to packing N hyperspheres in R^n in the unit hypercube so that they do not overlap and their common radius r is maximized; the set I is defined as $\{(i,j) \mid 1 \leq i < j \leq N\}$ and x_i^k represents the k-coordinate of the center of the sphere i. Notice that this formulation has $2nN$ linear constraints and $N(N-1)/2$ nonconvex constraints.

Let us point out that there are two main categories of studies dealing with packing problems. One of these approaches is to prove the optimality of suggested packings, either (purely) theoretically, or with the help of computers: consult [62] for detailed discussions and further references. On the opposite side, instead, heuristic techniques have been proposed to approximately solve many instances, even at quite large sizes. Approximate packings (i.e., packings determined by various methods without proof of optimality) are reported in the literature for up to 200 circles ($n = 2$). These numerical results have been obtained via several different strategies; for instance, using nonlinear programming solvers (e.g., MINOS, SNOPT), the minimization of an energy function [48], or billiard simulation [25] just to mention a few. Results for many such problems are reported at `www.packomania.com` that is regularly maintained and updated.

The packing problem can be formulated in different ways, and some of them usually have a slight advantage for local optimization methods. Here, we consider the formulation as a maximal dispersion one (see [2]) where, in particular, we refer to the following bound-constrained, max-min optimiza-

tion model of the point arrangement problem in the unit cube

$$\max_{\substack{x_i \in [0,1]^n, \\ 1 \leq i \leq N}} \quad \min_{(i,j) \in I} \quad \{ d_{ij} \mid d_{ij} = \|x_i - x_j\|_2 \}$$

This problem is equivalent to the following non-linear programming problem

$$\begin{aligned} \max_{x_1,\ldots,x_N \in \mathbb{R}^n,\, d \in \mathbb{R}} \quad & d \\ \text{subject to} \quad & \|x_i - x_j\|_2^2 \geq d^2 \qquad \forall (i,j) \in I \\ & x_i \in [0,1]^n \qquad\quad i = 1, \ldots, N \end{aligned} \tag{2.7}$$

where d is the minimum over all the squared Euclidean distances d_{ij}.

The second formulation involves a linear objective function subject to quadratic reverse-convex inequality constraints plus box constraints for the x_i variables. It is easy to show that a strong relation between this problem and the problem formulated in (2.6) exists. In fact, the optimal values of the objectives (2.6) and (2.7) are linked by the following relation

$$r^* = \frac{d^*}{2(d^* + 1)}$$

With a large number of spheres to be packed, the solution is very difficult to find. This difficulty is due to many reasons: on the one hand, it is clear that an optimal solution can be rotated, reflected, or the spheres can be reordered, and hence the number of equivalent optimal solutions blows up as the number of spheres increases. On the other hand, in several cases there even exists spheres in an optimal packing that can be moved slightly while retaining the optimality. Formally, we say that a sphere is free (rattler) if its centre can be moved towards a positive distance point without causing an overlapping with the others.

As mentioned above, approaches that use not only optimization models, but also the geometrical aspects of the problem are often more effective. Hence, in the next section, some of these useful geometrical characteristics will be investigated in detail.

2.4 Some geometrical descriptors

Several examples of geometrical descriptors are available in the literature. In particular, we restrict our attention to accurately describing a generic

configuration of N hard-spheres (atoms or particles) in the three-dimensional Euclidean space.

Steinhardt-Nelson order parameters

Bond orientation analysis have been extensively used in conjunction with molecular simulations in applications involving solid and liquid-crystalline phases [61]. In what follows, we assume that given a sphere i, a set of nearest neighbors $\mathcal{N}(i)$ is defined; any two spheres i and j are said to be connected by a *bond* if they are neighbors, i.e., if $i \in \mathcal{N}(j)$. The set of all bonds is called the bond network. The idea behind the bond orientation analysis is to derive scalar metrics from the information of the bond network (i.e., the set of bond vectors). The precise definition of the bond network is therefore crucial. There are other structure metrics derived from the bond network but the Bond-orientational Order Parameters (BOPs), introduced by Steinhardt et al. in [59], are the most commonly used. In this regard, Steinhardt expressed the angular dependencies of bonds (vectors connecting the central sphere with its neighboring spheres) in terms of the spherical harmonics, that is,

$$q_{lm}(i) = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} Y_{lm}\left(\theta_{ij}, \varphi_{ij}\right) \tag{2.8}$$

where θ_{ij} and φ_{ij} are the azimuthal and polar angles, specifying the orientation of a vector (bond) pointing from spheres i to j; Y_{lm} are the spherical harmonics, and $|\mathcal{N}(i)|$ is the number of neighbours of sphere i.

As θ_{ij} and φ_{ij} depend on the particular choice of the coordinate system, the order parameters $q_{lm}(i)$ also depend on the reference frame. Hence, rotational invariants of spherical harmonics, (i.e., independent of the coordinate system in which θ_{ij} and φ_{ij} are measured) can be constructed as

$$q_l(i) = \left(\frac{4\pi}{2l+1} \sum_{m=-l}^{l} |q_{lm}(i)|^2 \right)^{1/2}. \tag{2.9}$$

The order parameter $q_l(i)$ is a measure of the *local* order around particle i. The authors proposed to use some "suitable" set of bonds for the computation of q_l; specifically, they used a definition based on a cutoff radius of 1.2 times the diameter of a sphere. This way, each sphere that is closer to a given sphere i than a cutoff distance is assigned as the nearest neighbour

of such sphere. Notice that neighborhood definitions based on threshold are widely used but there are many different ways (see, e.g., [34]).

Unfortunately, as liquid has a short-range order, these order parameters cannot distinguish between liquid and solid phases. For this reason, the authors also proposed a rotationally invariant *global* order parameter, Q_{lm}, obtained by averaging $q_{lm}(i)$ over all N spheres as follows

$$Q_{lm} = \frac{1}{N} \sum_{i=1}^{N} q_{lm}(i) \tag{2.10}$$

with the rotationally invariant combinations

$$Q_l = \left(\frac{4\pi}{2l+1} \sum_{m=-l}^{l} |Q_{lm}|^2 \right)^{1/2} \tag{2.11}$$

Now, the global order parameter Q_l is close to zero for the liquid phase and deviates from zero for the crystalline phases. In conclusion, in the authors' opinion [59], it is also important to consider the third-order rotationally invariant combinations

$$W_l = \frac{\sum\limits_{\substack{m_1,m_2,m_3 \\ m_1+m_2+m_3=0}} \begin{vmatrix} l & l & l \\ m_1 & m_2 & m_3 \end{vmatrix} \times Q_{lm_1} Q_{lm_2} Q_{lm_3}}{\left(\sum\limits_{m=-l}^{l} |Q_{lm}|^2 \right)^{3/2}} \tag{2.12}$$

where the integers m_1 , m_2 and m_3 run from $-l$ to $+l$, but only combinations with $m_1 + m_2 + m_3 = 0$ are allowed. The coefficients are the Wigner $3j$ symbols [31].

In each case, the order parameters quantify the degree of crystalline order in the system; therefore, the order parameters assume nonzero (distinct) values in the crystalline phase, which reduce (mostly to zero) in the disordered phase. Usually a few, typically Q_4, Q_6, W_4 and W_6, of these averaged coefficients are considered in order to characterize a configuration. Table 2.1 reports BOPs for some typical structures. In our numerical experiments, BOPs descriptors proved to be the most effective way to represent solutions in a compact way.

Atomic clusters

Further examples of geometrical characteristics for atomic clusters are available from the literature. In [22] the authors defines the *coordination number* of a particle, c_i, as the number of particles located within a sphere of radius r centered around atom i, that is, $c_i = |\mathcal{N}(i)|$. From these coordination numbers four descriptors are derived, namely, the mean, the root-mean-square scatter, the minimum, and the maximum of atomic coordinations. The latter, as justified in [22], may be good candidates for describing the physical structure of a cluster. Two descriptors are also derived from the moments of inertia; these descriptors quantify the departure from spherical shape towards a more oblate or more prolate structure (see [10]).

Another geometrical invariant is reported in [17]; the *shape* of a cluster can be approximately described by shape factors representing the ratios of the lengths of the three axes of a ellipsoid with the same principle moments of inertia. Let $X \in \mathbb{R}^{3\times N}$ be the coordinates matrix, then the eigenvalues of XX^T correspond to the squared lengths of the equivalent ellipsoid axes, with the eigenvectors representing the directions of the axes. The *shape factors* are defined as ratios of eigenvalues

$$s_1 = \frac{\lambda_2}{\lambda_1}, \quad s_2 = \frac{\lambda_3}{\lambda_1} \tag{2.13}$$

where $0 \leq \lambda_1 \leq \lambda_2 \leq \lambda_3$.

Finally, in [49] it is shown how the energy can be decomposed into contributions due to the m nearest neighbor interactions, the *strain* due to deviations from the optimal separation in such pairs, and the non-nearest neighbor contribution

$$V = -m\epsilon + \sum_{\substack{i<j \\ r_{ij}<r_0}} [V(r_{ij}) + \epsilon] + \sum_{\substack{i<j \\ r_{ij}\geq r_0}} V(r_{ij}) \tag{2.14}$$

where r_0 is the cutoff that define neighbours and $r_{ij} = \|x_i - x_j\|_2$ is the separation between i and j. According to the authors, the first two terms in equations are the most sensitive to the structure and the balance between maximizing m and minimizing the strain is a key factor in determining the relative stability of different geometries [50]. Altering the range of the potential affects the strain term for any given geometry, and thereby shifts the balance between different motifs.

Geometry	Q_4	Q_6	W_4	W_6
fcc	0.19094	0.57452	-0.159317	-0.013161
hcp	0.09722	0.48476	0.134097	-0.012442
icosahedral	0	0.66332	0	-0.169754
liquid	0	0	0	0

Table 2.1: 3D bond-orientational order parameters for four typical structures; (fcc) denotes the face-centred cubic geometry; (hcp) denotes the hexagonal close packed geometry

Packing problems

There are many possibilities to define short geometrical descriptors for packing problems, some of which have already been proposed in the literature. Among these descriptors, we might cite the number (average, minimum, maximum) of contacts (spheres tangent to a single sphere), the number of contacts with the border of the container, the number of rattlers, symmetry indices, etc. As we noticed that they were very successful in discriminating configurations in atomic clusters, we have chosen to try the BOPs descriptors even for sphere packing – indeed these descriptors capture in a very precise way the relative position of spheres and can summarize regular patterns, or the absence of any regularity in quite an effective way.

2.5 Exploring the funnels

As mentioned earlier, the choice of a suitable local optimizer is of highest importance in our approach. In particular, we are interested in problem-specific methods that can fast locate the minimum of a funnel, in order to preserve as much as possible the set of features during the descent.

2.5.1 Cluster Surface Smoothing

The PES can be viewed as a collection of various funnels; each funnel represents an atomic cluster configuration (e.g., icosahedral, decahedral, close packed) and contains a huge amount of basins of attraction with the same motif. Nowadays, it is fully recognized that exploration of the energetic landscape of a molecular is more efficiently performed if this landscape is trans-

formed by means of local searches. Several approaches have been proposed, e.g., the methods described in [18] and [41] are based on the introduction of a two-phase refinement search to be employed in place of a standard one in a Basin Hopping scheme. Recent approaches, instead, tend to improve the refinement phase by designing algorithms that go very deep in energy by means of a clever use of local optimization and special purpose modifications to the cluster geometry.

For instance, Funnel Hopping (FH) algorithm proposed in [11] is a GO method used to locate the lowest-energy structures of LJ clusters. The basic idea of this algorithm is to insert a second refinement optimization phase between the first gradient-based local optimization and the global optimization phase. The goal of this second phase is to locate the minimum of the funnel that contains the current configuration with the least cost. Generally speaking, after a first numerical optimization performed with the limited memory quasi-Newton method, the internal part of the cluster is assumed to be well optimized, (due to the "pressure" of the atoms in the outer shell) while the atoms belonging to the surface are still not well organized and contribute to the growth of the value of potential energy.

Hence, the second refinement optimization phase of FH, called Cluster Surface Smoothing optimizer (CSS), aims at locating the motif's minimum energy by rearranging the surface atoms through a sequence of surface perturbations, surface atom relocation and local optimizations. We refer to [11] and to [42] for a deeper look into CSS, particularly about how the surface of a cluster is rearranged at each step. Similar approaches can be found, e.g., in [63] and [9].

2.5.2 Monotonic Basin Hopping

In spite of its simplicity, MBH is known to be a quite efficient algorithm for rapidly exploring funnel bottoms. Note that when multiple funnel bottoms exist, it is usually necessary to run MBH many times from different initial solutions in order to detect the funnel bottom corresponding to the global minimum. Choosing the correct neighborhood size is crucial in MBH and strongly determines its efficiency; if it is too narrow, every local minimum becomes a funnel bottom, and MBH tends to a basic Multistart method. On the opposite side, if the neighborhood is too large, only the global optimum is a funnel bottom, but finding a path which ends at the global minimum is very hard and, again, MBH behaves like a pure Multistart method.

In order to apply the MBH method to our packing problem, we need to reformulate it as a mathematical programming problem for which an efficient local search procedures exist. It is easy to see that problem (2.7) has a linear objective function, but reverse convex [27] quadratic constraints. Thus this problem is non convex and highly multimodal. However, with respect to general problems (2.6) with nonconvex constraints, for which even finding feasible solutions may be an extremely hard task, here feasible solutions are easy to find. As a local optimizer for MBH we adopted SNOPT. Finally, we observe that the GO method used for sphere packing in order to explore the funnels of the problem is radically different from that used in atomic clusters.

2.6 Experiments

In this section, we experimentally evaluate how FS-MSL performs for the two optimization problem types introduced above. The aim of this section is to show that thanks to carefully designed feature extraction procedures and local descent methods, two quite different and challenging GO problems could be solved with a significant computational saving thanks to the implementation of a clustering-based decision.

2.6.1 Experimental setup

In order to assess the effectiveness and efficiency of our method, we have employed an experimental setting which is shared between the scenarios of atomic cluster optimization and sphere packing, described respectively in Sections 2.3.1 and 2.3.2. The idea behind our experiments is first to start an efficient Multistart–like method for both problems based on the repeated execution of a very effective, problem-specific, descent procedure. Our aim in this phase was to determine if the proposed method is capable of obtaining the same quality, that is, reaching the same optima, while saving a large percentage of local searches. Metrics that involve counting the number of calls to an underlying local gradient-based optimizer are widely used in the literature, as they provide a convenient way of evaluating the efficiency of a GO algorithm, being independent from the testing hardware. In this Section, we call a "local search" a single execution of a standard local descent method which, in our experiments, was either L-BFGS [36] or SNOPT [24]. Indeed,

the first is used as underlying numerical optimizer by CSS for atomic clusters (see Section 2.5.1), while the second is the local solver used by MBH for the local optimization of sphere packing instances. Both can be seen as a bottleneck of the overall computational effort.

Our experimental strategy works as follows. First, we ran a number of independent trials (CSS and MBH, respectively, for what concerns cluster structure prediction and sphere packing). For each of these trials, we gathered the optimum reached and the number of local search calls performed before stopping. Then, in order to perform a fair comparison, we reused the same starting points to feed our proposed method, FS-MSL. Note that FS-MSL has a non deterministic behavior, since its evolution does not just depend on the sampled points but also on the order in which batches of sampled points are processed throughout the iterations. To take this fact into account, we randomly permuted the set of generated points – the same used to start Multistart (MS) from – before running our algorithm, and repeated the process 10 times in order to investigate its average behavior.

In order to measure the efficiency and effectiveness of our method, both the quantity of local searches performed as well as the quality of the returned optimum should be taken into account. As it is usual in GO literature, the relevant measure to consider is the average number of local searches per success: if a method, run in an independent way for a certain number of times, calls the local search routine L times in total and, out of the total runs performed, "hits" the putative global optimum H times, we can use the ratio L/H as the *average local search calls per success* index. This measures the average effort expected in order to "see" the global minimum for the first time; this index should be suitably re-defined when the number of hits is null. We define the *hit rate* $\overline{H}$, the local search savings $\overline{S}$ and the *average gain* $\overline{G}$ as

$$\overline{S} = 1 - \frac{L_F}{L_M}, \quad \overline{H} = \frac{H_F}{H_M}, \quad \overline{G} = 1 - \frac{L_F}{H_F} \cdot \frac{H_M}{L_M} \tag{2.15}$$

where H_M and H_F denote the percentage of hits achieved respectively by MS and FS-MSL, while L_F and L_M represent the total number of calls to a local optimizer by the two methods. In our experiments, H_M is equal to 100%, as a single MS run was performed and we defined as hit the best optimum returned by this method (which, in almost all experiments, coincides with the putative optimum in the literature). A negative average gain indicates that our FS-MSL has been indeed beaten by the MS approach, and quantifies

the advantage of using MS against FS-MSL; in particular, $\overline{G}$ is set to -1.0 whenever our method fails to hit the best record obtained by MS on all of the 10 executions.

2.6.2 Numerical results

Our experimental analysis carefully considers the problem of tuning the parameters of FS-MSL and interpreting them; we refer to Section 2.2 for the corresponding notation. The aim of this phase is that of finding experimental settings which guarantee a good trade-off between the quantity of local searches performed and the quality of the returned optima. Indeed, a proper choice of parameters can significantly enhance the behavior of the method achieving a good balance between exploration and exploitation of the feature space. In practice, we would like to design a tuning procedure such that a high hit rate is observed (within the 10 independent runs) and a large gain is achieved. In this scenario, it is a common practice to treat these differences by aggregating all objectives into a scalar function (like the gain $\overline{G}$ previously defined); however, doing so usually results in incomplete exploration of the parameter space. In particular, as it often happens when balancing multiple criteria, it is important to be able to correctly identify potentially conflicting objectives in order to cover a large choice of parameter settings. The Pareto approach not only enables the user to identify existing trade-offs among competing criteria but it also identifies a set of potential solutions associated with several measures. We analyzed the two objectives $\overline{S}$ and $\overline{H}$ in order to separate the computational effort, given by the number of local searches performed, from the accuracy, given by the hit rate. The overall gain measure $\overline{G}$ is already a compromise between savings and accuracy.

In the investigations proposed we used a grid search in order to approximate the Pareto front in the bi–objective space. To accomplish this, we applied an extensive enumeration of combinations by selecting them in a manually specified subset of the multidimensional parameter space (the decision space). The main parameters in our method are:

- the batch size n_b (i.e., the number of new samples at each iteration)
- the value of σ, which governs the likelihood that a full optimization will be started from a point
- the number of "concentration steps" n_c, i.e., the number local descent steps performed in order to concentrate the sample.

Another parameter was also considered: the decision whether the BOPs features should be computed only over the innermost elements (the core) or over the whole solution. So, in summary, before collecting statistics on a wide range of instances for both problems, we ran a set of preliminary experiments only on a tiny number of configurations (we chose to set this number equal to 50), in order to define a small set of parameters which are not dominated, in a Pareto sense, by other choices.

Parameter calibration for atomic clusters and sphere packing

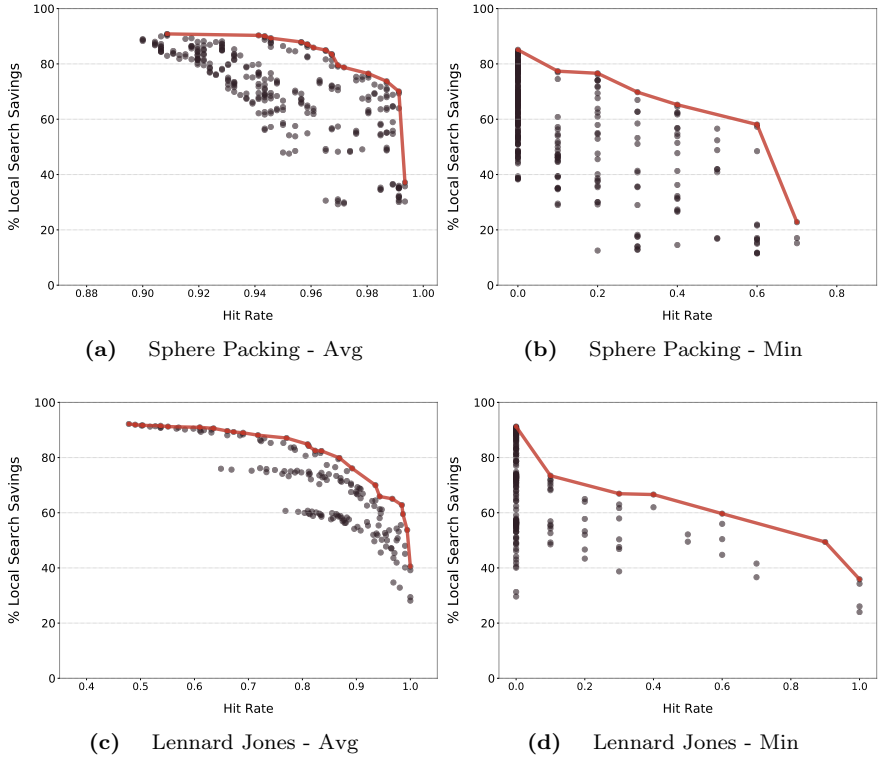
We remind that Lennard-Jones (LJ) cluster conformation has been investigated intensively and several putative minimum energy structures are available. According to the literature, see e.g. [57], while small sized LJ clusters are usually quite similar and mostly based on icosahedral structures, neighboring LJ clusters with sizes greater than 150 atoms may have different geometrical structures. Furthermore, even for clusters with similar or equivalent motifs, atoms on the outermost shell can be found to have different packing style; as an example, both LJ clusters with $N = 176$ and $N = 177$ atoms have icosahedral cores, but the outside atoms are packed in two different multilayer lattice structures. As a consequence, in order to test the optimization ability of a particular setting for this problem, we considered a set of LJ instances starting from 150 up to 200 atoms including icosahedron, decahedron and Marks decahedron structural conformations. In these preliminary experiments for atomic clusters, the batch size n_b was set to 50 while the value of σ and the number of concentration steps were varied in a grid. In order to carefully investigate the effectiveness of our method for avoiding expensive searches, we used a fine grid for σ starting at $\sigma = 0.01$ up to 0.5 in steps of 0.02. The number of concentration steps was varied between 0 and 2. In this scenario, we also considered some specific atomic cluster problem parameters like the decision C whether the BOP features should be computed only over the core or over the whole molecular and the decision A to consider only absolute values for BOP features or not, both binary parameters.

For the sphere packing problem, indeed, the literature reports that the fcc (face-centered cubic) and hcp (hexagonal close-packed) are the most dense known packings of equal spheres. However, computational results show that non-rigid pieces (rattlers) lead to irregular solutions even for small instances. In fact, provably optimal configurations are known for packings of up to 10

spheres only - see [54] and [55]. For this reason, we considered as enough representative for parameter calibration a set of instances starting from 20 up to 70 spheres. For sphere packing we tested three different batch sizes n_b: 10, 20 and 50. For atomic clusters n_b was to 50 as some early experiments showed that in this case changing this parameter was not very significant. The set of values for σ was chosen in the same way as in the Lennard-Jones case; the maximum number of concentration steps was raised to 4. For this problem, we also included the decision whether to consider only absolute values for BOP features or not; as the outer shell for sphere packing is forced by the packing constraint, we choose not to evaluate the effect of computing the BOP only for the internal spheres. Moreover the number of spheres in the experiments was relatively small, so that distinguishing between core and surface spheres is no more significant.

Figure 2.1 shows two possible selection procedures that can be used to support the decision process based on the Pareto principles making the right choice. We remark that the subset of optimal setting relating to the Pareto front are not dominated by any other choices of the parameters, so that the latter should allow FS-MSL to achieve the best performance for solving the optimization problems. In practice, by looking at graphical representations like this one, we obtained a smaller set of parameter choices on which further experiments and analysis were performed. As already mentioned above, to take into account the non-deterministic behavior of FS-MSL, a total of 10 runs for each experimental setting and instance are conducted. Each symbol (dots) in figures represents the hit rate and the savings $\overline{S}$ associated to a specific choice of FS-MSL parameters. In particular, in the left sub-figures, (a) and (c), the two represented objectives are the average savings and the average success rate over all instances, i.e., they were obtained by averaging the corresponding success rates and savings in runs performed with different instances of the problem; the right sub-figures, (b) and (d), instead, considers the pairs composed by the minimum achieved values for the savings and the success rate – in other words, in this figure the position of a dot corresponds to the worst behavior observed in running 10 times the algorithm for different instances of the problem. In all figures we emphasized (red dots) non-dominated points corresponding to Pareto points. In all the experiments, in order to define the hit rate, a tolerance equal to 10^{-6} was used for both problems.

The figure is just a pictorial proof of the fact that a relatively small set



(a) Sphere Packing - Avg **(b)** Sphere Packing - Min

(c) Lennard Jones - Avg **(d)** Lennard Jones - Min

Figure 2.1: Pictorial representation of the compromise between savings and hit rate for several parameter settings. Each point is associated to a specific parameter setting. In (a) and (c) the averages computed over 50 problem instances are reported, while in (b) and (d) we show the minimum (worst) values. Solutions on the approximation of the Pareto front (red line) are not dominated by any other solution (gray disks).

of parameter settings is sufficient to obtain a uniformly good behavior of our method for different problem instances.

Even restricting ourselves to Pareto non dominated solutions, however, we are eventually left with quite a large set of candidates. In order to reduce the set of parameter setting to experiment with, we chose to analyze only the non-dominated solutions on the Pareto front with hit rate greater than 0.9. The results are summarized in Table 2.2.

The best experimental settings might be chosen within those that appear on both the Pareto selection procedures, representing a good compromise between conflicting criteria. Notice that $\sigma = 0.07$ appears as a common choice for both problems with an impressive success rate (> 0.98) and good savings. We thought also that it might be interesting to see how those settings associated with higher savings perform in practice.

In accordance with these preliminary experiments, Table 2.2 reports (the rows in bold) the values for the batch size n_b, the threshold σ, the number of concentration steps n_c and, in addition, further specific problem-dependent parameters, that we then used in the numerical experiments.

Lennard Jones and Morse clusters

We first consider the experiments for Lennard Jones clusters of size $N = 150$ up to 200 atoms. In this section we would like to show the performance of the proposed algorithm for each size. Due to the excellent quality of the CSS optimizer, a MS method equipped with CSS, which we denote by MS-CSS was always able to provide, within the 1000 trials, the *putative* global optimum known in the literature. We would like to point out here that we are not referring to MS as an ideal GO procedure; however, using a global scheme equipped with CSS – a pure local optimizer with a powerful depth search capability, which does not attempt to jump outside the current funnel during the surface smoothing – yields a very effective GO method, capable of finding all known putative optima.

Figure 2.2 reports the *average gain* $\overline{G}$ (blue line) and number of hits (red dashed line) on ten different permutations of the FS-MSL trials, on the instances that were chosen in the tuning phase. When the observed gain is above the $\overline{G} = 50\%$ horizontal line, this means that our method allows to save more than half of the L-BFGS calls needed by the multi-start scheme. In Figure 2.2 (a), one CSS step is carried out as MSL's concentration step and the threshold parameter σ is set to 0.03. These results clearly show how our scheme is able to successfully predict the putative global optimum while saving roughly 54% of the computational effort. This happens also for extremely hard instances, even if the corresponding funnel is very narrow, getting rid of a large amount of redundant configurations.

In the experiment depicted in Figure 2.2 (b), the starting randomly generated points of MSL are concentrated only by running a single L-BFGS optimization: therefore, the concentration step (which precedes the cluster-

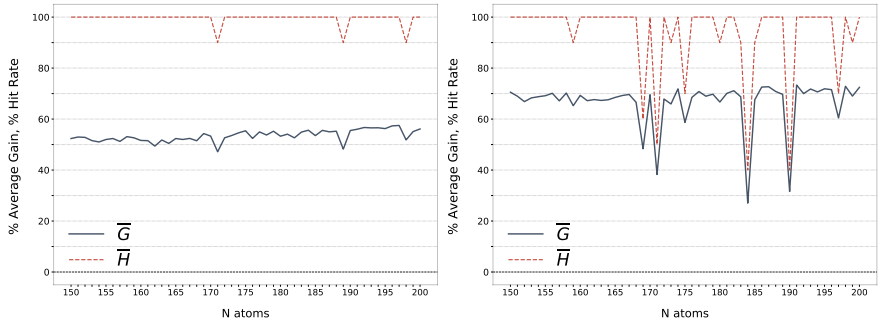
Lennard Jones Clusters

Avg						Min					
$\overline{S}$	$\overline{H}$	n_c	σ	A	C	$\overline{S}$	$\overline{H}$	n_c	σ	A	C
0.699	**0.935**	**0**	**0.03**	*****	*****	0.912	0.0	0	0.5		
0.658	0.943	1	0.1	*	*	0.734	0.1	0	0.03		*
0.650	0.966	1	0.09	*	*	0.669	0.3	1	0.15	*	*
0.627	**0.984**	**1**	**0.07**	*****	*****	0.665	0.4	0	0.03	*	*
0.595	0.986	1	0.05	*	*	**0.596**	**0.6**	**1**	**0.07**	*****	*****
0.537	**0.994**	**1**	**0.03**	*****	*****	**0.494**	**0.9**	**1**	**0.03**	*****	*****
0.405	**1.0**	**1**	**0.01**		*****	**0.358**	**1.0**	**1**	**0.01**		*****

Sphere Packing

Avg						Min					
$\overline{S}$	$\overline{H}$	n_c	σ	A	n_b	$\overline{S}$	$\overline{H}$	n_c	σ	A	n_b
0.858	**0.960**	**2**	**0.4**	*****	**10**	0.850	0.0	1	0.5		20
0.849	0.965	2	0.35	*	20	0.773	0.1	2	0.25		20
0.835	0.967	2	0.3	*	20	0.766	0.2	2	0.35	*	20
0.795	0.969	2	0.15		20	0.697	0.3	2	0.15		20
0.787	0.971	2	0.2	*	10	0.652	0.4	2	0.11		50
0.765	0.980	2	0.11		50	**0.580**	**0.6**	**2**	**0.07**		**50**
0.737	0.987	2	0.09		50	0.227	0.7	2	0.01		50
0.700	**0.991**	**2**	**0.07**		**50**						
0.371	0.993	4	0.01		50						

Table 2.2: This table reports a selection of the non dominated points on the Pareto frontiers obtained in the tuning phase for both problems. We chose to select only the ones whose average hit rate was greater than 0.9. (*) `True`.



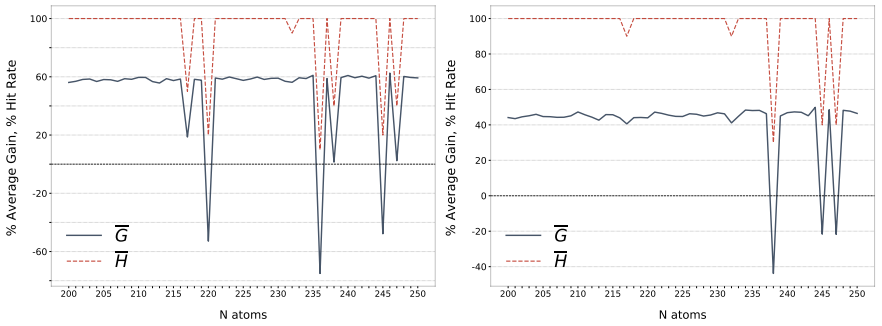
(a) $\sigma = 0.03,\ n_c = 1$ **(b)** $\sigma = 0.03,\ n_c = 0$

Figure 2.2: Gain profiles for the selected testing ground. In (a), the concentration phase consists of running a single CSS step, while in (b), randomly generated configurations are simply optimized once through L-BFGS.

ing and selection phase) does not employ CSS's surface smoothing technique to group the liquid-like configurations around their basins of attraction, and it boils down to a pure numerical optimization. As expected many hits are lost due to a shallower concentration, even if the average gain on all of the cluster sizes is the highest seen so far, namely $\overline{G} = 0.69$. Notice that an important design choice we made was to extract the fingerprint only from the atomic cluster's *core* as suggested by non-dominated solutions reported in Table 2.2. This should be motivated by the instability of the atoms lying on the outer shell and numerical evidence proves this intuition to be exact.

With these experiments, we showed how MSL is able to preserve the quality of an excellent GO scheme, while saving a large percentage of local searches. An analysis of these results from a chemical-physics point of view appeared in [4]. These results confirm how our method is able to predict the putative global optimum even at quite large size, and even for "magic number" instances well known in the literature for being hard to globally optimize, like, e.g., the $N = 98$ instance for Lennard Jones clusters [40]. In particular, Table 2.3 reports numerical results for "magic number" instances with sizes up to 150 atoms. Confirming our previous findings, our approach was able to locate all putative global minima in 100% runs saving roughly 50% of the computational effort. In order to better appreciate the quality of the method, in the table we also report the average number $\overline{LS}$ of local

searches needed to see the global optimum for the first time in 10 runs. It is worth noticing that these numbers can be significantly improved by reducing the batch size: in fact the stopping criterion used to compute $\overline{LS}$ was to stop after the putative optimum was found in the current batch. Thus, at least $n_b = 50$ local optimization runs are always executed, and from a selection of the resulting optima a complete CSS run is executed.



(a) $\sigma = 0.03,\ n_c = 1,\ A =$ `True` **(b)** $\sigma = 0.01,\ n_c = 1,\ A =$ `False`

Figure 2.3: Gain profiles for Lennard-Jones clusters with 200-250 atoms. In (a), our method was able to save roughly 48% of the computational effort using values for σ and n_c equal to 0.03 and 1, respectively. In (b), the experimental setting with the highest hit rate on the Pareto front is considered; as we expected, the average gain $\overline{G}$ is decreased to 41.19% but a higher hit rate was achieved.

Finally, in order to test the quality of our method and of the approach we followed to choose a small set of parameters for the algorithm, we extended our experiments to an out of sample set of larger instances; an overview of results for Lennard-Jones instances with $N \in [200, 250]$ is reported in Figure 2.3.

In [4], we also considered Morse clusters; we carried out experiments on Morse cluster instances from $N = 40$ up to 129 atoms, running 50 "pure" trials of FS-MSL for each configuration. By pure we mean that we were not simulating our scheme based on existent MS-CSS runs (as we did in the experiments for parameter calibration), and that each trial is totally independent from the others. Our method proved to be able to detect the putative optimum of a set of instances which are among the hardest to be solved. We refer to [4] for detailed numerical results.

Table 2.3: Gain observed for FS-MSL and average number of local searches before seeing the putative global optimum for the first time for "magic number" instances. The main parameters used in these experiments are $\sigma = 0.03$, $n_c = 1$, $n_b = 50$ and $A =$ `True`.

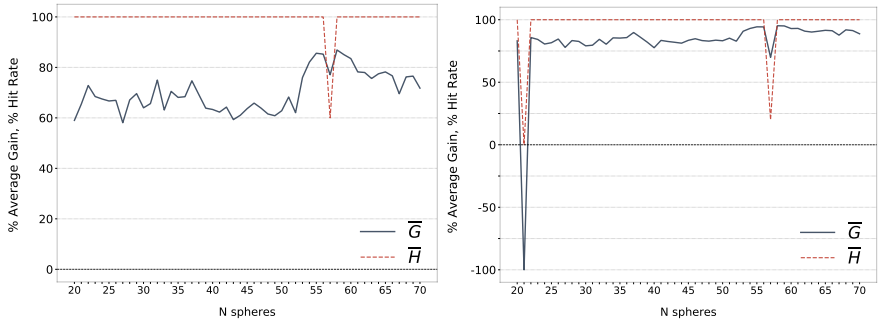
N	Geometry	Ref.	% $\overline{G}$	$\overline{LS}$
38	O_h	[20]	49.08	3326.1
75	D_{5h}	[20]	46.65	4158.7
76	C_s	[20]	47.90	1101.1
77	C_{2v}	[20]	48.46	718.4
78	C_s	[32]	48.41	310.8
98	T_d	[33]	54.25	376.5
102	C_{2v}	[19]	55.19	1420.0
103	C_s	[19]	55.27	3773.6
104	C_{2v}	[19]	56.64	1432.8
107	C_s	[32]	55.97	588.8

Sphere packing

The results obtained in atomic clusters were encouraging enough to motivate the use of the same algorithmic idea and even the same geometrical feature descriptor, the BOP four-valued array, for the sphere packing class of problems. As a local descent procedure, we employed the MBH method, equipped with SNOPT as a local search method. MBH is known to be a quite efficient algorithm to explore funnel bottoms with respect to a reasonably chosen neighborhood. After a few experiments, and drawing from previous experience, we choose as a neighborhood the hypercube centered at current configuration with edge length equal

$$\Delta = \frac{1}{\sqrt[3]{N}}$$

This choice is motivated by the desire to explore a funnel bottom without jumping outside the current region of attraction. MBH was run with a stopping criterion consisting on 1000 consecutive iterations without any improvement. In this setting, MBH was always able to provide, within the 1000 trials, the putative global optimum, with the only exception of instance $N = 57$.



(a) $\sigma = 0.07,\ n_c = 2,\ n_b = 50$ **(b)** $\sigma = 0.4,\ n_c = 2,\ n_b = 10$

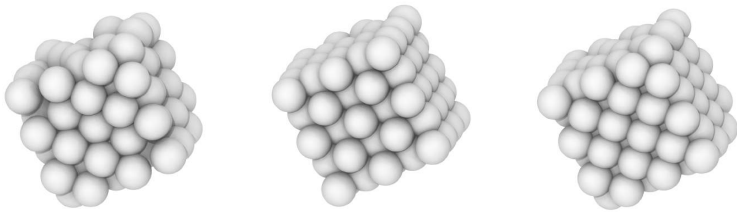
Figure 2.4: Average gain $\overline{G}$ (blue) and hit rate (red) for the sphere packing problem with $N = 20$ up to 70. In all experiments a single SNOPT step has been performed as MSL concentration step and two different σ values.

As we did with atomic clusters, we first ran a pure MBH method on the whole set of instances. After this run, we simulated 10 runs of FS-MSL generated by first randomly shuffling the sample of 1000 starting point used by MBH. After this shuffling procedure, our FS-MSL was used to simulate the decisions whether to start a full MBH descent or not from each concentrated sample point. A run of FS-MSL was considered as successful if the putative globally optimal value was reached for all instances except $N = 57$: in this case a run was considered as a successful one if the best configuration found by MBH was reached. Results for the sphere packing problem are reported in Figure 2.4. In Figure 2.4 (a), two SNOPT steps are carried out as concentration steps and the threshold parameter σ is set to 0.07.

These results clearly show how our scheme, with a relatively small computational effort, compared to the original MBH, is capable of finding all of the putative optima while saving roughly 70% of the computational effort. This happens also for extremely hard instances, where the putative optimum is seen only few times or, in the case of 57, when we use the best configuration as target (distance from the putative $-4.06669 * 10^{-6}$).

In the second setting, Figure 2.4 (b), the threshold σ is risen from 0.07 to 0.4 keeping the same number of concentration steps. The results are impressive: FS-MSL fails to hit the putative optimum only on instance $N = 21$, but the average gain on the remaining instances is the highest seen so

far with a value equal to 85.88 %.



(a) $N = 83$ **(b)** $N = 96$ **(c)** $N = 109$

Figure 2.5: New putative optimal configurations for sphere packing found by our algorithm.

After these initial numerical experiments devoted mostly to find reasonably good parameter settings, we extended our experiments and ran our FS-MSL method on larger instances, with the parameter settings obtained from Table 2.2.

Extending numerical experiments up to $N = 120$ spheres, we could find all putative globally optimal configuration. Moreover, we discovered new, improved, putative optima for the sphere packing problem with $N = 83, 96, 109$ spheres (see Figure 2.5); these new putative optima have been certified and inserted in the `www.packomania.com` repository. It is worth observing that in the sphere packing context, it was considered very unlikely that improved packings at these dimensions could be found. Our approach, thus, confirmed to be not only a way to save expensive local searches, but thanks to the capability of deciding to start a complete search only from selected promising points, it could deliver improved solutions even for widely studied instances.

2.7 Conclusions

What we have shown in this chapter is a general idea for computationally efficient algorithms which "learn" from their experience and avoid repeating expensive searches which might eventually lead to already discovered optimal configurations. The gain in efficiency (measured in local searches saved) obtained in the two different problem types presented is very significant, reaching even more than 85% in the sphere packing problem. Our aim was

to show that the decision whether to stop early or to go deep in search of a low level local optimum can be made through the idea of clustering based on a small set of selected features. Of course both the choice of the features as well as the choice of the deep exploration descent method are problem-dependent. But we have shown that even with two quite different problems, a common set of features can be identified which transforms a moderately efficient GO method into a very fast and reliable one.

The experiments we have done with two very different GO problems are in our opinion very promising for several reasons:

- we have shown that a very small number of features can be successfully used in order to efficiently distinguish high-dimensional solutions
- quite surprisingly, we found that we could use the same feature set for atomic clusters and for sphere packing
- we performed a set of experiments in order to select a very restricted set of parameters with excellent performance for both problems and for both criteria: local search savings and hit rate
- we introduced the idea of non dominated selection in the parameter space
- further experiments made with the parameters chosen this way enabled us to confirm many putative optima and to confirm the high computational savings
- for sphere packing, our method let us discover three new putative optima.

A possible extension of the method proposed here might be that of devising a way to be able to invert in some way the map from coordinates to features in order to design a method which might be able to generate new solutions based on sufficiently new features suggested by a clustering method.

Chapter 3

Enhancing the Performace of Memetic Algorithms

Evolutionary Algorithms (EAs) sample the objective function landscape at many different points, but unlike the Multistart approach where each base point evolves in isolation, points in an evolutionary strategy population influence one another by means of recombination. Memetic Algorithms (MAs) are commonly implemented as EAs endowed with a local search component. In this chapter we show that the cost of local searches in memetic approaches may be reduced with a clustering method by avoiding multiple rediscoveries of the same local optima. In addition, this clustering method can supply informations that can be used to maintain the diversity in the population. A detailed computational comparison between different variants of memetic Differential Evolution (DE) approaches is presented. The computational comparison reveals that our clustering-based approach usually outperforms the other approaches on a set of test functions with different characteristics chosen from the GO literature.

3.1 Introduction

Memetic Algorithms (MAs) represent one of the recent growing areas of research in evolutionary computation. The term MA is used to denote the synergy between evolutionary (or any population-based) approach and local

improvement procedures. This paradigm has emerged in the last quarter of previous century to become, nowadays, one of the most widely used solving approaches; this is supported by disparate applications ranging from machine learning to bioinformatics, from economics to physics just to mention a few. In particular, we refer to [46] for an excellent review of memetic algorithms.

As for what concerns the exploitation and exploration balance of these approaches, it is typically the case where a population-based component is used to guide the exploration of the search space, providing interesting solution to be refined via local improvement operators (e.g., heuristics, approximation algorithms, local search methods). However, in some applications, local searches may be computationally expensive and this holds, for example, when the number of variables increase. In this regard, the local search frequency, i.e., the proportion of members of the population from which a local search is started, may have a huge impact on the performance of the method. It is therefore clear that, in order to come up with efficient memetic solvers, one of the most relevant features to take into account is the balance between the global and the local phases.

In general, a population-based algorithm is supposed to work in high diversity conditions during the early iterations of the optimization, then progressively lose diversity in the proximity of the basins of attractions, and eventually lose all diversity when the global optimum is detected. The latter condition means that large part of the population may be represented by a unique basin of attraction. Clearly, a more intense and repeated local optimization provides greater chance of convergence to the minimum but limits the amount of evolution needed to locate unexplored basins. Hence, care should be taken to balance the computational budget allocated to the local optimization if the population is converging around previously found optima.

According to the literature, regarding the selection of individuals on which local improvement operator is applied, in [26] the authors propose to decrease the local search frequency for each individual by the number of duplicates contained in the population. This works around the problem of having redundant local searches on the same solutions. In [64] the authors proposed a method which uses EA to perform the exploration of expensive objective functions; then, every several generations, the algorithm would cluster the population using the k-means algorithm to identify the basins of attraction.

In this chapter, instead, we would like to generalize this approach towards using statistical techniques like Multilevel Single Linkage; if the population consists of several clusters, we can hope to easily detect solutions likely within the same basin of attraction upon which it may not be fruitful to apply local search. In our experimental analysis, we investigated the use of clustering on a quite simple but very effective memetic differential evolution schema. Our aim was to determine if the proposed strategy is capable of obtaining the same quality of the original method but at a fraction of the computational cost. We point out that this approach is completely independent with respect to the choice of the Evolutionary Algorithm (EA) and that many other combinations are possible.

3.2 Memetic Algorithms

The rationale behind MAs is to provide a more effective and efficient global optimization method by compensating EA in local exploitation. This hybridisation can achieve a good trade-off between the exploration of the domain search and the exploitation of found solutions, so, first, it is important to obtain good results in EA to offer interesting properties when applying them to multimodal optimization problems.

Recently, various MAs have been proposed based on different population-based global search mechanisms such as Particle Swarm Optimization [29], Ant Colony Optimization [16], or Differential Evolution [60]; in this work we restricted our attention to the latter. Like other EAs, DE is a population-based, stochastic global optimizer capable of working reliably in nonlinear and multimodal environments. Using a few parameters, DE exhibits an overall excellent performance for a wide range of benchmark functions. The advantages of DE, such as ease of use and high convergence characteristics make it, nowadays, a high-class technique for real-valued parameter optimization. Although DE was designed using the common concepts of EAs, such as multipoint searching, use of recombination and selection operators, it has some unique characteristics that make it different from many others in the family. In the next section, we discuss DE in details and, in order to test our strategy with effective memetic solvers, some examples of hybridization of DE with local search are presented. Without loss of generality, we assume here that we deal with minimization problems. Obviously, everything discussed here can easily be adapted to the maximization case.

3.2.1 Differential Evolution

Differential Evolution (DE) is a simple population-based stochastic method for GO, which performs extremely well on a wide variety of test and engineering problems. In 1995, Price and Storn developed DE to be a reliable and versatile continuous function optimizer that was also easy to use. Originally inspired by Genetic Algorithms (GAs) and geometrical arguments, DE manipulates candidate solutions with simple arithmetic operations such as addition, subtraction, and multiplication. Like nearly all EAs, DE comprises a population of p search variable vectors

$$P = \{x_1, \ldots, x_p\} \subset \mathbb{R}^n$$

where $x_i^{(j)}$ denotes the j-th component of the n-dimensional vector $x_i \in \mathbb{R}^n$. The method maintains and evolves the whole population through successive iterations of *mutation*, *crossover* and *selection*. The idea distinguishing DE from GAs is a very simple schema for recombining variable vectors consisting of differential mutation and arithmetic crossover. In particular, DE produces a population of *trial* vectors by adding a scaled, randomly sampled, vector difference to a third vector. If the resulting vector yields a objective function value lower than a predetermined member, the newly generated vector replaces the vector with which it was compared. Extracting distance and direction information from the population to generate random deviations results in an adaptive scheme with interesting convergence properties. The procedure of classic DE is discussed in detail to serve as a basis for theoretical analysis and algorithm development in later sections.

Let $f : \mathbb{R}^n \to \mathbb{R}$ be the objective function of the problem with given boundary constraints on search variable vectors collected in two n-dimensional vectors x_L and x_U. We consider the following problem

$$\min_{x_L \leq x \leq x_U} f(x) \tag{3.1}$$

Initialization. In order to establish a starting point for seeking, the population S must be initialized. Often there is no more knowledge available about the function to be optimized than the boundaries of the problem variables. In this case, the initial population may be randomly generated according to a *uniform* distribution, that is,

$$x_i^{(j)} = x_L^{(j)} + \mathcal{U}(0,1) \cdot (x_U^{(j)} - x_L^{(j)}) \qquad \forall j \leq n,\ \forall i \leq p$$

where $x_L^{(j)}$ and $x_U^{(j)}$ are the lower and upper bounds for the j-th component of the variable vector, respectively. As a rule, we will assume a uniform probability distribution for all random decisions unless otherwise stated.

Mutation. The aim of the differential mutation is to obtain an intermediate vector based on the current population P. Equation (3.2) shows how to combine three different, randomly chosen vectors to create a trial vector

$$v_i = x_{r_1} + F \cdot (x_{r_2} - x_{r_3}) \tag{3.2}$$

where r_1, r_2 and r_3 are distinct indices chosen at random in the set $\{1, \ldots, p\} \setminus \{i\}$. None of these points should coincide with the current *target* vector x_i. The *scale* factor, $F \in (0, 2)$ is a positive real number that controls the rate at which the population evolves. In classic DE, the scale factor is a single parameter used for the generation of all trial vectors, while in many adaptive schemes each individual x_i is associated with its own mutation parameter F_i.

Crossover. To complement the differential mutation search strategy, the trial vector u_i is found using the target x_i, the intermediate vector v_i and a crossover rule. The most common form of crossover is uniform and is defined as

$$u_i^{(j)} = \begin{cases} v_i^{(j)} & \text{if } \mathcal{U}(0,1) \leq \text{CR} \\ x_i^{(j)} & \text{otherwise} \end{cases} \tag{3.3}$$

where the superscript (j) represents the j-th component of respective vectors. The crossover probability, $\text{CR} \in [0, 1]$, roughly corresponds to the average fraction of vector components that are inherited from the mutation vector v_i.

Selection. DE uses simple one-to-one survivor selection where the trial vector u_i competes against the target vector x_i. The vector with the lowest *fitness* value survives into the next generation. The selection process may be outlined as

$$x_i = \begin{cases} u_i & \text{if } f(u_i) < f(x_i) \\ x_i & \text{otherwise} \end{cases} \tag{3.4}$$

where f is the function to be minimized.

The method reported above is one of the possible variants of DE, but different implementations of the basic scheme can be identified thanks to a simple taxonomy which takes the form of `DE/x/y/z`. Here `x`, which takes the values `rand` or `best`, denotes the base vector to be perturbed, `y` denotes the number of difference vectors considered for perturbation, i.e., the difference between two distinct randomly selected elements in the population, and `z` representing the crossover method. The latter value can be `bin`, indicating that the choice is made following a binomial probability distribution function, or `exp` when the choice on which components to keep from the current population is based on an exponential distribution The following are different mutation schemes frequently used in the literature:

1. Let r_1, r_2, r_3 be distinct and randomly generated in $\{1, \ldots, p\} \setminus \{i\}$; `DE/rand/1` scheme denotes

$$v_i = x_{r_1} + F \cdot (x_{r_2} - x_{r_3});$$

2. Let r_1, r_2 be distinct and randomly generated in $\{1, \ldots, p\} \setminus \{i\}$; `DE/target-to-best/1` scheme denotes

$$v_i = x_i + F \cdot (x_{best} - x_i) + F \cdot (x_{r_1} - x_{r_2})$$

   where x_{best} is the best member of the current population;

3. Let $r_1, \ldots, r_4$ be distinct and randomly generated in $\{1, \ldots, p\} \setminus \{i\}$; `DE/best/2` scheme denotes

$$v_i = x_{best} + F \cdot (x_{r_1} - x_{r_2}) + F \cdot (x_{r_3} - x_{r_4});$$

   where x_{best} is the best member of the current population;

3.2.2 Hybridization of DE

Although DE belongs to the elite EA class in consideration of its convergence properties, its overall performance does not meet the requirements for all classes of problems. In accordance with the earlier discussion, hybridization with a local search procedure can accelerate DE by improving its exploitation capability. In this regard, to propose a novel hybrid DE algorithm, a number of important aspects should be taken into account:

Local procedure employed. The choice of the local procedure (LS) may have a relevant impact on the performance of the approach. Most of the strategies refine the solutions of each generation by applying efficient LSs, like gradient descent or hill-climbers, but crossover operators for local refinement are also employed. A crossover operator is a recombination operator that produces new points around the population; for this reason, it may be considered as a move operator in a local search strategy.

Frequency of refinements. LSs can be applied to every member of the population or with some specific probability and with various replacement strategies. In particular, the frequency of refinements denotes the proportion of members of the population from which a LS is started. A careful selection should take into account two conflicting objectives: on the one hand to avoid the waste of computational effort caused by the multiple detection of the same local minimizer, on the other hand, to avoid not to start local searches from promising points. For example, the algorithm proposed in [47] uses simple a local hill-climbing techniques with simplex crossover operator (see [69]) to speed up convergence of only the best individual in the population, the rest of the search is done by classical DE. In [35], instead, a local search operator, i.e., random walk with direction exploitation, is applied to each DE individual with predefined fixed probability; the number of function calls used by each local search is also fixed.

Incorporation of domain-knowledge. As a consequence of the no free lunch theorem no efficient general purpose solver exists. It is clear that excessive reliance on purely stochastic evolutionary processes, with no expert guidance or external knowledge incorporation, will often lead to performance that is simply too slow for practical applications. These simple observations provided the insight for conceptualizing algorithms wherein the basic mechanisms of evolution should be augmented with problem-specific knowledge.

In what follows, we will restrict our attention to a memetic approach for the solution of continuous global optimization problems called Memetic Differential Evolution (MDE) [39]. Some interesting results are presented which concern the application of this method to standard GO test functions. In order to make the introduction of our approach clear, MDE and some

improved variants, namely G-MDE, D-MDE and H-MDE, will be described in more detail in the next sections. We point out that the latter was used as a base for investigating the effects of the use of clustering on memetic algorithms.

3.2.3 Memetic Differential Evolution

In [39] the authors explore the behavior of a quite standard DE algorithm applied to continuous objective functions transformed by means of a local procedure. As we already explained, the only difference with respect to the classical DE approaches is that a local search is performed form each newly generated point. In spite of the fact that MDE is a rather simple approach, it proved to be very efficient both over standard GO test functions and over some packing problems [39].

Algorithm 2 Memetic Differential Evolution (MDE)

Require: The population matrix P and the population size $p = |P|$;
the parameters $F \in (0, 2)$ and $\mathrm{CR} \in [0, 1]$ for DE

for all $i \in \{1, \ldots, p\}$ **do**
  Randomly choose $r_1, r_2, r_3 \in \{1, \ldots, p\} \setminus \{i\}$, all distinct;
  if $\mathcal{U}(0, 1) \leq \mathrm{CR}$ **then**
    $u_i^{(j)} \leftarrow x_{r_1}^{(j)} + F \cdot (x_{r_2}^{(j)} - x_{r_3}^{(j)})$
  else
    $u_i^{(j)} \leftarrow x_i^{(j)}$
  end if
  $q_i \leftarrow \mathcal{L}(f, \Omega, u_i)$
  if $f(q_i) < f(x_i)$ **then**
    $x_i \leftarrow q_i$
  end if
end for
return The new population matrix P

The MDE method is summarized in Algorithm 2; in the pseudocode, $\mathcal{L}$ is a local solver which, given an objective function f over Ω and a starting point $\bar{x}$, returns $x_* = \mathcal{L}(f, \Omega, \bar{x})$, a *local* minimizer for f where $f(x_*) \leq f(x)$ for all x in a neighborhood of x_*. Note that the selection mode is the same as in DE, where the procedure simply compares the current i-th member of the population x_i with the minimizer q_i, and replaces the former with the latter if $f(q_i) < f(x_i)$. This choice helps to drive the population towards

local minimizers improving the convergence characteristics of the original DE algorithm.

With respect to the aspects discussed in Section 3.2.2, the probability of starting a local search is equal to one for each newly generated point; no problem-specific knowledge is incorporated; the strategy can be classified as `DE/rand/1/bin`. As mentioned earlier, however, the strategy to call local search with a fixed frequency may have a strong impact on the performance of the method. In fact, a large amount of redundant computation may be incurred on evaluating the same solutions.

3.2.4 Some variants of MDE

In this section, we discuss three variants of MDE, namely the greedy variant (G-MDE), the distance variant (D-MDE), and the hybrid variant (H-MDE) introduced in [7]. Due to their different structures and exploration capability, a comparison with the latter may be a good indicator of the quality of our approach. According to the authors, indeed, G-MDE turns out to be a robust and efficient approach for single-funnel test functions while D-MDE, where the generation function is coupled with a diversification mechanism, appears to be the best choice for multi-funnel functions.

Generally speaking, all these methods can be viewed as a variant of the DE generation and/or selection mechanism. However, we emphasize that, in our numerical analysis, we choose not to investigate any sophisticated memetic algorithms, because even simple and easy to implement approaches are able to return very good results as reported in [6].

Greedy MDE (G-MDE)

In order to generate a trial point for the population member x_i , the procedure randomly selects another element of the population x_j and tries to predict the function value of the perturbed point by computing the difference between the objective function values at x_i and x_j, that is, $\Delta f = f(x_i) - f(x_j)$. According to the observed function values, G-MDE perturbs the current point along a direction which is, presumably, a descent one. This slight modification of the generation function of MDE is called *greedy* because it favors the directions $(x_j - x_i)$ which could lead to better function values if $\Delta f > 0$; otherwise, the method reverses the direction as follow

$$v_i = x_i + \varphi F \cdot (x_{r_1} - x_i) \tag{3.5}$$

where $\varphi = \text{sign}\,(\Delta f)$ and r_1 is uniform randomly selected in $\{1, \ldots, p\} \setminus \{i\}$.

In the G-MDE generation function, see Algorithm 3, the perturbations are applied to each population member x_i and the result of the perturbation is compared with x_i itself. The authors explain that such greedy function can be very effective for *single-funnel* functions, but, at the same time, claim that a too fast convergence to funnel bottoms may be a negative feature in terms of diversity maintenance.

Algorithm 3 The generation procedure for G-MDE

Require: The population matrix P and the population size $p = |P|$; the index of the evaluated point i; the parameters $F \in (0, 2)$ and $\text{CR} \in [0, 1]$ for DE

Randomly choose $r_1 \in \{1, \ldots, p\} \setminus \{i\}$;
if $f(x_i) > f(x_{r_1})$ **then** $\varphi = 1$
else $\varphi = -1$
end if
if $\mathcal{U}(0, 1) \leq \text{CR}$ **then**
    $v_i = x_i + \varphi F \cdot (x_{r_1} - x_i)$
else
    $u_i^{(j)} \leftarrow x_i^{(j)}$
end if
$q_i \leftarrow \mathcal{L}(f, \Omega, u_i)$
return The candidate vector q_i

Distance MDE (D-MDE)

In the D-MDE, in order to counterbalance the greediness of G-MDE, some diversity within the population is maintained. In particular, the generation function is still the greedy one, but now the selection function is described as follows

$$x_k^* = \begin{cases} q_i & \text{if } f(q_i) < f(x_k^*) \\ x_k^* & \text{otherwise} \end{cases} \tag{3.6}$$

where

$$x_k^* = \underset{\substack{x_k \in P \\ k \neq i}}{\arg\min}\ d(q_i, x_k) \tag{3.7}$$

with $d(\cdot, \cdot)$ as a distance (dissimilarity) measure between local minimizers. The distance measure employed by the authors is based on function values,

i.e., $d(q_i, x_k) = |f(q_i) - f(x_k)|$. However, other distance measures can be defined, e.g., the Euclidean distance, including some problem-specific ones.

Different from G-MDE, the selection function is not applied any more between the current population element x_i and the corresponding local minimizer, but between the member of the population x_k^*, nearest to the newly generated candidate q_i, and the candidate itself. The authors claim that the new selection function may guarantee a population more spread around the feasible region, since, e.g., there cannot be different copies of the same point within the population, thus causing a slower convergence of the algorithm but on the also an increase of the search area explored by the algorithm.

Algorithm 4 The selection procedure for D-MDE

Require: The population matrix P and the population size $p = |P|$; the index of the evaluated point i; the candidate vector q_i

1: $x_k^* \leftarrow \arg\min_{x_k \in P,\ k \neq i}\ d(q_i, x_k)$
2: **if** $f(q_i) < f(x_k^*)$ **then**
3: $\quad x_k^* \leftarrow q_i$
4: **end if**
5: **return** The new population matrix P

Hybrid MDE (H-MDE)

In the two previous subsections we have presented two different approaches which appear to be effective in different situations. Numerical results reported in [6, 7] show that G-MDE converges fast and appears to be suitable for single-funnel functions, while D-MDE 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. In [7] an hybrid approach which is able to self adapt to the properties of the function to be optimized is introduced; the latter uses both the greedy and the distance selection mechanisms previously discussed. In particular, the generation of the candidate points is always the greedy one. If $f(x_i) - f(x_j) > 0$, that is, $\varphi = 1$, the method also use the G-MDE selection function, thus forcing the greedy behavior. Otherwise, the D-MDE selection function is employed. This way, the H-MDE can avoid too fast convergence of the population towards regions which do not contain the global minimizer.

3.3 Clustering-based MDE

The main idea, in order to enhance the performance of a memetic approach, is based on the fact that it is unnecessary to start twice a local search on the same basin of attraction since the same local optimum will be rediscovered again and again. In this regard, newly generated points belonging to the

Algorithm 5 Clustering-based MDE (C-MDE)

Require: The population matrix P and the population size $p = |P|$;
the parameters $F \in (0, 2)$ and $\mathrm{CR} \in [0, 1]$ for DE

$\mathcal{S}_k \leftarrow \mathcal{S}_{k-1}$ $(\mathcal{S}_0 = P)$
$\mathcal{O} \leftarrow \emptyset$
for all $i \in \{1, \ldots, p\}$ **do**
  Randomly choose $r_1, r_2, r_3 \in \{1, \ldots, p\} \setminus \{i\}$, all distinct;
  if $\mathcal{U}(0, 1) \leq \mathrm{CR}$ **then**
    $u_i^{(j)} \leftarrow x_{r_1}^{(j)} + F \cdot (x_{r_2}^{(j)} - x_{r_3}^{(j)})$
  else
    $u_i^{(j)} \leftarrow x_i^{(j)}$
  end if
end for
for all $i \in \{1, \ldots, p\}$ **do**
  if $\nexists\ s \in \mathcal{S}_k : f(s) \leq f(u_i)$ **and** $\|s - u_i\| \leq \tau\,(|\mathcal{S}_k| + p)$ **then**
    $q_i \leftarrow \mathcal{L}(f, \Omega, u_i)$
    $\mathcal{O} \leftarrow \mathcal{O} \cup \{q_i, u_i\}$
  else
    $q_i \leftarrow u_i$
  end if
  if $f(q_i) < f(x_i)$ **then**
    $x_i \leftarrow q_i$
  end if
end for
$\mathcal{S}_k \leftarrow \mathcal{S}_k \cup \mathcal{O}$
return The new population matrix P

same basin of attraction are detected by means of clustering techniques, and only one local search is started for each basin. This way, the search process can reduce the cost of the local phase while providing precise local optima. Moreover, the strategy can avoid the collapsing of part of the population to a single point, which prevents further progress, thus improving the probability of convergence to a global minimizer. In fact, when the population only contains very similar solutions, the generation function will likely discover the same solutions, thus leading to diversity loss and potential premature convergence.

Different from MSL, DE replaces the sampling procedure of Multistart but the population remains independent from the past generated samples. We thus have two distinct sets of points; one of fixed size for DE, the population P to be evolved, and another one for the clustering procedure which contains all the past explored points, that is, the transformed sample $\mathcal{S}$. Algorithm 5 sketches the overall procedure; we will refer to this new clustering-based MDE method as C-MDE. Compared with the pseudocode reported in Algorithm 2, three crucial issues have to be discussed:

Clustering method. We recall that in MSL the decision to start a local search at the k-th iteration depends only on the threshold

$$\tau(K) = \pi^{-1/2} \left(\Gamma\left(1 + \frac{n}{2}\right) \mu(\Omega) \frac{\sigma \log K}{K} \right)^{1/n} \tag{3.8}$$

where $\sigma > 0$ is a parameter, $\mu(\Omega)$ is the Lebesgue measure of Ω, K is the total number of sample points and $\Gamma(\cdot)$ is the gamma function. A point is taken as the starting point for a local optimization if there is no other sample point, within the critical distance $\tau(k)$, with lower function value. We point out that the theoretical properties of MSL relied on a uniform sampling of the search space; our method results, instead, in a MSL with a sampling distribution generated by the DE and thus different from a uniform one. Intuitively, the non-uniformity of the sample does not prevent the global minimum to be discovered provided that the probability of generation of a point in its basin is different from zero. As long as at least a point is generated in the basin of a local minimizer, it will be discovered once K gets large enough and τ_k tends toward 0. Hence, we can replace the uniform distribution by another one biased toward regions of the search space where we hope the global optima is more likely to be. On the other hand, the non-uniformity also has implication on the probability with which the local search is applied to a given sample point. With uniform distribution, $\tau(\cdot)$ is computed in such a way that the probability of not having a point within distance $\tau(K)$, decrease to zero to guarantee some attractive properties as reported in Section 1.2.1. With non-uniform distribution, the threshold may be too large in high-density regions, e.g., around the basins of attractions, and too small in low-density regions. Though regions sampled with a low-density can be seen as less promising in the DE view, we have the effect of performing

local search with a higher probability on points sampled far away from the funnels already detected where, however, unexplored basins might exist. In this regard, C-MDE has the natural capability of promoting the exploration phase of the method avoiding a premature population collapsing.

Transformed sample. Since a population-based method replaces the sampling procedure, a set of transformed sample $\mathcal{S}$, which contains knowledge on the past minima discovered, is maintained to cluster the newly generated points. After wide experimentation, we decided to use the set $\mathcal{S}$ composed of :

1. All of the current members of the population P;
2. All of the local optima q_i discovered in previous iterations;
3. All of the previous generated points u_i from which the decision of starting a local search was taken.

Every descent path in the basins of attractions is conveniently approximated by the pair (u_i, q_i). We point out that the cardinality of $\mathcal{S}$ is non-decreasing although some garbage collection is made to avoid duplicates. As commented above, we used the classical threshold to decide the meaning of "close enough" and choose whether to start a local search from points in the current population. The size of the transformed sample for clustering grows less than linearly. Thus, in practice, keeping in memory all the minima detected up to the current iteration may be an appropriate choice. Conversely, if a huge number of points need to be stored, a mechanism to reduce the size of the set $\mathcal{S}$ is required. As an example, a fraction of points with the highest function value may be discarded at each iteration.

Selection mode. Note that DE algorithms can use both synchronous and asynchronous modes of survivor selection. In synchronous strategy the population is updated with all the new solutions at the same time, instead of updating the individuals just after being generated. Canonical DE and C-MDE implement a synchronous selection by maintaining a primary array for holding the current individuals and a secondary array to store the selected solutions for the next generation. Once the calculations with the current population members finish at a generation, the secondary array and the primary array are exchanged for resuming

calculations in the next generation. On the other hand, the variants of MDE use the asynchronous strategy. In this case, each newly generated point can replace its competitor (if better or equal) and become a member of the current population, subsequently taking part in the mutation of the other population members. Asynchronous update allows the improved solutions to contribute to the evolution immediately and can speed up the convergence faster than the synchronous batch mode update.

Note that our clustering-based schema, as reported in Algorithm 5, does not differ with respect to the generation and selection mechanisms of DE. Hence, all the clustering-based variants as described in Section 3.2.4 can be easily obtained; as an example the pseudocode of the greedy variant of C-MDE is sketched in Algorithm 6. In what follows, we will call all these variants CG-MDE, CD-MDE and CH-MDE, respectively.

3.4 Standard GO test functions

In this section we propose a few test problems commonly cited in the GO literature for which we performed an extensive set of tests in order to give a comparison of performance of the methods previously introduced. The test suite that we used consists of three highly multimodal benchmark functions, namely the Rastrigin, the Ackley and the Schwefel function, and some variants that will be discussed later; all test problems reported are box-constrained. Finding the minimum of the above mentioned functions, despite the use of gradient-based optimization, is a fairly difficult problem.

Test Problem 1. The *Rastrigin* function [67] is defined as

$$f_1(\mathbf{x}) = 10n + \sum_{i=1}^{n} \left(x_i^2 - 10cos(2\pi x_i)\right) \tag{3.9}$$

subject to $-5.12 \leq x_i \leq 5.12$.

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 search space. The *global* minimum is located at the origin $\mathbf{x}^* = (0, \ldots, 0)$ with $f_1(\mathbf{x}^*) = 0$.

Algorithm 6 The greedy variant of C-MDE (CG-MDE)

Require: The population matrix P and the population size $p = |P|$;
the parameters $F \in (0, 2)$ and $\mathrm{CR} \in [0, 1]$ for DE

$\mathcal{S}_k \leftarrow \mathcal{S}_{k-1}$ $(\mathcal{S}_0 = P)$
$\mathcal{O} \leftarrow \emptyset$
for all $i \in \{1, \ldots, p\}$ **do**
  Randomly choose $r_1 \in \{1, \ldots, p\} \setminus \{i\}$;
  if $f(x_i) > f(x_{r_1})$ **then** $\varphi = 1$
  else $\varphi = -1$
  end if
  if $\mathcal{U}(0, 1) \leq \mathrm{CR}$ **then**
    $v_i = x_i + \varphi F \cdot (x_{r_1} - x_i)$
  else
    $u_i^{(j)} \leftarrow x_i^{(j)}$
  end if
end for
for all $i \in \{1, \ldots, p\}$ **do**
  if $\nexists\ s \in \mathcal{S}_k : f(s) \leq f(u_i)$ **and** $\|s - u_i\| \leq \tau\,(|\mathcal{S}_k| + p)$ **then**
    $q_i \leftarrow \mathcal{L}(f, \Omega, u_i)$
    $\mathcal{O} \leftarrow \mathcal{O} \cup \{q_i, u_i\}$
  else
    $q_i \leftarrow u_i$
  end if
  if $f(q_i) < f(x_i)$ **then**
    $x_i \leftarrow q_i$
  end if
end for
$\mathcal{S}_k \leftarrow \mathcal{S}_k \cup \mathcal{O}$
return The new population matrix P

Test Problem 2. The *Ackley* function [3] is defined as

$$f_2(\mathbf{x}) = -20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)} + 20 + e \qquad (3.10)$$

subject to $-32.768 \leq x_i \leq 32.768$.

The Ackley function is also highly multimodal and a *single-funnel* one. While the barriers between local minimizers are lower with respect to the Rastrigin function, the nearest distance between local minimizers is not constant and it becomes smaller as we approach the global minimum. Moreover, the function has a global minimum located in a very tight basin. The *global* minimum is located at origin $\mathbf{x}^* = (0, \ldots, 0)$ with $f_2(\mathbf{x}^*) = 0$.

Test Problem 3. The *Schwefel* function [56] is defined as

$$f_3(\mathbf{x}) = -\sum_{i=1}^{n} x_i sin\left(\sqrt{|x_i|}\right) + \alpha n \tag{3.11}$$

subject to $-500 \leq x_i \leq 500$ and $\alpha = 418.982887$.

In comparison to Rastrigin function, Schwefel function adds the difficulty of being less symmetric and having the global minimum at the edge of the search space. Additionally, the Schwefel function is usually much more challenging with respect to the other two functions in view of its highly *multi-funnel* landscape. The *global* minimum is located at $\mathbf{x}^* = (420.9687, \ldots, 420.9687)$ where $f_3(\mathbf{x}^*) = 0$.

While widely used in the GO literature, these functions have some properties which may simplify the detection of the global minimizers. The first simplifying property is the separability. Both the Rastrigin and the Schwefel functions are *separable*. In this case, each variable is independent of the other variables; this fact can be sometimes exploited by global optimization approaches in order to reduce an n-dimensional problem into the solution of n one-dimensional ones. 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. Indeed, sometimes, GO methods use the center of the feasible box as a default starting point. Finally, the Rastrigin and Ackley functions are *symmetric* with respect to the origin.

In the literature, variants of the basic functions are proposed where high multimodality and the funnel properties are also maintained, but the simplifying features are removed. The variants we considered are:

$$f_i\left(DW(x - \bar{x})\right) \quad i = 1, 2, 3 \tag{3.12}$$

where $D \in \mathbb{R}^{n \times n}$ is a *diagonal* matrix of order n with positive diagonal elements; $W \in \mathbb{R}^{n \times n}$ is an *orthonormal* matrix of order n; $\bar{x} \in \mathbb{R}^n$ is a n-dimensional *shift* vector. Let I be the identity matrix of order n, the following combinations have been considered:

Separable. $D = W = I$, $\bar{x} = 0$; this is the basic version of the GO test function;

Rotated (R). $D = I$, W random orthonormal matrix, $\bar{x} = 0$; the orthonormal transformation eliminates separability. The rotation matrix W is generated from standard normally distributed entries by Gram-Schmidt ortho-normalization;

Rotated and Shifted (RS). $D = I$, W random orthonormal matrix, $\bar{x}$ randomly generated within the feasible region; the shift vector moves the global minimizer far away from the center of the feasible set;

Rotated, Shifted and Scaled (RSS). D with random diagonal elements in the interval $[1, 4]$, W random orthonormal matrix, $\bar{x}$ randomly generated within the feasible region; the diagonal transformation removes the symmetry of the problem with respect to permutation of the variables.

All the combinations have been applied to the basic functions and moderately small dimensions $n = 2, \ldots, 5$ were considered. Note that when we apply the orthonormal transformation, the search space is not rotated, i.e., the feasible region remains the same for all test problems. We point out that by applying such transformation on Schwefel function more and better minima may occur further away from $\mathbf{x}^* = (420.9687, \ldots, 420.9687)$ with unknown function values. In this case, a run was considered as a successful if the best configuration found among all the methods employed was reached.

In conclusion, the set of benchmark problems we employed in our experimentation is made up by 48 test functions. According to [7], some GO test functions as the Levy, the Sinusoidal and the Schaffers ones, have been discarded since they turned out to be quite simple ones with respect to the proposed approaches.

3.5 Experiments

In this chapter, we present the experimental results in a compact form and only performance profiles are used to evaluate and compare the methods investigated. The detailed numerical results can be found in the report [65].

3.5.1 Experimental setup

We have carried out different experiments to assess the performance of our approach based on clustering using the test suite described in Section 3.4.

The focus of this study was to compare the performance of the our method with the original one, in a variety of settings and involving different exploration features as described in Section 3.2.4. Our aim is to determine if the proposed approach is capable of obtaining the same quality, while saving a large percentage of local searches. Here, we used CG-MDE, CD-MDE and CH-MDE to denote the original methods powered by clustering techniques. Though classic DE uses only three control parameters, namely, population size p, scale factor F, and crossover probability CR, the choice of these parameters might be critical for the performance (e.g., see [37]). Moreover, we point out that C-MDE also requires an additional parameter σ which will be discussed later.

We summarize some choices common to all methods in what follows:

Feasibility. The problem of checking and possibly restoring feasibility is common in DE methods, even for problems defined on a box. In fact, the mutation operator may generate trial vectors whose components violate the predefined boundary constraints. A possible solution is to set the violating component to be the middle between the violated bound and the corresponding components of the parent individual, i.e.,

$$\begin{aligned} v^{(j)} &= (x_L^{(j)} + x^{(j)}) \ / \ 2 \qquad \text{if } v^{(j)} < x_L^{(j)} \\ v^{(j)} &= (x_U^{(j)} + x^{(j)}) \ / \ 2 \qquad \text{if } v^{(j)} > x_U^{(j)} \end{aligned}$$

where $v^{(j)}$ and $x^{(j)}$ are the j-th components of the mutation vector and the basic vector, respectively. However, in [58] a new approach to deal with box constraints has been recently introduced; the authors have showed that this technique performs well especially when the optimal solution is located near or on the boundary. The basic idea of this approach requires the calculation of an appropriate value for F given the direction $d = v_i - x_i$. If x_i is a feasible base point then the points $x_i + F \cdot d$ belong to the feasible region for all the values $F \in [0, F^*]$, where

$$F^* = \min \left\{ \min \left\{ \frac{x_L^{(j)} - x^{(j)}}{d^{(j)}} \;\middle|\; d^{(j)} < 0 \right\}, \ \min \left\{ \frac{x_U^{(j)} - x^{(j)}}{d^{(j)}} \;\middle|\; d^{(j)} > 0 \right\} \right\}$$

and the point $x + F^* \cdot d$ belongs to the boundary of the box. Feasibility with respect to all constraints is thus obtained by simply choosing F^*

as value for the scale factor in the current mutation function. The second strategy is the one used in all our experiments.

Scale factor. The scale factor is generally related to the convergence speed. In order to avoid premature convergence, it is crucial for the latter to be large enough and a reasonable choice, $F = 0.5$, is taken following the numerical experiments reported in [6, 7]

Crossover probability. No crossover is performed, CR = 1, unless otherwise stated. The reason for this choice is due to the fact that the crossover operation, at least the standard one discussed in Section 3.2.1, could improve the performance when applied to separable functions, but deteriorates the performance over non-separable functions.

Stopping criteria. The algorithm was stopped as soon as one of the following conditions was satisfied: 1) the sum of the differences in function value between the members of the population falls below a threshold (10^{-4} in the experiments). This condition is only necessary in order to prove that the population has shrunk into a single point, indeed, as only function values are taken into account, more than one point cluster might exist. This rule, by far, was the most common reason for stopping in all the performed experiments. 2) no change has been observed in the population (more precisely, in the objective values associated to the population elements) during the last 100 iterations of the algorithm. 3) the best observed value did not change during the last 20 000 local searches performed. These stopping criteria are the same used in [7].

3.5.2 Performance criteria

The algorithms presented in this work are based on multiple gradient-based local searches. Starting points for such local searches are generated with the aim of reaching the global minimizer with a limited computational effort. The basic operation through which the computational effort is measured is the local search call. In this regard, four performance criteria are selected from [7] to evaluate the performance of the algorithms. These criteria are described as follows:

%S denotes the percentage of successes over the set of the random runs;

LS/S denotes the average number of local searches per success over the set of random runs;

D denotes the average distance over the instances where a failure occurs, computed as

$$|f(x^*) - f(\hat{x})|$$

where $f(x^*)$ is the *global* minimum value attained at the global minimizer x^* and $f(\hat{x})$ is the *best* function value reached by the algorithm attained at some point $\hat{x}$;

SP denotes the success performance

$$\overline{\text{LS}} \times \frac{1}{\text{S}}$$

where $\overline{\text{LS}}$ is the number of local searches over only the successful runs. Of course, such criterion is meaningless for real GO problems where the global minimum value is not known, but, in this way, we are able to measure the computational effort of the algorithms to reach a global minimizer without taking into account the additional computational effort for stopping the algorithm themselves.

3.5.3 Numerical results and comparison

In this chapter, even if there are many variants of memetic DE schemes, we only compare our approach with the three versions proposed in Section 3.2.4. We would like to emphasize that these variants outperforms the original MDE and they are quite competitive even when compared with different DE schemes powered with local searches. In order to study the performance of our clustering-based scheme, we experimented using the test suite introduced in Section 3.4. We point out that the diversity of the test problems is adequate enough to make a general conclusion about the performance of the algorithms.

In our experimentation, we ran a set of 100 independent runs of each test problem to investigate the average behavior. Note that to evaluate the algorithms in a fair way, we used the same stopping criteria and the same set of initial random populations for each benchmark instance.

Here, we decided to present only the results obtained by using L-BFGS [36] as local search method. We point out that the use of different local solvers might also lead to (usually slightly) different results. As mentioned

earlier, we fixed $F = 0.5$ and $\mathrm{CR} = 1$ as the parameter setting for all algorithms following the numerical experiments performed in [7]; we have not performed a thorough computational investigation with other values.

All implementations in this work was written in Python - making use of the Numba [30] library for performance - using as local solver the L-BFGS-B 3.0 solver with default parameters (and the maximum number of variable metric corrections used to define the limited memory matrix fixed to 3) [45].

Sensitivities to the threshold parameter σ

The use of clustering techniques in memetic approaches may balance the exploration and the exploitation of the evolutionary process. Moreover, our method is also simple to use, by adding only one parameter to the original ones. In order to investigate the effect of the parameter σ on the performance of the proposed algorithm, see equation (3.8), a first set of experiments has been carried out fixing the value for the population size p and keeping all the remaining parameters unchanged. The value of the parameter σ has been varied in the following range

$$\sigma \in \left[\ 5 \cdot 10^{-6},\ 0.5\ \right]$$

In what follows, we propose performance profiles to compare the numerical results obtained; we refer to [15] for more details about this performance metric construction and to [65] for details about the numerical results. Generally speaking, the performance profiles provide a very useful and convenient means of assessing the performance of an algorithm relative to the best one on each test problems. As already mentioned above, our primary aim is to show that our clustering-based variants are able to outperform the original ones in terms of average number of local searches needed to reach the global minimizer, but, on the other side, we would like to show that this approach is also competitive in terms of the quality of the solution returned.

We analyzed the performance of each variant of the MDE method introduced in Section 3.2.4, namely the greedy, the distance and the hybrid one, comparing the latter with the same variants powered with clustering techniques, separately. Each curve gives the profile obtained for each method by solving the benchmark functions 100 times; moreover, we compared the performance of the methods using both LS/S and %S values (horizontal axis) as described in Section 3.5.2.

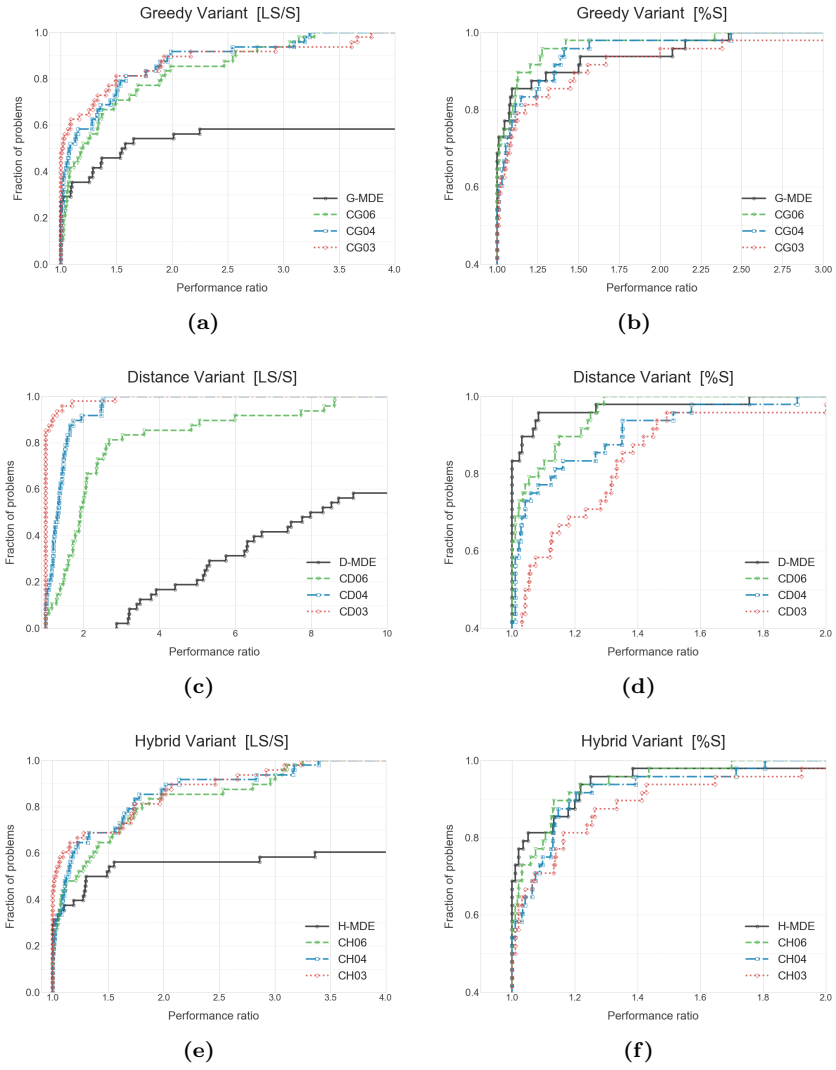


Figure 3.1: The performance profiles of the variants of MDE for the benchmark problems with a value of the population size equal to 10 and three different choices of the parameter σ: $5 \cdot 10^{-3}$ (C03), $5 \cdot 10^{-4}$ (C04) and $5 \cdot 10^{-6}$ (C06), respectively.

Here, we decided to depict the curves related to the original variant of MDE and the clustering-based alternatives for three different choices of the parameter σ:

$$\sigma \in \left\{ \ 5 \cdot 10^{-3}, \ 5 \cdot 10^{-4}, \ 5 \cdot 10^{-6} \ \right\}$$

The comparison of the methods for the first experimentation is summarized in Figure 3.1, where the value of the population size p is fixed to 10. As we can see from the performance profiles on the left side, the C-MDE methods outperform the variants of the original MDE, the black curve, in terms of the average number of local searches per success for all the variants and the values of the parameter σ considered. So it is clear that a large percentage of local searches can be saved by raising the value of σ, however, an extreme choice for this parameter might lead the method to becoming trapped in a local minimizer. In fact, the results on the right side of the figure show that, e.g., for values of σ that vary between $5 \cdot 10^{-3}$ to $5 \cdot 10^{-4}$, the red and blue curves, the C-MDE method is not able to preserve the same quality for any of the original variants investigated.

More interesting results, instead, are obtained for $\sigma = 5 \cdot 10^{-6}$, where our approach, the green curve, is able to balance exploration and exploitation in the majority of the cases, obtaining the same quality (or even better, e.g., see Figure 3.1 b and f) but at a fraction of the computational cost. In particular, as we can see from some of the experiments performed on the Schwefel's function - see Table 3.1, our clustering-based method appears to be more robust then G-MDE in terms of percentage of successes. A tentative explanation for these unexpected results on a very complicated function might be the following: the clustering methods are able to counterbalance the greediness of the G-MDE, thus exploiting information efficiently mantaining diversity in the population and promoting exploration. This "irregular" behavior of the CG-MDE method is a good feature with respect to multi-funnel functions where we need to avoid too fast convergence.

To investigate the sensitivity of the parameter σ to the variations of the population size, we decided to repeat all the experiments on the test suite raising the value of p up to 20. Note that due to the moderately small dimensions of the test problems - needed to successfully apply clustering techniques - the following choice of the population size

$$p \approx [\ 5n, 10n\ ]$$

can be considered quite large, thus proposing a new scenario with respect

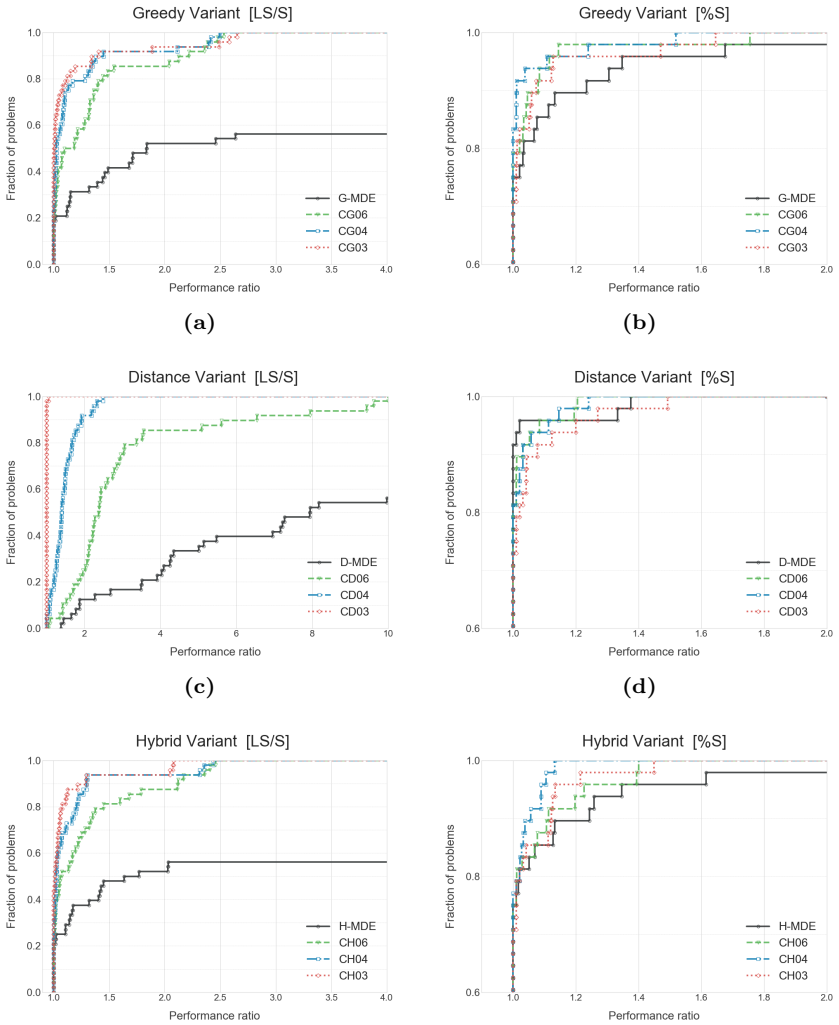


Figure 3.2: The performance profiles of the variants of MDE for the benchmark problems with a value of the population size equal to 20 and three different choices of the parameter σ: $5 \cdot 10^{-3}$ (C03), $5 \cdot 10^{-4}$ (C04) and $5 \cdot 10^{-6}$ (C06), respectively.

to the previous one. Again, as reported in Figure 3.2, the results show that the clustering-based methods are still significantly better than the original ones. Moreover, we point out that the quality of our approach is increased for all the values of σ investigated. Finally, some preliminary experiments have been performed with population size equal to 30, but we do not present detailed results as they show a similar behavior.

So it can be stated that the use of clustering techniques have enhanced the performance of the classical variants saving a large percentage of their computational cost; the numerical investigations also confirm that σ is a key parameter and the choice of a small value for the latter could be more reliable for unknown optimization problems. We point out that the effect should be studied in more detail also by varying the parameters of the DE schema which is beyond the scope of this work. Here, we have restricted our attention on the effect of the population size.

Sensitivities to the population size p

The performance of the DE is sensitive to the choice of the population size. On the one hand, increasing the population size will increase the diversity of possible movements, promoting the exploration of the search space; on the other hand, the probability to find the correct search direction decreases considerably. Furthermore, when we deal with memetic variants of DE, an even smaller population size is suggested as large population diminish the importance of local search method; especially if it is applied to the best individual only [47]. Most of the recommended limits found in the literature for the population size are within $5n$ or, more rarely, $10n$. Therefore, in order to investigate the sensitivity of our approach to the variations of the population size, we experimented our method fixing the parameter σ to the very attractive value of $5 \cdot 10^{-6}$ and varying the population following the rule $p = 5n$, where n is the dimension of the test function. Again, the detailed numerical results are reported in [65].

As reported in Figure 3.3, the results show that the clustering-based methods are still significantly better than the original ones. Hence, according to the the performance profiles depicted, we can summarize that (i) for the majority of GO test functions our approach is better than MDE if a proper value for σ is determined; (ii) the C-MDE provides the same (or better) quality but at a fraction of the computational cost; (iii) the C-MDE seems to be robust with respect to different population sizes; (iv) the C-MDE may

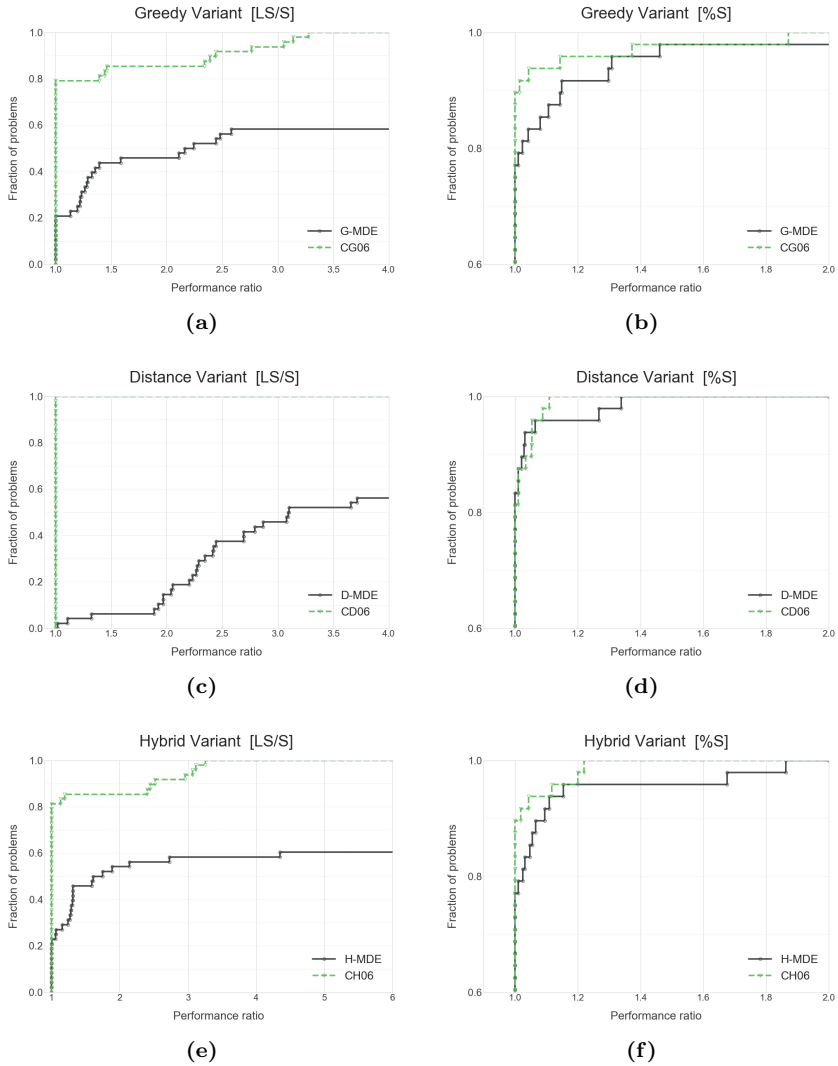


Figure 3.3: The performance profiles of the variants of MDE for the benchmark problems where the parameter σ is fixed to $5 \cdot 10^{-6}$ (C06) and the population size p is equal to $5n$ with n size of the problem.

avoid some negative phenomena like the population collapsing to a single point.

Comparison in term of success performance

As mentioned earlier, SP is useful for measuring the computational effort of the algorithms to reach the global minimizer without taking into account the additional computational effort for stopping the algorithm themselves. This performance measure is defined in Section 3.5.2 as a slight modification of the metric used in the CEC 2005 competition [23]. In [7] the authors used the success performance to compare their proposed variants of MDE with some recent results reported in the literature obtaining quite interesting results; thus, for the sake of completeness, we also propose a comparison with the latter criterion.

We report in Figure 3.4 the performance profile on the benchmark suite for values of the population size equal to 10 and 20. The experimental results show that our approach outperforms the variants of MDE again, and this time in terms of convergence velocity for almost all the test functions investigated. As reported in the experiments described, the greedy and hybrid clustering-based variants appear to be the best ones, and, as expected form the numerical results in [7], the performance of the hybrid C-MDE lies between the one of CG-MDE (very close) and the one of CD-MDE.

Therefore, we can claim that the use of clustering methods did not alter the natural behavior of the original variants of MDE investigated, while improving their performance in general.

3.6 Conclusion

In this chapter we showed that the cost of "CPU heavy" local searches in memetic approaches could be reduced with a clustering method by avoiding multiple rediscoveries of the local optima. In addition, this clustering method supplies information that can be used to maintain the diversity in the population. Driven by our personal experience, a detailed comparison between a new clustering-based memetic DE approach and different variants of memetic DE approaches is investigated. We point out that other ways to hybridize DE exists, however the variants proposed here, in spite of their simplicity, are proved to be very effective especially over well-known global optimization test functions.

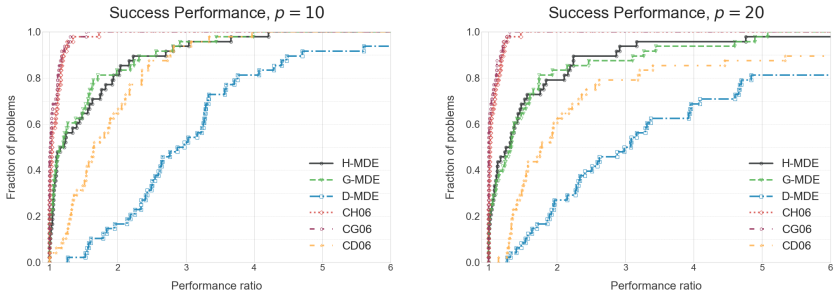


Figure 3.4: The performance profiles for the benchmark problems for each variant of MDE and clustering-based MDE with σ equal to $5 \cdot 10^{-6}$ (C06).

To evaluate the performance of our method, 48 single-objective functions with different characteristics are selected from the literature. A comprehensive set of experiments are conducted involving problems with moderately small dimension in order to assist the use of clustering techniques. The computational investigations confirm that our method, called C-MDE, is quite competitive even when compared with efficient variants of MDE, namely, the greedy variant (G-MDE), the distance variant (D-MDE), and the hybrid variant (H-MDE). In particular, according to our experiments, it appears that a proper choice of the parameter σ for C-MDE is of primary importance and can significantly enhance the behavior of the method achieving a good balance between exploration and exploitation of the search space.

In our future work, the effect of the clustering will be studied in more detail by varying the parameters of DE and the problem dimensionality. In addition, we believe that some other strategies to adaptively associate to each individual a different probability of starting a local search can be used in combination with the basic C-MDE schema. Another possible direction may be the one of applying clustering techniques to other memetic scheme such as PSO, ACO, GAs, etc.

Finally, the idea we proposed here is only a study of feasibility and some strategies to successfully apply C-MDE in large dimension are needed to aim of taking real-world problems. A first step toward this is presented in the next chapter.

Algorithm		G-MDE			CG-MDE		
Function	p	%S	SP	LS/S	%S	SP	LS/S
sch(2)-	10	**73**	105.09	1567.26	72	**41.84**	**121.17**
sch(2)-R	10	**81**	119.34	1453.09	56	**59.80**	**184.25**
sch(2)-RS	10	87	131.33	1264.14	**89**	**33.08**	**93.57**
sch(2)-RSS	10	96	117.40	829.90	**100**	**34.12**	**95.71**
sch(3)-	10	**55**	129.92	2285.45	49	**113.70**	**259.55**
sch(3)-R	10	**56**	276.79	2205.00	24	**261.98**	**538.50**
sch(3)-RS	10	62	110.56	1983.23	**75**	**69.60**	**167.56**
sch(3)-RSS	10	**66**	115.01	1886.52	63	**92.34**	**236.79**
sch(4)-	10	**41**	271.86	3169.27	38	**262.88**	**438.08**
sch(4)-R	10	13	769.23	9865.38	**27**	**337.45**	**644.00**
sch(4)-RS	10	19	495.84	6565.79	**36**	**220.83**	**424.92**
sch(4)-RSS	10	49	244.48	2574.08	**66**	**123.03**	**312.67**
sch(5)-	10	33	392.10	4069.70	**36**	**322.45**	**521.33**
sch(5)-R	10	13	863.91	10197.69	**28**	**395.79**	**682.18**
sch(5)-RS	10	16	566.41	7936.88	**19**	**450.69**	**877.05**
sch(5)-RSS	10	30	473.33	4163.00	**39**	**277.58**	**664.08**

Table 3.1: This table reports the numerical results for the multi-funnel Schwefel function of G-MDE and CG-MDE with the parameter $\sigma = 5 \cdot 10^{-6}$ (CG06). The dimension of the problems vary between 2 to 5 and the capitol letters denotes the transformations: (R) rotated, (RS) rotated and shifted and (RSS) rotated shifted and scaled.

Chapter 4

Scalability Study for C-MDE

In this chapter we propose a modification of the clustering-based MDE scheme introduced in Chapter 3, in order to successfully apply C-MDE in large dimension. In particular, a general purpose dimensionality reduction technique is used to reduce the dimension of the search space in which the clustering method is performed. A scalability study highlighted the robustness of the newly proposed algorithm. Different numerical results show that the performance of C-MDE are superior to, or at least comparable to, the state-of-the-art of MDE methods, the G-MDE, particularly over a set of well-known test problems carefully chosen from the GO literature.

4.1 Introduction

Among the stochastic approaches, evolutionary algorithms offer a number of advantages such as robust and reliable performance, global search capability as well as other supplementary benefits that make it often an attractive choice. In particular, the hybridization of EAs with local searches has proven to be very promising. Like other EAs, DE is a population-based, stochastic global optimizer capable of working reliably in nonlinear and multimodal environments. In the previous chapter we introduced a clustering-based MDE schema (C-MDE) with quite interesting properties but with experimental results over GO test functions with only moderately small dimension. Unfortunately, the difficulty of a problem generally increases with its dimen-

sionality; as the number of parameters or dimension increases, the search space also increases exponentially and, for highly nonlinear problems, this dimensionality may be a significant barrier for almost all optimization algorithms. Many real problems have high dimensionality, therefore, there is a need to discuss the methods to use for obtaining acceptable solutions. As mentioned in the introduction of this thesis, when the dimension becomes large, clustering is no more an option, as the feasible set is usually too large that no reasonable finite set of sampling points will ever be "uniform enough" to cover the region.

In this chapter, we propose a modification of the C-MDE scheme, in order to successfully apply it even in large dimension. In particular, the results obtained in Chapter 2 are encouraging enough to motivate the use of the same algorithmic idea but without requiring that the solutions of the corresponding optimization problem have any geometrical property. In this setting, a general-purpose dimensionality reduction technique is required to reduce the dimensionality of the search space.

4.2 Dimensionality reduction

When comparing different methods for dimensionality reduction, the most important criteria are the amount of distortion caused by the method and its computational complexity. A statistically optimal way of reducing dimensionality is to project the observations onto a lower-dimensional subspace while keeping relative distances as much unchanged as possible. One of the most widely used way to do this is the Principal Component Analysis (PCA); unfortunately it is quite expensive to compute for high-dimensional data sets. Moreover, traditional methods like PCA suffer from being based on linear models.

Nonlinear methods are often more powerful than linear ones, because the connection between the observations may be richer than a simple matrix multiplication. An effective technique that does capture nonlinear structure is t-SNE, which stands for t-distributed Stochastic Neighbor Embedding [70]. This technique can capture the nonlinear structure in high dimensional data, at least at a local level, meaning that if two points are close to each other in the high dimensional space, they have a high probability of being close together in the low dimensional embedding space. More recently, a new technique, very similar to t-SNE, called Uniform Manifold Approximation

and Projection (UMAP) has been proposed in [44]. UMAP appears to have some significant advantages over t-SNE; it is faster than t-SNE and captures global structure better than t-SNE. However, all these nonlinear models often comprise many parameters, whose identification requires large amounts of observations and computational time.

In this work, instead, a computationally simple method of dimensionality reduction that does not introduce a significant distortion in the observation set would be more desirable. In random projection, the original high-dimensional data is projected onto a lower-dimensional subspace using a random matrix whose columns have unit norm. This method has been found to be a computationally efficient, yet sufficiently accurate, method for dimensionality reduction of high-dimensional point sets [21].

4.2.1 Random Projection

Due to its computational efficiency, Random Projection (RP) techniques are useful in many settings. Roughly speaking, algorithmic applications of the random projection method involve projecting the input to a low-dimensional space at an appropriate stage of the algorithm. A natural setting is when the input data is in a high-dimensional space, and it is possible to preserve essential properties of the data (for the particular problem at hand) while reducing dimensionality.

In particular, in random projection the original n-dimensional vectors are projected to a d-dimensional subspace using a random $(d \times n)$ matrix R. Introducing matrix notation, let $X \in \mathbb{R}^{n \times K}$ be a coordinate matrix of K n-dimensional vectors, then

$$\phi(X) = \alpha R X \in \mathbb{R}^{d \times K}, \qquad \text{with } (d \ll n) \tag{4.1}$$

is the projection of the points onto a lower d-dimensional subspace, where $\alpha \in \mathbb{R}$ is a scaling factor that depends on the characteristics of the matrix R. Generally, the value of α is chosen to make the expected squared length of the projected vector equal to the expected squared length of the original one. Note that a linear mapping such as (4.1) can cause significant distortions in the reduced space if R does not have specific properties.

The key idea of random mapping arises from the Johnson-Lindenstrauss Lemma; if points in a vector space are projected onto a randomly selected subspace of suitably high dimension, then the distances between the points are approximately preserved. In this regard, Johnson and Lindenstrauss

[28] proved that any (K) points in Euclidean space could be embedded in $O(\epsilon^{-2} \log K)$ dimensions without distorting the distances between any pair of points by more than a factor of $(1 \pm \epsilon)$, for any $\epsilon \in (0, 1)$. This dimensionality reduction lemma is stated as follows:

Theorem 4.2.1 (**Johnson-Lindenstrauss Lemma**). *For any* $0 < \epsilon < 1$ *and any integer* K*, let* d *be a positive integer such that*

$$d \geq 4 \left(\epsilon^2/2 - \epsilon^3/3 \right)^{-1} \ln K \tag{4.2}$$

Then for any set X *of* K *points in* $\mathbb{R}^n$*, there is a map* $\phi : \mathbb{R}^n \to \mathbb{R}^d$ *such that for all* $u, v \in X$*,*

$$(1 - \epsilon)\|u - v\|^2 \leq \|\phi(u) - \phi(v)\|^2 \leq (1 + \epsilon)\|u - v\|^2 \tag{4.3}$$

Further this map can be found in randomized polynomial time.

In general, from this Lemma it follows that the projection preserves all pairwise distances of X in expectations, provided that R consists of i.i.d. entries (denoted by $\{r_{ij}\}_{i=1}^{d}{}_{j=1}^{n}$) with zero mean and constant variance; this is the only necessary condition for preserving pairwise distances. Hence, the choice of the random matrix R is one of the key points of interest and, in what follows, we will investigate two possible choices selected from the literature.

Gaussian random projection There are a variety of choices for the random matrix R that have similar properties. It is often convenient to let r_{ij} follow a symmetric distribution about zero with unit variance. A simple distribution is the standard normal, i.e.,

$$r_{ij} \sim N(0, 1) \tag{4.4}$$

Practically all zero mean, unit variance distributions of r_{ij} would give a mapping that still satisfies the Johnson-Lindenstrauss Lemma, see, e.g., [1]. The special case when R consists of normal entries is called the Gaussian Random Projections (GRP). Many theoretical results are known for GRPs, e.g., see [71] for further references.

Sparse random projection As mentioned earlier, the elements r_{ij} of R are often Gaussian distributed. In his recent work, Achlioptas [1] pro-

posed using the projection matrix R with i.i.d entries as

$$r_{ij} = \sqrt{s} \begin{cases} 1 & \text{with probability } 1/2s \\ 0 & \text{with probability } 1 - 1/s \\ -1 & \text{with probability } 1/2s \end{cases} \tag{4.5}$$

where $s = 1$ or $s = 3$. With $s = 3$, one can achieve a threefold speedup because only 1/3 of the data need to be processed, hence the name Sparse Random Projections (SRP). Since the multiplications with $\sqrt{s}$ can be delayed, no floating point arithmetic is needed meaning further computational savings. Sparse random projections have gained their popularity because are an alternative to dense Gaussian random projection matrix, but they guarantee similar embedding quality while being much more memory efficient and allowing faster computation of the projected data.

In practice, random projections are computationally very simple: forming the random matrix R and projecting the $(n \times K)$ coordinates matrix X into d-dimensions is of order $O(ndK)$, and if the data matrix X is sparse with about m nonzero entries per column, the complexity is of order $O(mdK)$ [51].

4.3 C-MDE with RP

In order to design an effective and efficient memetic algorithms for real-world GO problems, we need to take advantage of both the exploration abilities of EA and the exploitation abilities of local search by combining them in a well-balanced manner. In principle, the local search should be applied only to individuals that will productively take the search towards the global optimum. This is particularly important because application of local procedure on an ordinary individual may unnecessarily waste function (and gradient) evaluations and turn out to be expensive. In some schemes, the solutions with better objective function values are generally preferred for reproduction, as they are more likely to be in the proximity of a basin of attraction. In this regard, we have already shown (see Chapter 3) that the cost of the local search in a memetic approach may be reduced with a clustering method by avoiding multiple rediscoveries of the local optima. Unfortunately, for successful incorporation of clustering techniques in EAs,

Algorithm 7 C-MDE with Random Projection

Require: The population matrix P and the population size $p = |P|$;
the parameters $F \in (0, 2)$ and $\mathrm{CR} \in [0, 1]$ for DE

$\mathcal{S}_k \leftarrow \mathcal{S}_{k-1}$ $(\mathcal{S}_0 = P)$
$\mathcal{O} \leftarrow \emptyset$
for all $i \in \{1, \ldots, p\}$ **do**
  Randomly choose $r_1, r_2, r_3 \in \{1, \ldots, p\} \setminus \{i\}$, all distinct;
  if $\mathcal{U}(0, 1) \leq \mathrm{CR}$ **then**
    $u_i^{(j)} \leftarrow x_{r_1}^{(j)} + F \cdot (x_{r_2}^{(j)} - x_{r_3}^{(j)})$
  else
    $u_i^{(j)} \leftarrow x_i^{(j)}$
  end if
end for
for all $i \in \{1, \ldots, p\}$ **do**
  if $\nexists\ s \in \mathcal{S}_k : f(s) \leq f(u_i)$ **and** $\|R_k s - R_k u_i\| \leq \tau_d\,(|\mathcal{S}_k| + p)$ **then**
    $q_i \leftarrow \mathcal{L}(f, \Omega, u_i)$
    $\mathcal{O} \leftarrow \mathcal{O} \cup \{q_i, u_i\}$
  else
    $q_i \leftarrow u_i$
  end if
  if $f(q_i) < f(x_i)$ **then**
    $x_i \leftarrow q_i$
  end if
end for
$\mathcal{S}_k \leftarrow \mathcal{S}_k \cup \mathcal{O}$
return The new population matrix P

several issues must be resolved in order to deal with real-world GO problems and the most important one is the curse of dimensionality.

Drawing from ideas of Chapter 2, we assume that a dimensionality reduction map $\phi : \mathbb{R}^n \to \mathbb{R}^d$ from the n-dimensional solution space to a d-dimensional subspace is defined. This way the latter may be used to adapt the clustering method to work in a reduced space characterized by an arbitrary dimension ($d \ll n$). The primary difference between the newly proposed approach and the previously one presented in Chapter 2 is that we are no more required to look for *geometrical* GO problems. We also point out that the new C-MDE does not add any additional complexity or any additional parameter to the original scheme provided that a proper value of d (< 10) is chosen. So the pseudocode of this general purpose method is described in Algorithm 7 where, for a number of reasons explained earlier, the map $\phi(\cdot)$ is substituted by an appropriate random projection where a new matrix R_k is defined at each iteration.

4.4 Experiments

In order to show the feasibility of the newly proposed C-MDE, we compared it with the original greedy variant of memtic DE. Note that, due to the large computing times of D-MDE, we restrict our attention only to the G-MDE and the CG-MDE methods in view of their faster convergence as described in Section 3.2.4. Our aim is to show that C-MDE can be quite competitive with existing memetic evolutionary approaches in terms of the quality of the solution returned within a limited budget of local searches by using RP to reduce the dimensionality of the search space.

4.4.1 Test functions and settings

Among the previously discussed test functions, we selected three benchmark problems with different characteristics. For the Rastrigin and the Ackley function we tested the rotated, shifted and scaled version (RSS); for the Schwefel function, instead, we tested the separable one. In fact, the latter is usually much more challenging with respect to the other two functions in view of its highly multi-funnel landscape.

The test functions were studied at dimensions equal to 10, 30, 50, 70, 100, 300 and 500. As previously commented, we used different population sizes, namely 50 and 100, the latter more suitable for multi-funnel functions, where a higher degree of diversity is preferable. Both the population sizes were fixed according to the values proposed in [7] for G-MDE over similar test functions.

In each table we report four columns associated to each algorithm. In particular, i) %S denotes the percentage of successes over the set of random runs; ii) D denotes the average distance over the instances where a failure occurs, as described in detail in Section 3.5.2; iii) SD is the corresponding standard deviation where D = SD = 0 if no failure occurs; iv) LS/S denotes the average number of local searches per success over the set of random runs. In our experimentation, we ran a set 20 independent runs of each test problem using the same set of initial random populations to evaluate the different algorithms - in a similar way as done in Section 3.5.3. We decided to present the results obtained by using L-BFGS as local search method; the parameter σ is chosen equal to $5 \cdot 10^{-6}$ and all the remaining parameters are kept unchanged with respect to the previous experimentation with C-MDE. The dimension d is chosen equal to 3, in order to assist the use of

the clustering method while avoiding an excessive distortion; we have not performed a thorough computational investigation with other values. In our experiments, we used both Gaussian distributed random matrices for the population size equal to 50, and the much more memory efficient SRP for $p = 100$. Finally, the numerical investigation were performed with the same implementations used in Chapter 3.

4.4.2 Performance, comparison and discussion

The results reported in Table 4.1 summarize the experiments performed with $p = 50$ and GRP as random projection technique. For the Ackley and the Rastrigin function the results are quite consistent through the different dimensions. As expected, in view of the single-funnel nature of the functions, G-MDE performs quite well with a very high percentage of successes. The CG-MDE performs very similar to G-MDE, thus showing that in this case the clustering method is able to preserve the good performance of the greedy approach in terms of quality and velocity of convergence.

Moreover, the experimentation with the Rastrigin function shown that CG-MDE is even able to outperform the original greedy approach in terms of number of local searches needed to reach the global minimizer. With $n = 500$ both G-MDE and CG-MDE have not the best performance (no success occurs) but the D value reveals that, when not reaching the global minimizer, CG-MDE almost always stops at the best local minimizer, with an average distance value equal to 3.134.

The Ackley function is single-funnel. Thus, we expected a behavior similar to the Rastrigin function. In particular, we expected a good performance of CG-MDE with respect to G-MDE, the usual robustness (i.e., ability to reach the global minimizer) but at a fraction of the computational cost; it is remarkable that, in this case, G-MDE converges very fast to the global minimizer and its performance is always very close to (but no better than) our approach.

The results with the Schwefel function display some variability with respect to the two presented approaches. What we expected was a not very good performance of the G-MDE approach, which converges too fast and is not able to explore a larger portion of the feasible region. The majority of the results confirm this expectations: CG-MDE is quite robust, with a higher percentage of successes; in fact, as mentioned in Section 3.3, the clustering method supplies information that can be used to maintain the diversity in

the population.

Similar conclusions can be drawn about the performance of the algorithms when the population size is equal to 100 and SRP is used as random projection, i.e., CG-MDE outperforms G-MDE at every dimension for the Rastrigin and the Schwefel functions, and it gets quite good performance for the Ackley function (see Table 4.2). Moreover, the results show that the performance improvement is stable with respect to the variation of the problem dimension. From the experimental results of this section, we can conclude that the C-MDE with RP is superior to, or at least comparable to, G-MDE; furthermore, it exhibits a particularly significant property: the reliability at very large dimensions.

4.5 Conclusion

In this chapter we proposed a modification of the basic clustering-based MDE scheme in order to successfully apply C-MDE in large dimension. In particular we deal with the curse of dimensionality by using random projections to reduce the dimensionality of the search space thus adapting the clustering method to work in a smaller subspace of arbitrary dimension. We investigated the performance of the modified version of C-MDE algorithm using a benchmark suite consisting of functions carefully chosen from the GO literature. The experimental results showed that the proposed algorithm outperforms the classic greedy version of memetic DE (G-MDE) in terms of quality and number of local searches needed to reach the global minimizer in all experimental settings.

In our future study, we would like to verify the potential of some other dimensionality reduction technique. In this regard, some preliminary experiments with nonlinear methods were started; as an example, the results obtained by using the UMAP method are very precise but too slow for algorithmic applications.

Algorithm	G-MDE				CG-MDE + RP			
Function	%S	D	SD	LS/S	%S	D	SD	LS/S
rgn(10)-RSS	100	0.000	0.000	9922.50	100	0.000	0.000	**657.45**
rgn(30)-RSS	100	0.000	0.000	10455.00	100	0.000	0.000	**1058.95**
rgn(50)-RSS	100	0.000	0.000	10442.50	100	0.000	0.000	**1007.55**
rgn(70)-RSS	100	0.000	0.000	10215.00	100	0.000	0.000	**967.30**
rgn(100)-RSS	100	0.000	0.000	10072.50	100	0.000	0.000	**1089.80**
rgn(300)-RSS	100	0.000	0.000	11755.00	100	0.000	0.000	**1236.35**
rgn(500)-RSS	0	3.182	0.343	inf	0	3.134	0.312	inf
ack(10)-RSS	100	0.000	0.000	**270.00**	100	0.000	0.000	300.60
ack(30)-RSS	100	0.000	0.000	357.50	100	0.000	0.000	**354.15**
ack(50)-RSS	100	0.000	0.000	617.50	100	0.000	0.000	**337.85**
ack(70)-RSS	100	0.000	0.000	405.00	100	0.000	0.000	**393.95**
ack(100)-RSS	100	0.000	0.000	390.00	100	0.000	0.000	**342.80**
ack(300)-RSS	**100**	0.000	0.000	722.50	95	0.207	0.000	**369.63**
ack(500)-RSS	0	0.047	0.000	inf	0	0.112	0.085	inf
sch(10)-	100	0.000	0.000	13330.00	100	0.000	0.000	**2873.50**
sch(30)-	100	0.000	0.000	11532.50	100	0.000	0.000	**2881.30**
sch(50)-	100	0.000	0.000	12027.50	100	0.000	0.000	**2810.60**
sch(70)-	95	118.439	0.000	12013.15	**100**	0.000	0.000	**3154.00**
sch(100)-	100	0.000	0.000	12212.50	100	0.000	0.000	**3208.35**
sch(300)-	95	118.442	0.000	14726.31	**100**	0.000	0.000	**4070.00**
sch(500)-	95	118.445	0.000	13284.21	**100**	0.000	0.000	**4350.30**

Table 4.1: This table reports the numerical results for the test suite for G-MDE and CG-MDE with the parameter $\sigma = 5 \cdot 10^{-6}$. The population size is equal to 50 for all test functions.

Algorithm	G-MDE				CG-MDE + RP			
Function	%S	D	SD	LS/S	%S	D	SD	LS/S
rgn(10)-RSS	100	0.000	0.000	20065	100	0.000	0.000	**1191.95**
rgn(30)-RSS	100	0.000	0.000	21410	100	0.000	0.000	**1854.80**
rgn(50)-RSS	100	0.000	0.000	20655	100	0.000	0.000	**1805.45**
rgn(70)-RSS	100	0.000	0.000	21190	100	0.000	0.000	**1903.10**
rgn(100)-RSS	100	0.000	0.000	20930	100	0.000	0.000	**2031.65**
rgn(300)-RSS	100	0.000	0.000	21380	100	0.000	0.000	**2412.85**
rgn(500)-RSS	0	2.994	0.027	inf	0	2.982	0.014	inf
ack(10)-RSS	100	0.000	0.000	575.00	100	0.000	0.000	**563.75**
ack(30)-RSS	100	0.000	0.000	1230.00	100	0.000	0.000	**681.10**
ack(50)-RSS	100	0.000	0.000	2760.00	100	0.000	0.000	**691.95**
ack(70)-RSS	100	0.000	0.000	805.00	100	0.000	0.000	**802.20**
ack(100)-RSS	100	0.000	0.000	1840.00	100	0.000	0.000	**753.15**
ack(300)-RSS	100	0.000	0.000	940.00	100	0.000	0.000	**777.45**
ack(500)-RSS	0	0.047	0.000	inf	0	0.050	0.015	inf
sch(10)-	100	0.000	0.000	24565	100	0.000	0.000	**6436.50**
sch(30)-	100	0.000	0.000	22790	100	0.000	0.000	**5800.20**
sch(50)-	100	0.000	0.000	22920	100	0.000	0.000	**5907.90**
sch(70)-	100	0.000	0.000	22835	100	0.000	0.000	**5885.65**
sch(100)-	100	0.000	0.000	22695	100	0.000	0.000	**6159.30**
sch(300)-	100	0.000	0.000	23495	100	0.000	0.000	**7170.25**
sch(500)-	100	0.000	0.000	23730	100	0.000	0.000	**7102.15**

Table 4.2: This table reports the numerical results for the test suite for G-MDE and CG-MDE with the parameter $\sigma = 5 \cdot 10^{-6}$. The population size is equal to 100 for all test functions.

Chapter 5

Conclusion

In this research we dealt with the use of local searches within global optimization algorithms. Though we discussed different issues, we focused our attention on the strategies to decide whether to start or not a local search from a given point. More specifically, our aim was to avoid the waste of computational effort due to local searches which lead to already detected local minima or to local minimizers with a poor function value. For most GO methods local searches are called with a fixed frequency like, e.g., in the well-known Multistart method, but in some other approaches once a starting point is generated, a local search is not necessarily started. The idea of using statistical clustering techniques in order to decide whether to start a local optimization from a sampled point arose in the 80's and was a first example of that. Clustering methods provide a way to organize local information and focus the optimization effort on promising areas of the feasible space.

Unfortunately, in recent years, these approaches have not been used anymore, for a number of different reasons. First of all, local optimization is now quite a mature subject with many very efficient algorithms available, so that saving local searches, in many cases, is no more a fundamental requirement. Secondly, and most important, computing power and modern algorithms, e.g., evolutionary methods powered by local searches, give us the opportunity to solve very large dimensional GO problems and, when the dimension becomes large, clustering methods are no more an option as they compare solutions based on the value of their variables. Despite these defects, the idea of clustering for global optimization was really bright and therefore, in this research, we have proposed new ways to exploit it for GO.

In Chapter 2, we developed structural descriptors for representing the solutions of two well-known geometrical problems in a compact way, and adapted a clustering method to work in the restricted space induced by our choice of geometrical features. Large scale solutions are therefore compared and grouped based on their overall characteristics rather than the value of their variables. Thanks to our strategy, we were able to obtain the same quality of a very effective Multistart approach, while employing a tiny percentage of local searches to converge to the putative global optimum. Moreover, we extended the idea of MSL toward methods with sophisticated (and more expensive) "local" searches, e.g., the cluster surface smoothing method, that are able to preserve the set of features during the descent in the funnels of the problem. In Chapter 3, instead, we showed a different way to use clustering techniques applying them to population-based methods powered by local searches, i.e., memetic algorithms. It is quite natural to extend these evolutionary processes by performing a local search to each newly generated point and to base comparisons on the objective function values at local minima. However, the frequency of local search can have a tremendous impact on the performance of the method. First, we addressed questions concerning the hybridization of the differential evolution with local search while also exploring some variants; then we proposed a clustering-based approach, called C-MDE, that provided significant improvement with respect to the original approach. Our results showed that, considering standard GO test functions with moderately small dimensions, the clustering method was able to balance the exploration and the exploitation of the evolutionary process enhancing the performance of the method in general.

In Chapter 4, we combined the previous presented ideas into a single, general, method. In particular, in order to successfully apply C-MDE in large dimension, we dealt with the curse of dimensionality by using random projections to reduce the dimensionality of the search space, thus adapting the clustering method to work in a reduced space of arbitrary dimension (no geometrical descriptors are required). We investigated the performance of the proposed method using a benchmark consisting of challenging functions carefully chosen from the GO literature with quite interesting results.

In conclusion, we hope that this work will encourage further research in the use of clustering techniques within modern global optimization algorithms.

Appendix A

Publications

This research activity has led to several publications in international journals. These are summarized below.

International Journals

1. F. Bagattini, F. Schoen, **L. Tigli**. "Clustering methods for large scale geometrical global optimization", *Optimization Methods & Software*, vol. 34, 2019. [DOI: 10.1080/10556788.2019.1582651]
2. F. Bagattini, F. Schoen, **L. Tigli**. "Clustering methods for the optimization of atomic cluster structure", *The Journal of Chemical Physics*, vol. 148, 2018. [DOI: 10.1063/1.5020858]

Bibliography

[1] D. Achlioptas, "Database-friendly random projections: Johnson-lindenstrauss with binary coins," *Journal of Computer and System Sciences*, vol. 66, no. 4, pp. 671 – 687, 2003, special Issue on PODS 2001.

[2] B. Addis, M. Locatelli, and F. Schoen, "Disk packing in a square: A new global optimization approach," *INFORMS J. on Computing*, vol. 20, no. 4, pp. 516–524, 2008.

[3] T. Bäck and H.-P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evol. Comput.*, vol. 1, no. 1, pp. 1–23, Mar. 1993.

[4] F. Bagattini, F. Schoen, and L. Tigli, "Clustering methods for the optimization of atomic cluster structure," *The Journal of Chemical Physics*, vol. 148, no. 14, p. 144102, 2018.

[5] R. W. Becker and G. V. Lago, "A global optimization algorithm," in *Proceedings of the 8th Allerton Conference on Circuits and Systems Theory*, 1970, pp. 3–12.

[6] F. Cabassi and M. Locatelli, "A computational comparison of memetic differential evolution approaches," in *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO Companion '15. New York, NY, USA: ACM, 2015, pp. 1181–1184.

[7] ——, "Computational investigation of simple memetic approaches for continuous global optimization," *Computers Operations Research*, vol. 72, 02 2016.

[8] A. Cassioli, D. Di Lorenzo, M. Locatelli, F. Schoen, and M. Sciandrone, "Machine learning for global optimization," *Computational Optimization and Applications*, vol. 51, no. 1, pp. 279–303, 2012.

[9] J. Cheng and R. Fournier, "Structural optimization of atomic clusters by tabu search in descriptor space," *Theoretical Chemistry Accounts*, vol. 112, no. 1, pp. 7–15, 2004.

[10] ——, "Structural optimization of atomic clusters by tabu search in descriptor space," *Theoretical Chemistry Accounts*, vol. 112, no. 1, pp. 7–15, Apr 2004.

[11] L. Cheng, Y. Feng, J. Yang, and J. Yang, "Funnel hopping: Searching the cluster potential energy surface over the funnels," *The Journal of Chemical Physics*, vol. 130, no. 21, p. 214112, 2009.

[12] J. H. Conway and N. J. A. Sloane, *Sphere Packings, Lattices and Groups.*

[13] L. C. W. Dixon and G. P. Szegö, *Towards Global Optimization.* North-Holland, 1975.

[14] L. C. W. Dixon and G. P. Szegö, *Towards Global Optimization 2.* North-Holland, 1978.

[15] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Mathematical Programming*, vol. 91, no. 2, pp. 201–213, 2002.

[16] M. Dorigo, M. Birattari, and T. StÃ¼tzle, "Ant colony optimization â artificial ants as a computational intelligence technique," *IEEE Comput. Intell. Mag*, vol. 1, pp. 28–39, 2006.

[17] J. P. K. Doye, R. H. Leary, M. Locatelli, and F. Schoen, "Global optimization of morse clusters by potential energy transformations," *INFORMS J. on Computing*, vol. 16, no. 4, pp. 371–379, Sep. 2004.

[18] J. P. Doye, R. H. Leary, M. Locatelli, and F. Schoen, "Global optimization of morse clusters by potential energy transformations," *INFORMS Journal on Computing*, vol. 16, no. 4, pp. 371–379, 2004.

[19] J. P. Doye and D. J. Wales, "Magic numbers and growth sequences of small face-centered-cubic and decahedral clusters," *Chemical Physics Letters*, vol. 247, no. 4, pp. 339 – 347, 1995.

[20] J. P. Doye, D. J. Wales, and R. S. Berry, "The effect of the range of the potential on the structures of clusters," *The Journal of chemical physics*, vol. 103, no. 10, pp. 4234–4249, 1995.

[21] X. Fern and C. Brodley, "Random projection for high dimensional data clustering: A cluster ensemble approach," 2003, pp. 186–193.

[22] R. Fournier, "Theoretical study of the structure of silver clusters," *The Journal of chemical physics*, vol. 115, no. 5, pp. 2165–2177, 2001.

[23] C. García-Martínez, P. D. Gutiérrez, D. Molina, M. Lozano, and F. Herrera, "Since cec 2005 competition on real-parameter optimisation: a decade of research, progress and comparative analysis's weakness," *Soft Computing*, vol. 21, no. 19, pp. 5573–5583, Oct 2017. [Online]. Available: https://doi.org/10.1007/s00500-016-2471-9

[24] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM review*, vol. 47, pp. 99–131, 2005.

[25] R. Graham and B. Lubachevski, "Repeated patterns of dense packings of equal disks in a square," *The Electronic Journal of Combinatorics [electronic only]*, vol. 3, 07 2004.

[26] W. E. Hart, "Adaptive global optimization with local search," Ph.D. dissertation, 1994, uMI Order No. GAX94-32928.

[27] R. Horst and H. Tuy, "Global optimization - deterministic approaches, 3. auflage," 1996.

[28] W. B. Johnson, J. Lindenstrauss, and G. Schechtman, "Extensions of lipschitz maps into banach spaces," *Israel Journal of Mathematics*, vol. 54, no. 2, pp. 129–138, 1986.

[29] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948 vol.4.

[30] S. K. Lam, A. Pitrou, and S. Seibert, "Numba: A llvm-based python jit compiler," in *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, ser. LLVM '15, 2015, pp. 7:1–7:6.

[31] L. Landau and E. Lifschitz, *Quantum Mechanics*.

[32] R. H. Leary, "Global optima of lennard-jones clusters," *Journal of Global Optimization*, vol. 11, no. 1, pp. 35–53, 1997.

[33] R. H. Leary and J. P. K. Doye, "Tetrahedral global minimum for the 98-atom lennard-jones cluster." *Physical review. E, Statistical physics, plasmas, fluids, and related interdisciplinary topics*, vol. 60 6 Pt A, 1999.

[34] W. Lefebvre, T. Philippe, and F. Vurpillot, "Application of delaunay tessellation for the characterization of solute-rich clusters in atom probe tomography," *Ultramicroscopy*, vol. 111, pp. 200–6, 2011.

[35] T. W. Liao, "Two hybrid differential evolution algorithms for engineering design optimization," *Applied Soft Computing*, vol. 10, no. 4, pp. 1188 – 1199, 2010.

[36] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical programming*, vol. 45, no. 1, pp. 503–528, 1989.

[37] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Computing*, vol. 9, pp. 448–462, 2002.

[38] M. Locatelli, "On the multilevel structure of global optimization problems," *Computational Optimization and Applications*, vol. 30, pp. 5–22, 01 2005.

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

[40] M. Locatelli and F. Schoen, "Fast global optimization of difficult Lennard-Jones clusters," *Computational Optimization and Applications*, vol. 21, no. 1, pp. 55–70, 2002.

[41] ——, "Efficient algorithms for large scale global optimization: Lennard-jones clusters," *Computational Optimization and Applications*, vol. 26, pp. 173–190, 2003.

[42] ——, "Local search based heuristics for global optimization: atomic clusters and beyond," *European Journal of Operational Research*, vol. 222, no. 1, pp. 1–9, 2012.

[43] ——, *Global Optimization: theory, algorithms, and applications*, ser. MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics and the Mathematical Optimization Society, 2013, vol. MO15.

[44] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction," *ArXiv e-prints*, 2018.

[45] J. Morales and J. Nocedal, "Remark on "algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound constrained optimization"," *ACM Transactions on Mathematical Software*, vol. 38, no. 1, 2011.

[46] F. Neri and C. Cotta, "Memetic algorithms and memetic computing optimization: A literature review," *Swarm and Evolutionary Computation*, vol. 2, pp. 1 – 14, 2012.

[47] N. Noman and H. Iba, "Accelerating differential evolution using an adaptive local search," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 107–125, 2008.

[48] K. J. Nurmela and P. R. J. Östergård, "Packing up to 50 equal circles in a square," *Discrete & Computational Geometry*, vol. 18, no. 1, pp. 111–120, 1997.

[49] J. P. K. Doye and D. J. Wales, "Structural consequences of the range of the interatomic potential a menagerie of clusters," *J. Chem. Soc., Faraday Trans.*, vol. 93, pp. 4233–4243, 1997.

[50] ——, "Structural consequences of the range of the interatomic potential a menagerie of clusters," *J. Chem. Soc., Faraday Trans.*, vol. 93, pp. 4233–4243, 1997.

[51] C. H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala, "Latent semantic indexing: A probabilistic analysis," *Journal of Computer and System Sciences*, vol. 61, no. 2, pp. 217 – 235, 2000.

[52] A. H. G. Rinnooy Kan and G. T. Timmer, "Stochastic global optimization methods. Part I: Clustering methods," *Mathematical Programming*, vol. 39, pp. 27–56, 1987.

[53] ——, "Stochastic global optimization methods. Part II: Multi level methods," *Mathematical Programming*, vol. 39, pp. 57–78, 1987.

[54] J. Schaer, "On the densest packing of spheres in a cube," *Canad. Math. Bull*, vol. 9, p. 265â270, 1966.

[55] ——, "The densest packing of ten congruent spheres in a cube," *Intuitive geometry, (Szeged, 1991)*, vol. 63, pp. 403–424, 1994.

[56] H.-P. Schwefel, *Numerical Optimization of Computer Models*, 01 1981, vol. 33.

[57] X. Shao, H. Jiang, and W. Cai, "Parallel random tunneling algorithm for structural optimization of lennard-jones clusters up to $n = 330$," *Journal of Chemical Information and Computer Sciences*, vol. 44, pp. 193–199, 2004.

[58] M. Spadoni and L. Stefanini, "A Differential Evolution algorithm to deal with box, linear and quadratic-convex constraints for boundary optimization," *Journal of Global Optimization*, vol. 52, no. 1, pp. 171–192, 2012.

[59] P. J. Steinhardt, D. R. Nelson, and M. Ronchetti, "Bond-orientational order in liquids and glasses," *Physical Review B*, vol. 28, no. 2, p. 784, 1983.

[60] R. Storn and K. Price, "Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 23, 01 1995.

[61] K. J. Strandburg, Ed., *Bond–Orientational Order in Condensed Matter Systems*. New York: Springer-Verlag, 1992.

[62] P. G. Szabó, M. C. Markót, T. Csendes, E. Specht, L. G. Casado, and I. Garcãa, *New Approaches to Circle Packing in a Square: With Program Codes (Springer Optimization and Its Applications)*. Springer-Verlag New York, Inc., 2007.

[63] H. Takeuchi, "Clever and efficient method for searching optimal geometries of lennard-jones clusters," *Journal of chemical information and modeling*, vol. 46, no. 5, pp. 2066–2070, 2006.

[64] Y. Tenne and S. W. Armfield, *A Memetic Algorithm Using a Trust-Region Derivative-Free Optimization with Quadratic Modelling for Optimization of Expensive and Noisy Black-box Functions*. Springer Berlin Heidelberg, 2007, pp. 389–415.

[65] L. Tigli. (2019) saving local searches in global optimizations: supplementary material. [Online]. Available: webgol.dinfo.unifi.it/luca-tigli/c-mde.pdf

[66] A. A. Törn, "Cluster analysis using seed points and density-determined hyperspheres as an aid to global optimization," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 7, no. 8, pp. 610–616, 1977.

[67] A. Torn and A. Zilinskas, *Global Optimization*. Springer-Verlag, 1989.

[68] A. A. Törn, "A search clustering approach to global optimization," in *Towards global optimization 2*, L. C. W. Dixon and G. P. Szegö, Eds. North-Holland, 1978, pp. 49–62.

[69] S. Tsutsui, M. Yamamura, and T. Higuchi, "Multi-parent recombination with simplex crossover in real coded genetic algorithms," 1999.

[70] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," 2008.

[71] S. Vempala, "The random projection method," 01 2004.

[72] D. J. Wales and J. 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*, vol. 101, no. 28, pp. 5111–5116, 1997.