Università degli Studi di Padova

DIPARTIMENTO DI INGEGNERIA INDUSTRIALE
Corso di Laurea Magistrale in Ingegneria Elettrica

Tesi di laurea magistrale

A DETERMINISTIC METHOD FOR THE MULTIOBJECTIVE OPTIMIZATION OF ELECTROMAGNETIC DEVICES AND ITS APPLICATION TO POSE DETECTION FOR MAGNETIC-ASSISTED MEDICAL APPLICATIONS

Candidato:
Giampaolo Capasso

Relatore:
Prof. Piergiorgio Alotto

Anno Accademico 2014-2015

Contents

Sommario

Per la risoluzione di problemi di ottimizzazione singolo e multi obiettivo sono stati implementati con successo algoritmi di tipo stocastico, per i quali però le proprietà di convergenza sono dimostrate solo empiricamente. In questo lavoro di tesi viene quindi presentato un algoritmo deterministico di tipo Pattern Search, per il quale viene illustrata la dimostrazione di convergenza globale.

Inoltre vengono sviluppati degli algoritmi ibridi innovativi, costituiti da una componente deterministica di tipo Pattern Search, e da una componente stocastica, per migliorare le performance, in particolar modo per le applicazioni ingegneristiche che richiedono algoritmi particolarmente aggressivi, senza perdere le proprietà di convergenza.

Le performance vengono analizzate utilizzando delle funzioni test, che mostrano l'efficacia degli approcci utilizzati mediante un confronto con gli algoritmi tradizionali puramente stocastici.

Infine viene considerato un problema per l'ottimizzazione di un dispositivo elettromagnetico per il rilevamento di posizione, che ha applicazioni in campo medico.

Abstract

In recent years stochastic methods have been successfully developed and applied in order to solve single objective and multiobjective optimization problems, but despite their success they still lack strong mathematical justification.

In this work we present a Pattern Search optimizer, which being a deterministic method enjoys provable convergence properties.

Furthermore, we develop alternatives and extensions of the standard deterministic method, and innovative hybrid algorithms merging the Pattern Search with some stochastic approaches, in order to improve the performance without sacrificing the theoretical properties.

The effectiveness of this approach is shown through numerical examples.

Finally, we apply this method to a real case problem for the pose detection for magnetic-assisted medical applications in order to optimize the performance of the device.

Chapter 1

Introduction

Automatic optimization of electromagnetic devices became crucial in the last two decades, in order to comply to specific needs and requirements.

Inverse problems in electricity and magnetism require methods of solution where the aim is to ensure that the shape and the performance of devices are optimal by specific criteria.

Especially stochastic methods have been successfully implemented and applied, but despite their achievement in solving both single and multiobjective optimization problems, they lack a solid mathematical base, and their properties are mostly demonstrated only empirically [1] [2].

Therefore, we want to introduce a deterministic optimizer, a Pattern search method, uncommon in the area of electromagnetic devices, highlighting its main features and its convergence properties.

Thus, innovative hybrid algorithms obtained merging the deterministic optimizer with some metaheuristics such as Particle Swarm Optimization and Differential Evolution are developed, in order to improve the performance without sacrificing the theoretical properties.

From the performance analysis, this hybridization results into an efficient algorithm for the solution of electromagnetic design problems, and can be effectively implemented in a practical application, for the pose detection for magnetic-assisted medical devices.

1.1 Direct and inverse problems

In engineering science the direct problems are the ones where given the input or the cause of a phenomenon the aim is to find the output or the effect. Conversely in the inverse problems given an expected output or effect we want to compute the input or the cause, trying to understand their relation.

The design problems of electromagnetic devices belong to the category of the inverse problems. We can now describe the main features of inverse problems and some methods for their resolution [3].

While in the direct problems we have enough information to carry out an unique solution, there is no guarantee of having an unique solution in inverse problems. In fact given the input, the output and the operator modelling the input-output transformation respectively as x,y and A, the goal in the direct problem is to find the output y at point in the domain.

Assuming that the operator A is invertible the inverse problem for A is the direct problem for A^{-1}, while if A is not invertible the solution to the inverse problem does not exist. Furthermore, for the same given output y_1 we may have $x_1 = A^{-1}(y_1)$ and $x_2 = A^{-1}(y_1)$, with $x_1 \neq x_2$.

We can make a distinction from a mathematical point of view between well-posed and ill-posed problem.

The well-posed have the following features

- A solution always exists
- There is only one solution
- A small change of data leads to a small change in the solution

The ill-posed problems are those for which

- The solution may not exist
- There may be more than one solution
- A small change of data may lead to a big change in the solution

The inverse problems belong to the latter category, and show an high degree of insidiousness, because a solution may not exist for optimal design problems, and, on the other hand, if multiple solutions exist to a given problem they might be similar.

In engineering applications the aim is to design the geometry of a device in order to obtain a prescribed performance, which depends on the field.

1.2 Methods for solving inverse problems

1.2.1 Functional minimization

The solution to the inverse problem is performed using a suitable function $f(\boldsymbol{x})$ called objective function or cost function. Given $\boldsymbol{x}_0 \in \Omega \subseteq \mathbb{R}^n$, where Ω is the feasible region and $\boldsymbol{x}$ the unknown variables.

$$find \inf_{\boldsymbol{x}} f(\boldsymbol{x}) \quad \boldsymbol{x} \in \Omega \subseteq R^n \tag{1.1}$$

Where x_0 is the initial guess. The objective function may represent a performance depending on the field, or a residual between a computed and a known field value in order to find the design variables which deliver to the prescribed magnetic configuration.

Usually the objective function f is not known in analytical way, but is only known numerically as a set of values at sample points. Then, the solution can be obtained numerically, and requires a routine for calculating the field together with a routine for the minimization of the objective function.

The system is represented through a finite element model in two or three dimensions, and the minimization routine may be described as a step by step procedure as it is explained below.

Starting from the point $\boldsymbol{x}_0$, the iteration k updates the current design point $\boldsymbol{x}_k$ as

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha \boldsymbol{s}_k \tag{1.2}$$

Where α is a scalar displacement and $\boldsymbol{s}_k$ is the search direction in the feasible region. Given the new point $\boldsymbol{x}_{k+1}$ the finite element model is restarted and the evaluation of $f(\boldsymbol{x})$ is updated.

The computation ends when the stopping criterion is achieved, that criterion is defined by the parameter $\varepsilon > 0$.

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\| \leq \varepsilon \|\boldsymbol{x}_k\| \tag{1.3}$$

1.2.2 Rectangular systems of Algebraic equations

The numerical solution of field problems generally leads to a system of algebraic equations.

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{1.4}$$

Where $\boldsymbol{A}$ is a rectangular $m \times n$ matrix, $\boldsymbol{x}$ is the unknown n-vector and $\boldsymbol{b}$ the known m-vector. Usually, for the inverse problems the system is over-determined, $m > n$, because the number of condition to fulfil is greater than the degrees of freedom available, while for the direct problems, $m = n$, the matrix $\boldsymbol{A}$ is square and if $det(\boldsymbol{A}) \neq 0$, $\boldsymbol{A}$ is non singular, therefore $\boldsymbol{A}^{-1}$ exists and the system of equations has an unique solution for any $\boldsymbol{b}$.

In case $m > n$ and the rank of $\boldsymbol{A}$ is equal to n, we can look for a pseudo-inverse of $\boldsymbol{A}$, using numerical methods like Least-Squares or Singular Value Decomposition.

Least squares. A solution to the problem defined by equation 1.4 can be found minimizing the Euclidean norm of the residual $\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}$. This is defined as

$$r(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\| = \boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} - 2\boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{b} + \boldsymbol{b}^T\boldsymbol{b} \tag{1.5}$$

The gradient of the residual is

$$\nabla r(\boldsymbol{x}) = 2\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} - 2\boldsymbol{A}^T\boldsymbol{b} \tag{1.6}$$

We want to find the minimum point for the residual, that is the vector for which the gradient is equal to zero. This condition gives

$$\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b} \tag{1.7}$$

The matrix $\boldsymbol{A}^T\boldsymbol{A}$ is a square $n \times n$ matrix, and we can prove that the minimum vector $\boldsymbol{x}_{min} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b}$ suits the following condition for each n-dimensional vector $\boldsymbol{x}$

$$\|\boldsymbol{A}\boldsymbol{x}_{min} - \boldsymbol{b}\| \leq \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\| \tag{1.8}$$

Therefore, $\boldsymbol{x}_{min}$ is the least square solution, and the matrix $(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T$ is called the pseudo inverse of matrix $\boldsymbol{A}$.

From the numerical point of view the solving process using this method may not succeed for the magnification of ill condition when we pass from $\boldsymbol{A}$ to $\boldsymbol{A}^T\boldsymbol{A}$, and the round-off errors after calculating the entries of $\boldsymbol{A}^T\boldsymbol{A}$.

These issues may lead to instability and inaccuracy.

Singular-Value Decomposition. This method consists in the decomposition of matrix $\boldsymbol{A}$, assumed to be full column rank, into the product of three matrices, an orthogonal matrix $m \times m$ $\boldsymbol{U}$, a block diagonal matrix $m \times n$ $\boldsymbol{S}$, and an orthogonal $n \times n$ matrix $\boldsymbol{V}$. The result of the product is $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T$, where

$$\boldsymbol{S} = \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} \tag{1.9}$$

The diagonal entries of Σ are the singular values of $\boldsymbol{A}$, with $\Sigma = diag(\sigma_1, ..., \sigma_n)$.

In this case the solution to the least-square problem is given by $x_{min} = \boldsymbol{V}\boldsymbol{S}^{-1}\boldsymbol{U}^T\boldsymbol{b}$, with

$$\boldsymbol{S}^{-1} = \begin{bmatrix} \Sigma^{-1} & 0 \\ 0 & 0 \end{bmatrix} \tag{1.10}$$

Being $\Sigma^{-1} = diag(\sigma_1^{-1}, ..., \sigma_n^{-1})$. The matrix $\boldsymbol{V}\boldsymbol{S}^{-1}\boldsymbol{U}^T$ represents another pseudo-inverse of $\boldsymbol{A}$.

1.3 The need for multiobjective optimization

Most problems in engineering have several conflicting objectives to be satisfied, therefore the single-objective approach appears very limited for the nature of design problems, and usually consists in the transformation of all the objectives but one into constraints.

Therefore, multiple objective functions must be taken into account, and must be optimized simultaneously. In the formulation of the problem, a vector $F(\boldsymbol{x})$ holds the n_f objective functions, and considering n variables

$$given \quad \boldsymbol{x}_0 \in R^n, \quad find \inf_{\boldsymbol{x}}(F(\boldsymbol{x})), \quad \boldsymbol{x} \in \mathbb{R}^n \tag{1.11}$$

The problem is subject to n_c inequality, n_e equality constraints, and $2n$ side bounds.

$$g_i(\boldsymbol{x}) \leq 0, \quad i = 1, n_c \tag{1.12}$$
$$h_j(\boldsymbol{x}) = 0, \quad j = 1, n_e \tag{1.13}$$
$$l_k \leq x_k \leq u_k, \quad k = 1, n_v \tag{1.14}$$

The vector $\boldsymbol{F}$ defines a transformation from the design space $\mathbb{R}^n$ to the corresponding objective space $\mathbb{R}^{n_f}$.

The classical approach for the conversion from a multi to a single objective problem is done through a preference function $\psi(\boldsymbol{x})$, that is the weighted sum of the objectives

$$\psi(\boldsymbol{x}) = \sum_{i=i}^{n_f} f_i(\boldsymbol{x}) \tag{1.15}$$

With $0 < c_i < 1$, and $\sum_{i=i}^{n_f} c_i = 1$.

The weights allocation define the hierarchy attributed to the objectives, and influences the solution that will be computed, and will be unique.

The drawbacks of this method are mainly two, in fact the n_f objectives are often non-commensurable, having different physical dimensions, for these reasons the weighting procedure has no meaning from the physical point of view, and needs an appropriate and crucial scaling process for the choice of weights. Furthermore, this method has an a-priori bias, because the solution obtained can vary according the user decisions taken before the solving process.

On the contrary the Pareto optimization approach delivers a set of non dominated solutions, the Pareto front, those for which a decrease of a function is not possible without the simultaneous increase of at least one of the other functions. An example of Pareto front is shown in figure 1.1 [4]. Therefore, a family of equally good solutions is computed, without having any a-priori bias and without the need to mix functions with different unity measures.

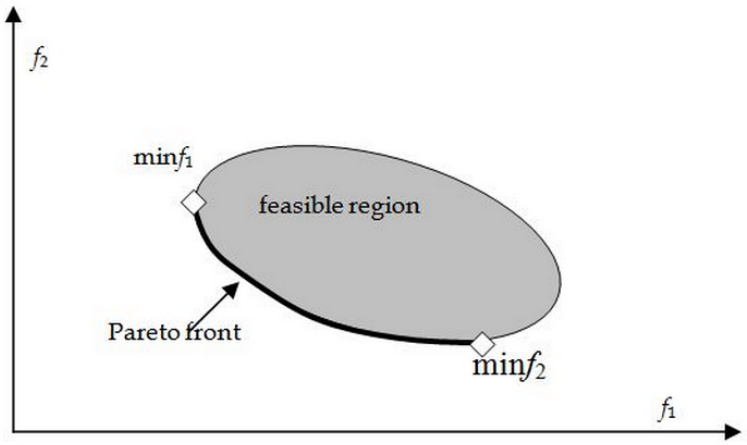

Figure 1.1: Pareto front

Finally, the decision maker will be able to decide, a-posteriori and having an higher level of information, which solution to pick from the front.

We want now to define the concepts of domination and the Pareto optimality [5].

1.3.1 Concept of domination

Assume that there are n_f objective functions. We use the operator $\triangleleft$ between two solutions i and j, $i \triangleleft j$, to point that solution i is better than j for a particular objective. On the other hand, when i is worse than j for a particular objective we write that $i \triangleright j$.

The definition of the problem through this operator covers also mixed optimization problem, with minimization for some objective functions, and maximization for others.

Definition 1. A solution $\boldsymbol{x}^{(1)}$ dominates the other solution $\boldsymbol{x}^{(2)}$ if the two following conditions are verified:

1) The solution $\boldsymbol{x}^{(1)}$ is no worse than $\boldsymbol{x}^2$ in all objectives, or $f_j(\boldsymbol{x}^{(1)}) \not\triangleright f_j(\boldsymbol{x}^{(2)})$ for all $j = 1,2,...,n_f$.

2) The solution $\boldsymbol{x}^{(1)}$ is strictly better than $\boldsymbol{x}^2$ in at least one objective, or $f_j(\boldsymbol{x}^{(1)}) \triangleleft f_j(\boldsymbol{x}^{(2)})$ for at least one $j \in \{1,2,....,n_f\}$.

In figure 1.2 [4] the difference between non dominated and dominated solutions is illustrated.

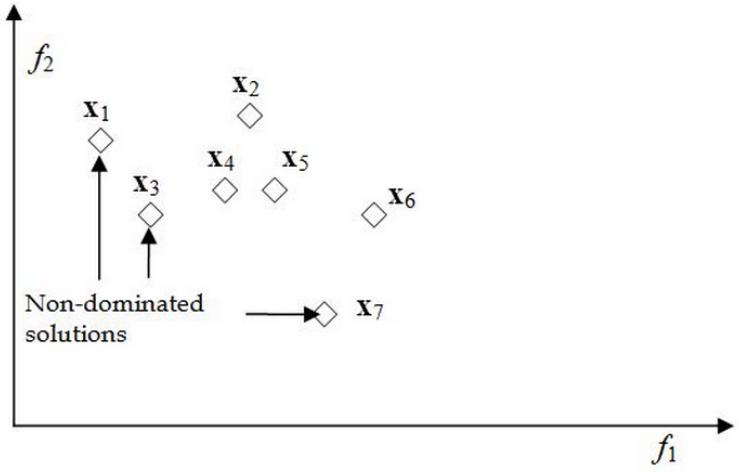

Figure 1.2: Non dominated vs dominated solutions

If any of the above condition is violated, the solution $\boldsymbol{x}^{(1)}$ does not dominate the solution $\boldsymbol{x}^2$, while if $\boldsymbol{x}^{(1)}$ dominates $\boldsymbol{x}^2$, the mathematical expression used is $\boldsymbol{x}^{(1)} \preceq \boldsymbol{x}^{(2)}$. Since the domination concept enables the comparison of multiple objectives solutions, the domination concepts is used by the multiobjective optimization methods to search for the non dominated solutions.

Furthermore, the definition 1 can be modified in order to introduce the notion of strong dominance relation as is illustrated below.

Definition 2. A solution $\boldsymbol{x}^{(1)}$ strongly dominates a solution $\boldsymbol{x}^{(2)}$, or $\boldsymbol{x}^{(1)} \prec \boldsymbol{x}^{(2)}$, if solution $\boldsymbol{x}^{(1)}$ is strictly better than solution $\boldsymbol{x}^{(2)}$ in all the n_f objectives.

If a solution $\boldsymbol{x}^{(1)}$ strongly dominates a solution $\boldsymbol{x}^{(2)}$, $\boldsymbol{x}^{(1)}$ also weakly dominates $\boldsymbol{x}^{(2)}$ but not vice versa.

The domination relation owns the following properties

- Reflexive. The relation is not reflexive because any solution $\boldsymbol{x}$ does not dominate itself
- Symmetric. The relation is not symmetric, because $\boldsymbol{x} \preceq \boldsymbol{y}$ does not imply $\boldsymbol{y} \preceq \boldsymbol{x}$. The opposite is true, if $\boldsymbol{x} \preceq \boldsymbol{y}$ then $\boldsymbol{y} \npreceq \boldsymbol{x}$
- Antisymmetric. Since the relation is not symmetric it cannot be antisymmetric as well
- Transitive. The relation is transitive, because if $\boldsymbol{x} \preceq \boldsymbol{y}$ and $\boldsymbol{y} \preceq \boldsymbol{z}$, then $\boldsymbol{x} \preceq \boldsymbol{z}$

1.3.2 Pareto-Optimality

Given a set of solutions, we can execute all possible pair-wise comparisons in order to find which solutions are non-dominated with respect to each other.

At the end, we will have a set of solutions, any two of which do not dominate each other. Obviously, for each solution outside this set, we can always find a solution in the set which dominates the former.

This set is called non-dominated set, and is defined below.

Definition 3. Among a set of solutions P, the non dominated set of solutions P' are those that are not dominated by any member of the set P.

From the notion of weak dominance we can also define the weakly non dominated set as follows

Definition 4. Among a set of solution P, the weakly non-dominated set of solutions P' are those that are not strongly dominated by any other member of the set P.

The cardinality of the weakly non-dominated set is greater than or equal to the cardinality of the non-dominated set as it is defined in 3.

Thus, the aim of a multiobjective optimization algorithm is to find a computationally efficient procedure to identify the non-dominated set from a population of solutions.

1.3.3 A practical example

Consider the electric circuit in figure 1.3. Given the values of the voltage source E_s and the impedance $Z_1 = R_1 + jX_1$, we want to find the values of the load impedance $Z_2 = R_2 + jX_2$ in order to maximize two objectives, the efficiency of the circuit η which is defined as the ratio between the power delivered to the load P_2 and the power source P_s, and the effective power delivered to the load P_2.

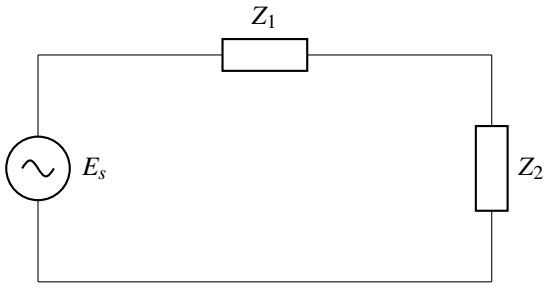

Figure 1.3: Electric circuit benchmark

The mathematical expressions of the two objective functions is given in the next equation.

$$\begin{cases} \eta = \dfrac{R_2}{R_2 + R_1} \\ P_2 = E_s^2 \dfrac{R_2}{(R_1 + R_2)^2 + (X_1 + X_2)^2} \end{cases} \tag{1.16}$$

We want now to show that the efficiency and the load power are two conflicting objectives.

Considering that in order to reduce the denominator the imaginary part of Z_2 must be equal to the opposite of the one Z_1, $X_2 = -X_1$, and that $R_2 = R_1 \dfrac{\eta}{1-\eta}$, we can express the power load as a function of the efficiency.

$$P_2 = \frac{E_s^2}{R_1} \eta (1 - \eta) \tag{1.17}$$

This relation is also represented in figure 1.4, setting for simplicity all the known values equal to one: $E_s = 1V$, $R_1 = 1\Omega$.

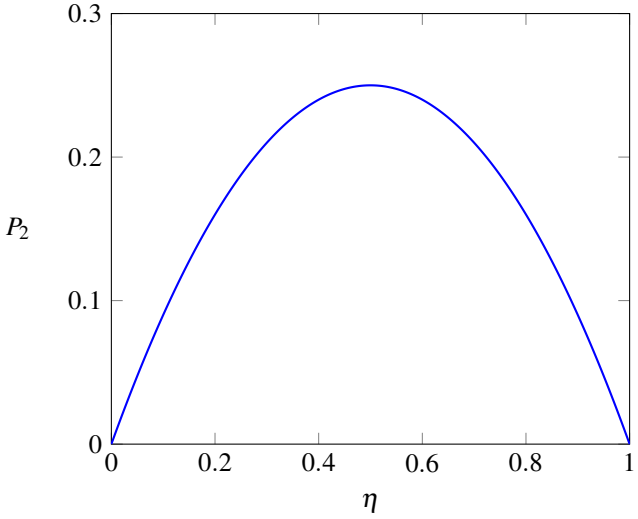

Figure 1.4: Power load as a function of the efficiency

As we can notice, an increase of η from 0 to 0.5, produces an improvement in the power load, which reaches its maximum value $P_{max} = 0.25$ for $\eta = 0.5$. After this point, an improvement in the efficiency

causes a deterioration in P_2. Therefore, we have two conflicting objectives, and we need to find a trade off between the two.

The Pareto front and the Pareto set are represented, respectively, In figure 1.5 and 1.6. We have considered unitary value also for the imaginary parts of the two impedance $X_1 = 1\Omega$, $X_2 = -X_1 = -1\Omega$.

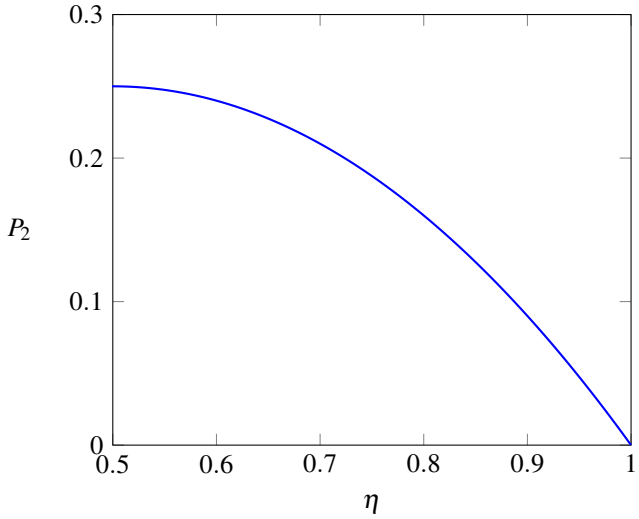

Figure 1.5: Pareto front for the benchmark problem

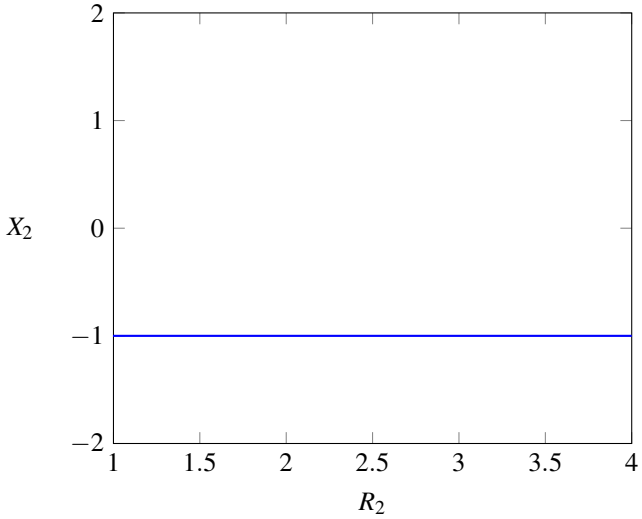

Figure 1.6: Pareto set for the benchmark problem

After all we have found a set of non dominated solutions, among which the decision maker will have to choose, deciding whether to privilege the efficiency of the system, the effective power transmitted to the load, or to find a balance between the two.

1.4 Methods for solving multiobjective optimization problems

1.4.1 Weighted sum method

Through this method a composite objective function $F(\boldsymbol{x})$ is created, summing all the normalized objective functions. This converts the problem into a single objective problem.

$$\min F(\boldsymbol{x}) = \sum_{m=1}^{n_f} w_m f_m(\boldsymbol{x}) \tag{1.18}$$

With $\sum_{m=1}^{n_f} w_m = 1$.

We can now consider the bi-objective case, supposing that both two the objectives need to be minimized. Thus, we have to find the minimum of the function $F(\boldsymbol{x}) = w_1 f_1(\boldsymbol{x}) + w_2 f_2(\boldsymbol{x})$.

Once the weights w_1 and w_2 are fixed, the search process is defined as a search for the point of the straight line defined by the equation $f_2 = -\dfrac{w_1}{w_2} f_1$ which is externally tangent to the Pareto front.

Therefore, only in the case of a convex Pareto front, all the non dominated solutions can be computed varying the weights, while if the front is non-convex some solutions are missed. Moreover, in the convex case it is difficult to identify the selection of weights which produce an uniform distribution of solutions on the Pareto front.

The next method illustrated, is able to deal also with non convex Pareto front.

1.4.2 ε-Constraint Method

This method keeps just one objective, and restricts the remaining ones within values defined by the user.

$$\begin{cases} \min f_\mu(\boldsymbol{x}) \\ f_m(\boldsymbol{x}) \leq \varepsilon_m \qquad m = 1,2,...,n_f \quad and \quad m \neq \mu \end{cases} \tag{1.19}$$

A suitable value for ε_m must be chosen for the m-th objective, requiring an a priori knowledge of appropriate range for the ε , and an unique solution is computed along the front.

Furthermore, the optimization procedure can be iterated with different values of ε_m in order to find several non dominated solutions.

This method works also with non convex Pareto front.

1.4.3 Goal-Attainment Method

This method minimizes the scalar quantity $\gamma \in \mathbb{R}$ with respect to the design vector $\boldsymbol{x}$, subject to

$$f_i(\boldsymbol{x}) - w_i \gamma \leq f_i^*, \quad i = 1,...,n_f \tag{1.20}$$

Where $f_i(\boldsymbol{x})^*$, with $i = 1,...,n_f$, are the objectives wanted by the user, which define the goal point $A = \{f_i^*\}$ that might be unfeasible.

Usually, the implementation of this method is done through the use of the following preference function.

$$\psi(\boldsymbol{x}) = \sup_i \frac{f_i(\boldsymbol{x}) - f_i^*}{w_i}, \quad w_i \neq 0, \quad i = 1,...,n_f \tag{1.21}$$

Also this method works for both convex and non-convex fronts.

1.5 Deterministic and stochastic methods

The methods illustrated in the previous section require a certain degree of problem knowledge, and essentially convert a multiobjective problem into a single objective one. For this reason we need to apply them several times to find the approximation of the Pareto front for a given problem, and this represents the main practical drawback.

We need therefore to keep the conflicting objectives separate, discarding the use of preference function, and using the knowledge we have about the problem necessary to pick a single solution a posteriori, that is after the optimization process, instead of a priori, before the process starts.

Furthermore, we need to use derivative-free methods, which are very appropriate when computing the derivatives is expensive, unreliable, or even impossible.

Two great families of algorithms are used for this purpose, the deterministic and stochastic methods.

Deterministic methods does not have stochastic elements, meaning that they do not use random variables, and the entire input and output relation of the model is conclusively determined, which means that given a certain input, it will always produce the same output.

Stochastic methods uses one or more stochastic elements, having random variables, meaning that the state is randomly determined, and each execution gives different results.

The deterministic method considered in this work is the Pattern search method, described in Chapter 2, while some of the most effective and popular stochastic algorithms, which belong to the family of evolutionary optimization, are illustrated in Chapter 3.

The two families of algorithms usually display different features and merits, in term of convergence speed, accuracy, robustness, and parallelization possibility.

Generally, the deterministic methods show

- High speed
- High accuracy, given to the strong local search or refinement component
- Poor robustness, because of the lack of a strong global search component
- Difficult to parallelize, given to the sequential behaviour

While the evolutionary methods

- Low speed
- Low accuracy, given to the lack of a strong local search or refinement component
- High robustness, because of a strong global search component
- Easy to parallelize, being population based algorithms

Given the diversity in the features and the qualities of the two kinds of algorithm, our aim is to find a right mix between the methods able to ensure a good performance for the speed, the accuracy, the robustness, and easy to parallelize.

The formulations of two new hybrid algorithms is carried out in Chapter 4, where also some alternatives for the Pattern Search method are described, and the assessment of their performance is illustrated in Chapter 5.

Lastly, in Chapter 6, a practical application is considered, regarding the pose detection for magnetic-assisted medical devices. The use of optimization algorithm is required, and the effectiveness of the multiobjective approach is pointed out.

Chapter 2

Pattern search methods

The Pattern search methods use the paradigm of direct-search methods of directional type, extending this approach to the multiobjective case through the concept of Pareto dominance. The polling procedure should generate as many points as possible in the Pareto front, thus representing an interesting alternative to the stochastic methods, being a deterministic derivative-free methodology.

In this chapter we describe this method both for the single and multiobjective case, demonstrating the global convergence in both two cases. The provable convergence properties is an advantage with respect to the stochastic methods, which lack a solid mathematical foundation.

2.1 Pattern search for Single-Objective optimization

Consider the following problem of minimizing a continuously differential function without computing or approximating the derivatives of f

$$min_{\boldsymbol{x}\in X} f(\boldsymbol{x}) \qquad X \subseteq \mathbb{R}^n \qquad f\colon \mathbb{R}^n \to \mathbb{R}$$

The pattern search methods (PS) solve the minimization problem computing and comparing the values of the function on the points lying on a grid or a lattice generated by a matrix $\boldsymbol{D}_k$, which holds on the vectors column the search directions responsible for the shape, and a step size parameter Δ_k which defines the distance between two consecutive points on the grid. This grid will be explored starting from an arbitrary point $\boldsymbol{x}_0 \in \mathbb{R}^n$ according to the steps of the following algorithm [6] [7].

for $k = 0, 1, 2...$

1) compute $\mathbf{s}_k = \Delta_k \boldsymbol{d}_k$, where $\boldsymbol{d}_k$ is a vector column of the matrix $\boldsymbol{D}_k$

2) Search step, generate the mesh $M_k = \{\boldsymbol{x}_k + \Delta_k \boldsymbol{D}_k z, z \in \mathbb{N}_0\}$ and compute the function in some points $\boldsymbol{x} \in M_K$. If $f(\boldsymbol{x}) < f(\boldsymbol{x}_k)$ declare the iteration successful $\to \boldsymbol{x}_{k+1} = \boldsymbol{x}$, skip the poll step and update the step size, else go to the poll step.

3) Poll step, compute the function just in the points around the poll center (current iteration) $P_k = \{\boldsymbol{x}_k + \Delta_k \boldsymbol{d}, \boldsymbol{d} \in \boldsymbol{D}_k\}$, if $f(\boldsymbol{x}_k + \Delta_k \boldsymbol{d}) < f(\boldsymbol{x})$ declare the iteration successful $\to \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta_k \boldsymbol{d}$, and go to the search step, else $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ and go to the search step. In both cases update the step size.

The step size must be updated according to the following rule

$$\Delta_{k+1} = \begin{cases} \theta \Delta_k, & \text{if the iteration was not successful} \\ \lambda \Delta_k, & \text{if the iteration was successful} \end{cases} \tag{2.1}$$

With $0 < \theta < 1$, and $\lambda \geq 1$.

We can develop several versions of the algorithm adapting the step size with different values of θ and λ, or imposing a greater decrease of the function instead of a simple one to declare the iteration successful, but the crucial point is the strategy adopted for the poll step. In fact we can adopt the complete strategy, in which the function is assessed in all the points around the poll step and the best one is chosen, or the dynamic strategy which computes and compares with the poll center one at a time

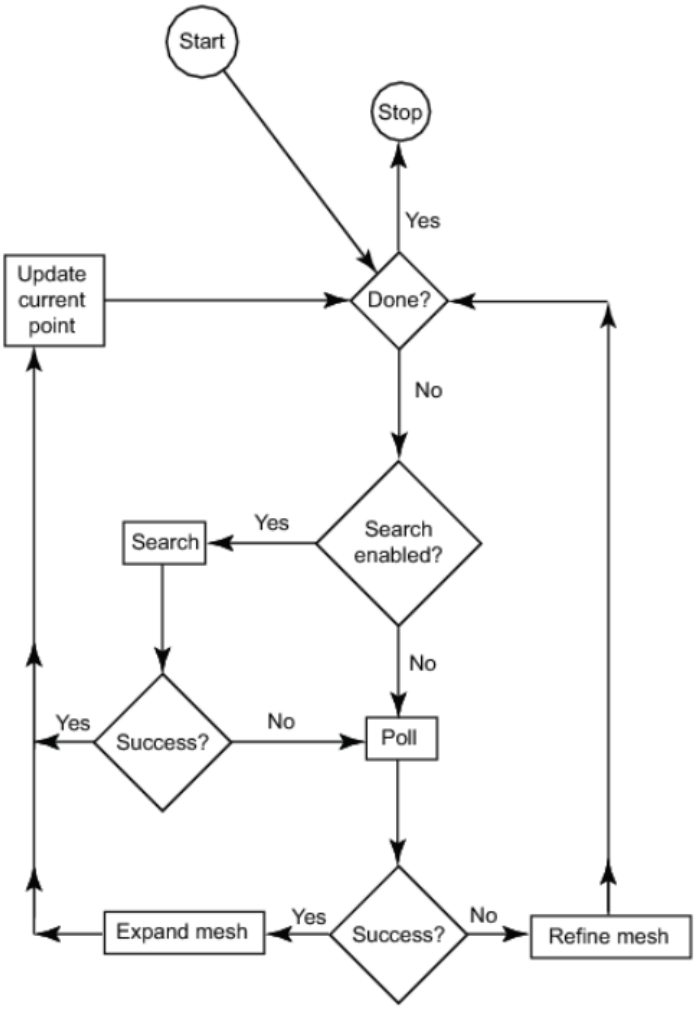

Figure 2.1: Pattern search flowchart

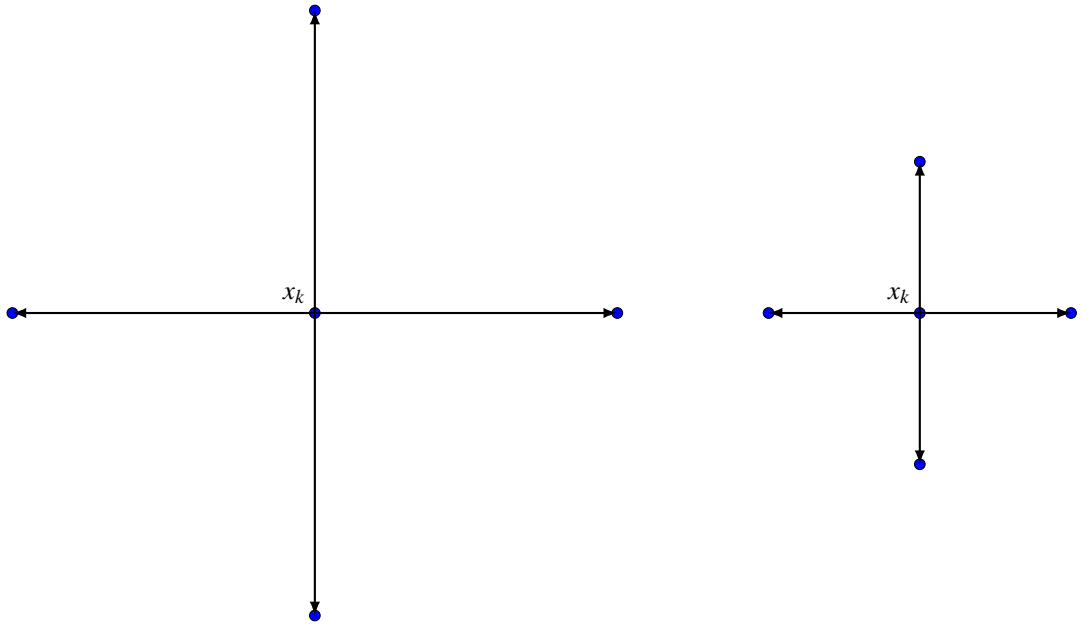

Figure 2.2: Example of polling with different step sizes

the points around, and as soon as a better point is found that becomes the new poll center. The search step does not have the aim to find a stationary point, but is used to explore the domain in all its parts.

2.2 Pattern search for Multi-Objective optimization

Consider the following Multi-Objective optimization problem

$$min_{\boldsymbol{x}\in\Omega}\boldsymbol{F}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), ..., f_m(\boldsymbol{x}))^T \qquad f_j\colon \mathbb{R}^n \to \mathbb{R}$$
$$\Omega = \{\boldsymbol{x} \in \mathbb{R}^n : \mathbf{l} \leq \boldsymbol{x} \leq \mathbf{u}\} \qquad \mathbf{l}, \mathbf{u} \in \mathbb{R}^n$$

Having several contrasting objectives we try to capture the whole Pareto front, computing all the non-dominated points. For that reason is crucial to define the concept of Pareto dominance.

$$\boldsymbol{x} \prec \boldsymbol{y} \iff \boldsymbol{F}(\boldsymbol{x}) \prec_{\boldsymbol{F}} \boldsymbol{F}(\boldsymbol{y})$$

Given two points $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, $\boldsymbol{x}$ dominates $\boldsymbol{y}$ ($\boldsymbol{x} \prec \boldsymbol{y}$) when all the objective functions for $\boldsymbol{x}$ are not strictly greater than for y but at least one which is strictly lower ($\boldsymbol{F}(\boldsymbol{x}) \prec_{\boldsymbol{F}} \boldsymbol{F}(\boldsymbol{y})$).

The next algorithm [8] tries to get the Pareto set excluding the dominated points with the function $filter$, which builds a set of disequations to discover the dominated points. For that purpose an iterative list L_k holding several items is initialized, every item consists in a point and a step size associated to the point. As in the single objective case a search step and a poll step are implemented, defining a directions matrix $\boldsymbol{D}_k$ and a step size Δ_k.

for $k = 0, 1, 2...$
1) Selection, order the list L_k and select the first item
$(\boldsymbol{x}, \Delta) \in L_k \to (\boldsymbol{x}_k, \Delta_k) = (\boldsymbol{x}, \Delta)$
2) Search step, assess the function in a finite set of points $\boldsymbol{x} \in M_k$ where M_k is the lattice $M_k = \{\boldsymbol{x}_k + \Delta_k \boldsymbol{D}_k z, z \in \mathbb{N}_0\}$. Build the list $L_{add} = (\boldsymbol{x}, \Delta_k)$, and call the function $filter$ to exclude the dominated points from the set $L_k \cup L_{add}$, $L_{filter} = filter(L_k, L_{add})$. Define a trial list L_{trial} putting $L_{trial} = L_{filter}$ or $L_{trial} \subseteq L_{filter}$. If $L_{trial} \neq L_k$ the iteration is successful because has found new non dominated points, put $L_{k+1} = L_{trial}$ update the step size and skip the poll step, else with $L_{trial} = L_k$ the iteration is not successful, therefore go to the poll step.

3) Poll step, search around the point of the selected item $P_k = \{\boldsymbol{x}_k + \Delta_k \boldsymbol{d} : \boldsymbol{d} \in \boldsymbol{D}_k\}$, define $L_{add} = \{(\boldsymbol{x}_k + \Delta_k \boldsymbol{d}, \Delta_k), \boldsymbol{d} \in \boldsymbol{D}_k\}$, call the function $filter$ $L_{filter} = filter(L_k, L_{add})$ and put $L_{trial} = L_{filter}$ or $L_{trial} \subseteq L_{filter}$. If $L_{trial} \neq L_k$ put $L_{k+1} = L_{trial}$, else $L_{k+1} = L_k$. In both cases update the step size, and go to the search step.

The step size is updated following the same rule of the single objective case (equation 2.1).

$$
\begin{array}{l}
\text{Set } L_3 = L_1 \cup L_2 \\
\textbf{for all } x \in L_2 \\
\quad \textbf{do} \begin{cases}
\textbf{for all } y \in L_3,\ y \neq x \\
\quad \textbf{do} \begin{cases} \textbf{if } y \prec x \\ \quad \textbf{then } \{L_3 = L_3 \backslash \{x\} \end{cases} \\
\textbf{if } x \in L_3 \\
\quad \textbf{then} \begin{cases}
\textbf{for all } y \in L_3,\ y \neq x \\
\quad \textbf{do} \begin{cases} \textbf{if } x \prec y \\ \quad \textbf{then} \begin{cases} L_3 = L_3 \backslash \{y\} \\ \textbf{if } y \in L_2 \\ \quad \textbf{then } \{L_2 = L_2 \backslash \{y\} \end{cases} \end{cases}
\end{cases}
\end{cases}
\end{array}
$$

Figure 2.3: Procedure to filter the dominated points from the set $L_1 \cup L_2$ assuming that L_1 is already formed by nondominated points.

An additional issue in this algorithm is the ordering strategy of the list L_k, from which depends the first item that will be selected. A strategy could be adding the new points on the bottom of the list, and the poll center in the last position in order to start the polling from a completely new point. Furthermore, to disseminate in a better way the points on the Pareto front and to reduce the size of the storage, we can choose $L_{trial} \subseteq L_{filter}$ in a way to exclude the points for which the objective functions assume values too close to the ones already found.

2.2.1 A worked example

To illustrate how the algorithm works for the multiobjective case we present its application to the ZDT2 test problem.

$$
ZDT2 : \begin{cases}
f_1(\boldsymbol{x}) = x_1 \\
g(\boldsymbol{x}) = 1 + \dfrac{9}{n-1} \sum_{i=2}^{n} x_i \\
h(f_1, g(\boldsymbol{x})) = 1 - (f_1 / g(\boldsymbol{x}))^2 \\
f_2(\boldsymbol{x}) = g(\boldsymbol{x}) h(f_1(\boldsymbol{x}), g(\boldsymbol{x}))
\end{cases} \tag{2.2}
$$

With $n = 2$ variables.

$$
\min \boldsymbol{F}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x})))^T
$$

$$
x_i \in [0, 1]
$$

In the following solving process no search step will be performed, and no ordering strategy is considered. The initial step size is $\Delta_0 = 0$, while the contraction and expansion factor are respectively $\theta = 0.5$ and $\lambda = 1$.

Initialization. We set the initial point $\boldsymbol{x}_0 = (0.5, 0.5)$, which corresponds to $(f_1(\boldsymbol{x}_0), f_2(\boldsymbol{x}_0)) = (0.5, 5.455)$. The poll step matrix is $\boldsymbol{D} = [\mathbf{I}_2 - \mathbf{I}_2]$, where $\mathbf{I}_2$ stands for the identity matrix of dimension 2. Thus, the starting iterate list of nondominated points is $L_0 = \{(\boldsymbol{x}_0; 1)\}$ where the step size associated to each point is also contained.

Iteration 0. The poll poll set is $P_0 = \{(0.5,0.5)+(1,0),(0.5,0.5)+(0,1),(0.5,0.5)+(-1,0),(0.5,0.5)+(0,-1)\}$. All these points are not feasible and the function is not evaluated for the P_0 set. Therefore the iteration is declared unsuccessful and the step size is contracted with the factor θ, $\Delta_1 = \theta\Delta_0 = 0.5$

Iteration 1. The iterate list is now $L_1 = \{(0.5,0.5;0.5)\}$. The poll set becomes $P_1 = \{(0.5,0.5)+(0.5,0),(0.5,0.5)+(0,0.5),(0.5,0.5)+(-0.5,0),(0.5,0.5)+(0,-0.5)\} = \{(1,0.5),(0.5,1),(0,0.5),(0.5,0)\}$. In this case all the points are feasible, thus

$$L_{add} = \{((1,0.5);0.5),((0.5,1);0.5),((0,0.5);0.5),((0.5,0);0.5)\}$$

The nondominated points are filtered from $L_1 \cup L_{add}$ using the function filter, giving as result

$$L_{filter} = \{((0,0.5);0.5),((0.5,0);0.5)\}$$

Therefore L_{trial} will coincide with L_{filter}. Since there were changes in the list the iteration is declared successful and the step size is maintained $L_2 = L_{trial} = L_{filter}$.

Iteration 2. At the beginning of the new iteration the algorithm selects a point from the two stored in L_2. Assume the point $(0,0.5)$ is selected, the poll set is $P_2 = \{(0,1),(0.5,0.5),(-0.5,0.5),(0,0)\}$. In this case not all the points are feasible, and the adding list becomes

$$L_{add} = \{((0,1);0.5),((0.5,0.5);0.5),((0,0);0.5)\}$$

The non dominated points are once again filtered from $L_2 \cup L_{add}$, just one of the poll points was nondominated, thus the iteration is successful, the step size is maintained, and the new list is

$$L_3 = L_{trial} = L_{filter}\{((0,0);0.5),((0.5,0);0.5)\}$$

As we have seen the iteration list may change its size at each iteration, showing a variable number of points in the temporary Pareto front in function of the iteration unlikely the population based stochastic algorithms where this number is fixed.

Usually the number of points steadily grows increasing the number of function evaluation, as we can see in the next figures where we represent both the Pareto set and the Pareto front of ZDT2 with 20 and 100 function evaluations.

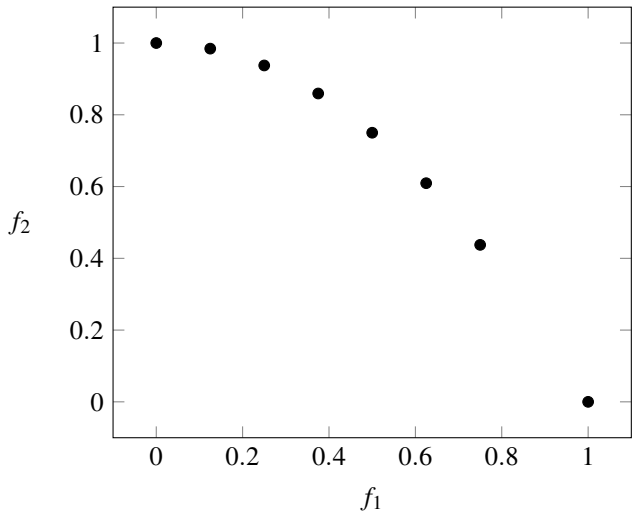

Figure 2.4: ZDT2 Pareto front with 20 function evaluations

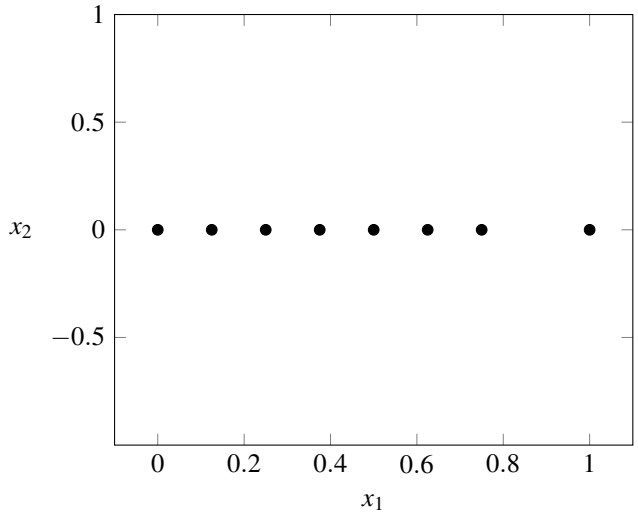

Figure 2.5: ZDT2 Pareto set with 20 function evaluations

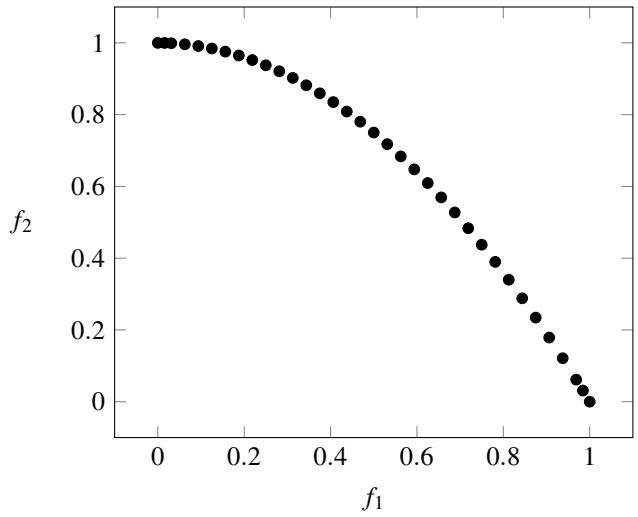

Figure 2.6: ZDT2 Pareto front with 100 function evaluations

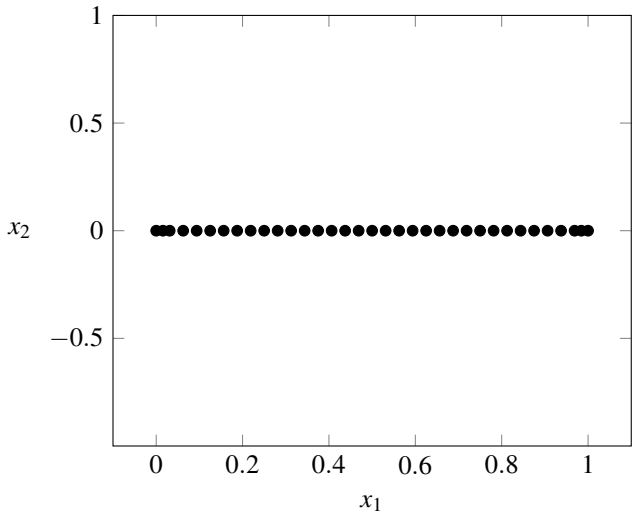

Figure 2.7: ZDT2 Pareto set with 100 function evaluations

2.3 Convergence analysis for the Single-Objective case

The purpose of this section is to prove that the Pattern Search method converges to a stationary point [6] [7] . The main requirements for this purpose regard the exploratory moves, that is the poll step procedure, the rule for the step size Δ update, and the set of directions included in the matrix $\boldsymbol{D}$.

The matrix $\boldsymbol{D}_k$ is defined in the following way

$$\boldsymbol{D}_k = \boldsymbol{B}\boldsymbol{C}_k \tag{2.3}$$

Where $\boldsymbol{B} \in \mathbb{R}^{n\times n}$ is the basis matrix, and $\boldsymbol{C}_k \in \mathbb{Z}^{n\times p}$, $p > 2n$, is the generatrix matrix, which can be partitioned into two components $\boldsymbol{C}_k = [\Gamma_k \quad \boldsymbol{A}_k]$, where $\boldsymbol{A}_k$ contains at least one column made by all zeros.

The exploratory moves must follow the two conditions

$$\begin{aligned} &1) \quad \mathbf{s} \in \Delta_k[\boldsymbol{B}\Gamma_k \quad \boldsymbol{B}\boldsymbol{A}_k] \\ &2) \quad if\, \min\{f(\boldsymbol{x}_k+\boldsymbol{y}), \boldsymbol{y} \in \Delta_k\, \mathbf{B}\Gamma_k\} < f(\boldsymbol{x}) \rightarrow f(\boldsymbol{x}_k+\mathbf{s}_k) < f(\boldsymbol{x}_k) \end{aligned} \tag{2.4}$$

While the step size must be updated according to the equation 2.1.

The set of directions used for the PS method has to be able to give an information comparable to the one we could have computing the gradient of the function, that is a proper idea of the local trend around the point. This condition is formalized below.

Condition 1
Given a points sequence $\{\boldsymbol{x}_k\}$, the sequences of directions $\{\boldsymbol{p}_k^i\}$, $i = 1, ..., r$ are limited and such that

$$\lim_{k\to\infty} \|\nabla f(\boldsymbol{x}_k)\| = 0 \Longleftrightarrow \lim_{k\to\infty} \sum_{i=1}^{r} \min\{0, \nabla f(\boldsymbol{x}_k)^T \boldsymbol{p}_k^i\} \tag{2.5}$$

That simply means the directions set must be such that all the directional derivatives along the search directions can assume non negative values if and just if the algorithm is close to a stationary point. If this condition was not satisfied it would not be possible to ensure the existence of at least one descent direction, and then to ensure the convergence of the method.

There are several classes of directions that satisfy the last condition, an example of particular relevance is the set of directions spanning positively $\mathbb{R}^n$, which must be used in the algorithm, and is defined below.

Definition 5. The directions $\{\boldsymbol{p}^1,...,\boldsymbol{p}^r\}$ span positively $\mathbb{R}^n$ if given a point $\boldsymbol{y} \in \mathbb{R}^n$, it can be expressed as a linear combination with non negative coefficients of $\{\boldsymbol{p}^1,...,\boldsymbol{p}^r\}$

$$\sum_{i=1}^{n} \beta_i \boldsymbol{p}^i \quad \beta_i \geq 0, \quad i = 1,...,r \tag{2.6}$$

The cardinality of these sets of directions is at least $n+1$ in $\mathbb{R}^n$, that is also the minimum size of matrix Γ.

The figure 2.8 [6] is an example of poll steps generated with the same step size and a matrix $\Gamma = [\mathbf{I} - \mathbf{I}]$ with four search directions in $\mathbb{R}^2$, while the figure 2.9 an example [9] of search step (blue circles) in a lattice.

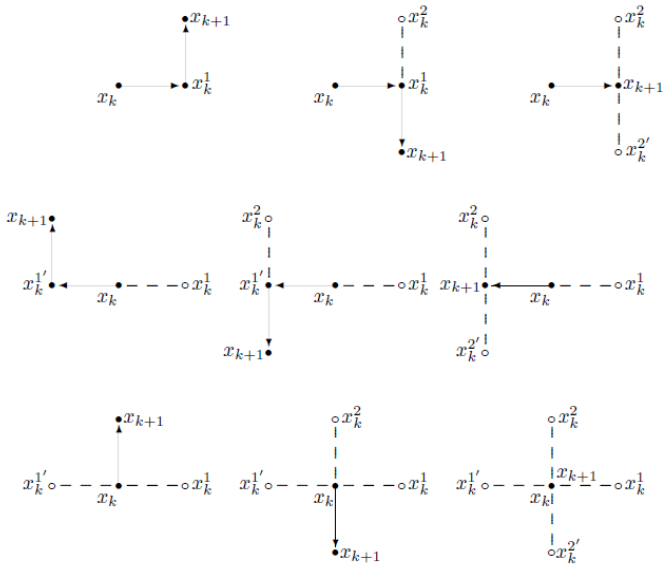

Figure 2.8: Possible poll steps in $\mathbb{R}^2$

To prove the global convergence of the PS method, that is the convergence from an arbitrary starting point, the general theorem of the global convergence will be illustrated (theorem 1), but first we define the two assumptions required.

Assumption 1. *The function $f : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function.*

Assumption 2. *Given the point $\boldsymbol{x}_0 \in \mathbb{R}^n$, the level set $l_0 = \{\boldsymbol{x} \in \mathbb{R}^n : f(\boldsymbol{x}) \leq f(\boldsymbol{x}_0)\}$ is compact.*

Theorem 1. *Let $\{\boldsymbol{x}_k\}$ be a sequence of points and $\{\boldsymbol{p}_k^i\}$ with $i = 1,...,r$ some sequences of directions, and the assumptions 1 and 2 be verified. If the following conditions are verified:*

1) $f(\boldsymbol{x}_{k+1}) \leq f(\boldsymbol{x}_k)$

2) the sequences $\{\boldsymbol{p}_k^i\}$ satisfy the condition 1 on the search directions.

3) some sequences of points $\{\boldsymbol{y}_k^i\}$ and of positive scalars $\{\varepsilon_k^i\}$ exist, and are such that:

$$f(\boldsymbol{y}_k^i + \varepsilon_k^i \boldsymbol{p}_k) \geq f(\boldsymbol{y}_k^i) - o(\varepsilon_k^i)$$
$$\lim_{k\to\infty} \varepsilon_k^i = 0$$
$$\lim_{k\to\infty} \|\boldsymbol{x}_k - \boldsymbol{y}_k^i\| = 0$$

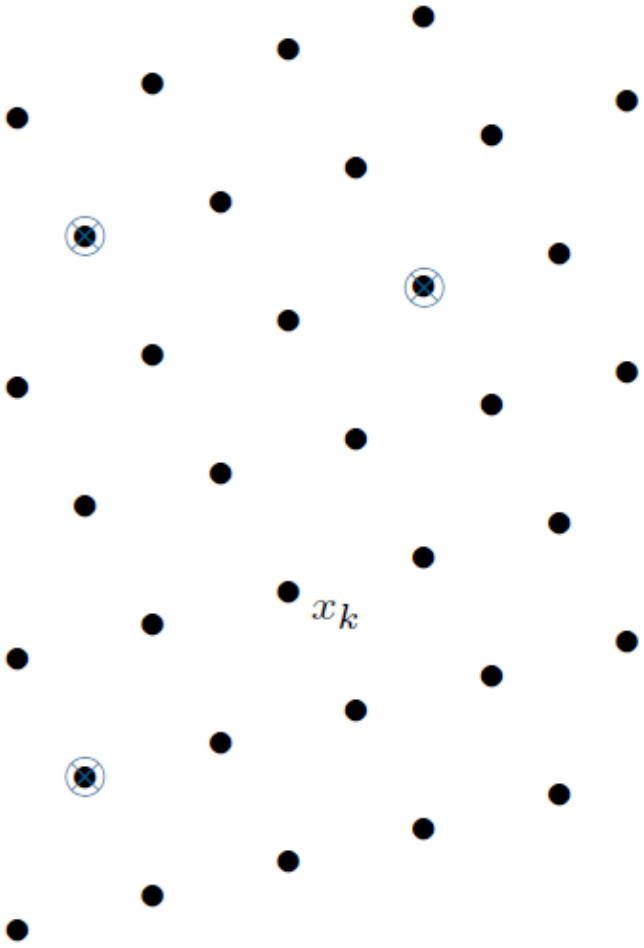

Figure 2.9: Search step and lattice

Then

$$\lim_{k\to\infty}\|\nabla f(\boldsymbol{x}_k)\| = 0 \tag{2.7}$$

To prove this theorem for the PS method the first step is to show, as we do in the next theorem, the existence of a sequence of scalar tending to zero for $k \to \infty$, that is in our case the sequence of the step size Δ_k.

Theorem 2. *Let the assumption 2 be verified. Then, if the exploratory moves follow the procedure described in the equation 2.4, and the step size is updated according to the equation 2.1, we can find an infinite set of index K such that*

$$\lim_{k\to\infty,k\in K}\Delta_k = 0 \tag{2.8}$$

The algorithm generates a sequence of points $\{\boldsymbol{x}_k\}$ such that $f(\boldsymbol{x}_{k+1}) \leq f(\boldsymbol{x}_k)$, therefore $\{\boldsymbol{x}_k\} \in l_0$ which is a compact set for the assumption 2. Furthermore, for every k the current iteration belongs to an entire lattice $M_k = \{\boldsymbol{x}_0 + \Delta_k \boldsymbol{D}_k z, z \in \mathbb{N}\}$, with $\boldsymbol{x}_0 \in \mathbb{R}^n$. The intersection of this lattice with the compact set l_0 is made by a finite set of points. That means there must be at least one point $\boldsymbol{x}_*$ such that $\boldsymbol{x}_k = \boldsymbol{x}_*$ for an infinite number of times. In the PS method is impossible to come back to a point that was already assessed in the previous iterations, but what can happened is that $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ when the iteration is not successful. Therefore, a subsequence $\{\boldsymbol{x}_k\}_K$ exists, such that for every $k \in K$, $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$, $\boldsymbol{x}_k = \boldsymbol{x}_*$ and $s_k = 0$. For the updating rule of the step size described in the equation 2.1 we have that

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

This result is crucial for the global convergence of the PS method which is proved in the next theorem.

Theorem 3. *Let $\{\boldsymbol{x}_k\}$ be the sequence generated by the algorithm Pattern search, and the directions $\{\boldsymbol{p}_k^i\}$ that correspond to the vectors column of the matrix $\boldsymbol{B}\Gamma_k$ satisfy the condition A for $i = 1, ..., 2n$. Then, if the exploratory moves procedure is done as in the equation 2.4 and the step size is updated according to the equation 2.1, an infinite set of index K exists, such that*

$$\lim_{k\to\infty,k\in K}\|\nabla f(\boldsymbol{x}_k)\| = 0$$

We need to prove the three requirements of theorem 1.

The second condition is verified for the hypothesis on the directions search, while the fact that $f(\boldsymbol{x}_{k+1}) \leq f(\boldsymbol{x}_k)$ is always true because the algorithm does not accept iterations that do not improve the objective function. Furthermore, in the theorem 2 was shown that exists an infinite set of index K such that

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

That implies the existence of a subsequence $\{\Delta_k\}_{K_1}$ with $K_1 \subseteq K$ such that for every $k \in K_1$, $\Delta_{k+1} < \Delta_k$. For the machinery of the algorithm for every $k \in K_1$ the previous iteration was unsuccessful, therefore we have

$$f(\boldsymbol{x}_k + \frac{\Delta_k \boldsymbol{p}_k^i}{\theta}) \geq f(\boldsymbol{x}_k) \quad with \quad \Delta_{k-1} = \frac{\Delta_k}{\theta}$$

Finally if we put for every $k \in K$

$$\begin{cases} \boldsymbol{\varepsilon}_k^i = \frac{\Delta_k}{\theta} \\ o(\boldsymbol{\varepsilon}_k^i) = 0 \\ \boldsymbol{y}_k^i = \boldsymbol{x}_k \end{cases}$$

Also the third condition is verified, that means the PS method converges to a stationary point

$$\lim_{k\to\infty,k\in K}\|\nabla f(\boldsymbol{x}_k)\| = 0$$

Under some more restrictive hypothesis described below on the exploratory moves procedure and the step size update is possible to prove the previous theorem in its stronger version.

$$\begin{aligned} &1) \quad \boldsymbol{s} \in \Delta_k[\boldsymbol{B}\Gamma_k \quad \boldsymbol{B}\boldsymbol{A}_k] \\ &2) \quad if \min\{f(\boldsymbol{x}_k+\boldsymbol{y}),\boldsymbol{y} \in \Delta_k\boldsymbol{B}\Gamma_k\} < f(\boldsymbol{x}) \rightarrow f(\boldsymbol{x}_k+\boldsymbol{s}_k) \leq \min\{f(\boldsymbol{x}_k+\boldsymbol{y}),\boldsymbol{y} \in \Delta_k\boldsymbol{B}\Gamma_k\} \end{aligned} \tag{2.9}$$

$$\Delta_{k+1} = \begin{cases} \theta\Delta_k, & \text{if the iteration was not successful} \\ \Delta_k, & \text{if the iteration was successful} \end{cases}$$

With $0 < \theta < 1$, and $\lambda = 1$ we ensure that the step size Δ_k is decreasing. Therefore, is possible to prove the previous theorem without the need to identify an infinite set of index K. In fact for the step size we get

$$\lim_{k\to\infty} \Delta_k = 0 \tag{2.10}$$

From which descends

$$\lim_{k\to\infty} \|\nabla f(\boldsymbol{x}_k)\| = 0 \tag{2.11}$$

As far we have proved the convergence to a stationary point using just the poll step, but that does not imply the global convergence to a global minimum because the poll step by itself is not able to define an ergodic process. For that purpose the use of the search step is necessary, being it able to cross all the points of the domain. Higher will be the number of points considered in the search step higher will be the probability of convergence to a global minimum of the method which uses both the poll and the search step.

2.4 Convergence analysis for the Multi-Objective case

Many of the results achieved in the previous section are still valid and have great importance in the convergence analysis for the multi-objective case, but further definitions and concepts are required. In fact the concept of critical point has to be formalized according to the definition of Pareto dominance, because the simple condition on the gradient functions is no more sufficient.

While some definitions do not need change or generalization, like the description of the exploratory moves or the rule for the step size update, we need to generalize some elements already seen previously, like the compactness of the set l_0, and the step size Δ_k convergence to zero.

Assumption 3. *Given the point $\boldsymbol{x}_0 \in \mathbb{R}^n$, the level set $L(\boldsymbol{x}_0) = \bigcup_{j=1}^{m} l_j(\boldsymbol{x}_0)$ is compact, where $l_j(\boldsymbol{x}_0) = \{\boldsymbol{x} \in \mathbb{R}^n : f_j(\boldsymbol{x}) \leq f_j(\boldsymbol{x}_0)\}$, $j = 1,...,m$. All the objective functions f_j are bounded above and below in $L(\boldsymbol{x}_0)$.*

We need to extend the compactness assumption to the union of the level sets, so that we can show that the step size converges to zero as in the previous case. The intersection of a compact set $L(\boldsymbol{x}_0)$ with an integer lattice is finite, as well as the number of points we can add to the iterate list. Therefore, the number of successful iterations is limited, and being the cycle among these points impossible, the step size must go to zero.

For the convergence analysis [8] we need to analyze the behavior of the algorithm at limit points of sequences of unsuccessful iterates, for that purpose the concepts of refining subsequences and directions are defined, and will be used in the proof of the convergence theorem.

Definition 6. A subsequence $\{\boldsymbol{x}_k\}_{k\in K}$ of iterates corresponding to unsuccessful poll steps is said to be a refining subsequence if the iterates of step size $\{\Delta_k\}_{k\in K}$ go to zero.

The existence of at least a convergent refining subsequence is ensured under our assumptions.

Definition 7. Let $\boldsymbol{x}_*$ be the limit point of a convergent refining subsequence. If the limit exists $\lim_{k\in K'} \frac{\boldsymbol{d}_k}{\|\boldsymbol{d}_k\|}$, where $K' \subseteq K$ and $\boldsymbol{d}_k \in \boldsymbol{D}_k$, and if $\boldsymbol{x}_k + \Delta_k \boldsymbol{d}_k \in \Omega$, for sufficiently large $k \in K'$, then this limit is said to be a refining direction for $\boldsymbol{x}_*$.

Furthermore, the notion of critical point has to be changed. While in the single objective case it has the property that moving away from it in any feasible directions the objective function is not improved, in the multi-objective case moving away from it a dominating point is reached, that means at least one of the objective functions gets worse. This idea is formalized through the concepts of Clarke tangent vector for the feasible directions, and of Pareto-Clarke critical point for the points on the Pareto front.

Definition 8. A vector $\boldsymbol{d} \in \mathbb{R}^n$ is said to be a Clarke tangent vector to the set $\Omega \subseteq \mathbb{R}^n$ at the point $\boldsymbol{x}$ in the closure of Ω if for every sequence $\{\boldsymbol{y}_k\}$ of elements of Ω that converges to $\boldsymbol{x}$ and for every sequence of positive real numbers $\{t_k\}$ converging to zero, there exists a sequence of vectors $\{\boldsymbol{w}_k\}$ converging to $\boldsymbol{d}$ such that $\boldsymbol{y}_k + t_k \boldsymbol{w}_k \in \Omega$.

The concept of tangent cone is generalized above with the concept of Clarke tangent cone, which is the set $T_\Omega^{Cl}(\boldsymbol{x})$ of all Clarke tangent vectors to Ω at $\boldsymbol{x}$. The hypertangent cone $T_\Omega^H(\boldsymbol{x})$ is the interior of the Clarke tangent cone, when the interior is not empty, obviously the Clarke tangent cone is the closure of the hypertangent cone.

The next one is the last definition required, the Clarke-Jahn generalized derivatives of a function represent the upper bound value of the directional derivatives on the directions $\boldsymbol{d}$ of the hypertangent cone in a point $\boldsymbol{x}$.

Definition 9. If we assume that $\boldsymbol{F}(\boldsymbol{x})$ is Lipschitz continuous near $\boldsymbol{x}$, that is each $f_i(\boldsymbol{x})$, for $i = 1, ..., m$, is continuous near $\boldsymbol{x}$, we can define the Clarke-Jahn generalized derivatives of the individual functions along the directions $\boldsymbol{d}$ in the hypertangent cone to Ω at $\boldsymbol{x}$

$$f_i^\circ(\boldsymbol{x};\boldsymbol{d}) = \limsup_{\boldsymbol{x}'\to\boldsymbol{x},t\to 0^+} \frac{f_i(\boldsymbol{x}'+t\boldsymbol{d}) - f_i(\boldsymbol{x}')}{t} \quad with \quad \boldsymbol{x}' \in \Omega, \boldsymbol{x}'+t\boldsymbol{d} \in \Omega \tag{2.12}$$

With this definition is possible now to introduce the concept of Pareto-Clarke critical point, that is the crucial point to prove the theorem for the global convergence analysis.

Definition 10. Let $\boldsymbol{F}$ be Lipschitz continuous near a point $\boldsymbol{x}_* \in \Omega$. Then $\boldsymbol{x}_*$ is a Pareto-Clarke critical point of $\boldsymbol{F}$ in Ω if, for all directions $\boldsymbol{d} \in T_\Omega^{Cl}(\boldsymbol{x}_*)$, there exists a $j = j(\boldsymbol{d}) \in \{1, ..., m\}$ such that $f_j^\circ(\boldsymbol{x}_*, \boldsymbol{d}) \geq 0$.

Therefore, the concept of Pareto-Clarke point coincides with the concept of Pareto minimizer, because there is no direction in the tangent cone which is descent for all the objective functions.

Finally we can show the convergence of the PS method for multi-objective optimization, first we prove the method produces a limit point for which there is no direction in the hypertangent cone descent for all the objective functions, eventually this result will be extended also to the directions of the tangent cone.

Theorem 4. *Consider a refining subsequence $\{\boldsymbol{x}_k\}_{k\in K}$ converging to $\boldsymbol{x}_* \in \Omega$ and a refining direction $\boldsymbol{d}$ for $\boldsymbol{x}_*$ in the hypertangent cone $T_\Omega^H(\boldsymbol{x}_*)$, then, if F is a Lipschitz continuous near $\boldsymbol{x}_*$, there exists a $j = j(\boldsymbol{d}) \in \{1, ..., m\}$ such that $f_j^\circ(\boldsymbol{x}_*, \boldsymbol{d}) \geq 0$.*

If we consider the refining direction $\boldsymbol{d} = \lim_{k\in K''} \frac{\boldsymbol{d}_k}{\|\boldsymbol{d}_k\|} \in T_\Omega^H(\boldsymbol{x}_*)$ for $\boldsymbol{x}_*$, with $\boldsymbol{d}_k \in D_k$ and $\boldsymbol{x}_k + \Delta_k \boldsymbol{d}_k \in \Omega$ for all $k \in K'' \subseteq K$.

Given the following disequation, for $j \in \{1, ..., m\}$

$$f_j^\circ(\boldsymbol{x}_*;\boldsymbol{d}) = \limsup_{\boldsymbol{x}'\to\boldsymbol{x},t\to 0^+} \frac{f_j(\boldsymbol{x}'+t\boldsymbol{d}) - f_j(\boldsymbol{x}')}{t} \geq \limsup_{k\in K''} \frac{f_j(\boldsymbol{x}_k + \Delta_k \boldsymbol{d}_k) - f_j(\boldsymbol{x}_k)}{\Delta_k \|\boldsymbol{d}_k\|}$$

Being $\{\boldsymbol{x}_k\}_{k\in K}$ a refining subsequence, for every $k \in K'' \quad \boldsymbol{x}_k + \Delta_k \boldsymbol{d}_k$, corresponding to an unsuccessful iteration, is a dominated point for the list L_k. Therefore, for each $k \in K''$ an index $j = j(k)$ function of the iteration k exists, such that

$$f_{j(k)}(\boldsymbol{x}_k + \Delta_k \boldsymbol{d}_k) - f_j(\boldsymbol{x}_k) \geq 0$$

The number of indexes j, corresponding to the number of objective functions, is a finite number, while there are infinite indexes $k \in K''$. Therefore, there is an index $j = j(\boldsymbol{d})$, function of the direction, for which exists an infinite set of indexes $k \in K''' \subseteq K''$ such that

$$f^{\circ}_{j(\boldsymbol{d})}(\boldsymbol{x}_*; \boldsymbol{d}) \geq \limsup_{k \in K'''} \frac{f_{j(\boldsymbol{d})}(\boldsymbol{x}_k + \Delta_k \boldsymbol{d}_k) - f_{j(\boldsymbol{d})}(\boldsymbol{x}_k)}{\Delta_k \|\boldsymbol{d}_k\|} \geq 0$$

In order to extend this result to the directions of the Clarke tangent cone, and therefore to prove that $\boldsymbol{x}_*$ is a Pareto minimizer, we need to impose the density of the set of refining directions. The ultimate theorem is illustrated below.

Theorem 5. *Consider a refining subsequence $\{\boldsymbol{x}_k\}_{k \in K}$ converging to $\boldsymbol{x}_* \in \Omega$, assuming that $\boldsymbol{F}$ is Lipschitz continuous near $\boldsymbol{x}_*$, if the set of refining directions for $\boldsymbol{x}_*$ is dense in $T^{Cl}_{\Omega}(\boldsymbol{x}_*)$ and the hypertangent cone $T^{H}_{\Omega}(\boldsymbol{x}_*)$ is non empty, then $\boldsymbol{x}_*$ is a Pareto-Clarke critical point.*

In the multiobjective case, unlike the single objective, the poll step by itself ensures the convergence to a Pareto minimizer, therefore the search step is not necessary, but can be implemented in order to disseminate the search within the domain.

2.5 Pattern search with random generated polling

As it was explained in Chapter 2.3 the polling matrix must have at least $n+1$ vectors in $\mathbb{R}^n$ to ensure the convergence, because at least for one direction the function is decreasing. In order to cut the computational cost is possible to reduce the number of vectors randomly generating the polling directions and without imposing the positive spanning property. This alternative mechanism increases the number of unsuccessful iteration, therefore is necessary to introduce the notion of probabilistically descent set of directions because at least one of them is needed to make an acute angle with the negative gradient with a sufficient probability value uniformly across the iterations. We can prove that this probabilistic set of directions ensures the convergence with an high probability, but the expansion parameter must be $\lambda > 1$ [10].

In particular for the practical implementations will be used a polling matrix of random vectors uniformly distributed in the unit sphere, with the parameters $\theta = 0.5$ and $\lambda = 2$. In this way the algorithm loses its deterministic features, but still have solid theoretical backgrounds unlike the traditional stochastic methods.

Chapter 3

Stochastic methods for multiobjective optimization

3.1 Overview

Stochastic optimization algorithms have been increasingly more popular in the last two decades, playing a major role in solving optimization problems in industrial engineering. These algorithms generate random variables, may have random objectives, problems and constraints, therefore, unlike the deterministic algorithms, each run is different from the other being the solving process defined random.

Furthermore, they do not need information on the gradient, solve the problems without the need to explore the search space thoroughly, and incorporate naturally noisy functions while standard deterministic methods require perfect information about the objective function, that is why they are often preferred to solve high dimension problems despite the lack of guarantees about the convergence to the global optimum.

The simplest methods for stochastic optimization are the Direct Random Search methods, they are easy to code, only need information about the objective function, and may be an interesting example to explain the main feature of the working principle of these random processes given their simplicity, that is why we decide to examine in detail the Blind Random Search [11]. This algorithm defines a sampling process for the variable x of the objective function f which does not take into account and is not influenced by the previous samples, thus the information gathered in the search process are not used to adapt the search strategy. This method is the only one in stochastic optimization where there is no adjustable coefficients that need to be tuned by the user, that is why the recursive implementation is very simple, and is presented below.

$$min_{\boldsymbol{x}\in X} f(\boldsymbol{x}) \qquad X \subseteq \mathbb{R}^n \qquad f\colon \mathbb{R}^n \to \mathbb{R}$$

0) Initialization: choose an initial value of $\boldsymbol{x}$, say $\boldsymbol{x}_0 \in X$, either randomly or deterministically. If random, usually a uniform distribution on X is used. Calculate $f(\boldsymbol{x}_0)$. Set $k = 0$.

1) Generate a new independent value $\boldsymbol{x}_{new}(k+1) \in X$, according to the chosen probability distribution. If $f(\boldsymbol{x}_{new}(k+1)) < f(\boldsymbol{x}_k)$, set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_{new}(k+1)$. Else, take $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$.

2) Stop if the maximum number of evaluations has been reached or the user is otherwise satisfied with the current estimate for $\boldsymbol{x}$ via appropriate stopping criteria; else, return to 1) with the new k set to the former $k+1$.

The convergence of the algorithm is almost sure under general conditions, but the convergence rate is very low also for domains of size not very large, given the curse of dimensionality. A more sophisticated algorithm which takes into account in the sampling the position of the current best estimate is the Localized Random Search, which is able to exploit the information obtained about the shape of the

function. The machinery of this algorithm is described below.

0) Initialization: pick an initial guess $\boldsymbol{x}_0 \in X$ either randomly or with prior information. Set $k = 0$.

1) Generate an independent random vector $\boldsymbol{d}_k \in \mathbb{R}^n$ and add it to the current $\boldsymbol{x}$ value. Check if $\boldsymbol{x}_k + \boldsymbol{d}_k \in X$. If $\boldsymbol{x}_k + \boldsymbol{d}_k \notin X$, generate a new $\boldsymbol{d}_k$ and repeat or, alternatively, move $\boldsymbol{x}_k + \boldsymbol{d}_k$ to the nearest valid point within X. Let $\boldsymbol{x}_{k+1} = \boldsymbol{x}_{new}(k+1)$ equal $\boldsymbol{x}_k + \boldsymbol{d}_k \in X$ or the aforementioned nearest valid point in X.

2) If $f(\boldsymbol{x}_{new}(k+1)) < L(\boldsymbol{x}_k)$, set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_{new}(k+1)$, else set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$.

3) Stop if the maximum number of evaluations has been reached or the user is otherwise satisfied with the current estimate for $\boldsymbol{x}$ via appropriate stopping criteria; else, return to Step 1 with the new set to the former $k+1$.

In this case we can tune the method setting the distribution of the deviation vector, possibly reducing the variability of $\boldsymbol{d}_k$ as k increases.

Given the adaptation of the search process due to the exploitation component the Localized Random Search has an higher convergence rate than the Blind Random Search.

Generally a stochastic algorithm is made of a random or an opportunistic initialization if some data are available a priori, a search with a probabilistic component which may or not be adjusted, a comparison in order to see if an improvement in the solution was found, and finally the check of the stopping condition. The crucial component is the search and the way it is done, because higher is the exploitation of the previous samplings done, higher is the rate of convergence, that is why the metaheuristic procedures have gained a lot of importance.

Meta means beyond or of an higher level, while heuristic means to solve by trial and error, thus by experience, therefore we refer to metaheuristic as a master strategy that produces solutions beyond the ones usually found in the search for optimality using and driving other heuristics.

The two fundamental components of metaheuristic algorithms are exploitation and exploration, the first has the duty to localize the quest in the regions where a good solution was found, the latter generates some random and sparse trial solutions in order to explore the search space, a good balance of this components improves the convergence rate. Also for them there is no guarantee that an optimal solution can be reached, but a good combination of the major components usually ensures the global convergence.

Often these methods are naturally inspired, for instance the Particle Swarm Optimization (PSO), which belongs to the Swarm Intelligence field, simulates the movements in a bird flock, and the Differential Evolution, which belongs to the field of evolutionary computation, is inspired by biological evolution simulating the natural selection mechanism in order to improve the solutions computed.

These two algorithms are already very popular and often used for engineering optimization, that is why they are taken as references for the performance comparison with the Pattern Search and the innovative algorithms developed in Chapter 4.

In this chapter these two methods are described in details.

3.2 Particle Swarm Optimization

Particle Swarm Optimization is a swarm-based evolutionary algorithm introduced by Eberhart and Kennedy [12], it is a population based evolutionary algorithm and is inspired by natural concepts like bird flocking or fish schooling, where each particles or candidate solutions explore the environment for food or the search space for a good solution, exchange information by acoustical means or exchange information by sharing position of promising location. Therefore, each particle moves in the search space with a dynamic adjustment of the velocity which depends on the experience of the particle and the experiences of the other particles exploring the search space. This machinery was inspired by the flocking behavior, given the possibility to communicate and exchanging information, this model is controlled by the following three basic rules.

- Separation, avoid crowding neighbors (short range repulsion)
- Alignment, steer towards average heading of neighbors

- Cohesion, steer towards average position of neighbors (long range attraction)

The mathematical formulation which defines the motion of the N particles is described by the following equations, updating the velocity $\boldsymbol{v}_i$ and the position $\boldsymbol{x}_i$ of each particle i.

$$\boldsymbol{v}_i(t+1) = w\boldsymbol{v}_i(t) + c_1 ud[\boldsymbol{p}_i(t) - \boldsymbol{x}_i(t)] + c_2 Ud[\boldsymbol{p}_g(t) - \boldsymbol{x}_i(t)] \tag{3.1}$$

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \Delta t \boldsymbol{v}_i(t+1) \tag{3.2}$$

The positive constants c_1 and c_2 are called acceleration coefficients, ud and Ud are random functions in the range [0,1], and w is the inertia coefficient, while $\boldsymbol{p}_i$ represents the personal best of the particle i and $\boldsymbol{p}_g$ the global best position ever.

In the equation 3.1 we can identify the sum of three components, the inertial component $w\boldsymbol{v}_i(t)$ proportional to the weight coefficient w, the cognitive element $c_1 ud[\boldsymbol{p}_i(t) - \boldsymbol{x}_i(t)]$ which depends on the information gathered by the single particle i, and the social element $c_2 Ud[\boldsymbol{p}_g(t) - \boldsymbol{x}_i(t)]$ which uses the best value found by all the particles.

The movement of particles [13] is illustrated in figure 3.1

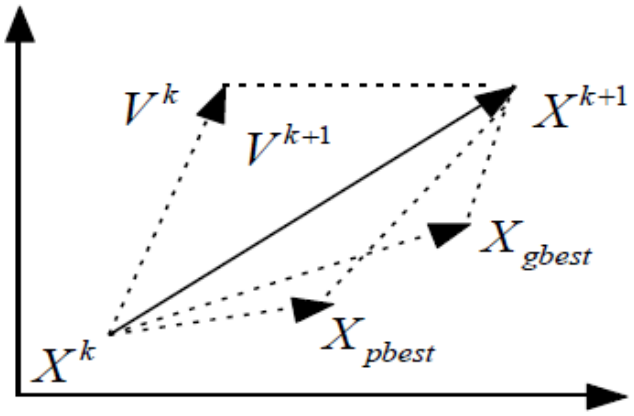

Figure 3.1: Particle motion in the swarm

The tuning of the parameters can influence the search process, an high value of the inertia weight w facilitates global exploration, while a low one local exploitation. The factor c_1 multiplies the cognitive component, that is the exploitation of its own experience, therefore is named individual factor, c_2 the societal factor influences the social component that is the information sharing and cooperation among particles.

Many efforts have been made to find the parameters values which optimize the performance of the algorithm, the numerical experiments [14] show that the inertia should be linearly decreasing with the iterations between 1 and 0.3, the populations size in the range between 20 and 30, and the individual and social factor linked by the following equation.

$$c_2 = c_1 + \alpha \tag{3.3}$$

With α between 1 and 1.5.

At last, we can discuss the convergence of PSO [15], knowing that we can only ensure the convergence to the best position visited by all the particles of the swarm. In order to assure convergence to the local or global optimum we need the two following conditions.

- The global best solution found at the time $t+1$ can be no worse than the global best found at t. This is the monotonic condition
- The algorithm must be able to generate with a nonzero probability a solution in the neighborhood of the optimum from any solution of the search space

Despite the fact that PSO satisfies the monotonic condition, once the algorithm reaches the state where x is equal to the personal best which in turn is equal to the global best for all the particles, no further improvement will be made. Since this state may be reached before the global best achieves a minimum, whether be local or global.

Therefore, the algorithm is said to converge prematurely, this means it is not a local nor a global search algorithm since there is no guarantee of convergence to a local or a global minimum from an arbitrary initial state.

3.3 Differential Evolution

The Differential Evolution is a stochastic population based algorithm introduced by Storn and Price [16]. It is an evolutionary algorithm which tries to improve a candidate solution operating the procedure of figure 3.2 [17]. Consider an optimization problem with a function of D real parameters, a population size equal to N. The parameters vector is $x_{i,G}$, where G is the generation number.

$$\boldsymbol{x}_{i,G} = [x_{1,i,G}, x_{2,i,G}, ..., x_{D,i,G}] \tag{3.4}$$

with $i = 1, 2, ..., N$

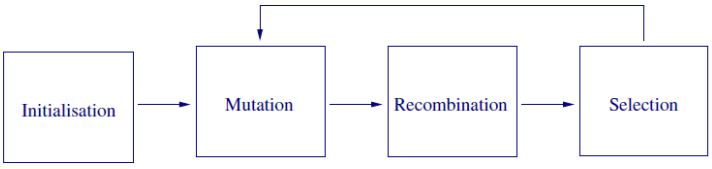

Figure 3.2: DE procedure

During the initialisation, given the lower and upper bounds for each parameter x_j^L and x_j^U, we select randomly and uniformly in the domain the initial parameter.

$$x_j^L \leq x_{j,i,1} \leq x_j^U \tag{3.5}$$

The mutation expands the search space on each of the N parameter vectors, adding a difference vector. We can select randomly three vectors $\boldsymbol{x}_{r1,G}$ $\boldsymbol{x}_{r2,G}$ and $\boldsymbol{x}_{r3,G}$, with different values for the three indexes $r1$ $r2$ and $r3$, and defining the donor vector as the weighted difference of two of the vectors to the third through the mutation factor MF.

$$\boldsymbol{v}_{i,G+1} = \boldsymbol{x}_{r1,G} + MF(\boldsymbol{x}_{r2,G} - \boldsymbol{x}_{r3,G}) \tag{3.6}$$

Where the mutation factor F is a constant in the range [0,2]. Many alternative options are possible for this step, like choosing the best vector of parameters instead of $\boldsymbol{x}_{r1,G}$, or using more vectors for the weighted sum.

The recombination mix successful solutions from the previous generation with current donors, the trial vector $\boldsymbol{u}_{i,G+1}$ is developed from elements of the target and donor vectors. We define a crossover ratio CR, that is the probability the elements of the donor vector enter the trial vector.

$$u_{j,i,G+1} = \begin{cases} v_{j,i,G+1}, & \text{if} \quad rand_{j,i} \leq CR \quad \text{or} \quad j = I_{rand} \\ x_{j,i,G}, & \text{else} \end{cases} \tag{3.7}$$

Where I_{rand} is a random integer from [1,2,...,D].

Finally we compare the target vector and the trial vector, the best, the one for which the target function assumes the lowest value, passes to the next generation.

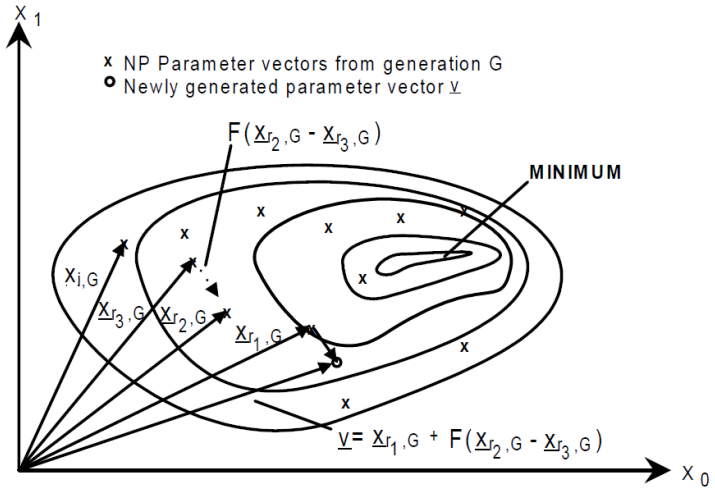

Figure 3.3: Generation of the donor vector $\boldsymbol{v}$ in a two dimensional example with the objective function showing its contour lines

$$\boldsymbol{x}_{i,G+1} = \begin{cases} \boldsymbol{u}_{i,G+1}, & \text{if} \quad f(\boldsymbol{u}_{i,G+1}) \leq f(\boldsymbol{x}_{i,G}) \\ \boldsymbol{x}_{i,G}, & \text{otherwise} \end{cases} \tag{3.8}$$

Afterwards, the three steps, mutation, recombination and selection, continue until the condition on the stopping criterion is satisfied.

The DE method has been shown to be effective on a large range of optimization problems, and more accurate than many other methods.

However there is no proof of convergence.

3.4 PSO and DE for multiobjective optimization

3.4.1 Non dominated sorting

In solving a multiobjective problem the aim is to find a set of Pareto-optimal points instead of a single point, therefore since PSO and DE are population based methods it seems natural to use them in order to capture a number of solutions in the set simultaneously.

For the search effectiveness the nondominated sorting procedure is introduced, this is a ranking selection used to emphasize good points classifying the entire population into several fronts [18].

Furthermore, given that the population of points should capture multiple Pareto optimal solutions is important to sort each front in decreasing crowding order, in order to avoid points clustering computing all the Pareto front and enabling a better points distribution along the front.

A possibility to assess the sparseness given an individual B, and being A and C the individuals before and after B, is the definition of the crowding distance $D_c(B)$ as follows [19].

$$D_c(B) = \sum_{i=1}^{N_{obj}} |f_i(A) - f_i(C)| \tag{3.9}$$

Where $f_i(A)$ and $f_i(C)$ are the objective function values for the ith objective. For the individuals on border D_c is not defined, but is set equal to infinite to ensure their unconditional selection to the next generation to keep the wideness of the front.

The above approach just takes into account the size of the neighborhood without considering the distribution of individual B sparseness in neighborhood. Therefore, in order to avoid the elimination of some well distributed individuals and the retention of some with a poor distribution, equation 3.9 is modified, introducing the term O that is the center point between A and C.

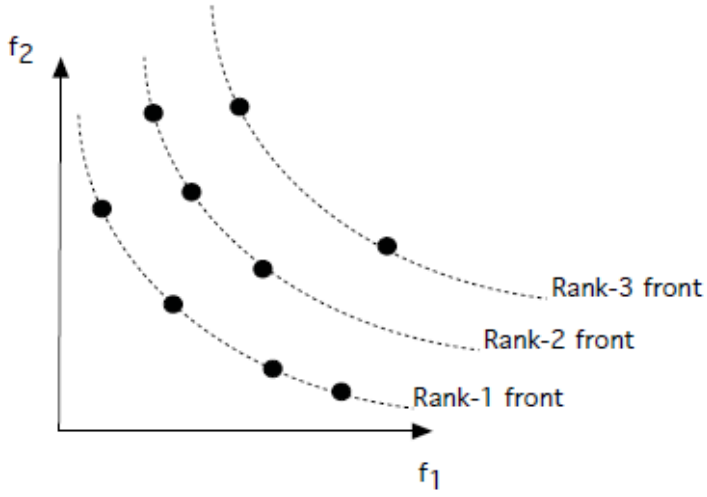

Figure 3.4: Non-dominated sorting procedure

$$D_c(B) = \sum_{i=1}^{N_{obj}} (|f_i(A) - f_i(C)| - |f_i(B) - f_i(O)|) = \tag{3.10}$$

$$\sum_{i=1}^{N_{obj}} (0.5|f_i(A) - f_i(C)| + \min[|f_i(A) - f_i(B)|, |f_i(B) - f_i(C)|])$$

Where $f_i(O)$ and $f_i(B)$ are the values of center point O and the individual B for the ith objective function.

In equation 3.10 the term $|f_i(A) - f_i(C)|$ reflects the neighborhood size, while the term $|f_i(B) - f_i(O)|$ represents the distance between the individual an the center of neighborhood, thus taking into account the distribution.

3.4.2 MOPSO

There are many different proposals for the extension of the PSO method to the multiobjective optimization, given its relative simplicity and its effectiveness in a wide variety of applications. We need to change the original scheme in order to apply the PSO strategy to the multiobjective case, where we want to achieve three main goals.

- Maximize the number of elements computed in the Pareto front
- Minimize the distance between the Pareto front computed and the real one
- Maximize the spread of the solutions computed in order to improve the distribution uniformity of the solutions found

The Multiobjective Particle Swarm Optimization [15], MOPSO, needs to find an efficient way to select the leader particles in order to give preference to nondominated solutions over the dominated ones, maintain the nondominated solutions found during the search to report solutions that are non dominated with respect to all the past populations preferably with a good distribution along the front, maintain diversity in the front to avoid clustering or convergence to a single solution. Furthermore, each particle needs a set of different leaders among which just one can be selected to update its position. This set of leaders is stored in an external archive, which can be defined, updated and bounded according to many different strategies.

The basic sequence of the MOPSO consists in the initialization of the swarm, the following set of leaders initialization with the nondominated particles from the swarm. Eventually one leader is selected for each swarm particle according to a quality measure, the particle position is evaluated and its corresponding personal best is updated, the replacement occurs when this particle is dominated or if they are both nondominated with respect to each other. Then the particles as well as the set of leaders have been updated, and finally the quality measure of the set of leaders is recalculated. The above process is repeated for a fixed number of iterations.

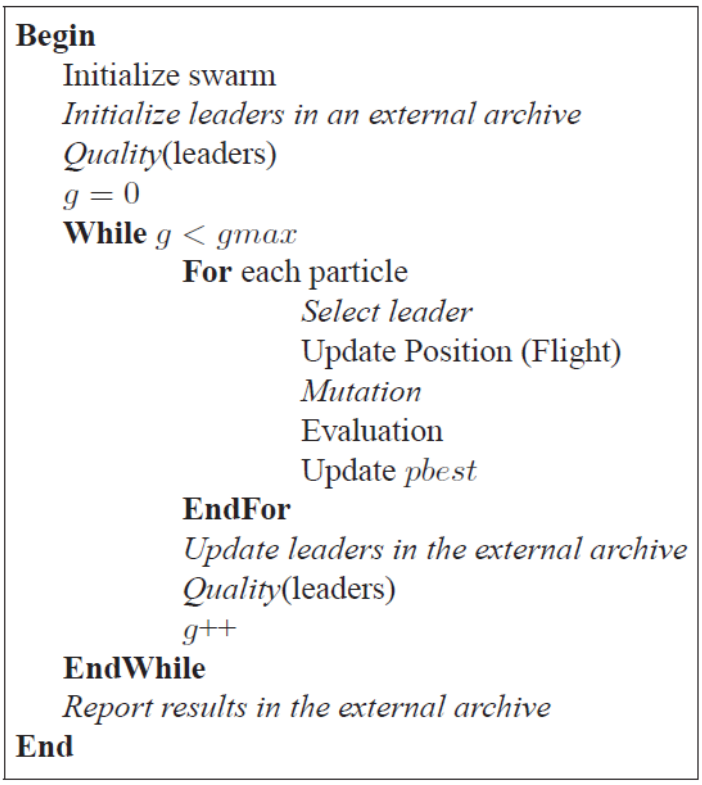

Figure 3.5: MOPSO algorithm pseudocode

Given the process above the crucial issues we must address consist in selecting and updating the leaders, whether we should do it in a random way or introducing a further criterion to promote diversity given the fact that the nondominated solutions are equally good, and in selecting the particles which should stay in the external archive from one iteration to another. The most linear approach is to introduce a quality measure which states how good is a leader, in this way we can select just one leader from the analysis of every nondominated solutions. Such feature can be implemented relating the quality to a density measure, which is referred to the closeness of the particles within the swarm.

Furthermore, we have to consider the use of an external archive in order to retain solutions, this archive enables the entrance of a solution if it is not dominated by the other points in the archive or if it dominates any of the solutions within the archive which will be eventually deleted.

The obvious outcome of this process is the size increase of the archive, which must be updated at every iteration. Therefore, this update may become very expensive from the computational point of view, and we must find a way to limit this computational cost. That is why the archive is bounded, and an additional criterion is introduced to decide which nondominated solutions should remain once the archive becomes full.

Another reason why the diversity is a fundamental feature for MOPSO is that without it the algorithm may lose its fast convergence, and have a premature convergence, that is the convergence to a local optimum. In order to avoid it we have to preserve the diversity which enables the algorithm to keep its fast convergence.

One way for the preservation is an appropriate use of the inertia weight, which controls the impact the influence of the previous history of velocities has on the current velocity, hence it also represents a trade-off between global and local exploration. A big coefficient boosts global exploration searching in new areas, while a small value facilitates local exploration in the current search area. For all these motivations the inertia weight varies during the optimization process through a linear reduction.

Furthermore, in order to avoid the stagnation of the swarm that occurs when the velocities of the particles are almost zero and lead the whole swarm to be trapped in a local optimum, the mutation of a single particle if the mutated particle becomes the new global best may be necessary, because the global best individual attracts all the members of the swarm leading it away from the current location.

For all the aims presented above the non-dominated sorting procedure is implemented in the MOPSO, in this approach once a particle has updated its position it is not compared just with the personal best of the particle, but all the personal best of the swarm and all the new positions recently obtained are stored in one set, then the procedure selects among them to conform the next swarm. Furthermore, there is a selection of the leaders from the set using a density estimation of neighborhood.

Another important issue to address in this algorithm is how avoid the out of bounds particles in order to reduce the function evaluations wasting through the clamping and reseeding processes. In figure 3.6 are represented three cases, one with the rebound on the same side of the bound, one with the passage on the other side, and the last one with a random positioning in the domain.

3.4.3 MODE

The combination of an external Pareto archive and the Differential Evolution algorithm is another possible approach to deal with multiple objectives problems. This evolutionary mechanism for multiobjective optimization [20], namely MODE (Multiobjective Differential Evolution) keeps the simplicity of DE for the single objective case, and needs to introduce only one more parameter besides the mutation factor (MF), the crossover rate (CR) and the population size (NP), that is the maximum size of the Pareto Front.

First the DE is called and the next population is determined. At each iteration the current Pareto set is compared with another already stored in an external file. Eventually a ranking operation is executed between the previous population elements and the current Pareto set, according to their strength which is defined as we have already seen with a non dominated sorting procedure.

Then, given the first NP ranked elements, two solutions are randomly chosen from the previous population, whose difference is added to the element from the rank in order to perform the mutation as defined in equation 3.6. Afterwards the mutated vector suffers a crossover (equation 3.7) in order to increase diversity.

During the initialization the first population is randomly generated and the unfeasible solutions are deleted. Then the other solutions are ranked, creating the first Pareto set. The new vectors in the current generation are compared with the external Pareto set, a ranking and selection process for the solutions is performed, and only the non dominated ones survive in the external archive.

If the number of elements is greater than the maximum size for the Pareto front defined at the beginning, a cluster reduction is performed.

The whole optimization process is described in figure 3.7.

This process runs until it fulfils one stop criterion.

Also in the multiobjective case the Differential Evolution method retains its relative simplicity due to the low number of control parameters, low computational requirements and an essential code easy to modify and update.

3.5 No Free-Lunch theorem

We can finally try to ask whether there is a best evolutionary algorithm which would always give better results for all the possible optimization problems. That would also mean that some variation and selection operators would always perform better than the others despite the given problem.

The answer to this question is that there is no best evolutionary algorithms, the result is known as "No Free-Lunch" theorem [21].

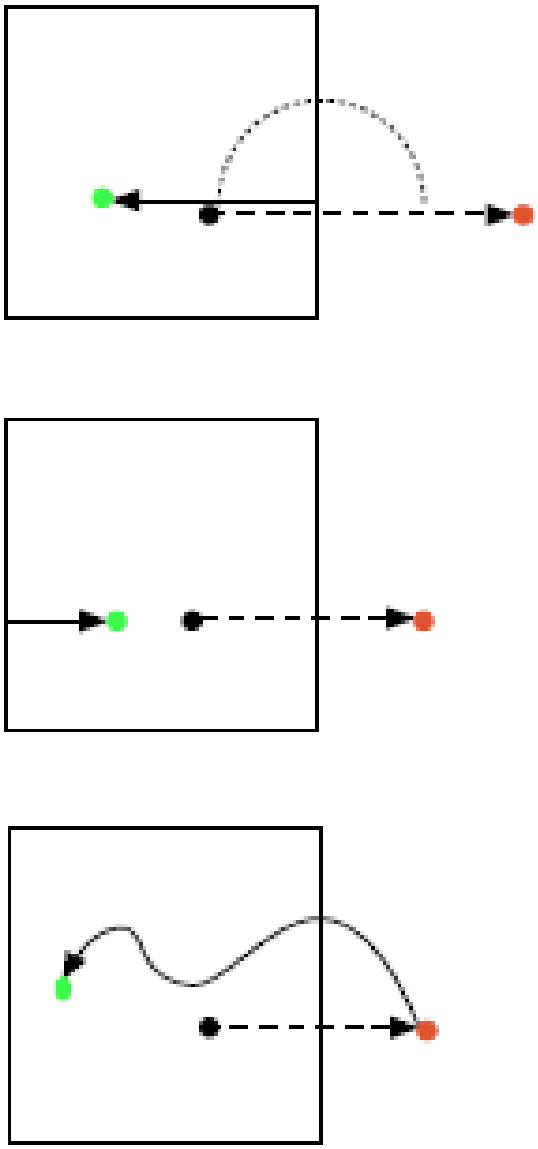

Figure 3.6: Clamping and reseeding

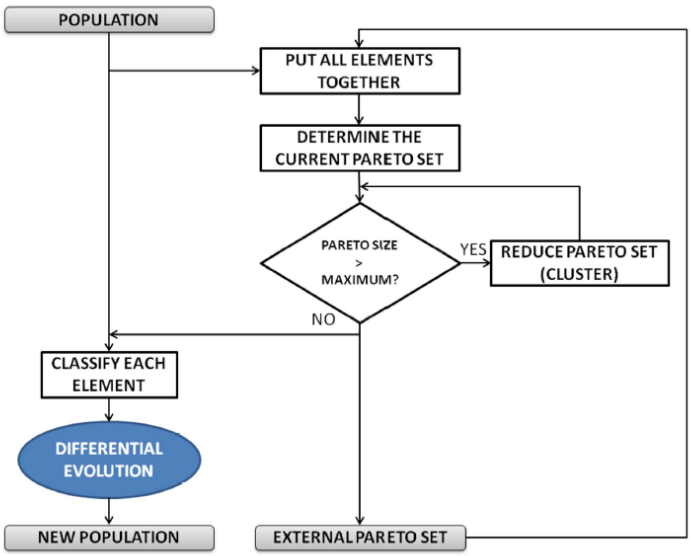

Figure 3.7: MODE algorithm

Consider an algorithm a, represented as a mapping from previously unvisited sets of points to a single previously unvisited point ξ_k in the search space composed of all the feasible points. Let $P(\xi_k|f,k,a)$ be the conditional probability of visiting point ξ_k when the algorithm a iterates for k time the objective function f. Thus, for any pair of algorithms a_1 and a_2

$$\sum_f P(\xi_k|f,k,a_1) = \sum_f P(\xi_k|f,k,a_2) \tag{3.11}$$

From the equation 3.11 follows that all optimization algorithms have the same mean performance across all the possible objective functions, being the sum of conditional probabilities of visiting point ξ_k the same over all the possible objective functions, regardless the algorithm chosen.

This outcome is valid both for evolutionary and deterministic algorithms.

Another consequence of this result is that if an algorithm gains something in term of performance on one class of problem, that gain is necessarily lost by the same algorithm in the performance for the remaining problems.

However, engineers are only interest in a subset of problems, then is possible to select or create an algorithm which outperforms the others for a particular class of problem.

Chapter 4

Alternatives and hybrid algorithms

The key aim of this chapter is to develop some innovative algorithms modifying and improving the standard PS method, and to create two hybrid algorithms mixing the PS with the two stochastic methods Particle Swarm Optimization (PSO) and Differential Evolution (DE)

4.1 Poll step strategy

Consider the poll step of the PS method described below.

Poll step: search around the point of the selected item $P_k = \{\boldsymbol{x}_k + \Delta_k \boldsymbol{d}, \boldsymbol{d} \in \boldsymbol{D}_k\}$, define $L_{add} = \{(\boldsymbol{x}_k + \Delta_k \boldsymbol{d}, \Delta_k),\ \boldsymbol{d} \in \boldsymbol{D}_k\}$, call the function $filter\ L_{filter} = filter(L_k, L_{add})$ and put $L_{trial} = L_{filter}$ or $L_{trial} \subseteq L_{filter}$. If $L_{trial} \neq L_k$ put $L_{k+1} = L_{trial}$, else $L_{k+1} = L_k$. In both cases update the step size, and go to the search step.

Two strategies can be used and implemented for the poll step, the fixed order strategy which computes the function in all the points around the poll center and if a non dominated point is found declares the iteration successful, or the dynamic strategy which considers one by one the points and as soon as a non dominated point is found the iteration is declared successful.

In figure 4.1 and 4.2 [22] are shown the calculations performed respectively with the dynamic and fixed strategy. While in the dynamic case when the first feasible point and non dominated point is found the polling around $\boldsymbol{x}^1$ is interrupted, in the second case all points around $\boldsymbol{x}^1$ are computed, performing a complete polling, and only at the end the non dominated points are filtered.

4.2 Search directions

In order to span positively $\mathbb{R}^n$ the number of search directions must be at least equal to $n+1$. An alternative to reduce the number of search directions, still ensuring the convergence, is a random matrix set directions generation. Therefore, the set of directions changes at each iterations, and can be shown that to maximize the convergence rate the random directions should be uniformly distributed on the unit sphere, but being the number of unsuccessful iterations higher the step size expansion must be greater than 1.

In the algorithm developed the random matrix has on the columns an orthonormal basis of the space $\mathbb{R}^n$, we indicate this set of vectors with the letter σ.

An orthonormal set of vectors $\boldsymbol{d}_1, \boldsymbol{d}_2, ..., \boldsymbol{d}_k \in R^n$ is normalized, having $\|\boldsymbol{d}_i\| = 1$ for $i = 1, ..., k$, orthogonal with $\boldsymbol{d}_i \perp \boldsymbol{d}_j$ for $i \neq j$, meaning that the vectors are mutually perpendicular. This set must be linearly independent, and if it is a vector space basis for the space it spans, such a basis is called an orthonormal basis. The n vectors column of matrix σ represent an orthonormal basis for $\mathbb{R}^n$.

In the first option matrices with n search directions are used, created with a random orthonormal matrix generator, where θ is the step size contraction coefficient and λ the expansion coefficient.

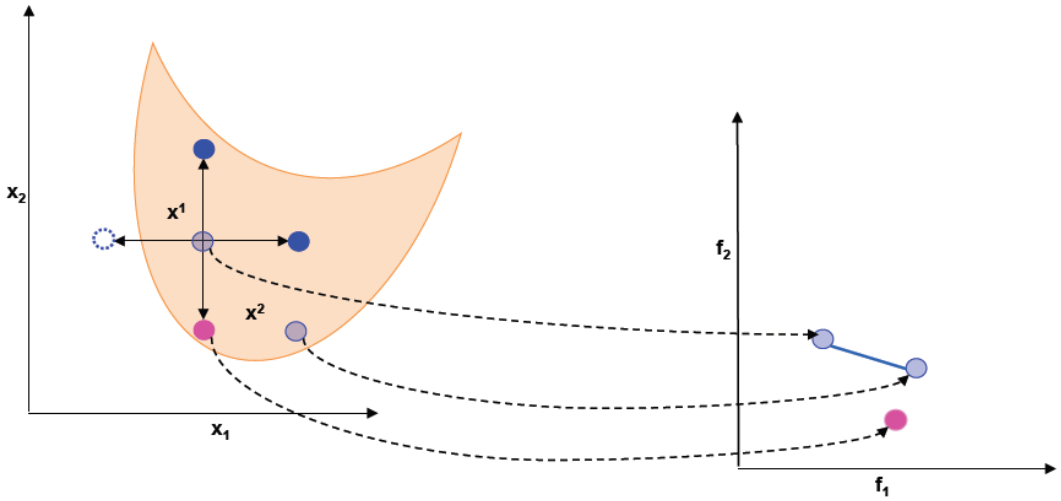

Figure 4.1: Dynamic strategy

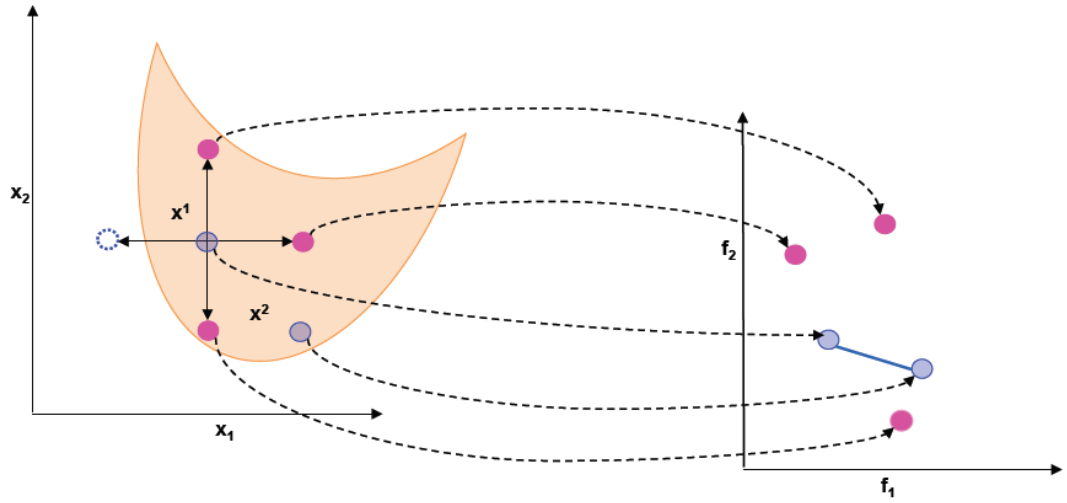

Figure 4.2: Fixed order strategy

$$\begin{aligned} \boldsymbol{D}_k &= [\sigma] \\ \theta &= \frac{1}{2} \\ \lambda &= 2 \end{aligned} \tag{4.1}$$

In the second option the number of search directions is equal to $2n$, therefore no advantages is achieved in term of function evaluations, but it may be useful when the shape of the Pareto set is much variable. In this case the step size expansion coefficient does not have necessarily to be higher than 1.

$$\begin{aligned} \boldsymbol{D}_k &= [\sigma - \sigma] \\ \theta &= \frac{1}{2} \\ \lambda &= 1 \end{aligned} \tag{4.2}$$

A generic form of orthonormal matrix in $\mathbb{R}^2$ is illustrated below

$$\boldsymbol{D}_k = \begin{bmatrix} cos(\alpha_k) & sin(\alpha_k) \\ sin(\alpha_k) & -cos(\alpha_k) \end{bmatrix} \tag{4.3}$$

Where the angle α_k is generated randomly at each generation, and in this simple way the random polling matrix is defined.

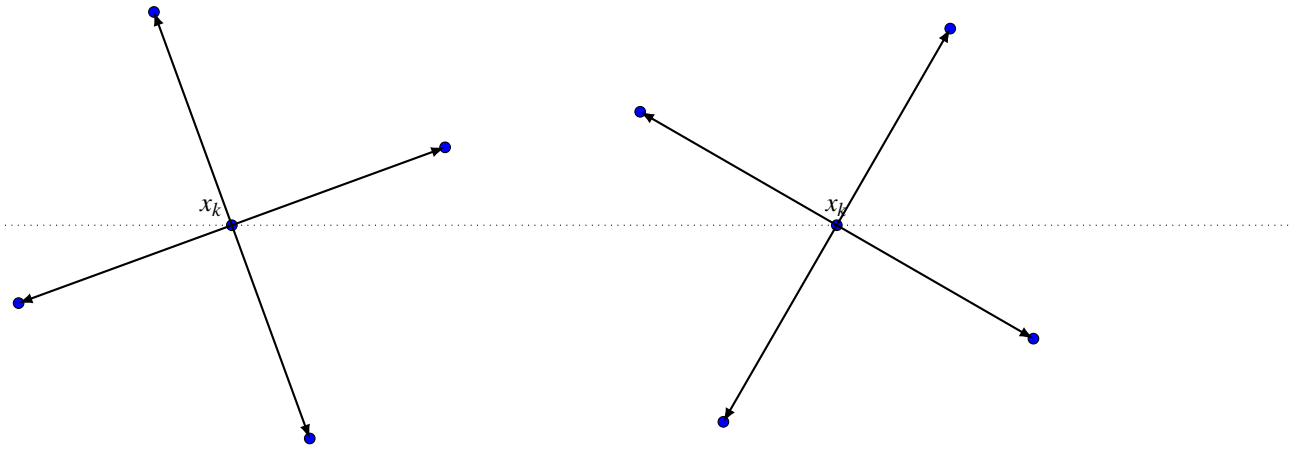

Figure 4.3: Random D_k generations

Another alternative for the directions matrix which takes into account the winning directions was developed, memorizing the last two for which the iteration is declared successful. Below the functioning of the algorithm is described.

1) Iteration k : store the winning direction $\boldsymbol{v}_k$
2) Iteration $k+1$: store the winning direction $\boldsymbol{v}_{k+1}$
3) Iteration $k+2$: $\boldsymbol{d}_{k+2} = \boldsymbol{v}_k + \boldsymbol{v}_{k+1}$. Define with the vector $\boldsymbol{d}_{k+2}$ an orthonormal basis γ of the space $\mathbb{R}^n$. Use as directions matrix $\boldsymbol{D}_{k+2} = [\gamma - \gamma]$.

We can now consider an example in $\mathbb{R}^2$ in order to show how the new polling matrix is defined.
If at the beginning the matrix $\boldsymbol{D}_0$ is defined through the identity matrix I

$$\boldsymbol{D}_0 = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \tag{4.4}$$

And we obtain for the winning directions in 0 and 1 respectively

$\boldsymbol{v}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $\boldsymbol{v}_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

Therefore, the vector which defines the orthonormal basis is

$\boldsymbol{d}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

Through the normalization of this vector, and the definition of the orthonormal basis we obtain the new directions matrix

$$\boldsymbol{D}_2 = \begin{bmatrix} 0.707 & -0.707 & -0.707 & 0.707 \\ 0.707 & 0.707 & -0.707 & -0.707 \end{bmatrix} \tag{4.5}$$

If at the iteration k the vector column obtained $\boldsymbol{d}_k$ is a zero vector, the new polling matrix is defined as the one at the beginning.

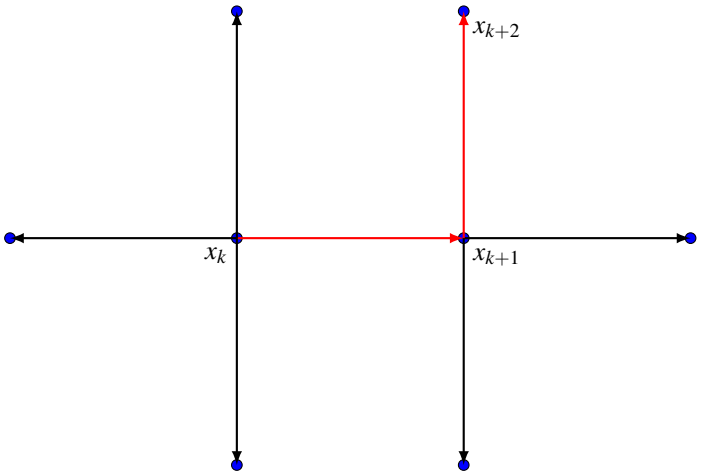

Figure 4.4: Store winning directions

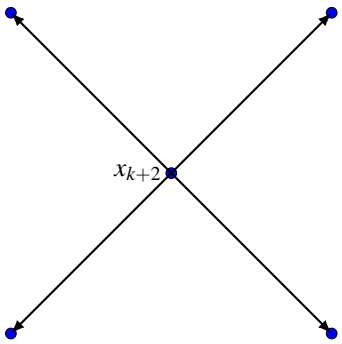

Figure 4.5: New search directions

Such method seems to be effective for problems with an irregular Pareto set, but does not show an improvement for a wide class of problems.

4.3 Adaptive step size strategy

In the standard Pattern Search method to each point in the temporary Pareto set is associated a step size. In order to increase the accuracy of the poll step an alternative is proposed in which a step size is associated to each search directions of each point in the temporary Pareto set, therefore to each point is associated a vector with $2n$ elements, if $2n$ is the number of search directions. Obviously the

matrix directions $\boldsymbol{D}_k$ containing on its vectors column the search directions $\boldsymbol{d}_i$ must be the same for each iterations. Therefore, the actual polling matrix is $\boldsymbol{D}_k^*$.

$$\boldsymbol{D}_k^* = \begin{bmatrix} \Delta_{k,1}\boldsymbol{d}_1 & \Delta_{k,2}\boldsymbol{d}_2 & \cdots & \Delta_{k,2n}\boldsymbol{d}_{2n} \end{bmatrix} \tag{4.6}$$

With this alternative strategy the convergence is ensured both in the single and the multiobjective case, below is presented the convergence analysis in the single objective case.

Indicating with Δ_i the step size associated to the direction i, with m the number of unsuccessful poll steps, with n the number of successful poll steps in the standard case, and finally with $n - s_i$ the number of successful iterations for the direction i ($s \geq 0$ and $s \in \mathbb{N}$), given that the number of successful iterations for each directions is equal or lower than in the standard case, being the step size associated to each directions, we have for every i

$$\Delta_i = \Delta_0 \theta^m \lambda^{(n-s_i)} \leq \Delta_0 \theta^m \lambda^n = \Delta \tag{4.7}$$

Where Δ_0 is the initial step size. As it was proved in the theory with $0 < \theta < 1$, and $\lambda \geq 1$ we get $\lim_{m\to\infty} \Delta = 0$. Finally for the comparison theorem

$$\lim_{m\to\infty} \Delta_i = \lim_{m\to\infty} \Delta = 0 \tag{4.8}$$

Therefore, the convergence is ensured, because all the elements Δ_i of the step size vector tend to zero increasing the number of iterations.

An example of adaptive polling with the matrix D_k^* is illustrated in figure 4.6.

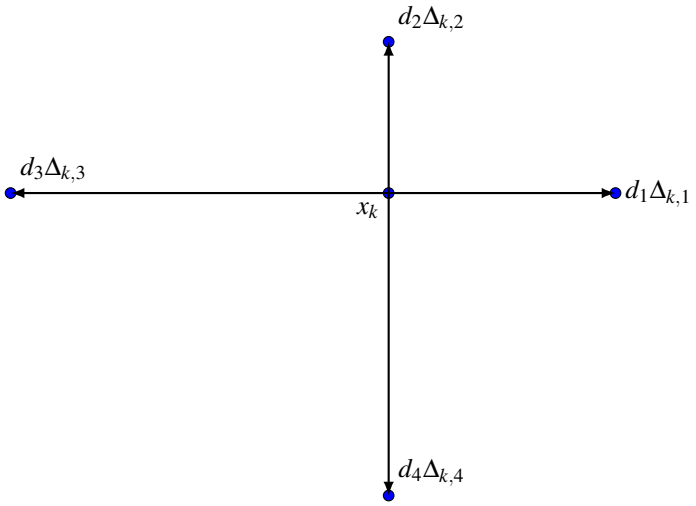

Figure 4.6: Adaptive poll step

4.4 Hybrid algorithm: PSO-PS

An hybrid algorithm which combines the Particle Swarm Optimization with Pattern Search was developed, the two components work in series, therefore after the execution of the PSO the output, consisting in the archive of the PSO, is used as list of initial points for the PS. That optimizer should keep the robustness of the PSO increasing the speed with the PS component, exploiting the strong global search component of PSO with the local search and refinement af a deterministic method like PS.

Furthermore, it is possible to modify in the settings the number of function evaluations for the PSO and the PS components, in order to adapt the optimizer to different classes of problems.

The convergence is ensured by the PS component, being PSO just used in order to provide the input for PS.

4.5 Hybrid algorithm: DE-PS

An hybrid algorithm consisting in a Differential Evolution and a Pattern Search component intertwined was developed, the machinery is described below.

Differential Evolution component

0) Initialization according to the population size N
1) Mutation
2) Recombination
3) Selection
4) Store in a non dominated sorting archive

if iteration > m (where m is set by the user)

Pattern Search component

5) Poll step around the first M points of the archive, with $M \leq N$
6) Store in a non dominated sorting archive
7) Reinitialize the population for the DE component with the first N points of the non dominated sorting archive

end if

8) go to step 1

For the convergence analysis we have to analyze the step 5, where the polling is performed.

If the polling around all the N points does not find any non dominated points the step size is reduced according to the parameter θ, while in case the procedure is successful the expansion parameter is $\lambda = 1$. Therefore, given that the Differential Evolution scheme works improving the Pareto set at each iteration, without adding new dominated points, also in this case the convergence is ensured by the PS component.

In practice, given its fast decrease a minimum value for the step size, which can be modified in settings, is imposed, and the execution continues until it meets the stop criterion based on a maximum value of function evaluations.

The combination of these two algorithms is done in order to obtain a greedy algorithm, suitable for engineering applications.

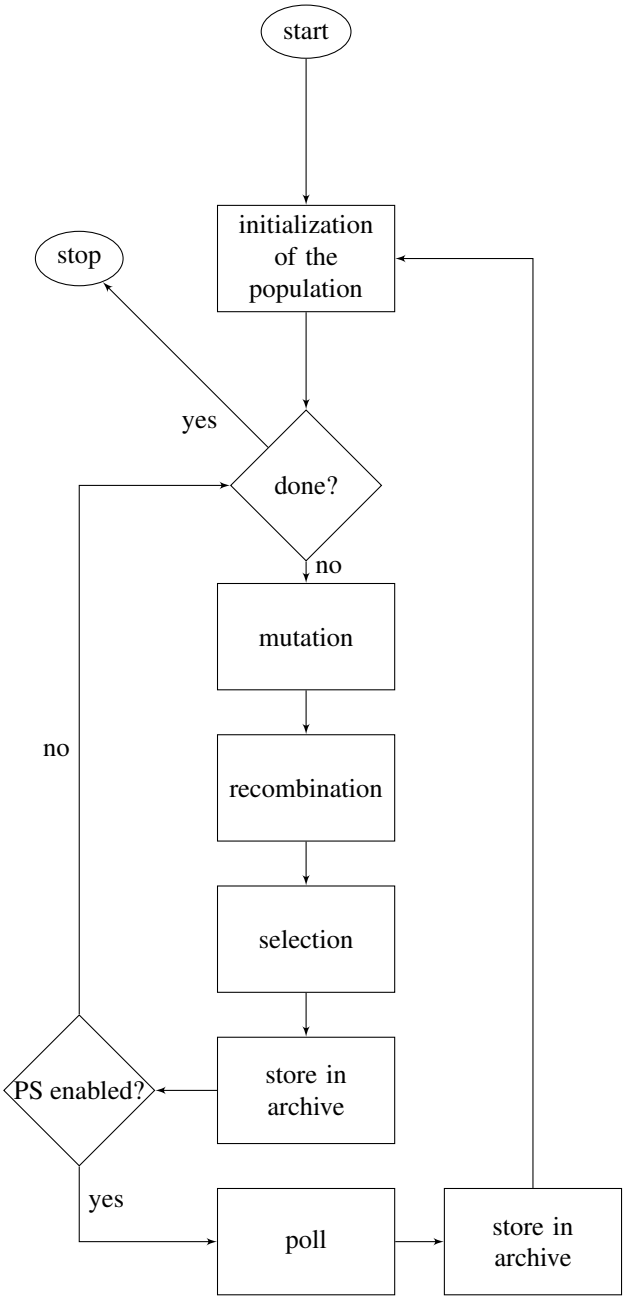

Figure 4.7: DE-PS flowchart

Chapter 5

Comparison of multiobjective algorithms: metrics and performance

5.1 Overview

In chapters 2 and 3 we have presented a pure deterministic Pattern search algorithms (PS), and two stochastic algorithms Particle Swarm Optimization (PSO) and Differential Evolution (DE) applicable to multiobjective problems, while in chapter 4 we have introduced several innovative algorithms changing the strategy for the step size update and for the polling (PS(adaptive)) like in sections 4.1 and 4.3, or creating hybrids between PSO and PS (PSO-PS) section 4.4, and among DE and PS (DE-PS) as in section 4.5.

The aim of this chapter is to compare all these algorithms measuring the performance for a set of test functions, in order to show what is the best option for multiobjective optimization, and assess the pure deterministic and new hybrid algorithms reliability.

Comparing the performance of multi-objective optimization algorithms is not trivial and is still a subject of considerable research, but there are two tasks that a multiobjective optimization algorithms must do well [5]

- Convergence as close to the true Pareto-optimal region as possible
- Maintain as many widely spread non-dominated solutions as possible

Therefore, we should consider benchmark problems where each of the above tasks is difficult to achieve.

The features are

- Convexity or nonconvexity in the Pareto-optimal front
- Discontinuity in the Pareto-optimal front
- Non-uniform distribution of solutions in the search space and in the Pareto-optimal front

A set of problems with these different features is presented in the next section, while the metrics to assess the closeness of the front computed to the real one and its uniform distribution are illustrated in section 5.3.

5.2 Test problems

Overall six test functions are considered, of which four belongs to the Zitzler-Deb-Thiele's (ZDT) test problems [5] [23].

These problems have two objectives which have to be minimized.

Minimize $f_1(\boldsymbol{x})$

Minimize $f_2(\boldsymbol{x}) = g(\boldsymbol{x})h(f_1(\boldsymbol{x}), g(\boldsymbol{x}))$

In the four problems all the variables lie in the range [0,1], they differ according to the way the functions $f_1(\boldsymbol{x})$, $g(\boldsymbol{x})$ and $h(\boldsymbol{x})$ are defined.

ZDT1. This is a problem with 30 variables (n=30) having a convex Pareto-optimal front.

$$ZDT1: \begin{cases} f_1(\boldsymbol{x}) = x_1 \\ g(\boldsymbol{x}) = 1 + \dfrac{9}{n-1}\sum_{i=2}^{n} x_i \\ h(f_1, g(\boldsymbol{x})) = 1 - \sqrt{f_1/g(\boldsymbol{x})} \end{cases} \tag{5.1}$$

This is the easiest of the problems considered, having a continuous Pareto-optimal front and a uniform distribution across the front.

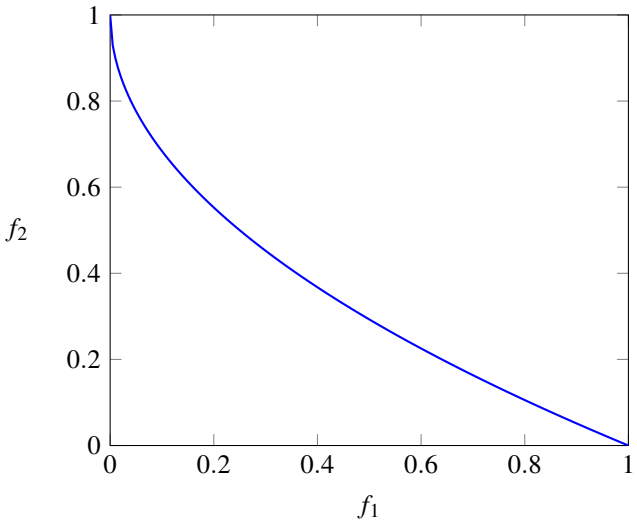

Figure 5.1: ZDT1 Pareto front

ZDT2. This is a problem with 30 variables (n=30) having a nonconvex Pareto-optimal front.

$$ZDT2: \begin{cases} f_1(\boldsymbol{x}) = x_1 \\ g(\boldsymbol{x}) = 1 + \dfrac{9}{n-1}\sum_{i=2}^{n} x_i \\ h(f_1, g(\boldsymbol{x})) = 1 - (f_1/g(\boldsymbol{x}))^2 \end{cases} \tag{5.2}$$

The main difficulty with this problem is that the Pareto-optimal region is non convex, but it has an uniform distribution of solutions.

ZDT3. This is a problem with 30 variables (n=30) having a number of disconnected Pareto-optimal fronts.

$$ZDT3: \begin{cases} f_1(\boldsymbol{x}) = x_1 \\ g(\boldsymbol{x}) = 1 + \dfrac{9}{n-1}\sum_{i=2}^{n} x_i \\ h(f_1, g(\boldsymbol{x})) = 1 - \sqrt{f_1/g(\boldsymbol{x})} - (f_1/g(\boldsymbol{x}))sin(10\pi f_1) \end{cases} \tag{5.3}$$

The main difficulty with this problem is that the Pareto-optimal region is discontinuous.

ZDT6. This is a problem with 10 variables (n=10) having a nonconvex Pareto-optimal front. Furthermore, the density of solutions across the Pareto optimal region is non-uniform.

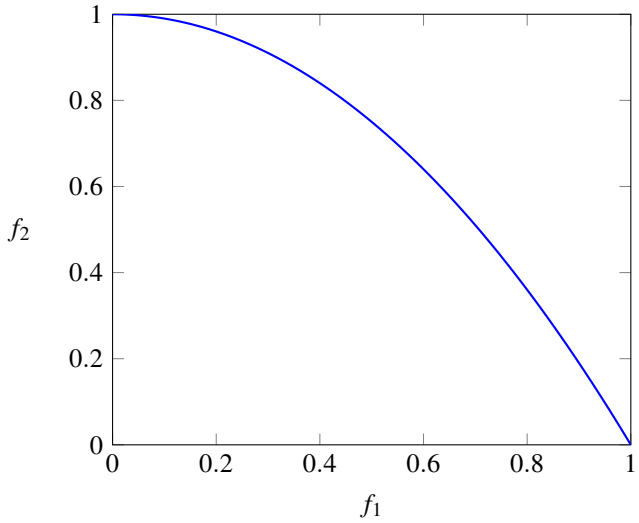

Figure 5.2: ZDT2 Pareto front

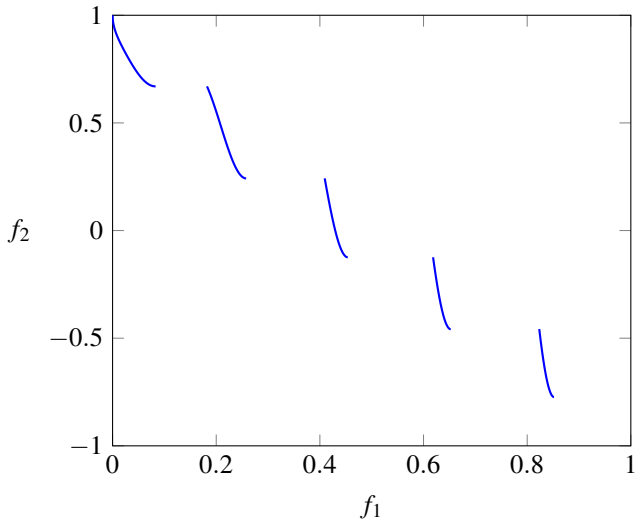

Figure 5.3: ZDT3 Pareto front

$$ZDT6: \begin{cases} f_1(\boldsymbol{x}) = 1 - exp(-4x_1)sin^6(6\pi x_1) \\ g(\boldsymbol{x}) = 1 + 9[(\sum_{i=2}^{n} x_i)/9]^{0.25} \\ h(f_1, g(\boldsymbol{x})) = 1 - (f_1/g)^2 \end{cases} \tag{5.4}$$

The adverse density of solutions across the front, together with the nonconvex nature of the front may cause troubles for many algorithms to converge to the true Pareto-optimal front.

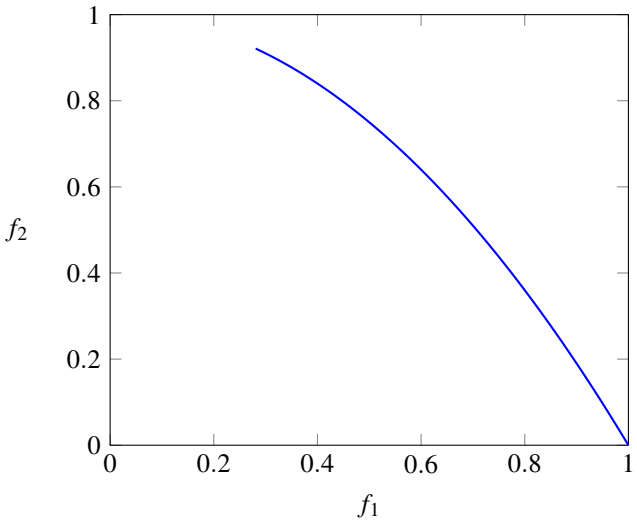

Figure 5.4: ZDT6 Pareto front

The remaining two test problems (CEC1 and CEC2) are taken from the CEC09 algorithm contest [24], and are both unconstrained problems with n=30 variables having as search space $[0,1] \times [-1,1]^{n-1}$. The two objectives to be minimized are f_1 and f_2.

CEC1.

$$CEC1: \begin{cases} f_1(\boldsymbol{x}) = x_1 + \frac{2}{|J_1|} \sum_{j \in J_1} [x_j - sin(6\pi x_1 + \frac{j\pi}{n})]^2 \\ f_2(\boldsymbol{x}) = 1 - \sqrt{x_1} + \frac{2}{|J_2|} \sum_{j \in J_2} [x_j - sin(6\pi x_1 + \frac{j\pi}{n})]^2 \end{cases} \tag{5.5}$$

Where j belongs to J_1 if it is odd and $2 \leq j \leq n$, while j belongs to J_2 if it is even and $2 \leq j \leq n$.

CEC2.

$$CEC2: \begin{cases} f_1(\boldsymbol{x}) = x_1 + \frac{2}{|J_1|} \sum_{j \in J_1} y_j^2 \\ f_2(\boldsymbol{x}) = 1 - \sqrt{x_1} + \frac{2}{|J_2|} \sum_{j \in J_2} y_j^2 \end{cases} \tag{5.6}$$

Where j belongs to J_1 if it is odd and $2 \leq j \leq n$, while j belongs to J_2 if it is even and $2 \leq j \leq n$ and

$$y_j = \begin{cases} x_j - [0.3x_1^2 cos(24\pi x_1 + \frac{4j\pi}{n}) + 0.6x_1] cos(6\pi x_1 + \frac{j\pi}{n}) & j \in J_1 \\ x_j - [0.3x_1^2 cos(24\pi x_1 + \frac{4j\pi}{n}) + 0.6x_1] sin(6\pi x_1 + \frac{j\pi}{n}) & j \in J_2 \end{cases} \tag{5.7}$$

The functions CEC1 and CEC2 have the same Pareto front, and have been chosen for the variability of the Pareto set.

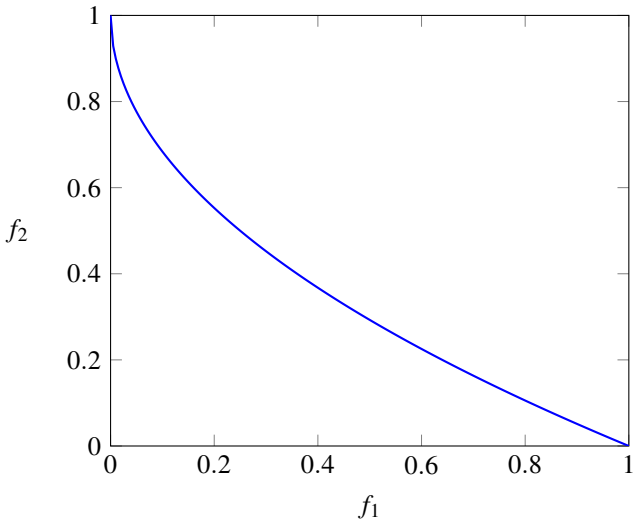

Figure 5.5: CEC1 and CEC2 Pareto front

5.3 Performance metrics

As we have already mentioned there are two distinct goals in multiobjective optimization, discovering solutions as close to the Pareto front as possible, and finding solutions as diverse as possible in the front. These two goals are orthogonal to each other, requiring the first a search towards the Pareto-optimal region and the second a search along the Pareto front.

5.3.1 Error Ratio

Given two algorithms A and B in order to assess which is the best one in term of closeness to the real Pareto front we introduce the Error Ratio (ER) metric. This metric counts the number of solution of A which are strictly dominated by the solutions of B.

$$ER_{AB} = \frac{\sum_{i=1}^{|A|} e_i}{|A|} \tag{5.8}$$

Where $e_i = 1$ if the point i in A is strictly dominated by at least one of the point of B, and $e_i = 1$ otherwise. The metric ER_{AB} takes a value between zero and one, an $ER_{AB} = 0$ means all solutions found by A are nondominated by the ones from B, while an $ER_{AB} = 1$ means that all the solutions are dominated by the ones computed by B. Smaller is the value the better is the performance of A compared to B.

5.3.2 Spacing

Furthermore, we introduce four parameters in order to assess the distribution of solutions across the Pareto front. We assume that the n objective function values computed have been sorted by increasing order, and we define the set D which contains the $n-1$ euclidean distances d_i with $i \in 1\{1,2,...,n-1\}$ between consecutive points in the computed Pareto front. Being the test functions considered problems with two objectives, the concept of consecutive points is defined without ambiguity.

$$d_i = \sqrt{(f_{i,1} - f_{i+1,1})^2 + (f_{i,2} - f_{i+1,2})^2} \tag{5.9}$$

We can now compute two measures derived from the set D, the mean μ which represents an indicator of the density of the solutions across the front, and the standard deviation σ, an indicator of the spacing, that is the uniformity of the distribution.

Table 5.1: Points distribution along the front (2000 function evaluations).

Algorithm	δ	μ	σ	σ^*
DE-PS	**0.2558**	**0.0801**	0.0705	0.9064
PSO-PS	0.5219	0.3129	0.1551	0.6124
PS(adaptive)	0.4125	0.2682	0.1337	0.5278
PS	0.2635	0.1563	0.0815	**0.5182**
DE	0.2909	0.1409	**0.0695**	0.5431

$$\mu = \frac{\sum_{i=1}^{n-1} d_i}{n} \tag{5.10}$$

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n-1}(d_i - \mu)^2} \tag{5.11}$$

The algorithms we will compare compute a different number of solutions, having a completely distinct working principle. While the stochastic methods are population based, thus computing a fixed and known number of points for each problems, the deterministic method has a steadily increasing number of points at each iteration, and the final number will not be fixed nor predictable.

That is why we need to introduce a metric able to compare the quality of the distribution of Pareto fronts computed with different number of points. The relative standard deviation σ^* fulfils this scope, being the ratio between the standard deviation and the mean.

$$\sigma^* = \frac{\sigma}{\mu} \tag{5.12}$$

A further aim of an optimizer is to avoid the making of "holes" in the computed front in order not to lose important information, for this reason we introduce a metric to indicate the maximum distance in the Pareto front, that is the maximum euclidean distance between consecutive solutions. The maximum spread δ is the maximum value in the set D.

$$\delta = \max(D) \tag{5.13}$$

5.4 Numerical results

The results presented here were obtained for two function evaluations cases, with 30000 and 2000 function evaluations. The first case represents a standard number of evaluations to deal with very large dimension problems, but such a high number of function evaluations is not realistic for many electromagnetic device design problems. For this reason we also consider the case with 2000 function evaluations, which employs a feasible amount of time to deliver a feasible solution and represents a measure of the aggressiveness of the algorithm, which is crucial for engineering optimization.

All the values in the tables consists in the average obtained by the algorithms in all the benchmark problems.

In Tables 5.1 and 5.2 the parameters regarding the spacing are illustrated, while Tables 5.3 and 5.4 show the error ratios resulted by the comparison of the algorithms. In particular the error ratio ER_{ij} in column i and row j, represents the number of points computed by the algorithm in column i which are dominated by the algorithm in row j. For instance the value in column 3 and row 2 is the relative value of solutions computed by the algorithm PS(adaptive) dominated by the solutions computed by the algorithm PSO-PS. Where E_{ij} is lower than E_{ji} it is marked in bold, meaning that the comparison between the algorithm in column i and the one in row j has seen the algorithm in column i as the winner.

The hybrid algorithm DE-PS wins the comparison in term of closeness to the real Pareto front, being the one with the lower Error Ratio when is compared with all the other algorithms. Also the DE algorithm, which constitutes a reference for its convergence speed is defeated, especially in the case with 2000 function evaluations, proving its aggressiveness which is crucial in solving real cases

Table 5.2: Points distribution along the front (30000 function evaluations).

Algorithm	δ	μ	σ	σ^*
DE-PS	**0.1273**	0.0205	0.0237	1.1302
PSO-PS	0.1589	0.0495	0.0543	1.1661
PS(adaptive)	0.1601	0.0296	0.0342	1.1292
PS	0.2031	0.0351	0.0633	1.4568
DE	0.1425	**0.0167**	**0.0233**	**1.1225**

Table 5.3: Error ratio (2000 function evaluations).

Algorithm	DE-PS	PSO-PS	PS(adaptive)	PS	DE
DE-PS		0.7222	0.5000	0.7095	0.9719
PSO-PS	**0.2502**		**0.0786**	**0.1936**	**0.3236**
PS(adaptive)	**0.3212**	0.5111		**0.3506**	0.4217
PS	**0.2379**	0.6361	0.5069		**0.4223**
DE	**0.0058**	0.5972	**0.3000**	0.4428	

Table 5.4: Error ratio (30000 function evaluations).

Algorithm	DE-PS	PSO-PS	PS(adaptive)	PS	DE
DE-PS		0.4626	0.5000	0.3730	0.5359
PSO-PS	**0.2554**		**0.2732**	**0.3127**	**0.1340**
PS(adaptive)	**0.2495**	0.4284		0.4185	**0.1821**
PS	**0.2145**	0.3477	**0.2408**		**0.0536**
DE	**0.1948**	0.6227	0.5325	0.5340	

optimization problems. Furthermore, it proves to have the lowest value for the maximum distance between consecutive points in the front.

Anyway the DE algorithm follows closely in term of performance, therefore, in the following figures we show the fronts computed by these two algorithms with a budget cost of 2000 function evaluations.

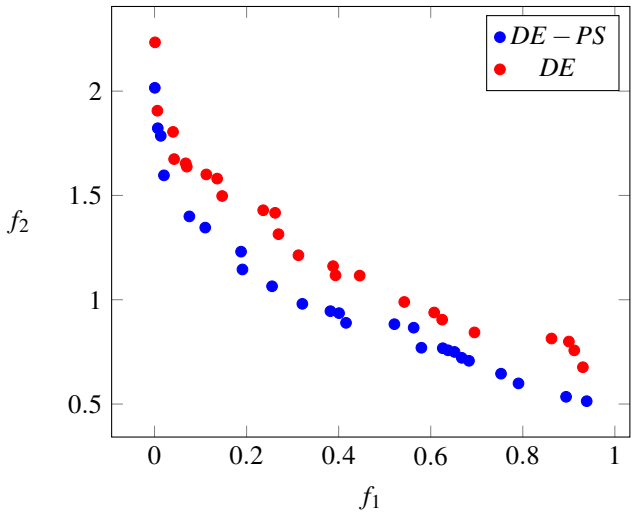

Figure 5.6: ZDT1 Pareto fronts computed.

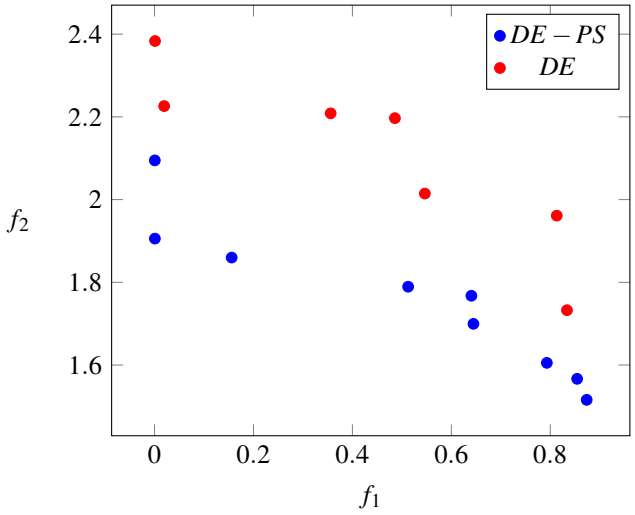

Figure 5.7: ZDT2 Pareto fronts computed.

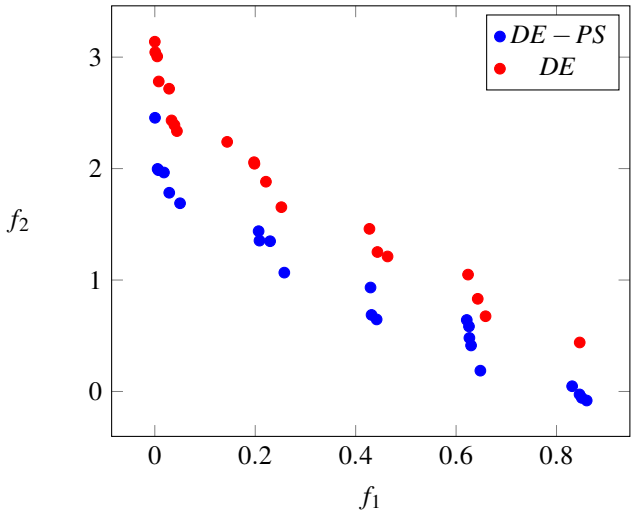

Figure 5.8: ZDT3 Pareto fronts computed.

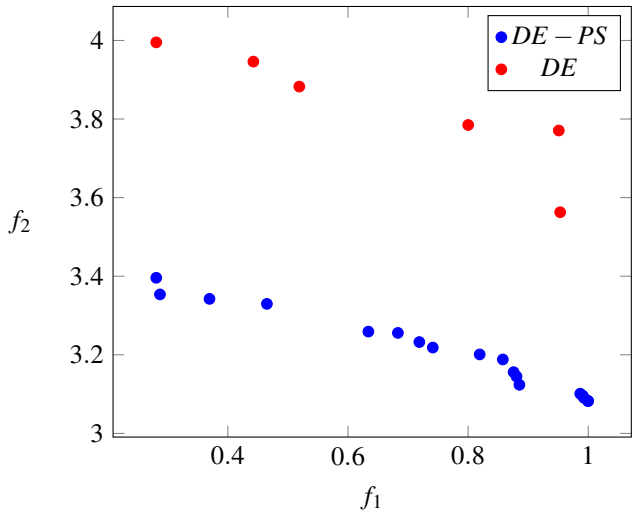

Figure 5.9: ZDT6 Pareto fronts computed.

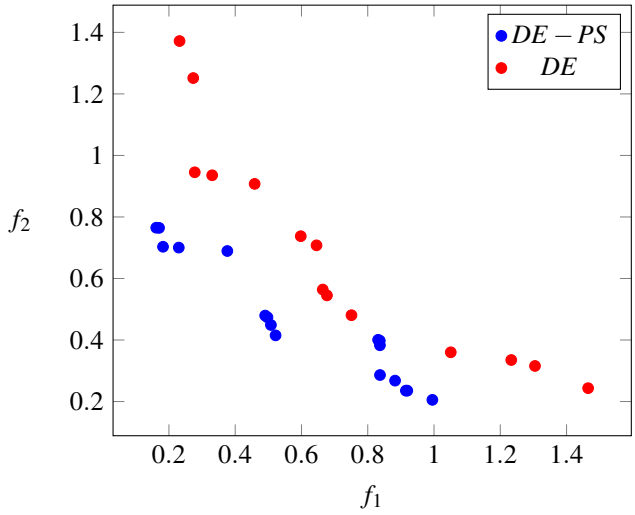

Figure 5.10: CEC1 Pareto fronts computed.

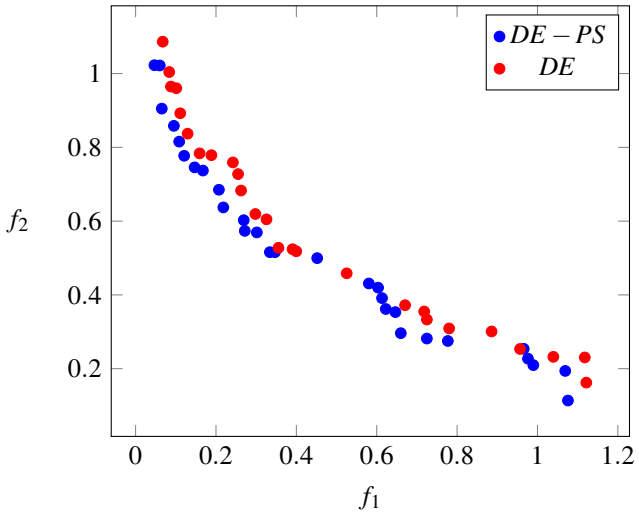

Figure 5.11: CEC2 Pareto fronts computed.

Chapter 6

Pose detection for magnetic-assisted medical devices

6.1 Overview

The pose detection for magnetic-assisted medical devices has very important applications in some surgical procedures. In fact, intra-medullary nails are used to stabilize and align some classes of fractures. An example is shown in figure 6.1 [25]. Eventually, the identification of the position and the orientation of the drill holes hidden by the tissue and the bone is done by X-Ray, creating several disadvantages.

An alternative for the pose detection method is represented by the use of a permanent magnet localized inside the drill hole, together with a set of external Hall sensors able to measure the magnetic field generated by the magnet.

From this set of measures, carried out in several positions, we can reconstruct the position and the orientation of the drill hole, solving an inverse problem. This procedure would be by far cheaper both in economic and time consumption term than the X-Ray one, furthermore it would nullify the health problems caused by the X-Ray.

The reconstruction of the position and the orientation of the permanent magnet from a set of magnetic measures is an inverse problem, which can be solved with an optimization algorithm. The mathematical model used to describe the magnetic field is explained in the next section.

Furthermore, the aim in this chapter is to find the best sensors configuration, between the planar and the circular configuration, and the optimum number of sensors for the pose detection, introducing a certain degree of noise in the Hall sensors measures.

6.2 Mathematical model

In this section we want to describe the model used for the computation of a magnetic field produced by a permanent magnet, which is used to define the function to be minimized in the optimization process. Therefore, we show the method for the calculation of the axial and transverse components of the intensity of the magnetic field of current loops and thin-wall air coils (solenoids). [26].

The coil has parallel walls and is wound tightly in an axial direction, having the cross section of the coil an arbitrary shape. Consider now a planar current loop of a general shape as in figure 6.4.

For the definition of the problem a scalar magnetic potential φ_m is used, so that we can compute the components of the magnetic field.

$$\mathbf{H} = -\nabla\varphi_m \tag{6.1}$$

The expressions for the components of the magnetic field intensity H_x, H_y, H_z in Cartesian coordinates can be written as

$$H_x = -\frac{\partial\varphi_m}{\partial x} \quad H_y = -\frac{\partial\varphi_m}{\partial y} \quad H_z = -\frac{\partial\varphi_m}{\partial z} \tag{6.2}$$

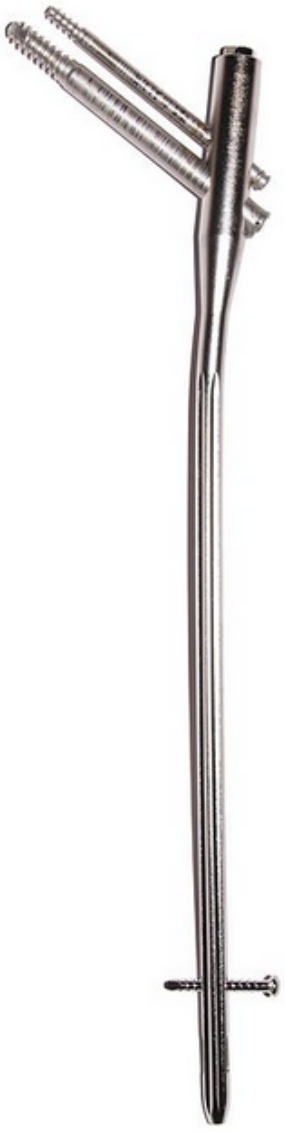

Figure 6.1: Intra-medullary nail

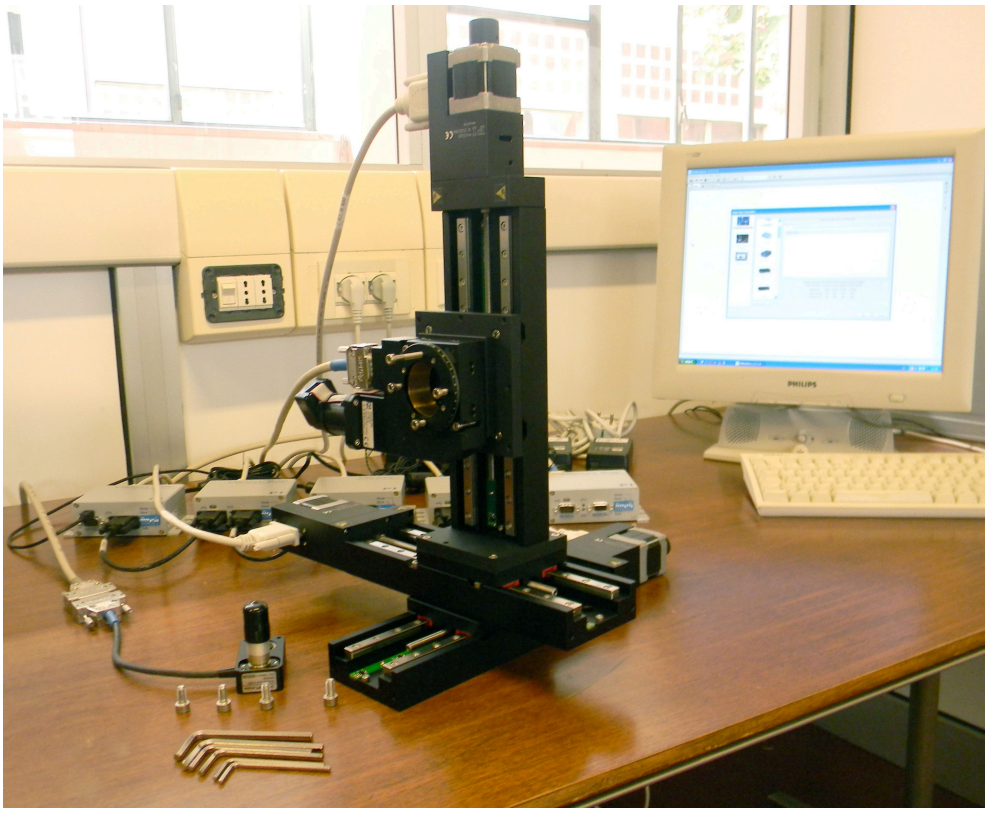

Figure 6.2: Device for the pose detection

Similar relations are also applied in the description of the radial and axial components in the cylindrical coordinate system. If the problem is cylindrically symmetric as it is our case, being the permanent magnet a cylinder, the tangential component is equal to zero.

$$H_r = -\frac{\partial \varphi_m}{\partial r} \quad H_\varphi = 0 \quad H_z = -\frac{\partial \varphi_m}{\partial z} \tag{6.3}$$

The scalar potential of the magnetic field, excited by an elementary current loop in the location specified by the radius vector is ρ

$$\varphi_{mdip}(\rho) = \frac{m\rho}{4\pi\rho^3} + const \tag{6.4}$$

Where $\mathbf{m} = I\mathbf{dS}$ is the magnetic dipole moment.
Then the scalar potential of a dipole oriented in the z-direction is

$$\varphi_{mdip} = \frac{1}{4\pi} \frac{z}{(r^2+z^2)^{\frac{3}{2}}} (IdS) + const \tag{6.5}$$

Through a surface integration we can add the potentials of all the dipoles over the effective area of the loop in order to compute the resultant scalar

$$\varphi_{loop}(x,y,z) = \frac{1}{4\pi} \iint_S \frac{z}{(r^2+z^2)^{\frac{3}{2}}} (IdS) \tag{6.6}$$

Consider now the case of a circular current loop and a circular solenoid as in figure 6.5 and 6.6.
We want to compute the magnetic field using a method based on elliptic integrals for circular loop and solenoid. It solves the equation for the vector potential within a cylindrical coordinate system, which comprises in these cases only the tangential component.
The vector potential of the circular current loop is

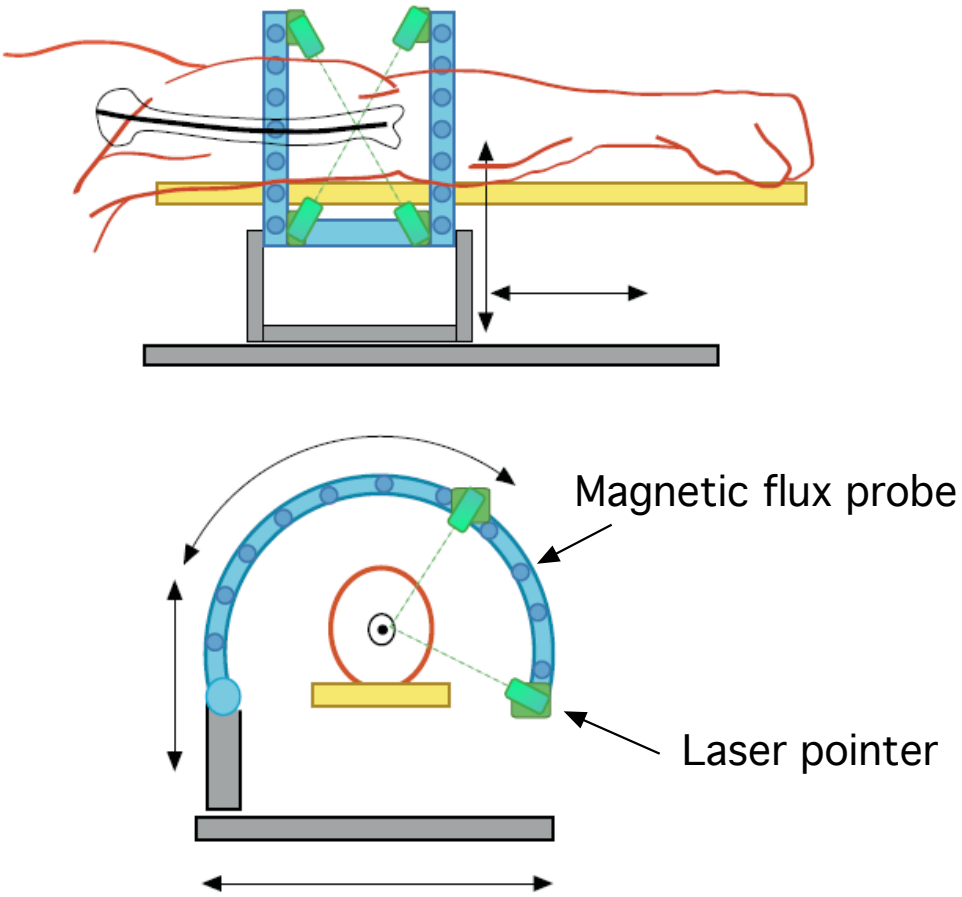

Figure 6.3: Magnetic configuration and pose-detection

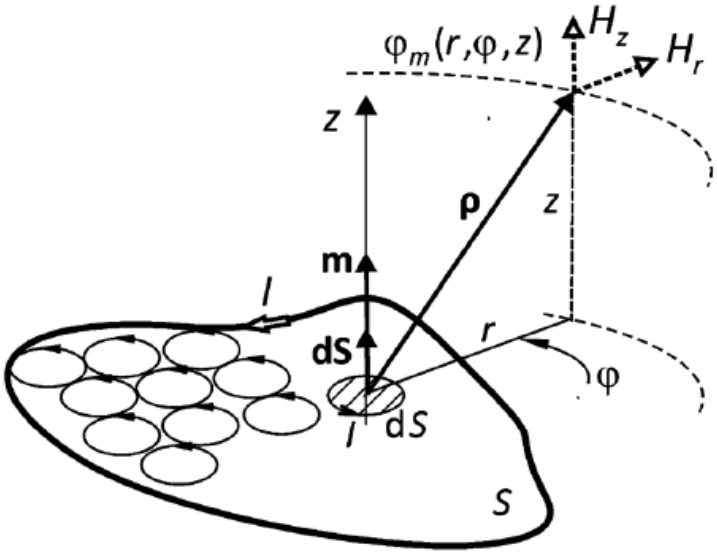

Figure 6.4: Planar current loop

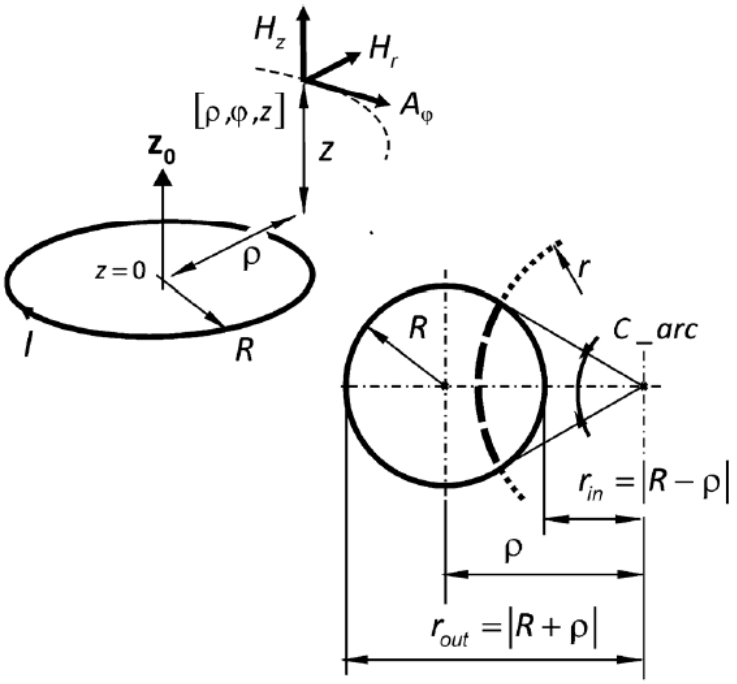

Figure 6.5: Circular current loop

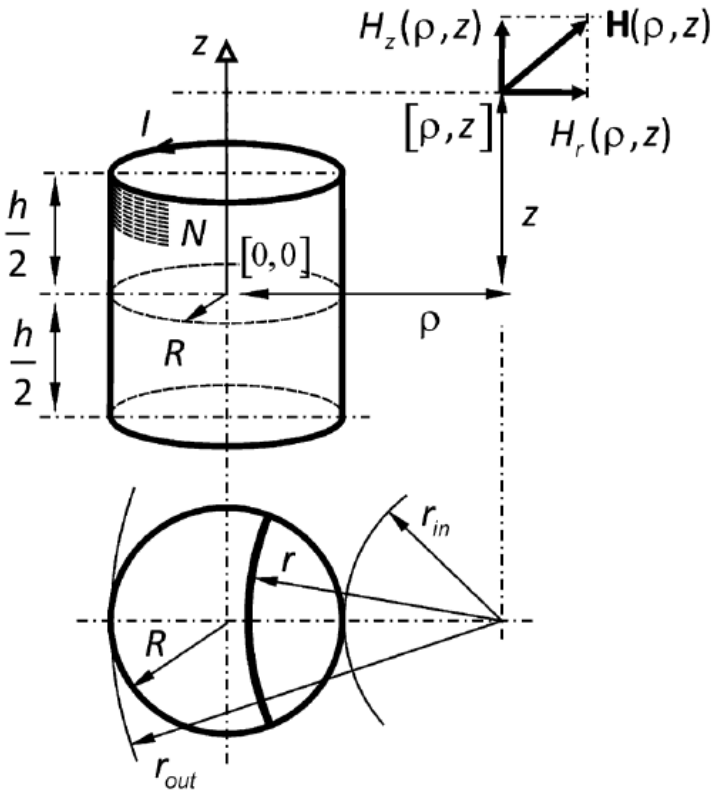

Figure 6.6: Circular solenoid

$$A_\varphi(R,I,\rho,z) = \frac{I}{4\pi}\int_0^{2\pi} \frac{Rcos(\varphi)}{\sqrt{R^2+\rho^2-2R\rho cos(\varphi)+z^2}}\,d\varphi \tag{6.7}$$

While the vector potential of the circular solenoid is

$$A_\varphi(R,h,I,\rho,z) = \frac{NI}{4\pi h}\int_{\frac{-h_x}{2}}^{\frac{h_x}{2}}\int_0^{2\pi} \frac{Rcos(\varphi)}{\sqrt{R^2+\rho^2-2R\rho cos(\varphi)+(z-\xi)^2}}\,d\varphi d\xi \tag{6.8}$$

The axial and the radial components of the magnetic field intensity is given by the respective derivatives of the vector potential.

$$H_r = -\frac{\partial A_\varphi}{\partial z} \quad H_z = \frac{1}{\rho}\frac{\partial \rho A_\varphi}{\partial \rho} \tag{6.9}$$

Being the equations 6.7, 6.8 and 6.9 not suitable for direct numerical quantification we need to introduce the defined general elliptic integral which is valid also for the following steps.

$$cel(k_c,p,a,b) = \int_0^{\frac{\pi}{2}} \frac{acos^2(\varphi)+bsin^2(\varphi)}{cos^2(\varphi)+psin^2(\varphi)}\frac{1}{\sqrt{cos^2(\varphi)+k_c^2 sin^2(\varphi)}}\,d\varphi \tag{6.10}$$

The parameter k_c denotes the complementarity modulus of the elliptic integral.

Finally, we can write the algorithm implemented for the calculation of the axial component H_z of the magnetic field intensity of the circular solenoid, which suits for the numerical quantification.

$$\begin{aligned} H_z(R,N,I,h,\rho,z) &= \frac{NI}{h}\frac{1}{2\pi}\left[Aux1\left(-\frac{h}{2}\right) - Aux1\left(\frac{h}{2}\right)\right] \\ Aux1(\xi) &= \begin{cases} \dfrac{z-\xi}{R+\rho}\sqrt{\dfrac{R}{\rho}}k\cdot cel\left(k_c,\left(\dfrac{R-\rho}{R+\rho}\right)^2,1,\dfrac{R-\rho}{R+\rho}\right) & if \quad \rho \neq 0 \\ \dfrac{\pi(z-\xi)}{\sqrt{R^2+(z-\xi)^2}} & if \quad \rho = 0 \end{cases} \end{aligned} \tag{6.11}$$

For the computation of the radial component H_r the following algorithm is implemented.

$$\begin{aligned} H_r(R,N,I,h,\rho,z) &= \frac{NI}{h}\frac{1}{2\pi}\left[Aux2\left(-\frac{h}{2}\right) - Aux2\left(\frac{h}{2}\right)\right] \\ Aux2(\xi) &= \begin{cases} \dfrac{R}{\rho}k\cdot cel(k_c,1,1,-1) & if \quad \rho \neq 0 \\ 0 \quad if \quad \rho = 0 \end{cases} \end{aligned} \tag{6.12}$$

For both two the components we have

$$k = \sqrt{\frac{4R\rho}{(R+\rho)^2+(z-\xi)^2}} \quad k_c = \sqrt{1-k^2} \tag{6.13}$$

6.3 Geometrical configurations of the sensors

In this section we want to describe and analyze the sensor configurations, the possible alternatives, and the main issues in pose detection.

Our aim is to detect the position and the inclination of a cylindrical permanent magnet through a set of measurements of the magnetic field generated by the cylinder, carried out by the sensors.

Two kinds of configuration are considered, a planar one, where the sensors are placed on a plane shaping a mesh, and the distance between the plane and the barycentre of the cylinder is equal to d as

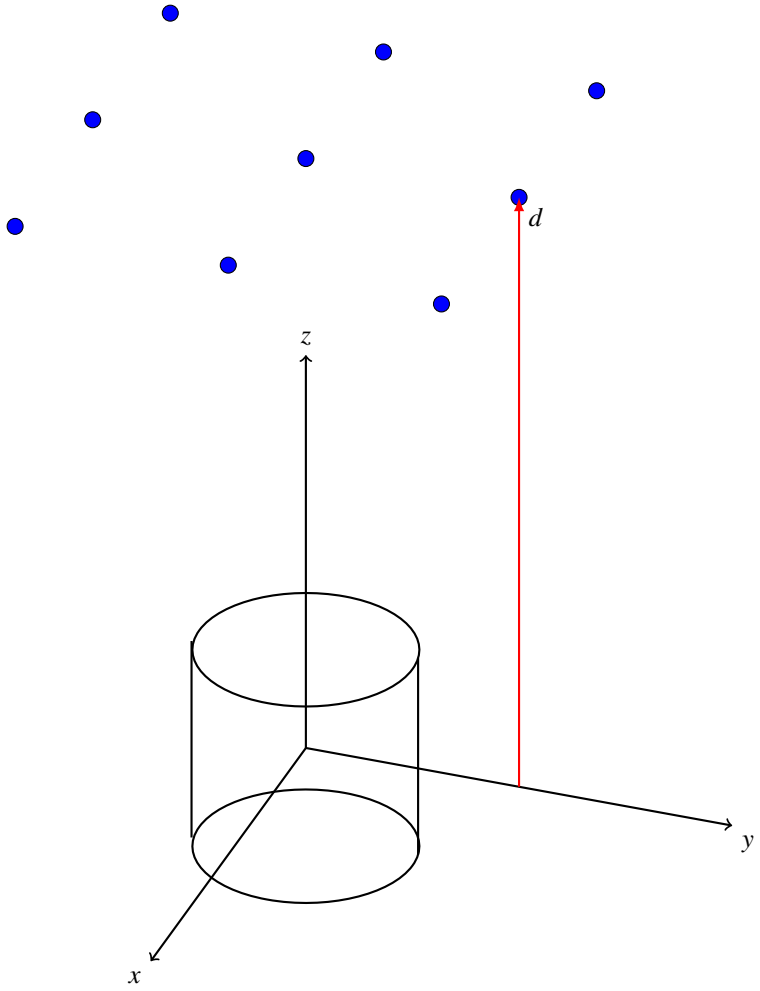

Figure 6.7: Cylinder and planar sensors

it is shown in figure 6.7. Therefore, in this case the sensors must be put just above or below the arm. Furthermore, we consider a circular configuration, where the sensors are arranged on a circle defined by the radius r as illustrated in figure 6.8, these sensors are placed all around the magnet like a bracelet for the arm.

These two fixed configurations are considered, having as only variables the distance d or the radius r and the number of sensors which shapes the mesh and the circle. We want to compare these two geometries in order to deliver a significant indication to find the most efficient configuration for the pose detection.

The main issue to consider is the robustness of the geometry, in fact each measure is affected by a noise component, this noise causes an error in the localization of the magnet, the greater is the error the lower is the robustness.

This question is crucial, because without the introduction of a noise the localization is carried out perfectly with all the sensors configurations, therefore this allows us to identify the best solution.

Thus, for each sensor given the ideal measures for each component in Cartesian coordinate $(B_{x,id}, B_{y,id}, B_{z,id})$, we introduce a random noise having as average the 5% of the ideal measure. Therefore, the real measures (B_x, B_y, B_z) are computed as it is described below.

$$\begin{cases} B_x = B_{x,id} + 2(rand - 0.5)0.05B_{x,id} \\ B_y = B_{y,id} + 2(rand - 0.5)0.05B_{y,id} \\ B_z = B_{z,id} + 2(rand - 0.5)0.05B_{z,id} \end{cases} \tag{6.14}$$

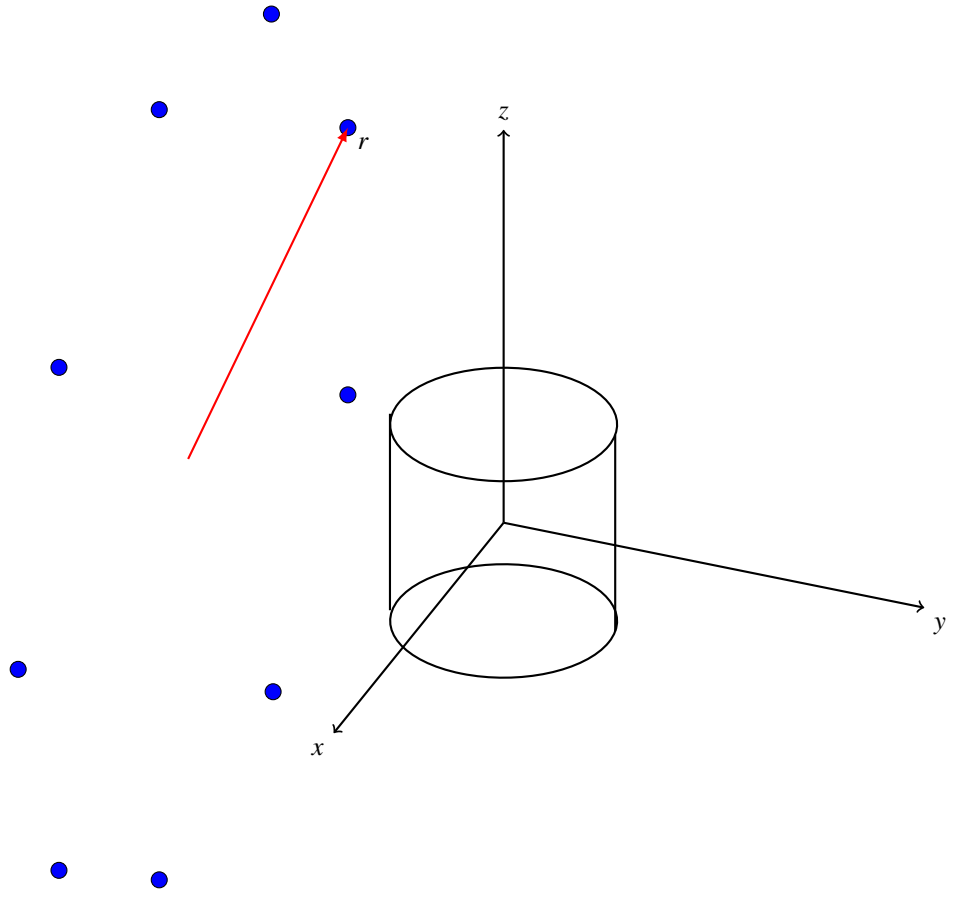

Figure 6.8: Cylinder and circular sensors

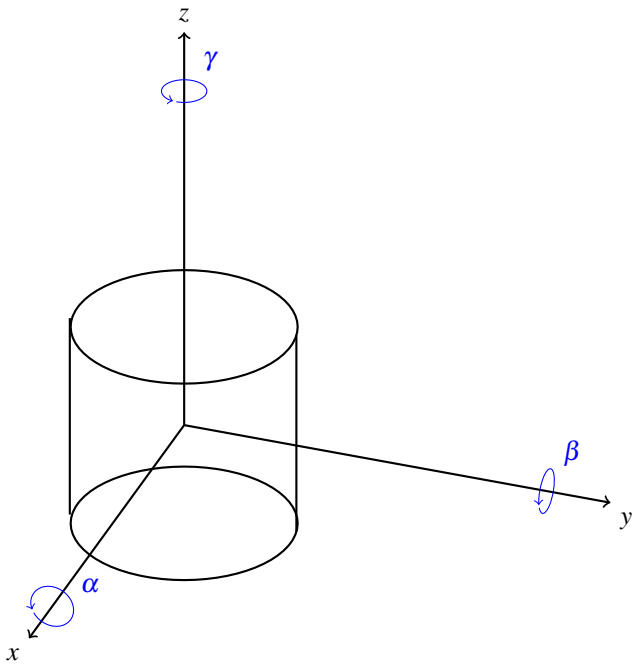

Figure 6.9: Cylinder rotations

Where *rand* is a function which returns a single uniformly distributed random number between 0 and 1.

Given the random component introduced in the measures each run in the solving process delivers a different error in the pose-detection, for this reason for each configuration the execution is repeated ten times, and the average error is considered.

The position of the permanent magnet can be described in the Cartesian space through the coordinates of its barycentre (x_b, y_b, z_b) and its rotations around the axes x,y,z defined respectively by the rotation angles α,β and γ, as in figure 6.9. Being the magnet considered a cylinder, the rotation around z is unnecessary for the definition of the position, therefore the angle γ can be neglected, and the magnet location is defined just using five variables $(x_b, y_b, z_b, \alpha, \beta)$.

From now, we take the inclination of the cylinder in figure 6.9 as a reference, being described by the rotation angles $\alpha = 0$ and $\beta = 0$.

Finally, we want to define the error ε, that is the index used to assess the robustness of the configuration, and the procedure used to obtain it.

We consider the position of the cylinder which is fully determined by the set of variables $(x_b, y_b, z_b, \alpha, \beta)$, where in the real case the rotation angle around x assumes values in a certain range, $\alpha \in [-20°, 20°]$, while it can freely rotate around y, meaning that $\beta \in [-180°, 180°]$.

Then, using the mathematical model described in section 6.2 we can compute the magnetic field in the n sensors, defining n ideal measures $(B^i_{x,id}, B^i_{y,id}, B^i_{z,id})$ for each sensor i. Afterwards, we transform this ideal measures in the real ones considered to detect the position using the formulas of equation 6.14.

The real measures obtained in this way (B^i_x, B^i_y, B^i_z) are the ones taken as input by the optimization algorithm, which search for the magnet position which produces the magnetic field required, having as search space $[x, y, z, \alpha, \beta]$. Therefore, the final outcome will describe the computed localization of cylinder, defined by the values $[x_c, y_c, z_c, \alpha_c, \beta_c]$ which define the barycentre and the inclination computed.

The first indicator of the shift of the computed magnet from the ideal one is the euclidean distance ε_1 between the two barycentres computed as shown below

$$\varepsilon_1 = \sqrt{(x_b - x_c)^2 + (y_b - y_c)^2 + (z_b - z_c)^2} \tag{6.15}$$

However, the distance ε_1 does not take into account the inclination of the magnet, which is crucial, being independent from the angles α and β. Thus, we consider also the distance between the centres of

the two upper faces of the cylinder computed and the ideal one. This value, called ε_2, is a function of the two rotation angles.

The indicator considered in the next section to assess the accuracy of the localization, that is the robustness of the configuration, is given by the average of the two distances

$$\varepsilon = \frac{\varepsilon_1 + \varepsilon_2}{2} \tag{6.16}$$

Obviously the higher is the noise introduced in the measures, the greater is the error ε, for this reason we compare the different configurations introducing a random noise with the same average value, equal to 5%.

6.4 Numerical results

As we have already illustrated in the section regarding the mathematical model, the radial and the z components of the magnetic field are

$$\begin{cases} B_z = \dfrac{J}{2\pi}\left(aux1\left(\dfrac{-h}{2}, r_1, r, z\right) - aux1\left(\dfrac{h}{2}, r_1, r, z\right)\right) \\ B_r = \dfrac{J}{2\pi}\left(aux2\left(\dfrac{-h}{2}, r_1, r, z\right) - aux2\left(\dfrac{h}{2}, r_1, r, z\right)\right) \end{cases} \tag{6.17}$$

Where the functions $aux1$ and $aux2$ are defined by the equations 6.11 and 6.12, J is the magnetization of the magnet, $r1$ the radius, and h the height.

For our numerical simulations the average values for the random noise is always 5%, and the following values have been used for the permanent magnet

$$\begin{cases} J = 1\ T \\ r_1 = 5\ mm \\ h = 10\ mm \end{cases} \tag{6.18}$$

In the planar configuration the sensors form a square with a side equal to 5 cm, therefore the position on the edges does not change, while increasing the sensor numbers the density in the mesh becomes higher.

The first numerical result obtained regards the average error as function of the distance in the planar configuration, and as function of the radius in the circular one.

In this process we have fixed the position of the permanent magnet, as well as the number of sensors that was put equal to four, and changed the values of the d and r, taking into account their physical lower limits, respectively 4 and 5 centimetres. Furthermore, for each value of d and r the solving process is repeated ten times, given the random nature of the noise introduced, and the average value for the error in the ten executions is considered, and represented in figure 6.10 and 6.11.

As we could expect the accuracy decreases for greater distances and radiuses, imposing to reduce them to their lower bounds in order to contain the error. Therefore, in the next analysis we will consider the configurations with $r = 5\ cm$ and $d = 4\ cm$.

We want now to vary the number of sensors, in order to see if this delivers a better robustness, and until when is convenient to increase the number.

For this analysis we have considered six different inclinations for the permanent magnet, defined by the following couples of angles

- $\alpha = 20°$ $\beta = 90°$
- $\alpha = 20°$ $\beta = 45°$
- $\alpha = 20°$ $\beta = 0°$
- $\alpha = 10°$ $\beta = 60°$
- $\alpha = 10°$ $\beta = 30°$

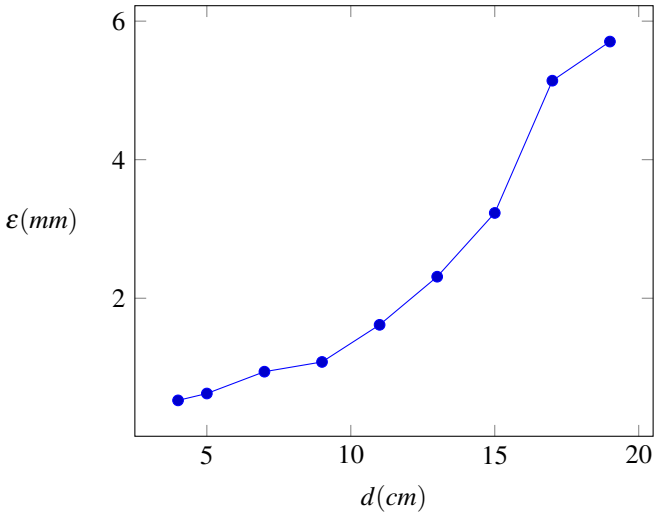

Figure 6.10: Error as function of distance in the planar configuration

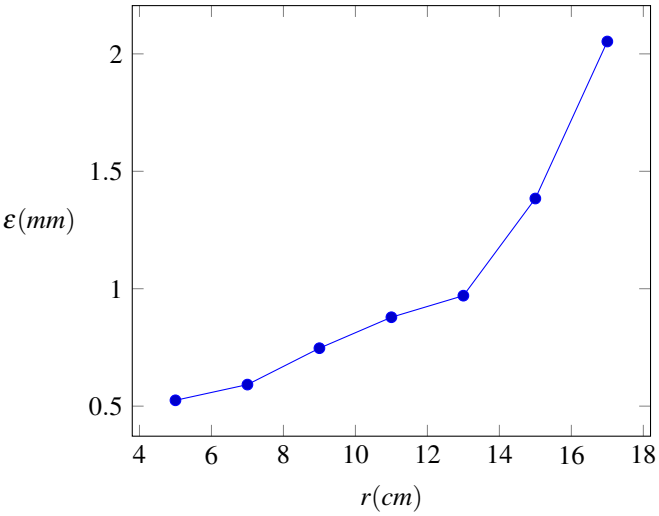

Figure 6.11: Error as a function of radius in circular configuration

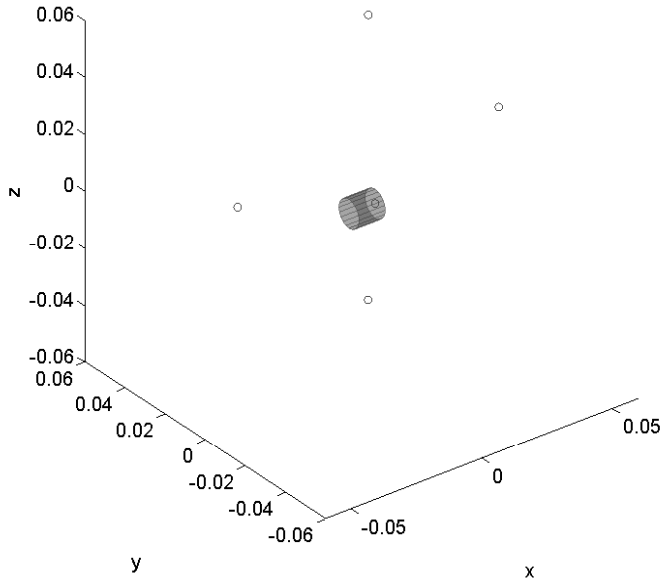

Figure 6.12: Circular configuration with 4 sensors in Matlab

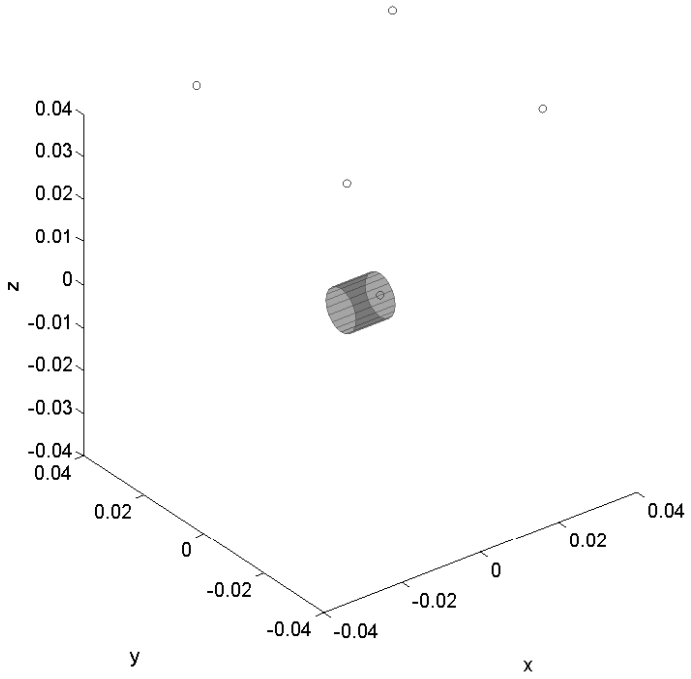

Figure 6.13: Planar configuration with 4 sensors in Matlab

- $\alpha = 0°$ $\beta = 90°$

Then, given one sensors configuration, for each inclination we execute ten times the optimization algorithm, which has as stopping criterion a number of function evaluations equal to 3000, which proved to be a value for which the objective function does not obtain further improvements.

We try to minimize the following function

$$\min \frac{\sum_{i=1}^{n} |\boldsymbol{B}_{m,i} - \boldsymbol{B}_{c,i}|}{n} \tag{6.19}$$

Where for each sensor i we compute the difference between the magnetic field measured $B_{m,i}$, obtained introducing the noise component, and the magnetic field computed $B_{c,i}$ by the algorithm which explores the five variables space $[x, y, z, \alpha, \beta]$. Then, we divide the sum of these differences for the total number of sensors n, in order to obtain an average value.

Empirically, the average value for which we obtain a decent localization for the cylinder, meaning that the magnetic configuration is physically possible, must respect the next condition

$$\frac{\sum_{i=1}^{n} |\boldsymbol{B}_{m,i} - \boldsymbol{B}_{c,i}|}{n} < 4 \cdot 10^{-5}\ T \tag{6.20}$$

Given a magnetization $J = 1\ T$ for the magnet.

Therefore, we have excluded these unfeasible results from the data used for the assessment of the error.

The error ε was computed considering the average value of the ten executions for each of the six different positions of the cylinder, and in turn computing the mean value of the six averages.

The final results are illustrated both for the planar and circular configuration in figure 6.14.

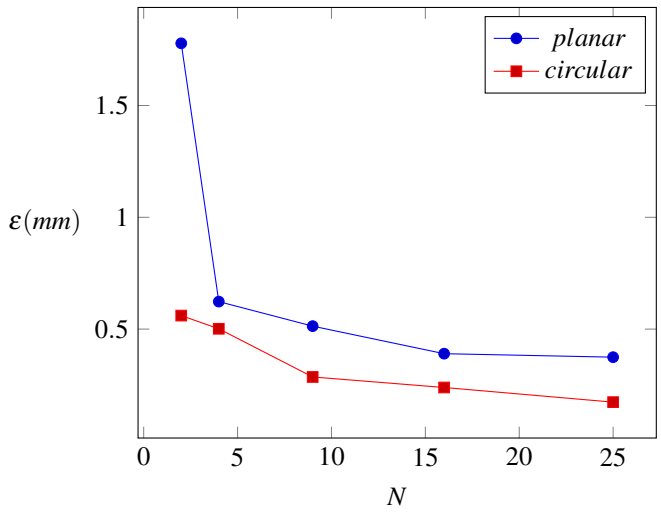

Figure 6.14: Error as a function of the number of sensors in the planar and circular configurations

As we can see the circular configuration always performs better than the planar one, an increase in the number of sensors improves the robustness, but from a number greater than nine there is just a weak amelioration.

Figure 6.15 shows the percentage of successful pose detection, that is the number of executions where the condition described in the equation 6.20 is respected.

In this case for both two the configurations when the number of sensors is greater than nine the percentage of success is above 90 %, and none of the two configurations seems to prevail on the other.

Finally, we want to study the performance for the worst case, that is the case where for each of the different inclinations of the magnet the error ε assumes the maximum value and respects the condition in 6.20. The figure 6.16 shows the average of the maximum for the six cylinder orientations.

Also here the circular configuration performs better, and it does not seem convenient to use more than 9 sensors.

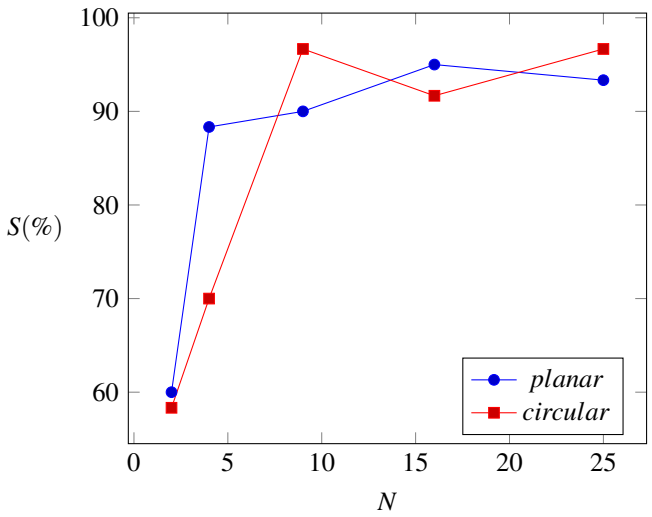

Figure 6.15: Successful cases in percentage as a function of the number of sensors in the planar and circular configurations

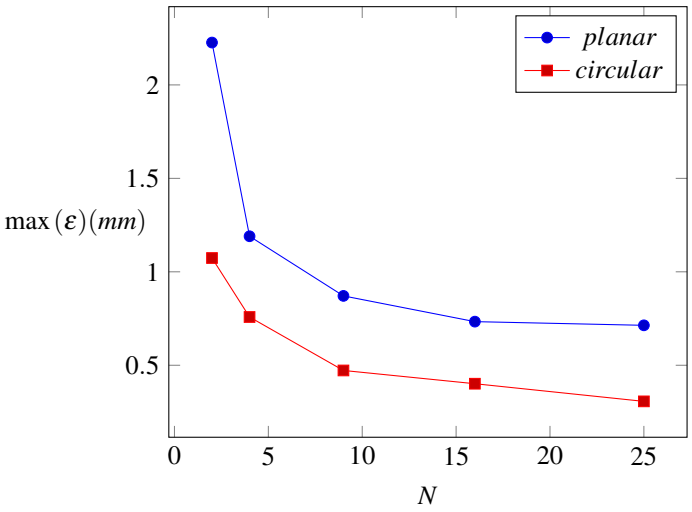

Figure 6.16: Worst case in the planar and circular configurations

Then, from these results, for a preliminary analysis we can deduce that a circular type configuration is preferable in terms of robustness, with a reliability above 90%.

It may be interesting to evaluate even more complex geometries for the sensors, to find out whether is possible or not to obtain a further improvement.

Conclusions

The Pattern Search method both for the single and multiple objectives optimization was illustrated, as well as its convergences analysis. Moreover, some innovative alternatives for the polling have been introduced which can be used as suitable approaches for specific classes of problems.

The hybrid method developed in this work composed of a Pattern Search and a Differential Evolution component has proven to outperform the traditional stochastic algorithms, being particularly aggressive, and therefore suitable for engineering applications. Furthermore, unlike the purely stochastic approaches, it keeps the convergence properties of the deterministic component, thus having strong theoretical basis.

In fact, the numerical examples have shown the effectiveness of the innovative method developed, that is a robust tool for multiobjective optimization.

A further improvement may be obtained analyzing the possibility to exploit some information of the function gathered by the Differential Evolution mechanism during the selection step, in order to adapt the following poll step in the Pattern Search component to create an even more greedy algorithm.

Finally, the algorithm was successfully implemented to an optimization problem for the design of an electromagnetic device for pose detection, particularly suitable for medical applications.

From the results gathered, the best configuration for the measuring sensors is the circular one, being the most robust and with an high percentage of successful pose detections, also indicating that beyond a certain value an increase in the number of sensors does not produce a significant performance improvement.

A possible development may be given by the study of configurations with more complex geometries, exploring other topologies, to see whether a sensible improvement in the robustness can be achieved at a relatively contained increase in the cost of the device.

Bibliography

[1] Y. Shi, R.C. Eberhart, "Empirical study of particle swarm optimization", *Proc. of the 1999 Congress on Evolutionary Computation*, vol.3, pp.1945-1950, 1999.

[2] J. Brest, S. Greiner, B. Boskovic, M. Mernik, V. Zumer, "Selfadapting control parameters in differential evolution: A comparative study on numerical benchmark problems", *IEEE Trans. on Evolutionary Computation*, vol.10, n.6, pp.646-657, 2006.

[3] P. Di Barba, *Multiobjective Shape Design in Electricity and Magnetism*, Springer, 2010.

[4] C. Noilublao, S. Bureerat, "Simultaneous topology, shape and sizing optimisation of skeletal structures using multiobjective evolutionary algorithms". In: dos Santos WP (ed) Evolutionary Computation, ISBN: 978-953-307-008-7, InTech, pp 487–508, 2009.

[5] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*, Wiley, 2001.

[6] V. Torczon, "On the convergence of pattern search algorithms", *SIAM Journal on Optimization*, vol.7, n.1, pp.1-25, February 1997.

[7] V. Piccialli, "Metodi per la soluzione di problemi non vincolati che non fanno uso di derivate", Dottorato in Ingegneria dei Sistemi, corso di Ottimizzazione Continua, Università degli Studi di Roma "La Sapienza".

[8] A. L. Custódio, J. F. A. Madeira, A. I. F. Vaz, L. N. Vicente, "Direct Multisearch for Multiobjective Optimization", *SIAM Journal on Optimization*, vol.21, n.3, pp.1109-1140, 2011,

[9] M. A. Abramson, C. Audet, J. Dennis, "Generalized Pattern Search Algorithms: unconstrained and costrained case", IMA workshop - Optimization in simulation based models, January 2003.

[10] S. Gratton, C. W. Royer, L. N. Vicente, Z. Zhang, "Direct search based on probabilistic descent", preprint 14-11, Dept. Mathematics, Univ. Coimbra, 2014.

[11] J. C. Spall, *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*, Wiley, 2003.

[12] J. Kennedy, R. Eberhart, "Particle swarm optimization", *Proc. of the IEEE Conf. on Neural Networks*, vol.4, pp.1942-1948, 1995.

[13] D. Tian, "A Review of Convergence Analysis of Particle Swarm Optimization", *International Journal of Grid and Distributed Computing*, vol.6, n.6, pp.117-128, 2013.

[14] P. Alotto, "Particle swarm optimization PSO", Università degli Studi di Padova.

[15] M. Reyes-Sierra, C. A. C. Coello, "Multi-objective particle swarm optimizers: A survey of the state-of-the-art", *International Journal of Computational Intelligence Research*, vol.2, n.3, pp.287–308, 2006

[16] R. Storn, K. Price, "Differential Evolution - a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces", Technical Report TR-95-012, ICSI, March 1995.

[17] K. Fleetwood, "An Introduction to Differential Evolution", http://www.maths.uq.edu.au/MASCOS/MultiAgent04/Fleetwood.pdf

[18] P. Alotto, "Single to multiobjective optimization (non dominated sorting)", Università degli Studi di Padova.

[19] H.J. Sun, C.H. Peng, J.F. Guo, H.S. Li, "Non-dominated sorting differential evolution algorithm for multi-objective optimal integrated generation bidding and scheduling", IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS 2009), Shangai, vol.1, pp.372-376, 20-22 November 2009.

[20] G. Caravaggi Tenaglia, L. Lebensztajn, "A Multiobjective Approach of Differential Evolution Optimization Applied to Electromagnetic Problems", *IEEE Transactions on Magnetics*, vol.50, n.2, February 2014

[21] D.H. Wolpert, W.G. Macready , "No Free Lunch Theorems for optimization", *IEEE Transactions on Evolutionary computation*, vol.1, n.1, pp.67-82, April 1997.

[22] A. L. Custódio, J. F. A. Madeira, A. I. F. Vaz, and L. N. Vicente, "Direct Multisearch for Multiobjective Optimization", CERFACS, 30 September 2011.

[23] E. Zitzler, K. Deb, L. Thiele, "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results", *Evolutionary Computation*, vol.8, n.2, pp.173-195, 2000.

[24] Q. Zhang, A. Zhou, S. Zhao, P. N. Suganthan, W. Liu, S. Tiwari, "Multiobjective optimization test instances for the CEC 2009 special session and competition", Technical Report CES-487, The School of Computer Science and Electronic Engineering, University of Essex, Tech. Rep., 2008.

[25] "Wikipedia, l'Enciclopedia Libera" http://www.wikipedia.com, 2015

[26] V. Pankrac, J. Kracek, "Simple Algorithms for the Calculation of the Intensity of the Magnetic Field of Current Loops and Thin-Wall Air Coils of a General Shape Using Magnetic Dipoles", *IEEE Transactions on Magnetics*, vol.48, n.12, pp.4767-4778, December 2012.