Tiago Alexandre Barrinha Cordeiro

Bachelor of Science

A global optimization algorithm using trust-region methods and clever multistart

Dissertation submitted in partial fulfillment
of the requirements for the degree of

Master of Science in
Mathematics and Applications

Adviser: Ana Luísa Custódio, Associate Professor,
NOVA University of Lisbon

Co-adviser: Maria do Carmo Brás, Assistant Professor,
NOVA University of Lisbon

Examination Committee

Chairperson: Professor Maria Isabel Gomes
Raporteur: Professor Paula Amaral
Member: Professor Ana Luísa Custódio

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

November, 2020

A global optimization algorithm using trust-region methods and clever multi-start

This document was created using the (pdf)LaTeX processor, based in the "novathesis" template[1], developed at the Dep. Informática of FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

Acknowledgements

I would like to thank my advisers, as they supported me throughout the development of the proposed algorithm with very helpful and fast feedback on every step of the way. Even through the physical restrictions and heavy workloads of this year, their availability and support allowed the completion of this work.

I am also grateful to the support of my family and friends, close and distant, who were present both in the stressful moments and the happy moments of this journey, giving me motivation and advise when needed.

Abstract

Global optimization is an important scientific domain, not only due to the algorithmic challenges associated with this area, but also due to its practical application in different areas of knowledge, from Biology to Aerospace Engineering.

In this work we develop an algorithm based on trust-region methods for solving global optimization problems with derivatives, using a clever multistart strategy, testing its efficiency and effectiveness by comparison with other global optimization algorithms.

Based on an idea applied to the resolution of problems in derivative-free optimization, this algorithm seeks to reduce the computational effort that the search for a global optimum requires, by comparing points that are relatively close to each other, using as comparison radius the one associated with the trust-region method, retaining only the most promising ones, which will continue to be explored. The proposed method has the added benefit of not only reporting the global optimum but also a list of local optima that may be of interest, depending on the context of the problem in question.

Keywords: Global Optimization; Trust-region Methods; Multistart Strategies; Optimization with Derivatives.

Resumo

A otimização global é um importante domínio científico, não só pelos desafios algorítmicos que lhe estão associados, mas pela sua aplicação prática em diferentes áreas do conhecimento, que vão desde a Biologia à Engenharia Aeroespacial.

Neste trabalho é desenvolvido um algoritmo baseado em métodos de regiões de confiança, para problemas de otimização global com derivadas, usando uma estratégia de multi-inicializações inteligente, sendo testada a sua eficiência e eficácia por comparação com outros algoritmos de otimização global.

Baseado numa ideia aplicada à resolução de problemas de otimização sem derivadas, este algoritmo procura reduzir o esforço computacional que a busca de ótimos globais requer, comparando pontos que se situam relativamente próximos usando como raio de comparação o raio associado ao método de região de confiança, e retendo apenas os mais promissores, que continuarão a ser explorados. O método proposto permite não só a obtenção do ótimo global mas também de uma lista de ótimos locais que podem ser de interesse, dependendo do contexto do problema em questão.

Palavras-chave: Otimização Global; Métodos de Regiões de Confiança; Estratégias de multi-inicializações; Otimização com Derivadas.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Optimization is an area of Mathematics that focuses on solving minimization or maximization problems within a function's domain, which can possibly be subject to restrictions to the values of the variables. While obviously different, we tend to generalize optimization problems to minimization problems, as a simple transformation of the function to its symmetric keeps the optimal solution, with a symmetric optimum value, allowing us to obtain the maximum of a problem.

Optimization problems are relevant not only in Mathematics, but also in many other scientific areas like Chemistry [13], Engineering or Economics [11], where often the goal is to identify a global minimum.

There are special cases where solving these global problems can be an easy task, like is the case of a convex function on a convex domain, where a single local and global minimum exists. As such, local optimization algorithms are enough to compute the global solution, even if for some classes of convex problems, challenges remain to develop efficient local algorithms. However, when in presence of nonconvex functions, as there can exist multiple local minima within the search domain, the identification of a global minimum is more difficult, justifying the need for global optimization algorithms.

A simple strategy for global optimization is to explore a local optimization solver, running it from different starting points. This approach is known as a multistart technique. The local optimization algorithms are typically iterative processes, starting from an initial point and following a descent direction in order to find a new point with a better objective function value. This procedure is repeated until some defined stopping criteria related to stationarity are met. By considering multiple starting points, in different areas of the feasible region, it is then expected to obtain a list of all local minima of the objective

function (if in finite number), from which the global optimum can be easily recovered. The additional information respecting to the computation of the different local optima is also a useful characteristic of these algorithms, as sometimes compromises must be made and more stable solutions may be preferable, even with worse objective function values. However, the multistart procedure can be time consuming and often searches considering different initializations will converge to the same optimum. Also, the chosen local solvers and multistart strategies may work better for some problems rather than others, as there is no guaranteed way to ensure a fast and accurate convergence for all the different kinds of problems one can encounter [24].

In this work we propose an algorithm for the global optimization of problems with second order differentiable objective functions with continuous variables, constrained to a bounding box. The motivation for the proposed method was a strategy successfully used on global derivative-free optimization [3], which is adapted in this work to incorporate trust-region methods in a global derivative-based optimization approach.

1.2 Related works

This section will be mainly focused on reviewing works related to trust-region methods, which have been extensively explored in optimization (see [2]). However, within the scope of global optimization, only few works incorporate this algorithmic class. We present some recent works related to this topic that provide useful insights on issues that should be considered when using a trust-region approach in a global optimization perspective.

The first of these works, [1], describes the use of a trust-region method as a solver for a specific class of functions presenting the so-called funnel structure. As described by the authors, functions with this structure are a "perturbation of an underlying function with a low number of local minima". Due to this reduced number of local minima, the underlying function is naturally easier to globally optimize. Thus, the focus of the algorithm is to efficiently generate a smooth approximation function that maintains the same overall shape of the original function to optimize. For that, Gaussian smoothing techniques are applied to the local minimization operator, after which a modified version of a trust-region method is used, incorporating a sampling algorithm, adapting the trust-region approach to global optimization.

Other applications of a trust-region method to global optimization problems is reported in [15], where a cubic separable term is introduced in the classic second order Taylor based model. The authors show that the usage of these cubic terms improves the algorithm's ability to reach the global minimum, being able to escape from certain local minima where the base trust-region method would be trapped. Although this modification of classical trust-region algorithms showed to be effective in finding the global optimum, our goal is to compute additional local minima, providing the end user with additional good quality choices.

With this aim, we will make use of a multistart approach which consists in providing different initializations to a local solver. Decisions about when and where to generate the new initializations need to be taken. Works like [14] cover these topics and provide insights on important issues that can influence the efficiency of the search and the quality of the solution found.

There are also heuristic methods that employ multistart approaches such as the one described in [22]. Although reaching promising results when tested, convergence has not yet been established. However, these types of methods are still quite important, as their combination with other algorithms can lead to good results.

Another work that adresses global optimization problems using derivatives is the one presented in [12]. This paper mentions an interesting technique where, instead of a multistart approach, the use of local search algorithms is paired with a Gaussian model that is added to the objective function to fill in possible attraction basins where the used solvers may end up in. Through several iterations of finding local optima and filling in basins, the algorithm eventually explores all possible basins, gathering a list of all local optima from which the global optimum of the objective function can be easily extracted.

As mentioned before, the present work is inspired by [3], an algorithm that focuses on a different area of optimization, which is derivative-free optimization. In this case, multistart is coupled with a merging strategy, giving up on searches that are not promising. Merging stages could be crucial in a multistart approach, since having several searches converging to the same local optimum can lead to a loss of efficiency of the algorithm. To overcome this issue, points that are sufficiently close to each other and expected to converge to the same local optimum are compared and only one is kept. These strategies will be detailed in Chapter 3.

1.3 Objectives and thesis organization

The main objective of this work is to develop a method capable of solving global optimization problems with second order differentiable objective functions and continuous variables. With this purpose, we introduce an algorithm that applies a clever multistart approach to a base trust-region method, by starting with multiple points and, at each iteration, by selecting one as a center to run a single iteration of the trust-region algorithm. A list of visited points is kept and updated along the optimization process, from which the center for the trust-region iteration is selected. Besides the initial launch of several points, the algorithm also incorporates a relaunching strategy, with new starting points being inserted into the list when certain criteria are met, in an attempt to ensure that the entire search domain is explored. Since such a strategy can generate too many points and have groups of points being driven towards the same attraction zones, a merging strategy is considered.

Given the structure of each iteration of the proposed algorithm, which considers a step with the solution of a trust-region subproblem, this thesis starts by revising the

trust-region class of optimization methods. Its general algorithmic structure and the corresponding convergence analysis is presented in Chapter 2. Afterwards, in Chapter 3, we detail the structure of the proposed algorithm and how it keeps similar convergence results to the ones of the base trust-region method. Chapter 4 reports extensive numerical tests, presenting the collection of problems used, the procedure followed in the definition of the different algorithmic variants, and the results obtained with these distinct algorithmic strategies that lead to the proposed algorithm. Comparisons based on different metrics with other global optimization solvers are also provided. The thesis ends in Chapter 5 with some conclusions and directions for future work.

Chapter 2

Trust-region Methods

2.1 Introduction

Trust-region methods are a class of iterative algorithms that use the information provided by the objective function and its derivatives to build a quadratic model that approximates the objective function with expected good precision within a certain small distance of a given point x_k. This class of algorithms is widely known and several authors have covered the topic [2, 9, 20].

At each iteration, a quadratic model is built based on the Taylor expansion of the objective function f around the selected point x_k, and it is used to approximate the value of $f(x_k + p)$:

$$f(x_k + p) \approx f(x_k) + \nabla f(x_k)^\top p + \tfrac{1}{2} p^\top \nabla^2 f(x_k) p$$

with the restriction that the chosen step p should be small enough, as the error associated with this model is $O(\|p\|^3)$, which is small as long as p is small. However, this is only true when the exact derivatives are used. Instead, approximations to the derivatives can be considered, mainly with the intent of facilitating their computation, but ultimately making the expected error associated with the model used to increase, being now $O(\|p\|^2)$ (see [20]).

By defining $m_k(p) = f(x_k) + g_k^\top p + \frac{1}{2} p^\top B_k p$, with $g_k = \nabla f(x_k)$ and B_k a symmetric matrix that approximates the Hessian, we obtain the quadratic models used by the method. It is now time to minimize $m_k(p)$, restricting the search to a small region around the current iterate, typically a ball of radius Δ_k, defined for the current iteration, and centered at x_k. The corresponding optimization problem, from which the direction p is computed, corresponding to the minimum, is defined as:

$$\begin{array}{lrcl} \text{minimize} & m_k(p) & = & f(x_k) + g_k^\top p + \frac{1}{2} p^\top B_k p \\ \text{subject to} & \|p\| & \leq & \Delta_k \end{array} \tag{2.1}$$

This is called the trust-region subproblem, and while it can bring some added complexity, it is easy to solve in many situations.

Validating the model comes next, resorting to a ratio ρ_k, that compares the agreement between the decrease obtained in the model and the variation achieved when considering the actual function values:

$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)} \tag{2.2}$$

Depending on the value of this ratio, the model is considered to be accurate enough to continue to be used, or a decision about increasing its quality is taken, which is related to the update of the radius Δ_k. The ratio ρ_k is additionally used to decide if the solution of the trust-region subproblem should be accepted as a new iterate.

The update of Δ_k depends on many factors, as we can see in Algorithm 2.1. The ratio ρ_k is compared with the constants μ_1 and μ_2, with $0 < \mu_1 < \mu_2 < 1$, being that if ρ_k is inferior to μ_1 then the trust-region radius should decrease by a given factor $0 < D < 1$, as the model did not predict accurately enough the decrease in real function values. Notice that Taylor models are guaranteed to accurately approximate smooth functions for points sufficiently close to the point where the model is built. Otherwise, if ρ_k is greater than μ_2 and the step length reaches the boundary of the current trust-region, meaning that $\|p_k\| = \Delta_k$, then this indicates that the current model is adequate but that the current radius could be limiting the progress of the algorithm. Therefore, we increase the radius by a given factor $I > 1$, while keeping it below a certain maximum, $\hat{\Delta}$, that guarantees that the norm of the step p_k, and as such the error associated with the model, will never increase too much.

Another comparison constant, $\eta \geq 0$, is used to define the acceptance of the new point, $x_k + p_k$, as the new center point for the computation of a new model in the following iteration, x_{k+1}. To establish the convergence of the algorithm it is simply required that $\eta \geq 0$. However, selecting small values for η helps to not discard potentially good new iterates, and so we often see η portrayed as a value in $\left[0, \frac{1}{4}\right)$. Other typical values considered for the different parameters are $\mu_1 = \frac{1}{4}$, $\mu_2 = \frac{3}{4}$, $D = \frac{1}{4}$ and $I = 2$. Notice also that having $\eta > 0$ means that sufficient decrease is required for the acceptance of a new point.

Algorithm 2.1 A basic trust-region method

Requires: Initial point x_0, $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, $0 < \mu_1 \leq \frac{1}{2}$, $\mu_1 < \mu_2 < 1$,$0 < D < 1 < I$, and $\eta \geq 0$
for k=0, 1, 2, ... **do**
  Solve the trust-region subproblem, (2.1), obtaining the direction p_k.
  Compute the ratio ρ_k.
  if $\rho_k < \mu_1$ **then**
    $\Delta_{k+1} = D * \Delta_k$
  else if $\rho_k > \mu_2$ **and** $\|p_k\| = \Delta_k$ **then**
    $\Delta_{k+1} = \min(I * \Delta_k, \hat{\Delta})$
  else
    $\Delta_{k+1} = \Delta_k$
  end if
  if $\rho_k > \eta$ **then**
    $x_{k+1} = x_k + p_k$
  else
    $x_{k+1} = x_k$
  end if
end for

2.2 The trust-region subproblem

The solution of the trust-region subproblem mentioned in Section 2.1 heavily depends on the characteristics of the matrix B_k used in the quadratic model. The most common methods to obtain an approximate solution of the trust-region subproblem include the Dogleg method, if the matrix B_k is positive definite, and the two-dimensional subspace minimization, for indefinite matrices (see [20]). Both strategies try to find solutions that achieve as much reduction as the Cauchy point, a definition that is crucial to prove convergence of the algorithm, as we will see in the next section. However, as this subproblem has been already thoroughly explored, we will not detail the different solution approaches and will instead resort to code already developed by the nonlinear optimization community (see [18]), covering all possible characteristics of the matrix B_k.

2.2.1 The Cauchy point

The Cauchy point, usually denoted by p_k^C, quantifies the sufficient reduction required in the model to obtain convergence of the trust-region method. This means that convergence will be guaranteed if the steps obtained from solving the trust-region subproblem, regardless of the technique considered, provide reductions similar to the reduction obtained with the Cauchy point.

The Cauchy point is the minimizer of the model m_k along the steepest descent direction $-g_k$. To compute it, we first consider a linear version of the trust-region subproblem, returning a direction that we denote by p_k^s. Second, we compute the optimal step length, τ_k, when following the direction p_k^s:

$$p_k^s = \arg\min_{p \in \mathbb{R}^n} f(x_k) + g_k^\top p \quad \text{s.t. } \|p\| \leq \Delta_k;$$

$$\tau_k = \arg\min_{\tau \geq 0} m_k(\tau p_k^s) \quad \text{s.t. } \|\tau p_k^s\| \leq \Delta_k;$$

Finally, we define the Cauchy point as

$$p_k^C = \tau_k p_k^s. \tag{2.3}$$

Solving the minimization problem for p_k^s is easy, as it is a linear problem. Its solution is obtained by taking the symmetric direction of the gradient and resizing it to the maximum allowed step length:

$$p_k^s = -\frac{\Delta_k}{\|g_k\|} g_k \tag{2.4}$$

Likewise, solving the second problem is also easy. We consider the two cases, $g_k^\top B_k g_k \leq 0$ and $g_k^\top B_k g_k > 0$. In the first case, we have the model function $m_k(\tau p_k^s)$ decreasing with τ, which means the minimum occurs at the limit of the trust-region. Since $\|p_k^s\| = \Delta_k$, this would mean that $\tau_k = 1$ when $g_k^\top B_k g_k \leq 0$. As for the second case, we know that $m_k(\tau p_k^s)$ is a convex quadratic function in τ, for which the minimum can be easily computed. Since we are still minimizing within the radius Δ_k, we must restrain the value of τ_k to either be the known minimizer of the quadratic function, $\frac{\|g_k\|^3}{\Delta_k g_k^\top B_k g_k}$, or 1 corresponding to the limit of the trust-region radius.

Concluding, the values that τ_k may assume are

$$\tau_k = \begin{cases} 1 & \text{if } g_k^\top B_k g_k \leq 0 \\ \min\left(\frac{\|g_k\|^3}{\Delta_k g_k^\top B_k g_k}, 1\right) & \text{otherwise} \end{cases} \tag{2.5}$$

As this step is inexpensive to calculate and for convergence purposes it is enough that the solution p_k of the quadratic model lies inside the trust-region and gives a reduction that can be quantified in terms of the Cauchy point (as we will see in Section 2.3), this method could be used as one of the strategies for solving the trust-region subproblem. However, improvement is still possible, as the properties of the approximation B_k to the Hessian matrix have not been explored. As we mention before, depending on the characteristics of the matrix B_k, other strategies have been considered in the literature [2, 9, 20].

2.3 Convergence analysis

Since the global convergence of the trust-region method depends on the model decrease obtained by the solution of the trust-region subproblem being proportional to the one obtained by the Cauchy point, we start this section by quantifying it. We follow closely

the line of reasoning of [20] and we will show that any solution p_k of the subproblem is required to provide a decrease in the model function that verifies

$$m_k(0) - m_k(p_k) \geq c_1 \|g_k\| \min\left(\Delta_k, \frac{\|g_k\|}{\|B_k\|}\right), \tag{2.6}$$

for $0 < c_1 \leq 1$. This condition is enough to establish the convergence of the trust-region method.

In [20], it is also noted that if Δ_k is the minimum in that expression, then we have a similar condition to the first Wolfe condition, that the desired reduction in the model is proportional to the gradient and the step size that is taken.

Theorem 1. *The Cauchy point satisfies (2.6) for* $c_1 = \frac{1}{2}$.

Proof. For simplicity, we omit the iteration index k during this proof.

Let's consider these three cases: $g^\top Bg \leq 0$, $g^\top Bg > 0$ and $\frac{\|g\|^3}{\Delta g^\top Bg} \leq 1$, and finally $g^\top Bg > 0$ and $\frac{\|g\|^3}{\Delta g^\top Bg} > 1$.

For the first case, we have

$$\begin{aligned} m(p^C) - m(0) &= m\left(-\frac{\Delta g}{\|g\|}\right) - f = -\frac{\Delta}{\|g\|}\|g\|^2 + \frac{1}{2}\frac{\Delta^2}{\|g\|^2} g^\top Bg \leq \\ &\leq -\Delta\|g\| \leq -\|g\| \min\left(\Delta, \frac{\|g\|}{\|B\|}\right) \leq -\frac{1}{2}\|g\| \min\left(\Delta, \frac{\|g\|}{\|B\|}\right). \end{aligned}$$

For the second case, as stated in Section 2.2.1, with equations (2.3), (2.4) and (2.5), $p^C = \tau p^s$, with $p^S = -\frac{\Delta}{\|g\|} g$ and $\tau = \frac{\|g\|^3}{\Delta g^\top Bg}$, which means that

$$\begin{aligned} m(p^C) - m(0) &= -\frac{\|g\|^4}{g^\top Bg} + \frac{1}{2} g^\top Bg \frac{\|g\|^4}{(g^\top Bg)^2} = -\frac{1}{2}\frac{\|g\|^4}{g^\top Bg} \leq \\ &\leq -\frac{1}{2}\frac{\|g\|^4}{\|B\|\|g\|^2} = -\frac{1}{2}\frac{\|g\|^2}{\|B\|} \leq -\frac{1}{2}\|g\| \min\left(\Delta, \frac{\|g\|}{\|B\|}\right). \end{aligned}$$

In the final case, we have $\frac{\|g\|^3}{\Delta g^\top Bg} > 1$ which is equivalent to $g^\top Bg < \frac{\|g\|^3}{\Delta}$ since $g^\top Bg > 0$. By applying it to the difference of the models, we obtain:

$$\begin{aligned} m(p^C) - m(0) &= -\frac{\Delta}{\|g\|}\|g\|^2 + \frac{1}{2}\frac{\Delta^2}{\|g\|^2} g^\top Bg \leq -\Delta\|g\| + \frac{1}{2}\frac{\Delta^2}{\|g\|^2}\frac{\|g\|^3}{\Delta} = \\ &= -\frac{1}{2}\Delta\|g\| \leq -\frac{1}{2}\|g\| \min\left(\Delta, \frac{\|g\|}{\|B\|}\right). \end{aligned}$$

which ends the analysis of all possible cases and as such finalizes the proof.

□

The decrease stated in condition (2.6) can be attained if our step size achieves a decrease that is at least a fraction $2 \geq c_2 > 0$ of the decrease obtained with the Cauchy point.

Theorem 2. *Let p_k be a step such that $\|p_k\| \leq \Delta_k$ and $m_k(0) - m_k(p_k) \geq c_2(m_k(0) - m_k(p_k^C))$, for $2 \geq c_2 > 0$. Then p_k satisfies (2.6) with $c_1 = \frac{c_2}{2}$.*

Proof.

$$m_k(0) - m_k(p_k) \geq c_2(m_k(0) - m_k(p_k^C)) \geq \frac{1}{2}c_2\|g_k\|\min\left(\Delta_k, \frac{\|g_k\|}{\|B_k\|}\right).$$

□

With this, we can now infer on the behaviour of the gradient throughout the iterations.

Theorem 3. *Let $\eta = 0$. Assume there is $\beta > 0$ such that $\|B_k\| \leq \beta, \forall k \in \mathbb{N}$. Let f be lower bounded on the set $S \overset{\text{def}}{=} \{x \mid f(x) \leq f(x_0)\}$ and Lipschitz continuously differentiable in a neighbourhood of a fixed radius $R_0 > 0$ of S, defined as $S(R_0) \overset{\text{def}}{=} \{x \mid \|x - y\| < R_0,\ y \in S\}$. Then*

$$\liminf_{k\to\infty} \|g_k\| = 0.$$

Proof. From Taylor's theorem, we have that

$$f(x_k + p_k) = f(x_k) + g(x_k)^\top p_k + \int_0^1 [g(x_k + tp_k) - g(x_k)]^\top p_k \, dt$$

Therefore, from the definition of m_k and with some algebraic manipulation, we get

$$\begin{aligned}
|m_k(p_k) - f(x_k + p_k)| &= \left|\frac{1}{2}p_k^\top B_k p_k - \int_0^1 [g(x_k + tp_k) - g(x_k)]^\top p_k \, dt\right| \leq \\
&\leq \left|\frac{1}{2}p_k^\top B_k p_k\right| + \int_0^1 \left|[g(x_k + tp_k) - g(x_k)]^\top p_k\right| dt \leq \\
&\leq \left|\frac{1}{2}p_k^\top B_k p_k\right| + \int_0^1 \|g(x_k + tp_k) - g(x_k)\|\|p_k\| dt \leq \\
&\leq \left|\frac{1}{2}p_k^\top B_k p_k\right| + \int_0^1 L\|x_k + tp_k - x_k\| dt \|p_k\| = \\
&= \left|\frac{1}{2}p_k^\top B_k p_k\right| + \int_0^1 L|t| \, dt \|p_k\|^2 = \\
&= \left|\frac{1}{2}p_k^\top B_k p_k\right| + L\|p_k\|^2 \frac{1}{2} \leq \frac{\beta}{2}\|p_k\|^2 + \frac{L\|p_k\|^2}{2}
\end{aligned} \tag{2.7}$$

where L is the Lipschitz constant for g on $S(R_0)$. We also assume that $\|p_k\| \leq R_0$ to ensure that x_k and $x_k + tp_k$ both belong to the set $S(R_0)$.

Let's suppose for contradiction that no subsequence of $\{\|g_k\|\}$ converges to zero. This means that there is $\epsilon > 0$ and a sufficiently large index $K \in \mathbb{N}$ such that

$$\|g_k\| \geq \epsilon, \quad \forall k \geq K \tag{2.8}$$

Using (2.6) and (2.8), we have that for indexes $k \geq K$,

$$m_k(0) - m_k(p_k) \geq c_1\|g_k\|\min\left(\Delta_k, \frac{\|g_k\|}{\|B_k\|}\right) \geq c_1\epsilon\min\left(\Delta_k, \frac{\epsilon}{\beta}\right) \tag{2.9}$$

We now build upon the ratio ρ_k with the following manipulation, using the previous equations (2.7) and (2.9) and the fact that $m_k(0) = f(x_k)$:

$$\begin{aligned} |\rho_k - 1| &= \left| \frac{(f(x_k) - f(x_k + p_k)) - (m_k(0) - m_k(p_k))}{m_k(0) - m_k(p_k)} \right| = \\ &= \left| \frac{m_k(p_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)} \right| \leq \frac{\|p_k\|^2 \left(\frac{\beta}{2} + \frac{L}{2}\right)}{c_1 \epsilon \min\left(\Delta_k, \frac{\epsilon}{\beta}\right)} \leq \frac{\Delta_k^2 \left(\frac{\beta}{2} + \frac{L}{2}\right)}{c_1 \epsilon \min\left(\Delta_k, \frac{\epsilon}{\beta}\right)} \end{aligned} \tag{2.10}$$

With the structure of the trust-region method in mind, we now bound the right side of (2.10) by limiting the values of Δ_k with a maximum $\bar{\Delta}$, defined as follows:

$$\bar{\Delta} = \min\left(\frac{c_1 \epsilon}{\beta + L}, R_0\right). \tag{2.11}$$

Notice that the term R_0 guarantees $\|p_k\| \leq \Delta_k \leq \bar{\Delta} \leq R_0$. It can also be proved that, since $c_1 \leq 1$, we have $\bar{\Delta} \leq \frac{\epsilon}{\beta}$. This means that $\forall \Delta_k \in (0, \bar{\Delta}]$, $\min\left(\Delta_k, \frac{\epsilon}{\beta}\right) = \Delta_k$, allowing the previous manipulation (2.10) to be expanded further:

$$|\rho_k - 1| \leq \frac{\Delta_k^2 (\frac{\beta}{2} + \frac{L}{2})}{c_1 \epsilon \min\left(\Delta_k, \frac{\epsilon}{\beta}\right)} = \frac{\Delta_k^2 (\frac{\beta}{2} + \frac{L}{2})}{c_1 \epsilon \Delta_k} = \frac{\Delta_k (\frac{\beta}{2} + \frac{L}{2})}{c_1 \epsilon} \leq \frac{\bar{\Delta} (\frac{\beta}{2} + \frac{L}{2})}{c_1 \epsilon} \leq \frac{1}{2}, \tag{2.12}$$

since $\bar{\Delta} \leq \frac{c_1 \epsilon}{\beta + L}$.

This, however, would make $\rho_k \geq \frac{1}{2} \geq \mu_1$. We then have $\Delta_{k+1} \geq \Delta_k$ whenever $\Delta_k \leq \bar{\Delta}$, which means that the reduction in Δ_k comes when $\Delta_k > \bar{\Delta}$. From this we conclude that

$$\Delta_k \geq \min(\Delta_K, \bar{\Delta} D), \quad \forall k \geq K \tag{2.13}$$

We now suppose that there is an infinite subsequence $\mathcal{K}$ in which $\forall k \in \mathcal{K}$, $\rho_k \geq \mu_1$. Then, for $k \in \mathcal{K}$ and $k \geq K$ we have

$$f(x_k) - f(x_{k+1}) = f(x_k) - f(x_k + p_k) \geq \mu_1 [m_k(0) - m_k(p_k)] \geq \mu_1 c_1 \epsilon \min\left(\Delta_k, \frac{\epsilon}{\beta}\right).$$

Since we know that f is bounded below and that at each iteration the function values either are kept constant or are decreased, the left-hand side converges to zero. This means that the right-hand side will also have to converge to zero and as such we can write

$$\lim_{k \in \mathcal{K},\, k \to \infty} \Delta_k = 0.$$

This contradicts the previous statement in (2.13). Therefore, the subsequence $\mathcal{K}$ cannot exist and so, for every subsequence $\mathcal{K}$ there must exist $k \in \mathcal{K}$ such that $\rho_k < \mu_1$. This implies that $\rho_k < \mu_1$ for all $k \in \mathcal{K}$ sufficiently large. Otherwise, it would be possible to consider a subsequence of $\mathcal{K}$ with the property $\rho_k \geq \mu_1$, which would lead to the previously established contradiction. So, since $\rho_k < \mu_1$ for all sufficiently large $k \in \mathcal{K}$, the

algorithm will reduce Δ_k by a constant factor of D, reaching the result $\lim_{k\to\infty} \Delta_k = 0$, again contradicting the same statement. This means that the initial assumption $\|g_k\| \geq \epsilon, \forall k \geq K$ is false, giving us the desired result. □

We can now prove a stronger version of this result, by imposing $\eta > 0$.

Theorem 4. *Suppose that $\eta > 0$ and that all the hypothesis of Theorem 3 are satisfied. Then*

$$\lim_{k\to\infty} g_k = 0$$

Proof. First, let's consider a positive index m with $g_m = g(x_m) \neq 0$. Using L again to denote the Lipschitz constant for g on $S(R_0)$, we have the following:

$$\|g(x) - g_m\| \leq L\|x - x_m\|, \forall x \in S(R_0).$$

Defining the constants ϵ and R as

$$\epsilon = \frac{1}{2}\|g_m\| > 0, \quad R = \min\left(\frac{\epsilon}{L}, R_0\right),$$

we can consider the ball $\mathcal{B}(x_m, R)$ defined as

$$\mathcal{B}(x_m, R) = \{x \mid \|x - x_m\| \leq R\},$$

which is contained in $S(R_0)$ and so Lipschitz continuity of the gradient within this ball is guaranteed. Hence,

$$x \in \mathcal{B}(x_m, R) \implies \|g(x)\| \geq \|g_m\| - \|g(x) - g_m\| \geq \frac{1}{2}\|g_m\| = \epsilon. \tag{2.14}$$

We can now infer on the sequence $\{x_k\}_{k\geq m}$. If it stays contained within the ball $\mathcal{B}(x_m, R)$, then we would have $\|g_k\| \geq \epsilon > 0$, $\forall k \geq m$. This can be proven false using a reasoning similar to the one of the previous proof and so we can conclude that the sequence $\{x_k\}_{k\geq m}$ will eventually leave the ball $\mathcal{B}(x_m, R)$.

Let $l + 1$, with $l \geq m$, denote the index of the first iterate after x_m to leave the ball $\mathcal{B}(x_m, R)$. Then $\|g_k\| \geq \epsilon$, $\forall k \in [m, l]$, $k \in \mathbb{N}$ and we can use assertion (2.9) to write

$$\begin{aligned} f(x_m) - f(x_{l+1}) &= \sum_{k=m}^{l} f(x_k) - f(x_{k+1}) \geq \\ &\geq \sum_{k=m, x_k \neq x_{k+1}}^{l} \eta[m_k(0) - m_k(p_k)] \geq \\ &\geq \sum_{k=m, x_k \neq x_{k+1}}^{l} \eta c_1 \epsilon \min\left(\Delta_k, \frac{\epsilon}{\beta}\right) \end{aligned} \tag{2.15}$$

where $x_k \neq x_{k+1}$ limits the sum to iterations where a non null step has been taken. Let's now explore the cases where $\Delta_k \leq \frac{\epsilon}{\beta}$, $\forall k \in [m, l]$, $k \in \mathbb{N}$, and where $\Delta_k > \frac{\epsilon}{\beta}$ for some k in the same conditions. In the first case, we have

$$f(x_m) - f(x_{l+1}) \geq \eta c_1 \epsilon \sum_{k=m, x_k \neq x_{k+1}}^{l} \Delta_k \geq \eta c_1 \epsilon R = \eta c_1 \epsilon \min\left(\frac{\epsilon}{L}, R_0\right), \tag{2.16}$$

while in the second case, we have

$$f(x_m) - f(x_{l+1}) \geq \eta c_1 \epsilon \frac{\epsilon}{\beta}. \tag{2.17}$$

Joining the two, we can safely say that

$$f(x_m) - f(x_{l+1}) \geq \eta c_1 \epsilon \min\left(\frac{\epsilon}{\beta}, \frac{\epsilon}{L}, R_0\right). \tag{2.18}$$

Let's now denote the lower bound of the function by a finite value $f^\star$. We know that $f(x_m) - f^\star \downarrow 0$, since the sequence $\{f(x_k)\}_{k=0}^{\infty}$ is decreasing and bounded below. Expanding on this difference, we also get

$$f(x_m) - f^\star \geq f(x_m) - f(x_{l+1}) \geq \eta c_1 \epsilon \min\left(\frac{\epsilon}{\beta}, \frac{\epsilon}{L}, R_0\right) = \eta c_1 \frac{1}{2} \|g_m\| \min\left(\frac{\|g_m\|}{2\beta}, \frac{\|g_m\|}{2L}, R_0\right) > 0, \tag{2.19}$$

from which we can retrieve that $g_m \to 0$, concluding the proof.

□

The previous result states that the gradient converges to zero, proving that we have reached a stationary point. Note that this does not guarantee convergence to a minimum. For that we need to introduce second-order information. If we guarantee that $B_k = \nabla^2 f(x_k)$ at every iteration, then second-order sufficient conditions are met and the stationary point reached is in fact at least a local minimum of the function (see [18]).

Chapter 3

Global Optimization based on Trust-region Methods and Clever Multistart

3.1 General structure

As mentioned before, our goal is to address global minimization problems, with continuous variables and second order differentiable objective functions, subject to bound constraints, which will guarantee the existence of a global minimum. We will resource to an iterative trust-region method, coupled with a clever multistart strategy, that includes merging steps for a better performance.

The main structure of the algorithm is organized in three steps: initialization/relaunching step, trust-region step, and stopping criteria step. The algorithm keeps a list of points, classified as active or inactive, and generated at any of the first two steps. Active points can be selected as centers for the trust-region step and will be the candidates to local minimizers. Inactive points flag subdomains of the feasible region where the objective function presents bad values.

After generating a set of initializations, the corresponding points are compared and classified as active or inactive (this classification will be detailed in Section 3.2). An active point is then selected for a trust-region step. Different criteria can be considered for this selection and will be detailed in the following sections. A quadratic Taylor-model is built for the point selected, locally representing the objective function, and minimized in a trust-region around the current iterate, considering the bounds defining the feasible region (see Chapter 2).

If the point resulting from the model minimization is accepted, according to the trust-region ratio (see Chapter 2), the list of points is updated by comparing it to the points already in the list. If the new point is close to some point already in the list, the corresponding function values are compared and only the best point is retained.

Although, if the new point is far from every point in the list, it will be accepted. Different measures of closeness can be adopted that use the trust-region radius associated to each point. If accepted, the new point will be added to the list, after being classified as active or inactive (see Section 3.2). After, stopping criteria are checked and, if not met, a decision about the need of performing new initializations is taken, possibly relaunching new points. Figure 3.1 summarizes the main steps of the algorithm.

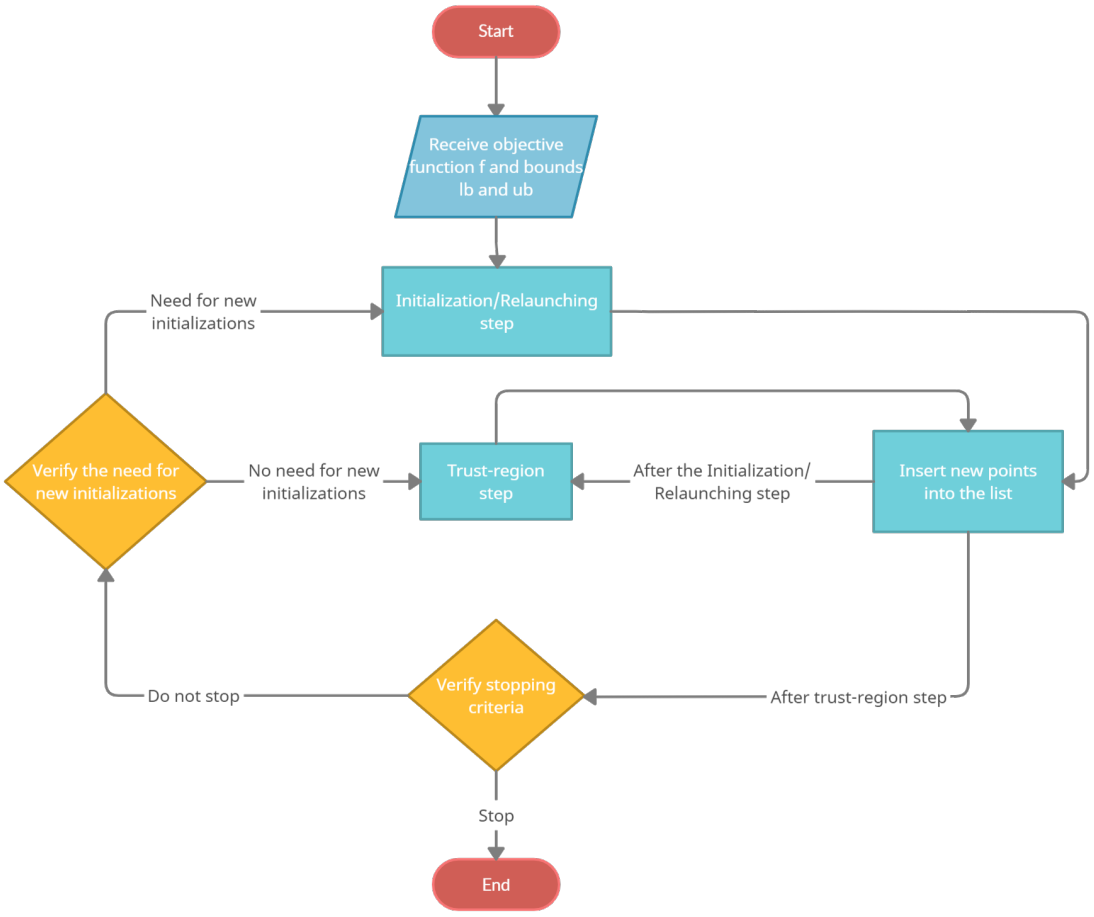

Figure 3.1: Flowchart illustrating the algorithmic steps.

In the following sections, detailed descriptions will be provided for the different procedures.

3.2 Adding points to the list

Anytime that a point is generated by the algorithm, it should be classified as active or inactive, possibly being added to the list. Points are saved in the list as tuples of form $(x; f_x; r_x; a_x; m_x)$, with $x \in \mathbb{R}^n$ representing the coordinates of the point, f_x the corresponding function value, r_x the associated trust-region radius, a_x a binary status classifying the

point as active or inactive, and m_x representing a binary status that indicates if the point has already been identified as a local minimizer.

It is important to define which points are worth keeping memory of and of those, which ones are also worth being used as centers for the trust-region step. There is no value in exploring points that are already identified as local minimizers. So, if a second-order stationarity condition is satisfied, m_x will be set equal to 1. These points will not be selected as model centers in the trust-region step.

Points that are isolated from every point in the list are added to it as active points, representing a part of the feasible region not yet explored. In this case, an initial trust-region radius is defined for the point. This can only occur at the initialization/relaunching step, since a point generated at the trust-region step always receives the trust-region radius resulting from the trust-region iteration.

When a point is comparable with another point in the list, the point with the worst function value will change its status to inactive. If a new point presents a better function value than at least one active point in the list, then this new point will be added to the list and the point already in the list will change its status to inactive. The status of the new point will be active, if no other point in the list comparable with it presents a better objective function value, or inactive, if there is another point in the list comparable with it that presents a better function value than the new point. In this last case, a merge has occurred. This merging procedure distinguishes the proposed algorithm from a local trust-region method coupled with a multistart strategy, since not all the trust-region searches will be conducted until the end.

Different comparison measures can be adopted to represent the proximity between two points. Natural choices would be to choose the minimum or the maximum between the trust-region radius of the points x, y under comparison, but other approaches could be taken. This would be translated into the use of function $m(r_x, r_y)$ in Algorithm 3.1, where r_i represents the trust-region radius of point i.

When a new point is compared to the list, an initial trust-region radius should be defined for it. If the new point was generated at an initialization/relaunching step, then it receives the initial trust-region radius. If the new point was generated at a trust-region step, then the trust-region center changes its status to inactive and the new point receives the trust-region radius, after being updated according to the rules described in Chapter 2.

This initial trust-region radius is used in function $m(\cdot, \cdot)$ to decide to which list points the new point should be compared to. It is also used in the definition of the final trust-region radius that will be the one of the new point when added to the list. The proposed formula avoids very large radius, resulting from the initialization/relaunching step, when the point that will be added to the list is going to replace a point which has already been explored and has a reasonably small trust-region radius.

Therefore, the final trust-region radius of the point being added to the list tries to maintain some of the spatial exploration provided by the points being inactivated. With this purpose, an auxiliary radius, r_{aux}, is defined, by adding the distance between the two points to the radius associated with the point that is going to be inactive. In this way, some information of the inactivated point is kept, allowing the computation of a more precise model when the list points already have small radius. However, in order to keep the convergence results of the base trust-region method, the minimum between this value and the radius obtained from the trust-region step should always be considered (see Section 3.6).

Figure 3.2 illustrates some possible cases for the updated radius, when the resulting radius r_{aux} is either larger or smaller than the radius provided by the trust-region step. Algorithm 3.1 formalizes the procedures described.

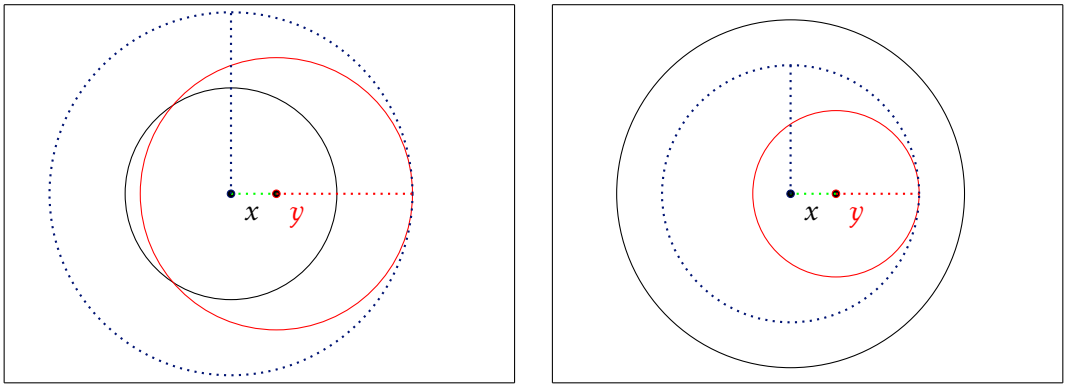

Figure 3.2: Resulting r_{aux} (dotted blue circle) from the comparison between the new point x (in black) and the list point y (in red). The resulting radius cannot be accepted on the left example, as it exceeds the original radius provided by the trust-region step, but it is accepted on the right.

Algorithm 3.1 Adding points to the list

Requires: list L of saved tuples, list L_{ini} of new tuples
for all $(x, f_x, r_x, a_x, m_x) \in L_{ini}$ **do**
  if $\min_{y \in L}(\|x - y\| - m(r_y, r_x)) > 0$ **then**
    if x is a local minimizer **then**
      $L = L \cup (x, f_x, r_x, 1, 1)$
    else
      $L = L \cup (x, f_x, r_x, 1, 0)$
    end if
  else
    $add_x = 0$; $r_{aux} = 0$; $a_x = 1$
    for all $y \in L$ **do**
      if $\|x - y\| \leq m(r_x, r_y)$ **then**
        if $f_x < f_y$ **then**
          $r_{aux} = \max\left(r_{aux}, \|y - x\| + r_y\right)$
          if $a_y = 1$ **then**
            $add_x = 1$
          end if
          $a_y = 0$
        else
          $a_x = 0$
        end if
      end if
    end for
    if $add_x = 1$ **then**
      $r_x = \min(r_x, r_{aux})$
      if x is a local minimizer **then**
        $L = L \cup (x, f_x, r_x, a_x, 1)$
      else
        $L = L \cup (x, f_x, r_x, a_x, 0)$
      end if
    end if
  end if
end for

3.3 Initialization/relaunching step

At the beginning of the optimization process, a point or a list of feasible points should be provided to the algorithm. A simple way of doing it is by considering the centroid of the feasible region, when a single initialization is intended, or to use a linestart approach, if more than one point is required. Linestart considers a line connecting the problem bounds and generates a given number of equally spaced points within this line, diagonally exploring the feasible region.

Additionally, at the end of each iteration, a decision is made about the need of new initializations. This could be related to the frequency of unsuccessful iterations or to

the number of active points, not yet identified as local minimizers, remaining in the list. Different strategies can be used for generating new points. Random uniform sampling or Latin hypercube sampling [17] are possible choices but would confer a random behavior to the algorithm.

Alternatively, low-discrepancy sequences, like Sobol sequences [21] or Halton sequences [10] can be used, allowing the algorithm to remain deterministic, while providing an efficient and thorough exploration of the feasible region. For reasonably small numbers of points, these low-discrepancy sequences provide a dense coverage of the entire region, unlike pure uniform random generation that may leave some areas poorly represented and others overly characterized (see Figure 3.3). Works detailing these sequences usually refer good results when applying them to numerical integration using quasi-Monte Carlo rules, which requires a dense coverage of the domain (see [5, 7, 19]).

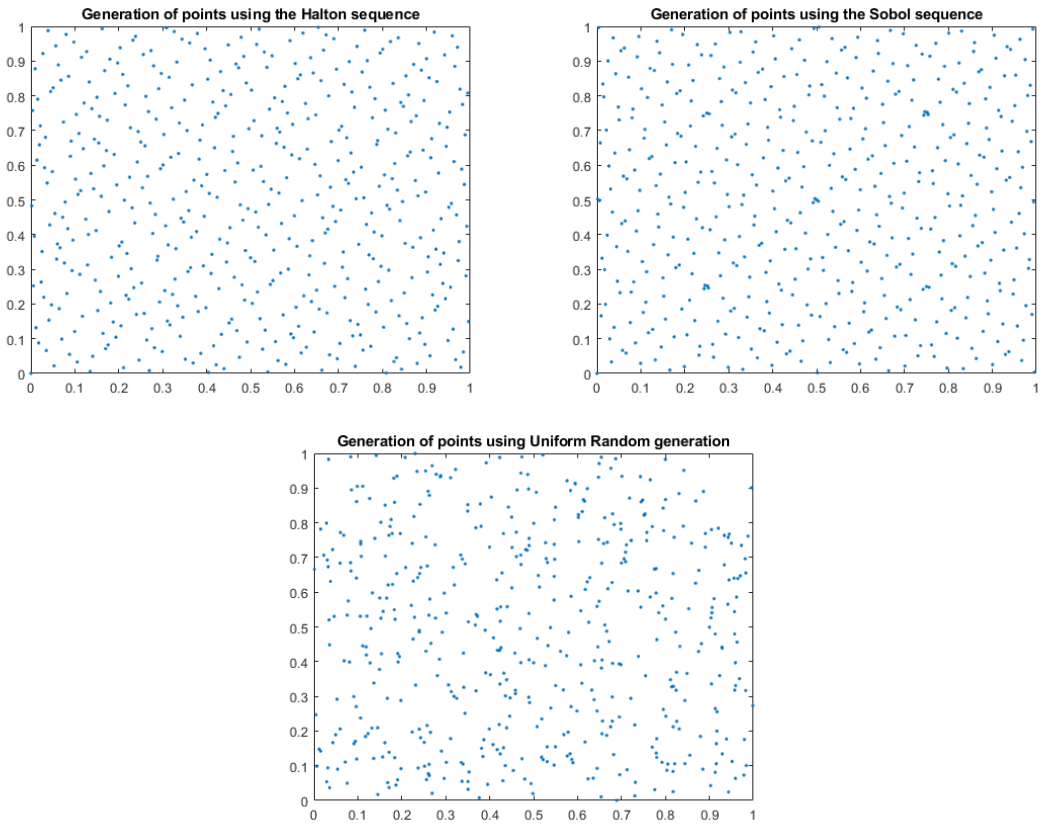

Figure 3.3: Spread of 500 generated points through the Halton sequence (top left), the Sobol sequence (top right) and uniform random generation (bottom).

In [3] the authors proposed the 2^n-centers method, which consecutively divides the box representing the feasible region into smaller boxes, using the centroid of each of the new boxes as initialization. This method scales poorly with the problem dimension, so it can be only valuable for problems with a very low number of variables.

3.4 Trust-region step

The first decision to be taken at this step is to select the point where to build the Taylor model and where to perform the trust-region iteration. Only active points not yet identified as local minimizers are worthy of being selected. Since the main goal is to identify the global minimum, a greedy approach will dictate the selection of the point presenting the lowest objective function value. Algorithm 3.2 details the procedure.

Algorithm 3.2 Selecting the model center

Requires: list L of saved tuples, r_{tol} minimum tolerance for trust-region radius
Select $x_{centers}$, subset of L where $\forall (x, f_x, r_x, a_x, m_x) \in x_{centers}, r_x > r_{tol} \wedge a_x = 1 \wedge m_x = 0$
$x_{centers} = \arg\min_{x \in x_{centers}}(f_x)$
if $|x_{centers}| > 1$ **then**
  $x_{centers} = \arg\max_{x \in x_{centers}}(r_x)$
end if
if $|x_{centers}| > 1$ **then**
  $x_{centers} = x_{centers}(1)$
end if

As we can see, ties are broken by selecting the point with the largest trust-region radius, meaning that the corresponding line of search has not yet been fully explored or is being successful in generating new points with a good agreement between the true function and the Taylor's model. If ties prevail, we simply choose the first point stored, as eventually all points in these conditions will be explored.

After choosing the model center, it is used in a single iteration of a trust-region method (see Algorithm 2.1). Changes in the basic trust-region step are not required. If the solution of the trust-region subproblem is accepted, then an attempt of adding the point to the list is performed, following the procedure detailed in Section 3.2.

If the point fails in being added as active to the list, the iteration is declared as unsuccessful, which is relevant when the criterion for initializing new searches is based on the number of consecutive unsuccessful iterations.

3.5 Stopping criteria step

When analyzing the limit behavior of an algorithm, stopping criteria should not be considered. However, in practical settings, some strategies should be defined, allowing to improve the numerical efficiency of the corresponding implementation.

Besides only allowing a maximum number of iterations for the algorithm, the iterative procedure will stop if all active points have been identified as being local minimizers, meaning that the norm of the corresponding gradient is below a given small positive threshold and the eigenvalues of the Hessian matrix are strictly positive, or if all points

have already been extensively explored, which traduces in a small value for the corresponding trust-region radius. This last stopping criteria is justified by the bounded feasible region. A local minimum may occur at the border of the feasible domain, where optimality conditions may fail to identify stationarity.

3.6 Convergence results

When proposing a new algorithm, more than exhibiting compelling numerical results reporting the corresponding numerical performance, the algorithmic structure should be analyzed, ensuring that the good numerical behavior was not obtained by chance.

The proposed algorithm relies heavily on the trust-region step. Trust-region methods have shown to be locally convergent in Section 2.3, regardless of the initialization provided. Thus, if a trust-region method is coupled with a simple multistart strategy, local convergence will still hold, for every sequence initialized at a different point. However, the proposed algorithm does not use a simple multistart strategy due to the procedure considered for adding new points to the list (see Section 3.2).

What makes the adding procedure problematic is the update of the trust-region radius. Under the conditions of Section 2.3, for ensuring convergence, any solution of the trust-region subproblem is required to provide a decrease in the model that satisfies equation:

$$m_k(0) - m_k(p_k) \geq c_1 \|g_k\| \min\left(\Delta_k, \frac{\|g_k\|}{\|B_k\|}\right), \tag{3.1}$$

where Δ_k represents the trust-region radius.

The dependence on the trust-region radius is clear. When an iteration of a trust-region method is performed, the approaches considered for solving the trust-region subproblem will ensure this decrease. Since in the procedure for adding a new point to the list, the new trust-region radius never exceeds the one resulting from the trust-region step, the previous inequality will still hold for the new radius. Local convergence is guaranteed, in similar conditions to the ones of Section 2.3, for every linked subsequence, where each point was computed from the previous one at a trust-region step or, when being added to the list, it inactivates the previous point in the linked sequence.

Chapter 4

Numerical Experiments

4.1 Performance assessment

This chapter reports the numerical experiments that were conducted during the algorithmic development, allowing to assess the performance of the code and to propose new algorithmic variants.

All the implementations were developed in MATLAB [16]. At the beginning, the algorithm was initialized with linestart, evenly spacing in the line joining the problem bounds as many points as the problem dimension. Additionally, the centroid of the feasible region was always considered as initialization. For the trust-region step, the base version of the algorithm considered typical parameters from the literature (see [20]), namely $\mu_1 = 1/4$, $\mu_2 = 3/4$, and $\eta = 1/10$ (see Algorithm 2.1). The initial value and the maximum size allowed for the trust-region radius were considered problem dependent, being defined as $\Delta_0 = \|ub - lb\|/10$, and $\hat{\Delta} = \|ub - lb\|/3$, where lb and ub denote the lower and upper bounds of the problem. For updating the trust-region radius, the factors $I = 6/5$ and $D = 1/4$ were used. These values result from a light calibration that was performed over the base version.

In the trust-region step, the solution of the trust-region subproblem, corresponding to the minimum of the quadratic Taylor-model inside the considered trust-region, was obtained with function *trust* from MATLAB. This function requires the gradient vector and the Hessian matrix as input, which define the current quadratic model. Since information about derivatives was not available for the problem collection and considering that the codification of this information was time-consuming and prone to error, the Matlab suite DERIVESTsuite (see [4]) was used, which considers finite-differences to estimate derivatives, through the function *derivest*. This function is used jointly with functions *gradest* and *hessian*, to produce the approximations to the first and second order derivatives of a

given function. Function *gradest* simply performs repeated calls to *derivest*, while function *hessian*, as the author explains, applies an "efficient computation of the off-diagonal elements of the Hessian matrix", improved by applying Romberg extrapolation. It is not possible to pass bounds to function *trust*, when solving the trust-region subproblem. To address the bound constraints, the resulting point was projected on the hyperrectangle that defines the feasible region.

When comparing two points, the largest ratio between the two points is used for comparison, by setting $m(r_x, r_y) = \max(r_x, r_y)$, promoting merging.

The algorithm would stop once that all active points have been identified as local minimizers, meaning that the norm of the corresponding gradient vector was below 10^{-5} and all eigenvalues of the Hessian matrix were larger than 10^{-8}, or when all active points presented a trust-region radius below 10^{-4}. A maximum number of 5000 iterations was allowed.

To select the best algorithmic variant, from the ones tested, and to assess the numerical performance of the algorithm by comparison with other solvers, performance profiles (see [6]) have been used , which will be described in the next section.

4.1.1 Profiles and metrics

Performance profiling is a technique that facilitates the assessment to the performance of solvers on a certain metric through a plot of a cumulative distribution function $\rho_s(\theta)$, related to performance ratios of solver s. It is particularly useful in large test sets, when analyzing large tables of results would be unpractical. However, it can also prove to be useful in test sets of small sizes, due to the benefits of the visual representation, that allows an easy assessment of data. Performance profiles also facilitate the comparison of a large number of solvers.

Let $t_s(p)$ denote the measured performance of solver $s \in \mathcal{S}$ on problem $p \in \mathcal{P}$ for a specific metric, assuming that $t_s(p) > 0$ and that lower values of $t_s(p)$ indicate a better performance. The performance ratio is defined by

$$\tau_{p,s} := \frac{t_s(p)}{min\{t_\sigma(p) : \sigma \in \mathcal{S}\}} \geq 1. \tag{4.1}$$

The cumulative distribution function for a solver $s \in \mathcal{S}$ is therefore defined by

$$\rho_s(\theta) = \frac{1}{|\mathcal{P}|} \times \left|\left\{p \in \mathcal{P} : \tau_{p,s} \leq \theta\right\}\right|, \tag{4.2}$$

with $\rho_s(\theta)$ being the probability that solver s is within a factor θ of the best performance over all solvers on the set $\mathcal{P}$. Thus, the value of $\rho_s(1)$ represents the probability of the solver s winning over the remaining ones. On the other hand, solvers with the largest probabilities $\rho_s(\theta)$ for large values of θ are the most robust (the ones that solve the largest number of problems in $\mathcal{P}$).

Different metrics can be considered to build performance profiles, like is the case of the amount of time spent by a solver to attain the stopping criterion. Despite the popularity of this metric, it is not enough to address the performance of a global optimization algorithm. Additionally to computational time, we will consider the minimum objective function value computed, the number of local minima found and the iteration where the global minimum is attained, as metrics to compare the different strategies and solvers. From these, priority will be given to the best objective function value found and to the number of points identified as local minimizers.

For the latter, larger values indicate better performance. Thus, when computing performance profiles for this metric, we set $t_s(p) = 1/\ t_s(p)$. Modifications are also required when analyzing functions that allow nonpositive values, where the ratio (4.1) could even not be defined. We will follow the procedure proposed in [23], where a translocation of all the function values is considered to guarantee strictly positive values, while maintaining the same relative distance among points, which allows the results to remain comparable.

It is important to notice that the hierarchical comparison of solvers based on performance profiles can not be done, unless only two solvers are plot (see [8]). Instead, one should only classify the best scenario, leaving the remaining cases unclassified.

4.1.2 Problem collection

As test set, we used the collection of problems reported in [3], selecting only second order differentiable functions. Table 4.1 details the problems, which in majority present identical bounds for all variables. The notation "-" stands for an unknown number of local or global minima.

Table 4.1: Problem set

Problem	Dimension	Lower Bound	Upper Bound	Number of known	
				Local minima	Global minima
aluffi_pentini	2	-10	10	2	1
bohachevsky	2	-50	50	-	1
branin_hoo	2	$[-5\ 0]^\top$	$[10\ 15]^\top$	3	3
cosine_mixture	2	-1	1	-	-
cosine_mixture	4	-1	1	-	-
dekkers_aarts	2	-20	20	3	2
exponencial	2	-1	1	-	1
exponencial	4	-1	1	-	1
fifteenn_local_minima	2	-10	10	15^2	1
fifteenn_local_minima	4	-10	10	15^4	1
fifteenn_local_minima	6	-10	10	15^6	1
fifteenn_local_minima	8	-10	10	15^8	1
fifteenn_local_minima	10	-10	10	15^{10}	1
goldsteen_price	2	-2	2	4	1
griewank	5	-600	600	-	1
griewank	10	-400	400	-	1
hosaki	2	$[0\ 0]^\top$	$[5\ 6]^\top$	2	1
kowalik	4	0	0.42	-	1
mccormick	2	$[-1.5\ -3]^\top$	$[4\ 3]^\top$	2	1
multi_gaussian	2	-2	2	5	1
neumaier2	4	0	4	-	1

Problem	Dimension	Lower Bound	Upper Bound	Number of known	
				Local minima	Global minima
neumaier3	10	-100	100	-	1
periodic	2	-10	10	50	1
poissonian	2	$[1\ 1]^\top$	$[21\ 8]^\top$	-	-
powell	4	-10	10	1	1
rastrigin	10	-5.12	5.12	-	1
rosenbrock	2	-5.12	5.12	1	1
shekel_45	4	0	10	5	1
shekel_47	4	0	10	7	1
shekel_410	4	0	10	10	1
shekel_foxholes	5	0	10	-	1
shekel_foxholes	10	0	10	-	1
shubert	2	-10	10	760	18
sinusoidal	10	0	180	-	1
sixhumpcamel	2	$[-3\ -2]^\top$	$[3\ 2]^\top$	6	2
sphere	3	-5.12	5.12	1	1
tenn_local_minima	2	-10	10	10^2	1
tenn_local_minima	4	-10	10	10^4	1
tenn_local_minima	6	-10	10	10^6	1
tenn_local_minima	8	-10	10	10^8	1
threehumpcamel	2	-5	5	3	1
transistor	9	-10	10	1	1
wood	4	-10	10	1	1

4.1.3 Solvers for comparison

Efficiency and efficacy of an algorithm can be stated when it is compared to state-of-art solvers for the same class of problems. Unfortunately, finding solvers for global optimization problems with second order differentiable functions was not an easy task. The algorithm introduced in [12] could be a promising resource for comparison. However, it is implemented in fortran90 and it is not easily convertible to MATLAB. Running the code in fortran is a challenging task for an inexperienced user, as mentioned by the authors when contacted about the availability of the computational implementation. For this reason we were forced to give up on the use of the aforementioned code.

The Global Optimization toolbox of MATLAB has two functions dedicated to global optimization, that ended up being our comparison solvers: *Globalsearch* and *Multistart* [22]. It is important to notice that the base license of MATLAB does not include these functions, requiring the purchase of the Global Optimization toolbox additionally to the Optimization toolbox, which is a disadvantage when compared with the proposed algorithm.

Globalsearch and *Multistart* use *fmincon* as the base local optimization solver. This last function is suited to find the minimum of a constrained nonlinear multivariable function using a gradient-based method. Nonlinearity can be presented in both the objective function and constraints. The default settings consider an interior-point method, but the user is allowed to switch to trust-region-reflective, sequential quadratic or active-set methods, according to the features of the problem to be solved.

Globalsearch function runs *fmincon* from a required starting point x_0. If this run converges, *Globalsearch* records the start point and the end point for an initial estimate

on the radius of a basin of attraction for x_0. The *Globalsearch* heuristic assumption is that basins of attraction are spherical.

The function uses the scatter search [22] algorithm to generate a set of potential starting points (1000 by default) within the problem bounds. If the problem is unbounded, artificial bounds are imposed. The objective function is then evaluated at a subset of these trial points, selected using a score function that sums the point's objective function value to a multiple of the sum of the constraints violations, meaning that for feasible points the algorithm just takes the objective function value as the score. By default, 200 out of the initial 1000 sampled points go through this process, with the remaining points being removed from the list of trial points. The point of this subset with the best score is selected to run *fmincon*.

The solutions to both runs of *fmincon* are then compared, and the smallest of these values is used as a threshold value, called *localSolverThreshold* within the code. The solutions obtained from *fmincon* in the first two runs, from x_0 and the best scored point out of the initial sample, are then used as centers for spheres that estimate basins of attraction, with radius corresponding to the distance to the initial points of each run. The steps of the algorithm are repeated for the remaining points of the subset of trial points until all points satisfying the following conditions have been explored, through running *fmincon*, or the maximum allowed time is attained:

- If point p is not in an existing basin, defined by the estimated spheres;
- If score(p)<*localSolverThreshold*;
- If point p is within bounds.

Within this last step, updates to internal variables are also performed to improve the estimation of existing attraction basins and include new attraction basins defined from convergence of trial points. The lowest value recorded for an explored basin within the allotted time is taken as the minimum value and is returned.

Multistart is a much simpler and straightforward method that focuses on running *fmincon* on any feasible point initially available. The user can provide a list of initial points to run the algorithm or the cardinality of a set of points to be randomly generated within the problem bounds. The local solver *fmincon* is run on each of these starting points, until stopping criteria are met, without any merging stages, saving each local run's result and continuing to the next point. Similarly to *Globalsearch*, *Multistart* continuously keeps track of the elapsed time, being the maximum run time and the exploitation of all starting points provided the two allowed criteria to stop the algorithm.

4.2 Initialization/relaunching step

As mentioned in Section 3.3, different strategies can be used for initialization and have been tested. Deterministic methods provided the best results, with the additional advantage of a deterministic behaviour. We only present the results obtained with the initialization through Halton and Sobol sequences, whenever consecutive unsuccessful iterations occur. In our experiments, we tested performing the initialization/relaunching step after a fixed number of consecutive unsuccessful iterations was obtained, ranging from two to five. In any case, the number of points considered as new initializations equaled the problem dimension.

The four metrics introduced in Section 4.1.1 were used to assess performance, prioritizing the minimum value obtained for the objective function, followed by the number of local minima identified.

We started the calibration of the algorithm by defining the number of consecutive unsuccessful iterations required to perform the initialization/relaunching step, when a deterministic method is used to generate new points. Despite the deterministic behavior of the algorithm, small variations can occur on the elapsed time. Thus, we ran our algorithm twenty times and considered the average time of these runs to report on the tables as well as to build the performance profiles.

When analyzing the two main metrics for the case where new points are generated with the Halton sequence, we realized that all the four variants reached the same minimum value for the objective function, reducing the corresponding performance profile to a straight line. For this reason, only the profile regarding the number of identified minima is presented in Figure 4.1.

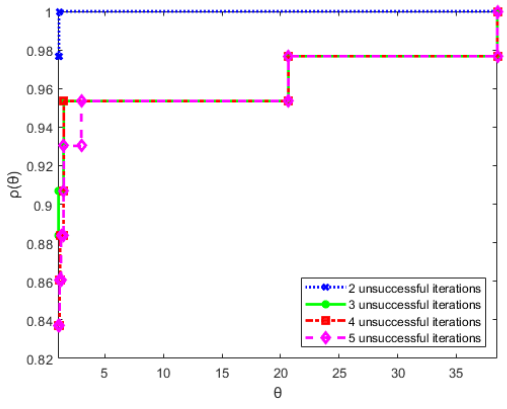

Figure 4.1: Performance profile for the number of identified minima when initializing new points through the Halton sequence, after a fixed number of consecutive unsuccessful iterations.

Analyzing this profile, we can conclude that performing the initialization/relaunching step after two consecutive unsuccessful iterations consistently returns the best results. In fact, this variant reaches $\rho(\theta) = 1$ for small values of θ, and therefore can be considered

the best strategy.

For the case of initialization through the Sobol sequence, Figure 4.2 reports the profiles for the two main metrics.

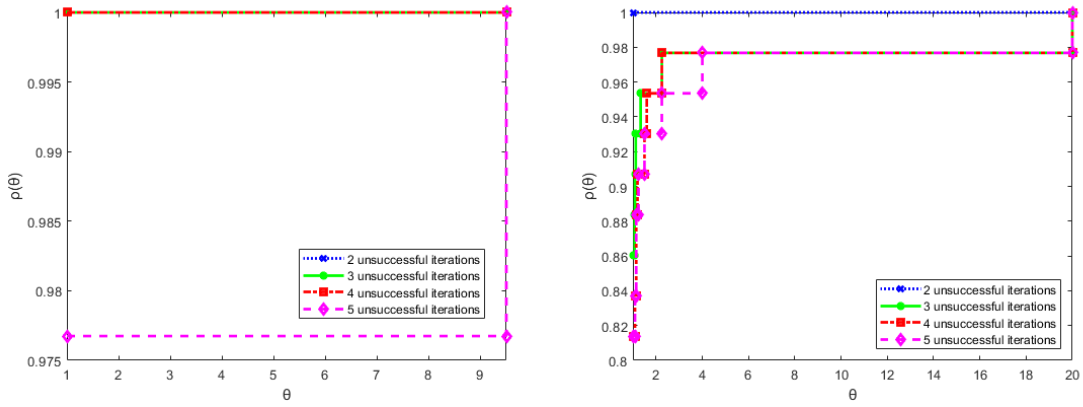

Figure 4.2: Performance profiles for the minimum objective function value (left) and the number of minima identified (right), when initializing new points through the Sobol sequence after a fixed number of consecutive unsuccessful iterations.

As we can observe from these profiles, the results for the four strategies tested are quite similar to the ones obtained using Halton sequence initializations, with a slight disadvantage of the variant that performs the initialization/relaunching step after five consecutive unsuccessful iterations, when the metric considered is the best objective function value. When comparing the number of minima, the best variant is again the strategy that considers two consecutive unsuccessful iterations before initializing new points.

Finally, we compared the best strategy for each initialization method, to decide which one to incorporate in our algorithm. The results on the two main metrics are reported in Figure 4.3.

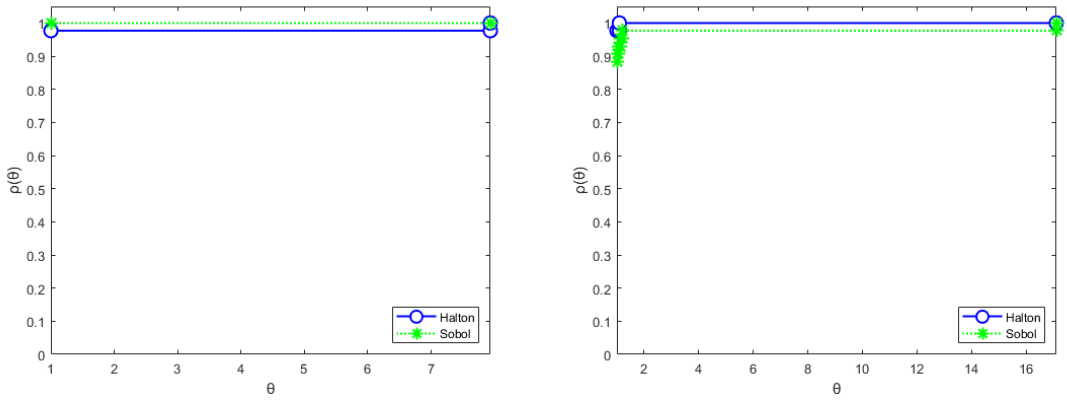

Figure 4.3: Performance profiles for the minimum objective function value (left) and the number of minima identified (right), when initializing through either Halton or Sobol sequences, after two consecutive unsuccessful iterations.

The results are extremely close. For the minimum value of the objective function, initializing through the Sobol sequence has an advantage, regardless a small one. On the other hand, when analyzing the profile for the number of local minima identified, the variant that uses the Sobol sequence is less efficient and improves slower with the increase of θ than the one that uses the Halton sequence strategy, that easily reaches the factor θ of the best solver, where it solves the problems with 100% probability. Considering the relevance of the objective function value found for global optimization, initialization through the Sobol sequence is the selected method.

Comparisons with *Globalsearch* and *Multistart* were performed to rank our method against these solvers. Results are reported in Table I.1 in the Annex. *Globalsearch* was initialized with the centroid of the feasible region, whereas *Multistart* initialized with a linestart and the centroid of the feasible region, similarly to our algorithm. Both solvers run with the default settings. For stopping, the elapsed time of our algorithm was supplied to the solvers, so that the search stops if this time is attained or when other internal stopping criterion is activated. Due to the randomness of *Globalsearch* and *Multistart*, we ran these two solvers twenty times and considered average values for all metrics to build the tables and the performance profiles. These settings will prevail along this chapter.

4.3 Improvements to the initialization/relaunching step

4.3.1 Based on active points

From the previous results, we notice that the algorithm appears to get stuck in local minima quite often, possibly meaning that the feasible region is not being completely explored. An illustrative example of this situation is problem *tenn_local_minima*, which corresponding results are presented in Table I.1, in the Annex. This problem has an incredible amount of local minima, varying with the problem dimension, as we can see in Figure 4.4 for the case of dimension two. Despite this, our algorithm rarely registers more minima than the dimension of the problem, as is the case for dimensions two and four, meaning that all starting points converged to nearby local minima without failure, satisfying second order optimality conditions and verifying the stopping criteria with a small number of new initializations. This means that there is an unexplored part of the feasible region that needs to be reached and analysed.

To overcome this issue, a possibility is to force the algorithm to initialize with a different criterion. The second order optimality conditions give us information about the active points on the list that have already converged to a local minimum. The selection of the center point for the trust-region step incorporates this information, discarding points that have already converged to a local minimum, to not waste time in needless calculations. However, when the list contains only a single active point not yet identified as a local minimizer, we can force the algorithm to initialize new points.

Besides the benefits of this new strategy in finding extra minima, when all starting

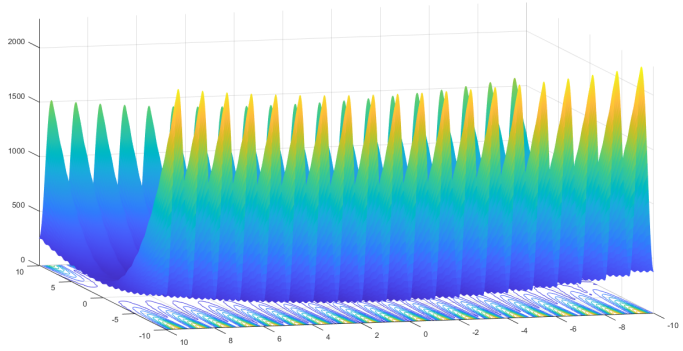

Figure 4.4: Plot of function *tenn_local_minima* for dimension two.

points tend to quickly converge to local minima, it also avoids relying only on the convergence to a single point, allowing the choice of potentially better new points in the feasible region. Therefore a new strategy is defined that considers to perform the initialization/relaunching step through the Sobol sequence when two consecutive unsuccessful iterations were performed or when the list of active points, not yet identified as local minimizers, is a singular set.

A comparison to the previous strategy was made in order to verify whether or not there was an improvement. The results are reported in Figure 4.5, for the two main metrics.

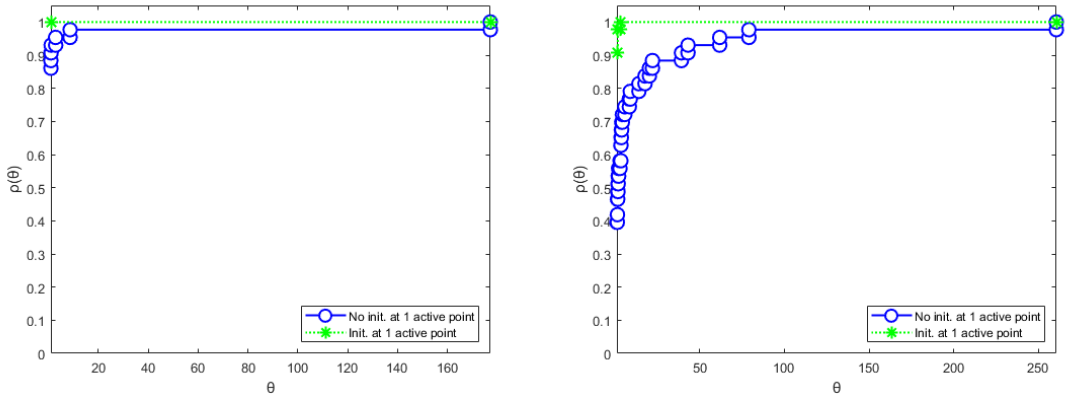

Figure 4.5: Performance profiles for the minimum objective function value (left) and the number of minima identified (right), when initializing through the Sobol sequence, after performing two consecutive unsuccessful iterations or combining this criterion with the existence of a single active point in the list, not yet identified as a local minimizer.

As stated in the above figures, the new strategy shows an improvement in the two metrics of higher importance, being better than the old variant in terms of the approximation obtained to the global minimum and achieving a remarkable improvement on the number of local minima identified. Detailed results for this strategy and a comparison to *Globalsearch* and *Multistart* can be found in Table I.2 of Annex I.

4.3.2 Balancing objective function value and spread

In global optimization there is always a compromise between evolving good quality solutions, already found, and exploring new parts of the feasible region. In the current version of the algorithm, the criterion for selecting the model center at a trust-region step is solely based on the minimum value for the objective function, in a greedy approach.

To include some exploration of the feasible region, we decided to modify the criteria for selecting the model center at a trust-region iteration, considering points far from the model center selected at the previous iteration, but that additionally present reasonable low objective function values.

With this goal, at the beginning of each trust-region step, we introduce the computation of a ratio, for every active point y in the list L_k, not yet identified as a local minimizer:

$$\text{ratio}(y) = \frac{\max\limits_{x\in\mathbb{X}}(f_x) - f_y}{\max\limits_{x\in\mathbb{X}}(f_x) - \min\limits_{x\in L_k}(f_x)} + \frac{\|y - x_k\|}{\|ub - lb\|},$$

where $\mathbb{X}$ represents the set of all evaluated points, L_k states for the list at the previous iteration, x_k represents the model center at the previous iteration, and ub and lb represent respectively, the upper and lower bounds of the problem. This ratio gives higher values to distant points from the previous model center with good objective function values.

However, selecting as model center the point with the highest of these ratios did not improve the results, as can be seen in Figure 4.6.

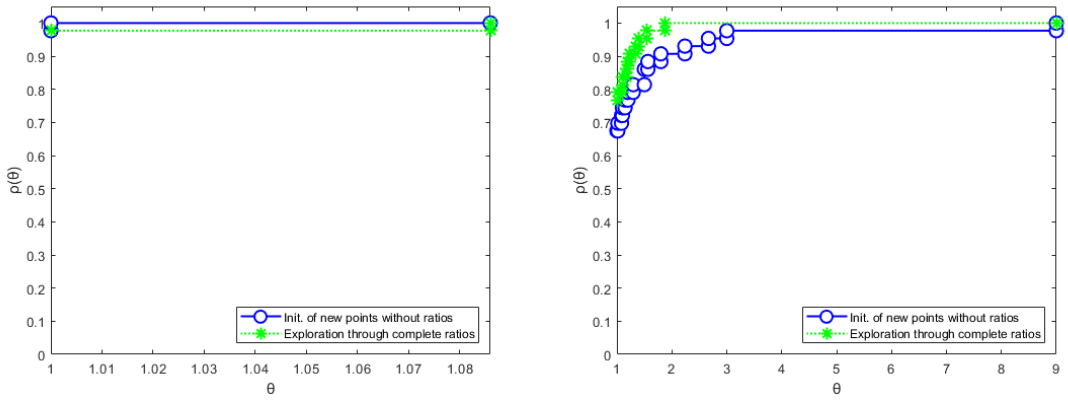

Figure 4.6: Performance profiles for the minimum objective function value (left) and the number of minima identified (right), when initializing through the Sobol sequence, with or without the new exploration method using ratios (two terms).

A second variant considers only the second term of the ratios, focusing on exploring distant points with no regard for function values. This proved to be a better strategy than the original version of the ratios, despite not being better than the variant defined in Section 4.3.1, with slightly worst results in terms of best objective function value found (see Figure 4.7 and Table I.3 in the Annex).

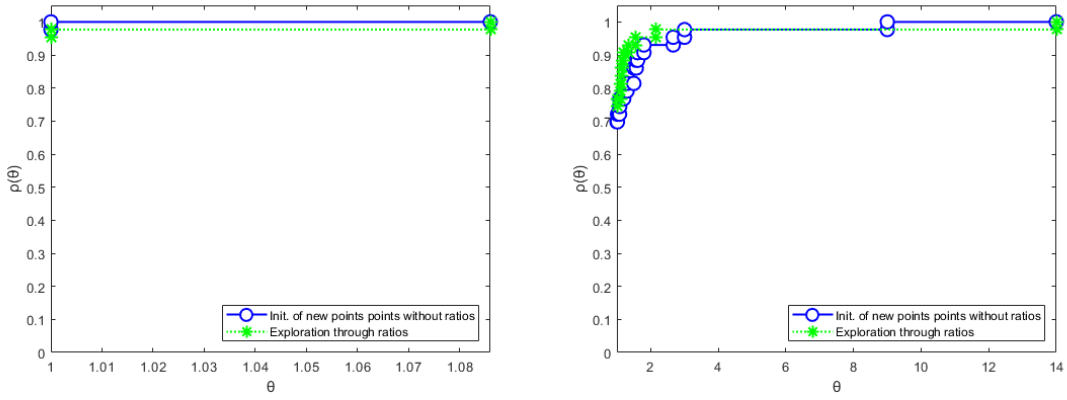

Figure 4.7: Performance profiles for the minimum objective function value (left) and the number of minima identified (right), when initializing through the Sobol sequence, with or without the new exploration method using ratios (one term).

From these two profiles, we can observe very close results, with the previous variant having an advantage in the obtained minimum objective function value. Despite this, the profile for the number of local minima identified has a more prominent advantage for low factors. In this case, the choice for the best variant was very influenced by the preference for the approximation to the global minimum, in opposition to the number of local minima found. Therefore, the strategy presented in Section 4.3.1, without the exploration through ratios, continues to be the best so far.

4.4 Conservative approach to merging

For the current best version, the algorithm seems to excessively perform merging iterations. This is the case of problems *transistor* and *fifteenn_local_minima*. The reason for this could be directly related to the radius defined for comparison of a point x to another point y in the list, denoted by the function $m(r_x, r_y)$ in Algorithm 3.1. In fact, all of the previous experiments considered this radius as the maximum of the two radius for points x and y. However, a more conservative method for merging can be applied, by considering the smallest radius of the two points when comparison is done, i.e., $m(r_x, r_y) = \min(r_x, r_y)$, for each $y \in L$.

In Figure 4.8, we compare the conservative merging method with the previous best algorithmic variant by plotting the profiles for the obtained minimum value and the number of local minima found.

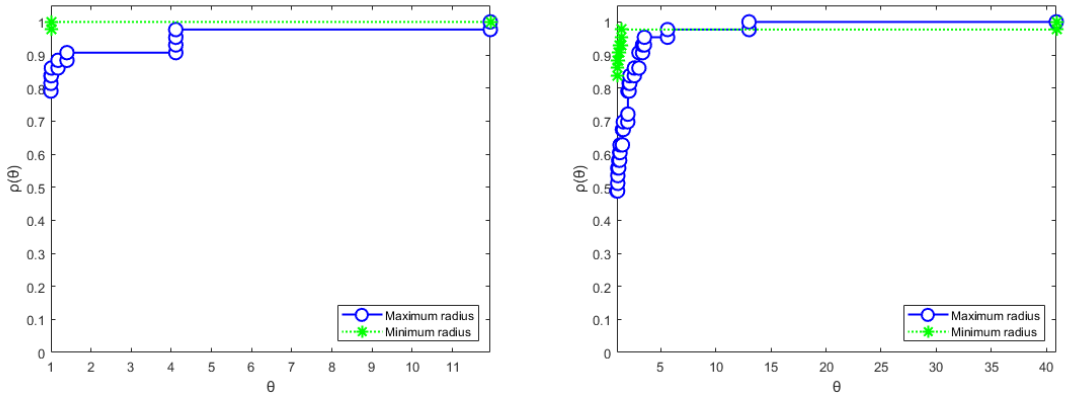

Figure 4.8: Performance profiles for the minimum objective function value (left) and the number of minima identified (right), considering the maximum and the minimum radius for point comparison.

We can conclude that this strategy clearly improves the performance of the algorithm in both metrics, despite suffering a slight loss of quality for high values of θ for the number of local minima identified.

After all the algorithmic variants tested, we are now at a point where we can define that the best variant of our iterative trust-region algorithm coupled with a smart *multi-start* strategy is the one that employs initializations through the Sobol sequence, upon two consecutive unsuccessful iterations of the trust-region method or if a single active point (that has not yet converged to a local minimum) remains in the list, considering a conservative strategy when comparing points on the merging stage.

4.5 Comparison with Globalsearch and Multistart

With the best variant of our algorithm defined, we compared its performance with the *Globalsearch* and *Multistart* algorithms. As mentioned before, the quality hierarchy of the solvers cannot be established when comparing more than two solvers on a profile. For this reason, we compared our algorithm with the MATLAB solvers one at each time.

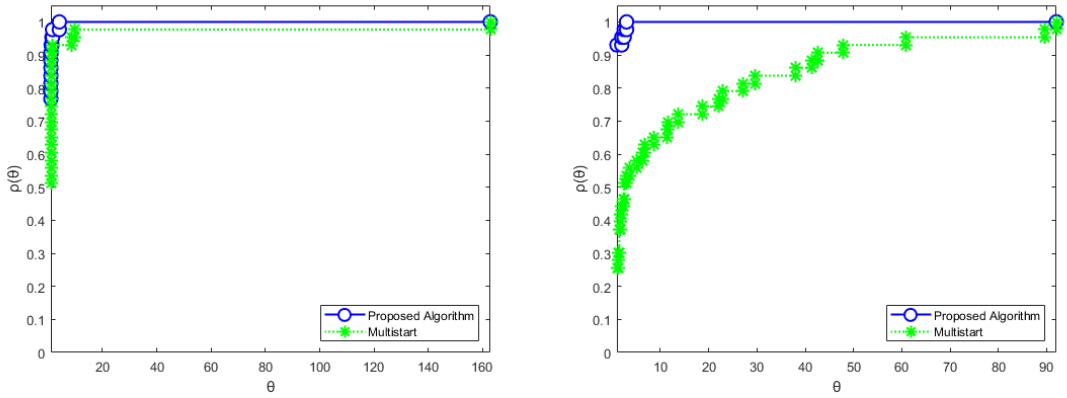

Figure 4.9: Performance profiles for the minimum objective function value (left) and the number of minima identified (right), when comparing our algorithm to *Multistart* function, with default settings and linestart initialization plus the centroid.

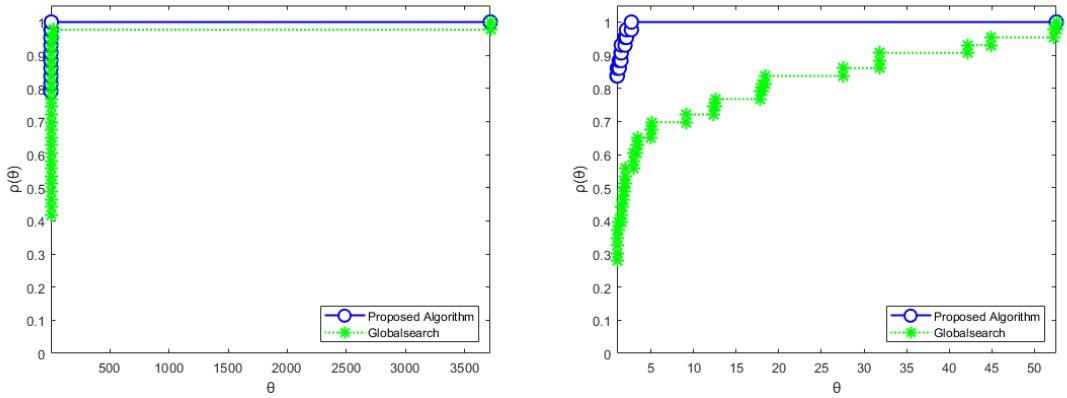

Figure 4.10: Performance profiles for the minimum objective function value (left) and the number of minima identified (right), when comparing our algorithm to *Globalsearch*, with default settings and initialization with the centroid of the feasible region.

The complete results can be found on Annex I, Table I.4. Figure 4.9 shows that our algorithm is slightly better in the estimation of the global minimum in comparison to the *Multistart* function and outperforms *Multistart* in the number of local minima found.

The comparison results to *Globalsearch* function are similar, allowing to conclude that this function is as efficient as our algorithm to reach the global minimum, but loses on the number of local minima detected. This can be justified due to our algorithm being able to identify and return local minima in the borders of the feasible region, where optimality conditions may not be met. This is evident, for example, in *hosaki* and *mccormick* problems (see Figure 4.11), where local minima corresponding to points $(0,0)^\top$ and $(-1.5,-3)^\top$, respectively, are correctly returned.

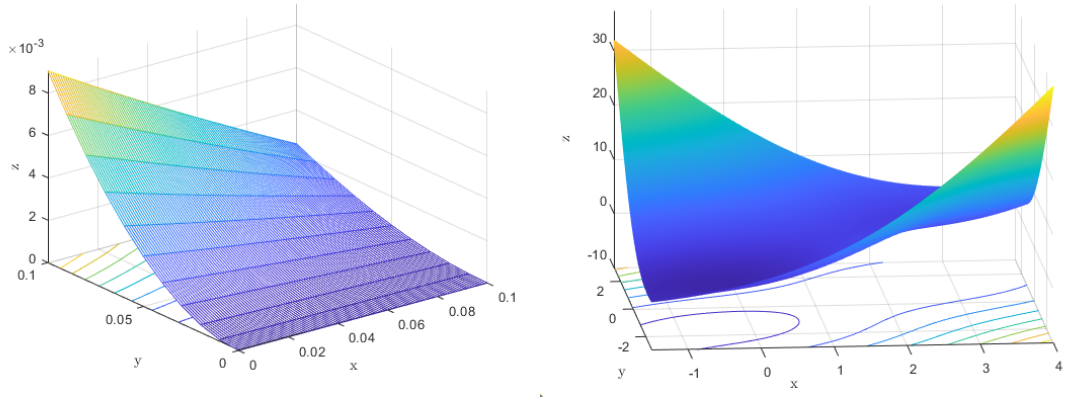

Figure 4.11: Plots of functions *hosaki* (left) and *mccormick* (right).

There is also the case of the problem *branin_hoo* in which the literature does not report all local minima correctly. Our algorithm returns one more local minimum that is not a border point and also verifies second order optimality conditions. This guarantees that the new reported minimum is indeed a local minimum that has not been previously accounted for in the literature.

Finally, there is the case of problem *powell* in which two local minima are returned by our algorithm, while the literature only reports one (which is correct since the function is convex). The second point returned is the result of convergence to the same minimum, stopped due to reaching a stage where the gradient is small enough and its Hessian recognized as positive definite. If smaller tolerances were considered for the classification of points as local minimizers, the point would evolve to the true minimum.

4.5.1 Remarks on previous comparisons

Comparing *Globalsearch* and *Multistart* with our algorithm, using their default settings is not completely fair due to the different methods of optimization considered. Therefore, new tests were performed with *Globalsearch* function, using the *trust-region-reflective* solver when *fmincon* is called. As the gradient of the objective function should be supplied by the user when *trust-region-reflective* method is considered, we resource to DERIVESTsuite [4] to estimate the gradient of each point, similarly to its use in the Taylor's expansion of our algorithm. The results of this experience can be seen in Table I.5 in the Annex. In fact, when we stop *Globalsearch* based on the computational times spent by our algorithm and force it to employ a trust-region-reflective method, having to estimate the gradient, the performance of this function decreases a lot.

Due to these results, we investigated the profile of MATLAB internal functions of our algorithm, in terms of percentage of time spent to run each function, during the optimization process. As expected, calls to the estimation of the derivatives, gradient and Hessian are time consuming, along with the trust-region step and the function that compares each new point to the list of points, denoted by Classification in Figure 4.12.

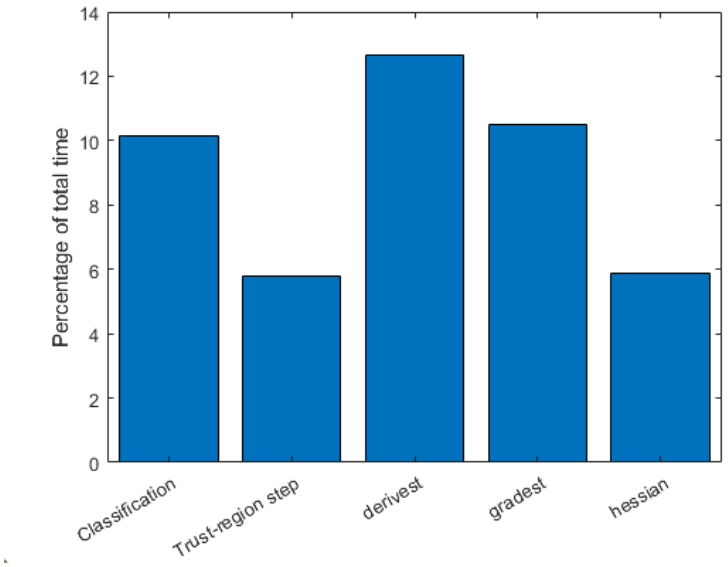

Figure 4.12: Barplot of the average percentage of total time spent on the five functions that consume the higher percentage of total time during the optimization process.

In a real application problem, the time spent in derivatives estimation can be dramatically reduced if functions for computing the true derivatives are provided (or by using a more efficient procedure for their numerical estimation).

Along with these results, we also ran *Multistart* providing as initializations all the points generated by our algorithm at the initialization/relaunching step. These points are generated according to our initialization criteria and provide a very good baseline that improves the results obtained with *Multistart*, but also make the run times very similar. This is indicative of the tradeoff between thorough exploration of the feasible domain and fast approximation to the global optimum. These results can be seen in Table I.6, in the Annex.

Chapter 5

Conclusions and Open Questions

In this thesis we proposed an algorithm for global optimization problems with continuous variables and second order differentiable objective functions, when subject to bound constraints. The algorithm is based on an iterative trust-region method coupled with a clever multistart strategy that prevents to explore all the generated initializations until the end. Points that are sufficiently close to each other are compared and only the best ones are retained.

It is well-known that trust-region methods locally converge. Since during the merging of points the trust-region radius of the kept point never exceeds the one resulting from the trust-region step, similar convergence results hold for the proposed method.

A similar approach has already been successful in the global solution of problems where derivatives of the function defining the problem are not available [3]. Numerical results, considering the comparison to *Globalsearch* and *Multistart* MATLAB functions, proved that the proposed algorithm is also competitive for derivative-based optimization, taking a small advantage in the detection of the global minimum and outperforming both solvers in the number of local minima found.

Although for the run time this algorithm is worse than the Matlab methods, mainly due to the time spent to estimate derivatives to build the trust-region quadratic model, it is important to notice that its implementation does not require the Global Optimization Toolbox, to which *Globalsearch* and *Multistart* belong. Additionally, the proposed algorithm is deterministic, always guaranteeing the same quality for the computed solution.

A research work is never finished and there are always new questions that deserve to be addressed. A possible avenue to improve the performance of the algorithm is the use of more efficient routines for the estimation of derivatives. The use of the solver to address derivative-based optimization problems related to the minimization of radial basis functions that appear in derivative-free optimization codes is also planned.

Bibliography

[1] B. Addis and S. Leyffer. "A trust-region algorithm for global optimization." In: *Comput. Optim. Appl.* 35 (2006), pp. 287–304.

[2] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. *Trust-region Methods*. Society for Industrial and Applied Mathematics, 2000. ISBN: 978-0-89871-460-9.

[3] A. L. Custódio and J. F. A. Madeira. "GLODS: Global and Local Optimization using Direct Search." In: *J. Global Optim.* 62 (2015), pp. 1–28.

[4] J. D'Errico. *Adaptive Robust Numerical Differentiation*. MATLAB Central File Exchange, 2020. URL: https://www.mathworks.com/matlabcentral/fileexchange/13490-adaptive-robust-numerical-differentiation.

[5] J. Dick and F. Pillichshammer. *Digital Nets and Sequences. Discrepancy Theory and Quasi-Monte Carlo Integration*. Cambridge University Press, Cambridge, 2010. ISBN: 978-0-521-19159-3.

[6] E. D. Dolan and J. J. Moré. "Benchmarking optimization software with performance profiles." In: *Math. Program.* 91 (2002), pp. 201–213.

[7] M. Drmota and R. F. Tichy. *Sequences, Discrepancies and Applications, Lecture Notes in Math.* Springer, 1997. ISBN: 3-540-62606-9.

[8] N. Gould and J. Scott. "A note on performance profiles for benchmarking software." In: *ACM Transactions on Mathematical Software* 43, Article 15 (2016).

[9] I. Griva, S. G. Nash, and A. Sofer. *Linear and Nonlinear Optimization*. Second Edition. Society for Industrial and Applied Mathematics, 2009. ISBN: 978-0-898716-61-0.

[10] J. Halton. "Algorithm 247: Radical-inverse quasi-random point sequence." In: *Communications of the ACM* 7 (1964), pp. 701–702.

[11] R. Horst, P. M. Pardalos, and N. V. Thoai. *Introduction to Global Optimization*. Second Edition. Springer, 2000. ISBN: 978-0-7923-6574-7.

[12] F. Lampariello and G. Liuzzi. "A filling function method for unconstrained global optimization." In: *Comput. Optim. Appl.* 61 (2015), pp. 713–729.

[13] R. H. Leary. "Global optimization on funneling landscapes." In: *J. Global Optim.* 18 (2000), pp. 367–383.

[14] M. Locatelli and F. Schoen. "Global optimization based on local searches." In: *Ann. Oper. Res.* 240 (2016), pp. 251–270.

[15] J. M. Martínez and M. Raydan. "Separable cubic modeling and a trust-region strategy for unconstrained minimization with impact in global optimization." In: *J. Glob. Optim.* 63(2) (2015), 319–342.

[16] *MATLAB version 9.8.0.1396136 (R2020a) Update 3*. Natick, Massachusetts: The Mathworks, Inc., 2020.

[17] M. D. McKay, R. J. Beckman, and W. J. Conover. "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code." In: *Technometrics* 21 (1979), 239–245.

[18] J. J. Moré and D. C. Sorensen. "Computing a trust region step." In: *SIAM J. Sci. and Stat. Comput.* 4 (1983), 553–572.

[19] H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*. Society for Industrial and Applied Mathematics, 1992. ISBN: 0-89871-295-5.

[20] J. Nocedal and S. J. Wright. *Numerical Optimization*. Second Edition. Springer, 2006. ISBN: 978-0-387-30303-1.

[21] I. M. Sobol. "On the distribution of points in a cube and the approximate evaluation of integrals." In: *U.S.S.R Comput. Maths. Math. Phys* 7 (1967), 86–112.

[22] Z. Ugray, L. Lasdon, J. Plummer, F. Glover, J. Kelly, and R. Martí. "Scatter search and local NLP solvers: A multistart framework for global optimization." In: *INFORMS J. Comput.* 19 (2007), pp. 328–340.

[23] A. I. F. Vaz and L. N. Vicente. "A particle swarm pattern search method for bound constrained global optimization." In: *J. Glob. Optim.* 39 (2007), pp. 197–219.

[24] D. H. Wolpert and W. G. Macready. "No free lunch theorems for optimization." In: *IEEE Trans. on Evol. Comput.* 1 (1997), pp. 67–82.

ANNEX I

Annex 1 Tables

In this Annex we present the detailed results obtained from the run tests performed on our algorithm, as detailed in Chapter 4, with the corresponding results obtained from *Globalsearch* and *Multistart* algorithms. The results presented are averages of twenty runs. *Globalsearch* and *Multistart* were run in each case with a maximum computational time equal to the average run times from the accompanying version of our algorithm. Default settings are considered, when comparing other solvers against the final variant of our algorithm. Exceptions occur in the last two tables, where a trust-region-reflective solver is used in conjunction with *Globalsearch* (Table I.5), and *Multistart* is ran with all the points considered as initializations by our algorithm (Table I.6) (and not only the ones from the first iteration).

In these tables, we use the following header notations: *Dim* referring to Dimension, *Min* referring to the obtained minimum objective function value, *Min_count* referring to the number of local minima identified, *It_min* referring to the iteration where the algorithm reached the minimum objective value, and *Time* referring to the run times reported in seconds.

All the runs were executed in a PC with Intel® Core™ i7-8550U CPU @ 1.80GHz 1.99GHz processor and 8GB of RAM.

Table I.1: Comparison of the algorithm considering Sobol sequences for initialization, after two consecutive unsuccessful iterations being performed, with Globalsearch and Multistart functions.

Problem	Dim	Known minimum	Nº of known local minima	Sobol initialization after 2 unsuccessful iterations				Globalsearch			Multistart		
				Min	Min_count	It_min	Time	Min	Min_count	Time	Min	Min_count	Time
aluffi_pentini	2	-0,352386037	2	-0,352386074	1	7	0,045	-0,352386074	1	0,049	-0,352386074	2	0,044
bohachevsky	2	0	-	0	1	1	0,082	0	1,9	0,092	0	3	0,036
branin_hoo	2	0,397887358	3	0,397887358	3	16	0,07	0,397887358	2	0,077	0,397887358	2	0,036
cosine_mixture	2	-2,2	-	-2,2	12	12	0,345	-2,199999292	9,1	0,237	-2,199995447	3	0,03
cosine_mixture	4	-4,4	-	-4,4	36	12	1,622	-4,399999712	18,35	0,402	-4,399990894	5	0,053
dekkers_aarts	2	-24777	3	-24776,51834	3	22	0,072	-7432,955503	1,3	0,078	-24776,51834	3	0,058
exponencial	2	-1	-	-1	1	1	0,038	-1	1	0,057	-1	1	0,024
exponencial	4	-1	-	-1	1	1	0,053	-1	1	0,068	-1	1	0,036
fifteenn_local_minima	2	0	225	0,198772614	3	2	0,118	0,03332555	3,2	0,128	0,110253015	3	0,057
fifteenn_local_minima	4	0	50625	0,398546993	5	2	0,223	0,223898855	5,65	0,219	0,398546993	5	0,111
fifteenn_local_minima	6	0	15^6	0,598321572	4	2	0,851	0,277048141	10,3	0,418	2,32E-12	7	0,262
fifteenn_local_minima	8	0	15^8	0,798096151	5	2	1,963	0,346997694	11,1	0,622	0,010987366	9	0,388
fifteenn_local_minima	10	0	15^10	0,010987366	8	14	1,645	0,318044539	10,85	0,774	0,010987366	11	0,553
goldstein_price	2	3	4	3	3	29	0,138	3	2	0,141	3	3	0,058
griewank	5	0	-	0	9	1	5,987	0	10,4	0,225	0	5	0,091
griewank	10	0	-	0	60	1	40,737	0	5,55	0,265	0	9	0,379
hosaki	2	-2,345811576	2	-2,345811576	3	5	0,209	-2,345811576	1,8	0,167	-2,345811576	2	0,034
kowalik	4	0,00030748	-	0,000307486	1	5	0,176	0,000307486	1,4	0,187	0,000307486	2	0,088
mccormick	2	-1,913222887	2	-1,913222955	3	12	0,177	-1,913222955	1	0,138	-1,913222955	2	0,033
multi_gaussian	2	-1,296954013	5	-1,296954046	2	3	0,051	-1,296954046	1	0,077	-1,296954046	2	0,041
neumaier2	4	0	-	7,50E-11	1	337	2,73	1,34E-07	5,7	1,677	1,50E-07	5	1,083
neumaier3	10	-210	-	-210	1	1	0,283	-210	1	0,15	-210	1	0,266
periodic	2	0,9	50	0,9	3	1	0,034	0,9	1	0,063	0,9	3	0,027
poissonian	2	71,25171938	-	71,33773046	3	17	0,217	71,25171938	2,6	0,221	71,25171938	2	0,048
powell	4	0	1	0	1	1	0,102	0	1,3	0,108	8,07E-14	3,15	0,113
rastrigin	10	0	-	0	3	1	0,369	0	5,95	0,378	0	7	0,237
rosenbrock	2	0	1	1,14E-17	1	17	0,121	2,05E-11	1	0,126	2,03E-11	1	0,077
shekel_45	4	-10,15319585	5	-10,15319968	4	12	0,214	-10,15319968	3,75	0,219	-10,15319968	4	0,088
shekel_47	4	-10,40281884	7	-10,40294057	4	7	0,221	-10,40294057	3,85	0,231	-10,40294057	4	0,122

Problem	Dim	Known minimum	Nº of known local minima	Sobol initialization after 2 unsuccessful iterations				Globalsearch			Multistart		
				Min	Min_count	It_min	Time	Min	Min_count	Time	Min	Min_count	Time
shekel_410	4	-10,53628373	10	-10,53640982	4	7	0,223	-10,53640982	4,55	0,233	-10,53640982	4	0,121
shekel_foxholes	5	-10,40560576	-	-2,700967461	3	6	0,25	-4,092211943	5,2	0,262	-2,700967461	4	0,104
shekel_foxholes	10	-9,861357063	-	-10,20879279	8	63	3,897	-1,800940973	2,1	1,226	-1,479761058	1	0,41
shubert	2	-186,7309	760	-10,50344304	3	23	0,219	-186,7309088	8,85	0,221	-24,92043558	3	0,039
sinusoidal	10	-3,5	-	-3,5	144	1	17,432	-3,5	6,05	0,425	-3,5	5	0,114
sixhumpcamel	2	-1,031628453	6	-1,031628453	3	3	0,075	-0,909203759	2,65	0,084	-1,031628453	3	0,031
sphere	3	0	1	0	1	1	0,05	0	1	0,066	0	1	0,032
tenn_local_minima	2	2,36E-31	100	3,109750807	3	3	0,045	9,65E-15	1	0,054	2,332143324	2,7	0,048
tenn_local_minima	4	1,18E-31	10^4	3,109584208	5	3	0,203	2,44E-14	3,85	0,185	2,44E-14	5	0,113
tenn_local_minima	6	7,85E-32	10^6	3,109528628	5	3	0,187	3,62E-15	2,6	0,189	3,62E-15	3	0,122
tenn_local_minima	8	5,89E-32	10^8	3,109500838	7	3	0,545	0,466484019	7,05	0,405	0,388758826	9	0,336
threehumpcamel	2	0	3	0	1	1	0,037	0	1	0,059	0	1	0,035
transistor	9	0	1	12,85215906	6	2807	39,453	6,464125064	7,6	1,87	0,002222034	6	1,328
wood	4	0	1	8,68E-18	1	12	0,127	5,68E-13	1	0,141	4,88E-14	1	0,144

Table I.2: Comparison of the algorithm considering Sobol sequences for initialization, after two consecutive unsuccessful iterations being performed, or when a single active point (not considered as local minimizer) remains in the list, with Globalsearch and Multistart functions.

Problem	Dim	Known minimum	Nº of known local minima	Sobol init. after 2 unsuccessful iterations and 1 act. point				Globalsearch			Multistart		
				Min	Min_count	It_min	Time	Min	Min_count	Time	Min	Min_count	Time
aluffi_pentini	2	-0,352386037	2	-0,352386074	1	7	0,087	-0,352386074	1,95	0,098	-0,352386074	2	0,036
bohachevsky	2	0	-	0	1	1	0,413	0	4,6	0,144	0	3	0,034
branin_hoo	2	0,397887358	3	0,397887358	4	24	0,145	0,397887358	3,7	0,148	0,397887358	2	0,034
cosine_mixture	2	-2,2	-	-2,2	12	104	0,589	-2,199999543	9,85	0,257	-2,199995447	3	0,028
cosine_mixture	4	-4,4	-	-4,4	153	1785	12,457	-4,399999632	18,2	0,387	-4,399990894	5	0,048
dekkers_aarts	2	-24777	3	-24776,51834	1	5000	14,119	-23537,69243	2,9	0,156	-24776,51834	3	0,055
exponencial	2	-1	-	-1	1	1	0,308	-1	1	0,286	-1	1	0,024
exponencial	4	-1	-	-1	1	1	5,206	-1	1	0,235	-1	1	0,036
fifteenn_local_minima	2	0	225	0,198772614	10	2	0,701	0,012821903	8,25	0,205	0,110253015	3	0,054
fifteenn_local_minima	4	0	50625	0,398546993	110	2	13,764	0,109145306	8,35	0,269	0,398546993	5	0,108
fifteenn_local_minima	6	0	15^6	0,598321572	247	2	46,782	0,293229082	9,75	0,415	2,32E-12	7	0,267
fifteenn_local_minima	8	0	15^8	0,798096151	215	2	64,258	0,393674967	9,55	0,54	0,010987366	9	0,39

Problem	Dim	Known minimum	Nº of known local minima	Sobol init. after 2 unsuccessful iterations and 1 act. point				Globalsearch			Multistart		
				Min	Min_count	It_min	Time	Min	Min_count	Time	Min	Min_count	Time
fifteenn_local_minima	10	0	15^10	0,010987366	141	13	82,691	0,253970442	13,3	0,844	0,010987366	11	0,553
goldstein_price	2	3	4	3	4	29	0,218	3	2,1	0,159	3	3	0,057
griewank	5	0	-	0	80	1	40,41	0	11,05	0,237	0	5	0,093
griewank	10	0	-	0	230	1	93,702	0	5,95	0,275	0	9	0,375
hosaki	2	-2,345811576	2	-2,345811576	2	5	0,364	-2,345811576	1,75	0,15	-2,345811576	2	0,034
kowalik	4	0,00030748	-	0,000307486	1	5	1,381	0,000307486	2,35	0,798	0,000307486	2	0,088
mccormick	2	-1,913222887	2	-1,913222955	2	12	0,267	-1,913222955	1	0,148	-1,913222955	2	0,034
multi_gaussian	2	-1,296954013	5	-1,296954046	3	3	0,126	-1,296954046	4	0,14	-1,296954046	2	0,041
neumaier2	4	0	-	4,06E-11	14	585	33,632	8,88E-08	6,5	2,797	1,50E-07	5	1,08
neumaier3	10	-210	-	-210	1	1	440,278	-210	1	0,149	-210	1	0,262
periodic	2	0,9	50	0,9	2	1	14,225	0,9	22,15	0,414	0,9	3	0,026
poissonian	2	71,25171938	-	71,25171938	3	84	5,494	71,25171938	3	0,353	71,25171938	2	0,046
powell	4	0	1	0	1	1	6,74	7,21E-15	3,95	0,204	0	5	0,129
rastrigin	10	0	-	0	782	1	77,679	0	10,85	0,627	0	7	0,236
rosenbrock	2	0	1	1,14E-17	1	16	0,228	2,05E-11	1	0,199	2,03E-11	1	0,074
shekel_45	4	-10,15319585	5	-10,15319968	5	12	2,301	-10,15319968	4,15	0,431	-10,15319968	4	0,085
shekel_47	4	-10,40281884	7	-10,40294057	7	7	2,474	-10,40294057	4,5	0,431	-10,40294057	4	0,119
shekel_410	4	-10,53628373	10	-10,53640982	8	7	2,568	-10,53640982	5,15	0,365	-10,53640982	4	0,117
shekel_foxholes	5	-10,40560576	-	-10,40561724	25	152	8,383	-4,757528968	8,55	0,619	-2,700967461	4	0,101
shekel_foxholes	10	-9,861357063	-	-10,20879279	29	32	480,567	-1,979334577	2,4	1,244	-1,479761058	1	0,405
shubert	2	-186,7309	760	-186,7309088	10	213	5,917	-186,7309088	10,7	0,264	-24,92043558	3	0,039
sinusoidal	10	-3,5	-	-3,5	491	1	139,035	-3,5	5,95	0,423	-3,5	5	0,113
sixhumpcamel	2	-1,031628453	6	-1,031628453	5	23	0,366	-1,031628453	4,8	0,17	-1,031628453	3	0,031
sphere	3	0	1	0	1	1	3,052	0	1	0,131	0	1	0,031
tenn_local_minima	2	2,36E-31	100	3,109750807	17	3	0,467	9,65E-15	3,25	0,153	9,65E-15	3	0,049
tenn_local_minima	4	1,18E-31	10^4	3,109584208	101	3	6,187	2,44E-14	5,85	0,231	2,44E-14	5	0,111
tenn_local_minima	6	7,85E-32	10^6	3,109528628	395	3	46,39	3,62E-15	5,75	0,293	3,62E-15	3	0,12
tenn_local_minima	8	5,89E-32	10^8	3,109500838	274	3	62,503	0,563668949	7,65	0,42	0,388758826	9	0,335
threehumpcamel	2	0	3	0	1	1	0,311	0	2,85	0,16	0	1	0,033
transistor	9	0	1	10,94380184	17	4005	66,964	13,54081815	8,6	1,687	0,002222034	6	1,315
wood	4	0	1	8,68E-18	1	11	1,205	5,68E-13	1	0,256	4,88E-14	1	0,177

Table I.3: Comparison of the algorithm balancing objective function value and spread with Globalsearch and Multistart functions.

Problem	Dim	Known minimum	Nº of known local minima	Balancing strategy using Sobol				Globalsearch			Multistart		
				Min	Min_count	It_min	Time	Min	Min_count	Time	Min	Min_count	Time
aluffi_pentini	2	-0,352386037	2	-0,352386074	1	7	0,073	-0,33476677	1,3	0,133	-0,352386074	2	0,045
bohachevsky	2	0	-	0	1	1	0,177	0	3,05	0,163	0	3	0,036
branin_hoo	2	0,397887358	3	0,397887358	4	24	0,113	0,418006002	3,45	0,13	0,397887358	2	0,036
cosine_mixture	2	-2,2	-	-2,2	13	112	0,674	-2,199999311	10,1	0,284	-2,199995447	3	0,03
cosine_mixture	4	-4,4	-	-4,4	140	903	7,286	-4,399999664	16,9	0,4	-4,399990894	5	0,052
dekkers_aarts	2	-24777	3	-24776,51834	3	26	0,129	-21060,04059	2,65	0,133	-24776,51834	3	0,058
exponencial	2	-1	-	-1	1	1	0,061	-1	1	0,067	-1	1	0,025
exponencial	4	-1	-	-1	1	1	0,453	-1	1	0,242	-1	1	0,036
fifteenn_local_minima	2	0	225	0,198772614	10	2	0,881	0,017490112	8,2	0,203	0,110253015	3	0,056
fifteenn_local_minima	4	0	50625	0,398546993	99	2	12,968	0,184366157	8,4	0,272	0,398546993	5	0,112
fifteenn_local_minima	6	0	15^6	0,598321572	216	2	46,01	0,222854793	10,15	0,444	2,32E-12	7	0,264
fifteenn_local_minima	8	0	15^8	0,798096151	179	2	63,362	0,215145397	9,3	0,552	0,010987366	9	0,392
fifteenn_local_minima	10	0	15^10	0,010987366	152	14	81,151	0,319885685	9,75	0,739	0,010987366	11	0,554
goldstein_price	2	3	4	3	4	29	0,192	3	2,05	0,155	3	3	0,059
griewank	5	0	-	0	129	1	37,41	0	10,35	0,225	0	5	0,092
griewank	10	0	-	0	107	1	82,542	0	5,1	0,24	0	9	0,37
hosaki	2	-2,345811576	2	-2,345811576	3	5	0,29	-2,345811576	1,9	0,151	-2,345811576	2	0,035
kowalik	4	0,00030748	-	0,000307486	1	5	0,321	0,000307486	1,8	0,335	0,000307486	2	0,089
mccormick	2	-1,913222887	2	-1,913222955	3	12	0,313	-1,913222955	1	0,142	-1,913222955	2	0,035
multi_gaussian	2	-1,296954013	5	-1,296954046	3	3	0,101	-1,296954046	3	0,112	-1,296954046	2	0,043
neumaier2	4	0	-	7,50E-11	1	337	3,004	1,16E-07	5,95	2,166	1,50E-07	5	1,064
neumaier3	10	-210	-	-210	1	1	0,311	-210	1	0,144	-210	1	0,259
periodic	2	0,9	50	0,9	18	1	0,319	0,9	19,9	0,329	0,9	3	0,027
poissonian	2	71,25171938	-	71,33773046	8	17	0,471	71,25171938	3	0,361	71,25171938	2	0,047
powell	4	0	1	0	1	1	0,295	1,75E-15	4,05	0,19	0	5	0,126
rastrigin	10	0	-	0	795	1	81,078	0	13	0,696	0	7	0,232
rosenbrock	2	0	1	1,14E-17	1	17	0,221	2,05E-11	1	0,203	2,03E-11	1	0,075
shekel_45	4	-10,15319585	5	-10,15319968	5	12	0,478	-10,15319968	4,15	0,385	-10,15319968	4	0,086
shekel_47	4	-10,40281884	7	-10,40294057	6	7	0,547	-10,40294057	4,7	0,409	-10,40294057	4	0,119
shekel_410	4	-10,53628373	10	-10,53640982	6	7	0,503	-10,53640982	6,4	0,343	-10,53640982	4	0,118
shekel_foxholes	5	-10,40560576	-	-10,40561724	25	217	6,247	-4,514929126	8,35	0,601	-2,700967461	4	0,103

Problem	Dim	Known minimum	Nº of known local minima	Balancing strategy using Sobol				Globalsearch			Multistart		
				Min	Min_count	It_min	Time	Min	Min_count	Time	Min	Min_count	Time
shekel_foxholes	10	-9,861357063	-	-10,20879279	29	63	223,207	-2,410314884	2,4	1,295	-1,479761058	1	0,404
shubert	2	-186,7309	760	-186,7309088	18	176	1,044	-186,7309088	10,9	0,27	-24,92043558	3	0,039
sinusoidal	10	-3,5	-	-3,5	638	1	81,875	-3,5	5,95	0,404	-3,5	5	0,111
sixhumpcamel	2	-1,031628453	6	-1,031628453	6	23	0,147	-1,031628453	4,95	0,155	-1,031628453	3	0,032
sphere	3	0	1	0	1	1	0,117	0	1	0,121	0	1	0,032
tenn_local_minima	2	2,36E-31	100	3,109750807	17	3	0,505	9,65E-15	3,45	0,153	9,65E-15	3	0,049
tenn_local_minima	4	1,18E-31	10^4	3,109584208	159	3	10,388	2,44E-14	6,05	0,249	2,44E-14	5	0,112
tenn_local_minima	6	7,85E-32	10^6	3,109528628	348	3	42,492	3,62E-15	5,5	0,282	3,62E-15	3	0,12
tenn_local_minima	8	5,89E-32	10^8	3,109500838	271	3	63,848	0,31098943	7,45	0,396	0,388758826	9	0,327
threehumpcamel	2	0	3	0	1	1	0,056	0	1,05	0,067	0	1	0,034
transistor	9	0	1	10,94380184	11	3049	69,382	4,421009843	9,2	2,246	0,002222034	6	1,303
wood	4	0	1	8,68E-18	1	12	0,444	5,68E-13	1	0,231	4,88E-14	1	0,177

Table I.4: Comparison of the proposed algorithm with conservative method for merging, with Globalsearch and Multistart functions.

Problem	Dim	Known minimum	Nº of known local minima	Conservative approach to merging using Sobol				Globalsearch			Multistart		
				Min	Min_count	It_min	Time	Min	Min_count	Time	Min	Min_count	Time
aluffi_pentini	2	-0,352386037	2	-0,352386074	1	9	0,099	-0,352386074	1,95	0,11	-0,352386074	2	0,036
bohachevsky	2	0	-	0	3	1	0,606	0	4,45	0,136	0	3	0,034
branin_hoo	2	0,397887358	3	0,397887358	4	24	0,128	0,397887358	3,75	0,136	0,397887358	2	0,034
cosine_mixture	2	-2,2	-	-2,2	19	992	5,581	-2,19999917	9,6	0,251	-2,199995447	3	0,028
cosine_mixture	4	-4,4	-	-4,4	304	3391	21,811	-4,399999656	17,1	0,355	-4,399990894	5	0,048
dekkers_aarts	2	-24777	3	-24776,51834	1	5000	14,168	-21060,04059	2,7	0,146	-24776,51834	3	0,055
exponencial	2	-1	-	-1	1	1	0,249	-1	1	0,252	-1	1	0,024
exponencial	4	-1	-	-1	1	1	5,447	-1	1	0,247	-1	1	0,035
fifteenn_local_minima	2	0	225	0,198772614	35	2	1,84	0,042310305	7,05	0,19	0,110253015	3	0,054
fifteenn_local_minima	4	0	50625	0,222427098	239	553	32,406	0,199753271	7,5	0,254	0,398546993	5	0,109
fifteenn_local_minima	6	0	15^6	0,598321572	266	2	59,106	0,166817061	8,35	0,39	2,32E-12	7	0,261
fifteenn_local_minima	8	0	15^8	0,397445014	206	4074	93,066	0,312334084	11,2	0,617	0,010987366	9	0,395
fifteenn_local_minima	10	0	15^10	1,22E-12	150	13	121,872	0,200984896	12,2	0,804	0,010987366	11	0,561
goldstein_price	2	3	4	3	4	29	0,234	3	2,05	0,164	3	3	0,057
griewank	5	0	-	0	207	1	45,511	0	11,5	0,243	0	5	0,091

Problem	Dim	Known minimum	Nº of known local minima	Conservative approach to merging using Sobol				Globalsearch			Multistart		
				Min	Min_count	It_min	Time	Min	Min_count	Time	Min	Min_count	Time
griewank	10	0	-	0	267	1	156,191	0	5,95	0,272	0	9	0,376
hosaki	2	-2,345811576	2	-2,345811576	3	5	5,224	-2,345811576	1,85	0,153	-2,345811576	2	0,034
kowalik	4	0,00030748	-	0,000307486	2	5	11,024	0,000307486	2,55	0,927	0,000307486	2	0,087
mccormick	2	-1,913222887	2	-1,913222955	3	12	5,224	-1,913222955	1	0,139	-1,913222955	2	0,033
multi_gaussian	2	-1,296954013	5	-1,296954046	3	3	0,127	-1,296954046	4,55	0,144	-1,296954046	2	0,041
neumaier2	4	0	-	3,05E-11	18	672	43,576	1,11E-07	6,15	2,586	1,50E-07	5	1,087
neumaier3	10	-210	-	-210	1	1	459,741	-210	1	0,16	-210	1	0,262
periodic	2	0,9	50	0,9	26	1	2,334	0,9	23,9	0,42	0,9	3	0,026
poissonian	2	71,25171938	-	71,25171938	10	43	5,556	71,25171938	3	0,376	71,25171938	2	0,047
powell	4	0	1	0	2	1	14,216	1,50E-15	4,25	0,202	0	5	0,128
rastrigin	10	0	-	0	627	1	123,854	0	12	0,666	0	7	0,237
rosenbrock	2	0	1	1,14E-17	1	18	0,298	2,05E-11	1,05	0,201	2,03E-11	1	0,075
shekel_45	4	-10,15319585	5	-10,15319968	5	13	2,985	-10,15319968	4,05	0,375	-10,15319968	4	0,086
shekel_47	4	-10,40281884	7	-10,40294057	7	8	3,3	-10,40294057	4,7	0,473	-10,40294057	4	0,121
shekel_410	4	-10,53628373	10	-10,53640982	10	8	3,429	-10,53640982	6,45	0,39	-10,53640982	4	0,117
shekel_foxholes	5	-10,40560576	-	-10,40561724	27	219	11,037	-4,78836344	7,85	0,596	-2,700967461	4	0,102
shekel_foxholes	10	-9,861357063	-	-10,20879279	27	54	499,44	-2,323943629	2,15	1,18	-1,479761058	1	0,404
shubert	2	-186,7309	760	-186,7309088	56	142	6,806	-186,7309088	11	0,265	-24,92043558	3	0,038
sinusoidal	10	-3,5	-	-3,5	12	1	361,358	-3,5	6,35	0,423	-3,5	5	0,111
sixhumpcamel	2	-1,031628453	6	-1,031628453	5	29	0,299	-1,031628453	4,95	0,172	-1,031628453	3	0,03
sphere	3	0	1	0	1	1	2,969	0	1	0,125	0	1	0,031
tenn_local_minima	2	2,36E-31	100	3,109750807	34	3	0,945	9,65E-15	3,7	0,159	9,65E-15	3	0,049
tenn_local_minima	4	1,18E-31	10^4	1,42E-12	213	74	17,664	2,44E-14	5,05	0,211	2,44E-14	5	0,111
tenn_local_minima	6	7,85E-32	10^6	3,15E-15	276	294	56,44	3,62E-15	5,25	0,277	3,62E-15	3	0,119
tenn_local_minima	8	5,89E-32	10^8	3,15E-21	198	88	84,597	0,233250519	7,2	0,416	0,388758826	9	0,332
threehumpcamel	2	0	3	0	3	1	0,282	0	2,85	0,151	0	1	0,033
transistor	9	0	1	0,002225648	15	4160	72,358	22,50999998	8,5	2,666	0,002222034	6	1,309
wood	4	0	1	8,68E-18	1	13	1,852	5,68E-13	1	0,231	4,88E-14	1	0,182

Table I.5: Comparison of the proposed algorithm with Globalsearch performing fmincon with a trust-region-reflective solver.

Problem	Dim	Known minimum	Nº of known local minima	Conservative approach to merging using Sobol				Globalsearch		
				Min	Min_count	It_min	Time	Min	Min_count	Time
aluffi_pentini	2	-0,352386037	2	-0,352386074	1	9	0,099	-0,352386074	1	0,175
bohachevsky	2	0	-	0	3	1	0,606	0	4,3	0,312
branin_hoo	2	0,397887358	3	0,397887358	4	24	0,128	0,397887358	3,05	0,148
cosine_mixture	2	-2,2	-	-2,2	19	992	5,581	-2,2	9,5	0,481
cosine_mixture	4	-4,4	-	-4,4	304	3391	21,811	-4,4	17,5	1,67
dekkers_aarts	2	-24777	3	-24776,51834	1	5000	14,168	-21060,04059	2,65	0,188
exponencial	2	-1	-	-1	1	1	0,249	-1	1	0,26
exponencial	4	-1	-	-1	1	1	5,447	-1	1	1,151
fifteenn_local_minima	2	0	225	0,198772614	35	2	1,84	0,016303932	16,55	0,535
fifteenn_local_minima	4	0	50625	0,222427098	239	553	32,406	0,051723251	16,8	1,628
fifteenn_local_minima	6	0	15^6	0,598321572	266	2	59,106	0,207973292	14,05	3,687
fifteenn_local_minima	8	0	15^8	0,397445014	206	4074	93,066	0,273244852	15,6	8,87
fifteenn_local_minima	10	0	15^10	1,22E-12	150	13	121,872	0,374114612	14,1	13,615
goldstein_price	2	3	4	3	4	29	0,234	4,35	2,05	0,233
griewank	5	0	-	0	207	1	45,511	0	9,45	1,308
griewank	10	0	-	0	267	1	156,191	0	5,7	4,368
hosaki	2	-2,345811576	2	-2,345811576	3	5	5,224	-2,345811576	1,25	0,285
kowalik	4	0,00030748	-	0,000307486	2	5	11,024	0,000307971	30,7	4,291
mccormick	2	-1,913222887	2	-1,913222955	3	12	5,224	-1,913222955	1	0,265
multi_gaussian	2	-1,296954013	5	-1,296954046	3	3	0,127	-1,296954046	3,3	0,147
neumaier2	4	0	-	3,05E-11	18	672	43,576	4,95E-05	11,6	12,175
neumaier3	10	-210	-	-210	1	1	459,741	-210	1,2	0,876
periodic	2	0,9	50	0,9	26	1	2,334	0,9	21,55	0,8
poissonian	2	71,25171938	-	71,25171938	10	43	5,556	71,25171938	11,75	0,539
powell	4	0	1	0	2	1	14,216	0	5,1	0,818
rastrigin	10	0	-	0	627	1	123,854	0	12,8	4,248
rosenbrock	2	0	1	1,14E-17	1	18	0,298	1,24E-17	1	0,304
shekel_45	4	-10,15319585	5	-10,15319968	5	13	2,985	-10,15319968	4	1,77
shekel_47	4	-10,40281884	7	-10,40294057	7	8	3,3	-10,40294057	4,7	2,142
shekel_410	4	-10,53628373	10	-10,53640982	10	8	3,429	-10,53640982	5,85	2,364
shekel_foxholes	5	-10,40560576	-	-10,40561724	27	219	11,037	-5,6950626	7,85	3,965

Problem	Dim	Known minimum	Nº of known local minima	Conservative approach to merging using Sobol				Globalsearch		
				Min	Min_count	It_min	Time	Min	Min_count	Time
shekel_foxholes	10	-9,861357063	-	-10,20879279	27	54	499,44	-3,295383485	3,35	19,261
shubert	2	-186,7309	760	-186,7309088	56	142	6,806	-186,7309088	13,15	0,377
sinusoidal	10	-3,5	-	-3,5	12	1	361,358	-3,5	5,75	5,053
sixhumpcamel	2	-1,031628453	6	-1,031628453	5	29	0,299	-1,031628453	5,1	0,299
sphere	3	0	1	0	1	1	2,969	0	1	0,309
tenn_local_minima	2	2,36E-31	100	3,109750807	34	3	0,945	0,15550353	5,65	0,264
tenn_local_minima	4	1,18E-31	10^4	1,42E-12	213	74	17,664	0,583065279	7,25	0,847
tenn_local_minima	6	7,85E-32	10^6	3,15E-15	276	294	56,44	0,699920314	7,85	2,421
tenn_local_minima	8	5,89E-32	10^8	3,15E-21	198	88	84,597	1,049674153	7,15	4,16
threehumpcamel	2	0	3	0	3	1	0,282	0	2,9	0,285
transistor	9	0	1	0,002225648	15	4160	72,358	52,98768297	8,4	65,132
wood	4	0	1	8,68E-18	1	13	1,852	8,29E-06	1,6	1,867

Table I.6: Comparison of the proposed algorithm with Multistart initialized with all the points generated in the initialization/relaunching step during the optimization process of the proposed algorithm.

Problem	Dim	Known minimum	Nº of known local minima	Conservative approach to merging using Sobol				Multistart		
				Min	Min_count	It_min	Time	Min	Min_count	Time
aluffi_pentini	2	-0,352386037	2	-0,352386074	1	9	0,099	-0,352386074	2	0,107
bohachevsky	2	0	-	0	3	1	0,606	0	10	0,202
branin_hoo	2	0,397887358	3	0,397887358	4	24	0,128	0,397887358	4	0,115
cosine_mixture	2	-2,2	-	-2,2	19	992	5,581	-2,19999984	24	0,455
cosine_mixture	4	-4,4	-	-4,4	304	3391	21,811	-4,39999992	202	6,247
dekkers_aarts	2	-24777	3	-24776,51834	1	5000	14,168	-24776,51834	3	0,999
exponencial	2	-1	-	-1	1	1	0,249	-1	1	0,133
exponencial	4	-1	-	-1	1	1	5,447	-1	1	2,789
fifteenn_local_minima	2	0	225	0,198772614	35	2	1,84	1,26E-13	63	0,893
fifteenn_local_minima	4	0	50625	0,222427098	239	553	32,406	2,04E-15	693	19,328
fifteenn_local_minima	6	0	15^6	0,598321572	266	2	59,106	2,01E-15	1393,25	59,279
fifteenn_local_minima	8	0	15^8	0,397445014	206	4074	93,066	1,51E-14	1530,95	93,262
fifteenn_local_minima	10	0	15^10	1,22E-12	150	13	121,872	0,010987366	1548,85	122,077
goldstein_price	2	3	4	3	4	29	0,234	3	4	0,249

Problem	Dim	Known minimum	Nº of known local minima	Conservative approach to merging using Sobol				Multistart		
				Min	Min_count	It_min	Time	Min	Min_count	Time
griewank	5	0	-	0	207	1	45,511	0	2044,45	44,914
griewank	10	0	-	0	267	1	156,191	0	2985,7	156,785
hosaki	2	-2,345811576	2	-2,345811576	3	5	5,224	-2,345811576	2	0,261
kowalik	4	0,00030748	-	0,000307486	2	5	11,024	0,000307486	5	6,616
mccormick	2	-1,913222887	2	-1,913222955	3	12	5,224	-1,913222955	2	0,248
multi_gaussian	2	-1,296954013	5	-1,296954046	3	3	0,127	-1,296954046	5,75	0,136
neumaier2	4	0	-	3,05E-11	18	672	43,576	1,96E-10	93,85	43,601
neumaier3	10	-210	-	-210	1	1	459,741	-210	1	460,38
periodic	2	0,9	50	0,9	26	1	2,334	0,9	40	0,502
poissonian	2	71,25171938	-	71,25171938	10	43	5,556	71,25171938	3	0,235
powell	4	0	1	0	2	1	14,216	0	309,35	14,267
rastrigin	10	0	-	0	627	1	123,854	0	2612,35	124,211
rosenbrock	2	0	1	1,14E-17	1	18	0,298	2,02E-11	1	0,318
shekel_45	4	-10,15319585	5	-10,15319968	5	13	2,985	-10,15319968	5	2,997
shekel_47	4	-10,40281884	7	-10,40294057	7	8	3,3	-10,40294057	7	3,376
shekel_410	4	-10,53628373	10	-10,53640982	10	8	3,429	-10,53640982	10	3,444
shekel_foxholes	5	-10,40560576	-	-10,40561724	27	219	11,037	-10,40561724	29	11,067
shekel_foxholes	10	-9,861357063	-	-10,20879279	27	54	499,44	-10,20879279	31	499,855
shubert	2	-186,7309	760	-186,7309088	56	142	6,806	-186,7309088	97	1,346
sinusoidal	10	-3,5	-	-3,5	12	1	361,358	-3,5	8120,95	363,456
sixhumpcamel	2	-1,031628453	6	-1,031628453	5	29	0,299	-1,031628453	5	0,214
sphere	3	0	1	0	1	1	2,969	0	1	0,757
tenn_local_minima	2	2,36E-31	100	3,109750807	34	3	0,945	7,82E-15	42	0,698
tenn_local_minima	4	1,18E-31	10^4	1,42E-12	213	74	17,664	1,78E-15	542,85	16,047
tenn_local_minima	6	7,85E-32	10^6	3,15E-15	276	294	56,44	1,75E-15	1172,3	56,594
tenn_local_minima	8	5,89E-32	10^8	3,15E-21	198	88	84,597	3,48E-15	1127,95	84,749
threehumpcamel	2	0	3	0	3	1	0,282	0	3	0,152
transistor	9	0	1	0,002225648	15	4160	72,358	4,33E-07	158	72,399
wood	4	0	1	8,68E-18	1	13	1,852	4,88E-14	1	1,866