POLITECNICO DI MILANO
DEPARTMENT OF ENERGY
Ph.D. in Electrical Engineering

Social Network Optimization for Electrical and Electromagnetic Advance Optimization Systems

Supervisor: Prof. RICCARDO ZICH
Tutor: Prof. FRANCESCO GRIMACCIA

Thesis of:
ALESSANDRO NICCOLAI
Matr. 878819

Academic Year 2019 - 2020

A Mamma e Babbo
Ai miei Nonni

Contents

List of Figures

List of Tables

Ringraziamenti

Ringrazio il professor Riccardo Zich che mi ha accompagnato per anni nel mio percorso universitario e di ricerca. Ringrazio, rispettando l'ordine alfabetico, tutti i membri del mio gruppo di ricerca per il sostegno ed i consigli: Alberto, Emanuele, Francesco, Marco e Sonia.

Approfitto dell'occasione per ringraziare anche tutte quelle persone che, pur non avendo direttamente contribuito allo sviluppo di questa tesi, mi hanno seguito ed aiutato nel corso degli anni.

Voglio dedicare un ringraziamento particolare a Nonna Gilda e Nonno Celestino, e a Mamma e Babbo. Ringrazio il resto della famiglia per il loro supporto.

Ringrazio tutti gli amici che mi sono sempre stati vicini: Alessia ed i suoi consigli fondamentali per la mia sopravvivenza in Cina, Alice ed i suoi suggerimenti, Andrea e le giornate a Ribera e le chiacchierate davanti ad una birra, Annalisa sempre presente dal primo anno di università e Mimmy e Carolina, le mie sorelline veronesi che, pur conoscendomi bene, continuano a manifestare un affetto profondo.

Ringrazio Lorenzo per l'amicizia che abbiamo costruito in questi anni e per le lunghe passeggiate in montagna. Ringrazio inoltre Nicolò che con le sue frequenti telefonate e il jogging a Verona ha mantenuto salda e viva la nostra amicizia.

Ringrazio Tobia e le lunghe serate su Skype, le digressioni filosofiche e giuridiche, i ponderati consigli e le spericolate sciate in montagna.

Infine, ringrazio Roberta per il suo amore e la sua pazienza.

Executive summary

In recent years, the use of Computational Intelligence techniques is growing in the engineering design: methods as Evolutionary Optimization Algorithms as well as Artificial Neural Networks are becoming more common and better known.

The research on Evolutionary Algorithms (EAs) is already established: many important journals and international conferences are devoted to this topic. This research changed its behaviour in the years: at the beginning the focus was on the creation of a general algorithm able to solve all the problems. Since the demonstration and the publication of the *No Free Lunch Theorem for Evolutionary Optimization* [1], the nature of the research is drastically changed: in fact, this theorem demonstrates the impossibility of implementing the best algorithm.

The new research trend can be divided into two main parts: on one hand, many researchers are devoting efforts in the implementation of effective algorithms for the solution of *multi-objective* and *many-objective* problems, *i.e.* problems in which there are more than one conflicting targets that should be solved at the same time. On the other hand, the trend is focused on the implementation of *ad-hoc* algorithms for the effective solution of a specific problem [2]. This second trend is less popular because it requires a highly multi-disciplinary approach to the research: both algorithmic experts and experts on the problem physics should interact. For what concern antenna design, the use of Evolutionary Optimization algorithms is very popular, due to the high non-linearities and the presence of many local minima [3]. Among all the antenna, the design of reflectarray is highly suitable for Evolutionary Optimization: in fact, for achieving special performance in terms of radiation pattern, no deterministic solutions are available.

Evolutionary Optimization algorithms are widely applied also in electrical engineering system. In [4] the Particle Swarm Optimization is used for designing the electricity distribution network: in particular, it is exploited for finding the optimal capacitor allocation in a network with high wind generators. In [5] the Genetic Algorithm has been applied in the optimiza-

tion of a permanent magnet synchronous generator: in this case, the system performance are calculated by means of an analytical model during the optimization and, only at the end of the process, the FEM software is used for assessing the solution.

The aim of this thesis is to analyse the entire optimization system, considering all its parts and understanding their effects on the final result of the optimization. This activity is rarely performed in literature and in the most of the cases, Evolutionary Algorithms are considered black boxes. This often lead to results that are not as performing as they can be.

The thesis is divided in four main chapters: after an Introduction, in Chapter 2 the optimization system is analysed and its parts are shown as well as their interactions. The idea is to understand the possible system architecture that can be used for different problem.

The third chapter is focused on Evolutionary Optimization Algorithm and, in particular, on Social Network Optimization. This algorithm has been designed, implemented, improved and tested in my PhD program. The analysis of the algorithm represents an activity that can be done on all the EAs and that is very useful for being able to have the better performance of the algorithm in different applications. Among the analyses performed, two of them should be underlined: the parametric analysis that has been correlated with a theoretical explanation of the convergence proprieties of the algorithm, and a deep comparison among different algorithms.

In particular, the comparison is done among seven EAs, two random algorithms, and four standard local optimization algorithms. Twelve different benchmarks have been used in this study.

In the following two chapters, the optimization systems for two different applications have been analysed: in both the system performance should be calculated with numerical methods, as Finite Element Methods. The direct optimization on FEM models is hardly applicable, so different approaches have been tests.

The first analysed problem is an high frequency electromagnetic system: the optimization of a reflectarray designed for having scanning capabilities. This problem is characterized by a very high number of design variables (148) and the objective function is highly non-linear.

The problem optimization has been approaches with an analytical model, and at the end of the process the final solution has been assessed with a commercial FEM software.

This problem presents more performance parameters that should be optimized at the same time: for facing this, all of them have been summarized in a single cost value by means of a linear combination. The coefficients of this combination have been investigated, as well as the feasibility function

used for managing the boundary of the search space.

Moreover, an analysis of the parameters of Social Network Optimization has been performed and, eventually, a comparison among a large set of EAs.

The second analysed problem is a low-frequency magnetic problem in which the design af a die mold is analysed.

Two different FEM software have been used: the first one is faster and less accurate. With this software a deep analysis on the search space has been performed and it results in an adaptive approach that is able to provide very good results.

The second software is Comsol Multiphysics, that is much more accurate, but the computational time required for the calculation of the system performance is very high. With this second software, the use of surrogate model is investigated, and the results compared with the first FEM software. With both the two FEM models, all the EAs used in this thesis have been compared.

Chapter 1

Introduction

Optimization is a key aspect of the design of engineering systems. It can be faced with different approaches accordingly to the system nature and its complexity: in many cases the *divide et impera* approach is enough for having an optimal or quasi-optimal solution of the problem. This fails completely when the interaction among the part of the systems is very strong.

This reflects the definition of complexity: a system is defined complex when there are some emergent proprieties or behaviour often created by the mutual interaction among its parts. Complex systems are very hard to be solved and often their solution cannot be achieved deterministically [6].

Another common design method is the *trial and error*: it is very often used in complex systems because the reiterated tentative design takes into account all the interacting parts of the systems as a whole. The main difficulty of this approach is due to the fact that it takes a lot of time and the final solution is often sub-optimal.

The complex models used for simulating engineering systems, that provide the capability to assess accurate phenomena, require a fast and effective optimization process, which objectives range from geometric and performance optimization up to joint optimization involving also the economic aspects.

Among all these approaches, those involving multiple goals are the most interesting because they allow to have an overview of the final object. However, they are the most complex ones since they often result in multiple equivalent sub optimal solutions. The involvement of conflicting goals makes necessary to identify appropriate objective functions to take into account most of the design requirements. In this context, advanced computational intelligence algorithms can be used to find out the optimized design, involving a large number of physical and geometric parameters, and to maximize the performance of electrical machines and energy-harvesting devices. These procedures are population-based iterative techniques which basically perform

an indirect synthesis by evolving the parameters of interest to identify one optimal solution in the design space, through properly defined single and multi-objective fitness functions [7].

The most popular evolutionary algorithms, i.e. Genetic Algorithm (GA) [8] and Particle Swarm Optimization (PSO) [9] have been combined in the last decade with a broad range of other soft computing techniques, like, for example, artificial neural networks [10], fuzzy systems [11], giving birth to a large discipline of hybrid methods which constitute the so-called Computational Intelligence.

In recent years, many other optimization algorithms have been implemented in order to have performances better than GA and PSO. These two algorithms defines the two most important classes of EAs: the Darwinian algorithms, that emulates the species evolution, and the Swarm Intelligence algorithms, that emulates the behaviour of groups of particles [12].

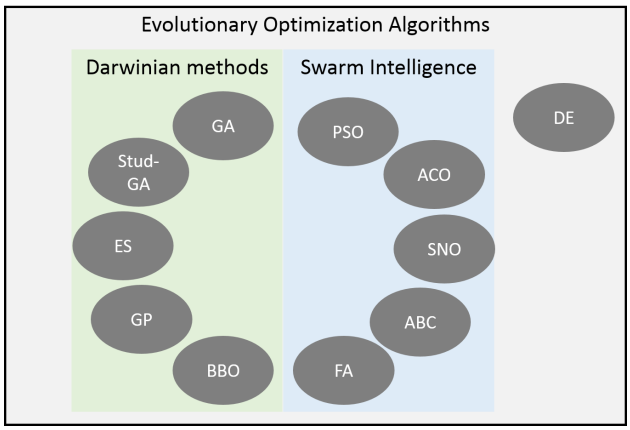


Figure 1.1: Classification of the main EAs.

Figure 1.1 shows the classification of the algorithms: the Darwinian algorithms includes Genetic Algorithms (GA), Stud-GA, the Evolutionary Strategies (ES), the Genetic Programming (GP) and Biogeography Based Optimization (BBO), while Swarm Intelligence algorithms includes Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Social Network Optimization (SNO), Artificial Bee Colony (ABC), and Firefly Algorithm (FA). Differential Evolutionary (DE) is not a biologically inspired algorithm.

Among the numerous algorithm, it is important to remember the Biogeography Based Optimization (BBO) [13] that has been also applied in the field on antenna optimization [14, 15]. Differential Evolution (DE) is another EA that shows very good performance in many problems [16], even if it is

not very widely applied because often its results are affected by a high standard deviation for non-separable problems [17]. Estimation of Distribution Algorithms (EDAs) have shown very good performances both in standard benchmarks [18] and in antenna optimization [19].

Performance improvements can be achieved properly modifying some operators of GA and PSO. An example of it is the Stud-Genetic Algorithm that outperforms the basic GA on a large set of benchmarks [20]. Modifications of the PSO have been widely applied in literature, like the Meta-Swarm PSO [21].

Other variations of the PSO regard the type of problems that is can face: for example, in [22] a specific implementation for discrete problems have been tested. PSO has been also modified for solving binary problems [23, 24].

Among all the antenna configurations, reflector antennas have been exploited for the high demand of radar and satellite communication, where a point-to-point connection was needed and, consequently, a high gain was required. There are many shapes of reflectors in literature: curved, planar, corner but the most utilized are the parabolic [25].

In order to reduce the space required by the reflector, especially in aerospace applications, Reflectarray Antennas have been introduced: in fact, they have low weight, low profile and the possibility to be easily folded [26].

Reflectarray Antennas (RAs) consist in a planar array made up of different re-radiating elements illuminated by a feed source (typically a horn antenna) placed in central or offset position. In these antennas, changing one or more geometrical parameter of each patch it is possible to control the phase of the re-radiated field changes and to obtain the desired radiation pattern. They can be designed to radiate a shaped, contoured or multiple beam [27, 28].

A recent problem in reflectarray antenna design is to improve their scanning capabilities, *i.e.* the possibility to modify the direction of the radiation pattern main beam. This can be achieved with different approaches, like active reflecting elements or with a re-orientation of the feeder [29].

For what concern antenna design, the use of Evolutionary Optimization algorithms is very popular, due to the high non-linearities and the presence of many local minima [3]. The optimization of beam-scanning reflectarray has been faced with different optimization techniques: for example, in [30] a bi-focal reflectarray antenna has been designed with a multi-objective implementation of PSO.

Evolutionary Optimization algorithms are widely applied also in electrical engineering system. In [4] the Particle Swarm Optimization is used for designing the electricity distribution network: in particular, it is exploited for finding the optimal capacitor allocation in a network with high wind

generators.

In smart grid context, the optimization becomes even more important. In fact, the optimization can be related to single building energy management [31], communication systems [32], demand management [33], load forecasting [34], and many other applications.

Also in the field of electrical machines, the application of Evolutionary Optimization Algorithms gives very good performance. In [35] the Particle Swarm Optimization has been applied to the optimization of a permanent magnet linear generator, simulated by means of an analytical model.

In [5] the Genetic Algorithm has been applied in the optimization of a permanent magnet synchronous generator: in this case, the system performance are calculated by means of an analytical model during the optimization and, only at the end of the process, the FEM software is used for assessing the solution.

In literature, a large space is also devoted to the application of existing algorithms to design problems: in this case, the attention is focused on the results of the optimization in terms of system performance or on the comparison of different existing techniques. For example, in [36] different variations of the Differential Evolution are tested in antenna problems, showing the high optimization capabilities of the algorithm.

In order to increase the reliability of the results of the optimization, more complex systems are tested. In [37] and [38] a new combination between Evolutionary Algorithms and surrogate models is tested for achieving a proper trade-off between accuracy of the result and total computational time required. In [39] the trade-off between computational time and accuracy is achieved with different surrogate models, while in [40] the ensemble approach is used.

Computational intelligence techniques have been widely applied in recent years both to power systems and to robotics. For example, Biogeography-Based Optimization (BBO) for emission dispatch problems [41], real-coded GA for reactive power control and smart grid optimization [42], multi-objective PSO for navigation of robots [43]. Moreover, some search-based optimization techniques have been also applied to manage battery SoC and generator delivered power in hybrid electric vehicles [44], but still limited examples can be found for electrical machine design, e.g. Differential Evolution (DE) [45] and Genetic-Swarm hybrid algorithms [46]. All these methods are mainly based on iterative procedures with a strong stochastic base, and their performance must be evaluated in terms of speed of convergence and computational burden. In fact, these techniques are suitable when the device structure is complex, as is the case of electrical machine design, which often requires time-consuming and non-linear FEM simulations. To address this issue, surrogate

models are used to speed-up global evolutionary search [47].

Many attempts have been made using Computational intelligence techniques both in power systems and, more recently, in robotics. For example, Biogeography-Based Optimization (BBO) for emission dispatch problems [41], real-coded GA for reactive power control and smart grid optimization [42], multi-objective PSO for navigation of robots [43].

Most of these techniques have been used to optimize complex configurations of various electric motors and generators, giving birth to a particularly broad and detailed literature. For example, in [48] a multiobjective approach is proposed to have higher power density and space utilization in planar motors; in [49] evolutionary algorithms are used for optimizing the design of permanent magnet motors with different winding configurations and cooling systems; in [50], a novel automated design procedure is applied to the optimization of synchronous reluctance machines. All these methods are mainly based on iterative procedures with a strong stochastic base, e.g. Differential Evolution (DE) [45], Particle Swarm Optimization (PSO) [51, 52], and hybrid algorithms [46], and their performance must be evaluated in terms of speed of convergence and computational burden. In fact, these techniques are suitable when the device structure is complex, as in the case of electrical machine design, which often requires time-consuming and non-linear FEM simulations. To address this issue, surrogate models are often used to speed-up global evolutionary search [47, 53, 54]. A special attention is paid to the study of linear machines and in particular the optimization of Tubular Linear Generator (TLG), which are taking place in many energy harvesting applications [55–57]. The optimization of the produced power is crucial in such a type of problems which require the solution of different interacting physical domains.

In this context, the optimization of Antennas Arrays is often faced with EAs [58]. For example, Genetic Algorithms (GA) have been widely applied to antenna optimization [3], both in binary problems and in real-value problems [59]. Also Particle Swarm Optimization (PSO) [60] has been widely used in antenna optimization in its basic implementation [61–63] or with some variants as Meta-Swarms [21], Black-Hole PSO [64] and others [65,66]. Most of the problems faced with PSO are real-value problems: in fact, its operators have been designed and are suitable for this kind of problems. There are some modifications of PSO that have been introduced to deal with binary problems [23, 24]. They are often indicated by bPSO, but their application is not so wide as for GA.

Since optimization is an important tool for system design and it is widely applied in many fields, some preliminary applications of SNO and some parts of this thesis have been already published. In [67] a comparison between

SNO and other EAs is performed and then the former is applied to the task allocation problem in a Wireless Sensor Network. A preliminary analysis on that problem has been presented in one of the most important conferences on Evolutionary Computation [68]. For what concerns the analysis of working principle of the algorithm, a book chapter has been published [69]. SNO has been tested on the problem of parameter matching for Photovoltaic five parameters model [70]. In [71] SNO has been applied to the optimization of a Tubular Permanent Magnet Linear Generator, and it has been preliminary compared with Genetic Algorithm and Particle Swarm Optimization on the TEAM25 problem. Finally, several conference papers have been presented on the use of SNO in antenna applications. In these, the analytical model presented in this thesis has been iteratively improved and its accuracy has been increased with respect to the FEM software [72, 73].

This thesis is structured as follows: Chapter 2 contains a critical analysis of the optimization system. Chapter 3 presents the implementation, the analysis and the assessment of Social Network Optimization. In Chapter 4 the optimization of a beam-scanning reflectarray has been investigated analysing many parts of the entire process, and in Chapter 5 a standard electromagnetic benchmark, the TEAM25 problem, has been addressed with advanced evolutionary techniques, analysing the effect of different FEM software. Finally, in Chapter 6 some conclusions are drawn.

Chapter 2

Optimization Systems

The *optimization system* is the structure composed by several elements that is used to solve efficiently an *optimization problem*.

The optimization problems are mainly composed by three elements:

- some design (or decision) variables: they are the set of geometrical or physical parameters that can influence the behaviour of the systems that should be optimized. There are three main types of design variables: real variables can assume any vale within a range; discrete or integer variables cannot have fractional part; binary variables can assume only one of the two values $\{0, 1\}$;

- a set of constraints, that makes only a part of the search space feasible. The constraints can be equality or inequality. The simplest inequality constraints are the ones that defines the lower and upper limits for the design variables;

- the objective function, that calculates the performance of the analysed system and return some performance parameters.

The optimization problems can be linear or non-linear, depending on the definition of the objective function. For linear problems there are some specific techniques that have been implemented: these algorithms exploit the linearity of the problem for finding the optimal set of design variables (global optimum of the objective function) [74].

For what concern non-linear problems, algorithms used to solve them are classified under the name of *non-linear programming* (NLP) [75, 76].

A generic NLP is based on a transition rule: it defines how the *candidate solutions* (the tested feasible points) moves into the search space.

There are two sub-classes of NLP algorithms: the point-based, in which at each iteration only one candidate solution is tested, and the population-based, in which a set of solutions are tested in the same iteration [77].

The *point-based* algorithms are based on an initial guess ($\mathbf{x}_0$) that is moved into the search space by means of the transition rule. Due to high influence of the initial guess, these algorithms should be started many times for a proper identification of the global best of the objective function (multi-start algorithms). The main characteristics of these algorithms are:

- Local search: the search area is highly influenced by the starting guess; thus, it is possible to state that the optimization is performed only in a sub-part of the entire search domain. Due to this reason, these algorithms are often blocked in local minima of the function.

- Low memory required: due to the fact that only one solution is stored at each time, the required memory is very low.

- No parallelism is possible for a single trial: while analysing only one trial (start) of the algorithm, at each iteration only one point is evaluated, and it is not possible to parallelize the iterations.

- For these algorithms it is often possible to develop an easy theory for their working mechanism.

On the other hand, *population-based* algorithms are based on an initial population of candidate solutions ($\overline{\mathbf{x}}_0$) that is evolved in the search space creating at each iteration a new population. Due to the presence of many points in the initial population, these algorithms are much less affected by the specific selection of the initial population. Their main characteristics are [76]:

- They can be considered global optimizer because they can optimize in one trial in the entire search space. This is also achieved by the introduction of stochastic operators in the optimizer, thus given an initial population, different runs of these algorithms give different results.

- High memory required: at each iteration a population of candidate solutions should be stored in the memory; for high-dimensional problems (that often requires also high dimensional populations) this fact can be a limitation.

- They are suited for parallel processing. This is due to two reasons: the first one is related to the presence of a population of individuals that

should be evaluated at each iteration. These can be easily parallelized [68]. The second one is an implicit parallelism of population-based algorithms [78, 79].

- There are no easy theories that can explain the working principles of population-based algorithms. The convergence is generally proven by using a set of tests on standard benchmarks.

It is important to remember the *No Free Lunch Theorem*, that states that all the algorithms perform the same if averaged on all the possible functions [1]. This theorem can be bypassed because no one is interested in all the possible objective problems, but only in a narrow class.

However, this theorem originated a new research field: in fact, instead of finding the best generic algorithm (that is demonstrated not existing), the research has been focused on the customization of the algorithms, often using heuristics.

This customization is particularly interesting in Evolutionary Optimization Algorithms (EAs) because they have three very important advantages with respect to point-based algorithms:

- They naturally provide more than one single final solutions: this aspect is fundamental in multi-objective optimization but can be really important in the practical use of the algorithm because they provide to the designer a set of different (quasi) optimal solutions among which selecting the final design. This is very important when some performance indexes cannot be easily inserted in the objective function.

- EAs are very flexible in their use: they have no requirements in terms of derivability or continuity of the function; moreover, they can deal easily with constraints.

- The operator of EAs can be easily customized for the specific optimization problems (see the applications of Genetic Algorithm to the Travelling Salesman Problem [80]).

An interesting use of EAs is related to *hierarchical optimization*: different levels of optimization can be designed exploiting the best features of population-based and point-based optimization. An example of this application can be the use of EAs for finding feasible parts of the research domain for highly constrained problems and then apply point-based algorithms for finding the optimal solution.

Another interesting application of the combined use of EAs and point-based algorithms is the research of interesting parts of the domain with EAs and then a fine-tuning of the solution with point-based algorithms.

A practical and effective use of EAs is composed by the three following steps:

1. A first run for analysing the main features of the objective function This is aimed to find the best algorithm that can optimize the system and the best way for dealing with constraints or more performance parameters.

2. Performing the so-called *improvement runs* that are aimed to improve the design of the optimization system in all its detail.

3. A final run for finding the final optimal solution and then this solution should be analysed.

Figure 2.1 shows a possible schematization of optimization system architecture. Its two pillars are the optimization problem and algorithm. Both have several elements that should be designed to improve the overall system performance.

These two pillars interact by means of two connections: one from the algorithm to the problem composed by the box domain condition and the variable decodification; the second one from the problem to the algorithm and it is composed by the cost function evaluation.

The part composed by the optimization problem, the box domain condition, the variable decodification and the cost function is called in this thesis objective function.

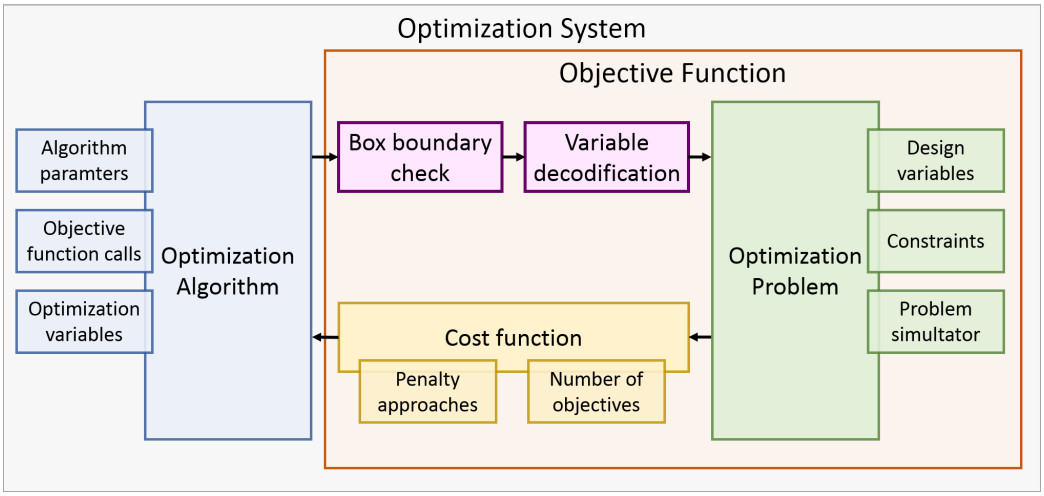


Figure 2.1: Schematization of a possible optimization system architecture.

In the following of this chapter the optimization environment is analysed with much more details: the two main blocks (the optimization algorithm and the problem) will be inspected and then their relation is analysed.

2.1 Optimization problem

The optimization problem is one of the two most important elements of the optimization system. It is the function that solves the real system: the input is a set of design variables $\mathbf{d}$ and the output is a set of performance parameters $\mathbf{p}$:

$$\mathbf{p} = f(\mathbf{d}) \tag{2.1}$$

The optimization problem can be divided in more parts, all affecting the optimization process. The first one is the system simulator used for solving the physical system: it can be an analytical model implemented in a programming language, a numerical model (such as Finite Element Method models), an experimental setup, or a surrogate model.

Each one of them have several impacts on the optimization process. The accuracy of the model with respect to the real system is a key feature that should be considered: the optimizers can find a final solution that has not good performance in the real system because the accuracy for that solution is low.

In this framework it is important to keep into considerations the hypothesis of the adopted models: in some cases these can be added into the optimization system as constraints; in others, they can be neglected in a first step and then considered by the designer selecting from the final population of the algorithm the most interesting and feasible solution.

The accuracy is very important also for surrogate models: in this case also the sampling points distribution used for the creation of the model is critical because it influences a lot the model capability to be accurate in evaluating the candidate solution of the optimizer. The surrogate models are analysed in the last section of this chapter.

Another aspect that should be considered is the computational time required by the model because it is indicative of the total optimization time. In fact, the computational effort of the optimizer is often negligible with respect to the one required for the evaluation of the candidate solutions. Due to this reason, in this thesis the number of objective function calls has been always selected as termination criterion for the search.

The model can also influence the non-linearities of the optimization function, but this is a very hard aspect to be considered since it depends on the specific problem analysed.

The optimization problems often contain several constraints in the solution. Different constraints can be found in the optimization problems: equality constraint or inequality. The first type is often hard to be directly approached, thus in many cases it is reduced to an inequality constraint relaxing the requirement.

The simplest type is a limitation on the upper and lower bound of the design variable. This constraint, called box constraint, is almost present in every engineering problem and it can be easily managed because the optimization algorithms very often works with a predefined limits of their optimization variables. All the possible set of design variable within the box domain is called *search space*.

The second type of constraint regards limitations of some functions of design variable (*e.g.* combination of them): this is another common type of constraint. It can be managed in the optimization system in two different ways: the simplest approach is to associate a cost component to the violation of the constraint (*penalty approach*). Another way is to design specific algorithm operators for creating at every iteration feasible solutions: the applicability of this approach is highly influenced by the type of optimization algorithm and by the specific constraint.

The third type of constraint are limitations on specific outputs of the system (*e.g.* imposing the passing band of a filter in the minimization of its in-band ripple). In this case the problem can be approached creating more cost components (multi-objective approach) or with multi-layer optimization.

In general, it is always better to select the design variables such that the number of constraints is reduced, if the problem allows it.

The output of an optimization problem is a set of performance parameters: they can contain one or more objectives and, if constraints are present, their violation level.

This set of parameters is then given to the *cost function* that process it for returning to the optimization algorithm the cost value to be minimized.

2.2 Cost function

The cost function is one of the two interfaces between the optimizer and the optimization problem: it translates the performance parameters into one or more cost values. Mathematically it is expressed as:

$$\mathbf{c} = g(\mathbf{p}) \tag{2.2}$$

If the output is only a single cost value, the problem is a *single-objective* optimization, otherwise it is called *multi-objective* optimization. Often, if the

number of costs to be optimized is greater than 4 or 5, the problem is called *many-objective*.

In this thesis, the attention is focused only on single-objective problems, even if all the EAs have been implemented in literature also for multi-objective and many-objective problems.

Single-objective algorithms can optimize also several conflicting objectives at the same time with a scalarization approach: a weighting factor is considered, and the final cost is a weighted sum of the different objectives.

A proper definition of the cost function can be used for dealing with the constraints. This is called penalty approach because the solutions in which one or more constraint are violated have a second cost term.

This second cost term can be either constant or proportional to the violation level: this last case is the most common because the optimizer is pushed toward the feasible region of the search space.

It is important to design properly the penalization term for avoiding two possible conditions: the first one is when a feasible solution have a higher cost than an unfeasible one. The second condition is the creation of significant local minima that are not present in the original function.

The cost function can take into account also the *multi-layer optimization*, in which the optimizer firstly solves a part of the problem and then a second one. Even in this case, it is important to avoid that bad scaling of the two problems lead to undesired artificial local minima.

2.3 Optimization algorithm

The optimization algorithm is the second fundamental pillar of the optimization system: it is an iterative process that modifies a set of solutions for finding the global best of the function. It solves the following problem:

$$\mathbf{x}_{opt} = \arg\min_{\mathbf{x}\in\mathbb{D}}(F(\mathbf{x})) \tag{2.3}$$

where $\mathbf{x}$ is the set of optimization variables, F is the objective function and $\mathbb{D}$ is the optimization variables domain. The problem here considered is unconstrained because in this formulation the constraints are included in the objective function cost definition.

The algorithms differs one from the other by the number of candidate solutions at each iteration (point-based for 1 candidate solution at time, population-based for more), and by the *transition rule*, that is the transformation of the candidate solutions from an iteration to another.

Figure 2.2 shows a possible schematization of a generic optimization algorithm. This schematization can be valid both for point-based and for

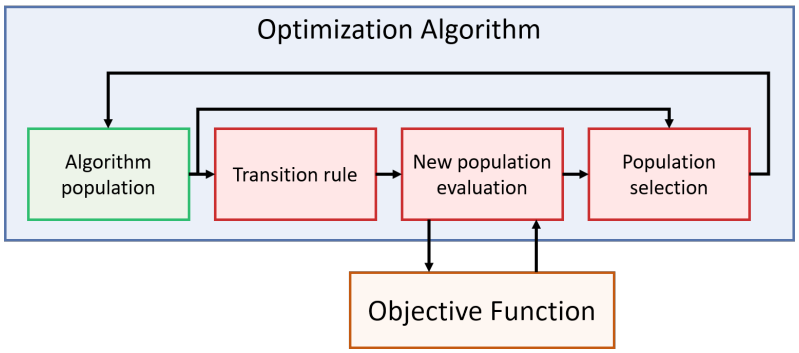


Figure 2.2: Schematization of a generic optimization algorithm.

population-based algorithms, even if the figure presents the typical nomenclature of population-based algorithms.

There are three very important operators in the algorithm: the transition rule that is the real heart of the algorithm. The population evaluation is the interacting part between the optimization algorithm and the objective function. Finally, the population selection is the operator that keeps the population size constant: in fact, at each iteration, it selects from the initial population and the new population the solutions that will belong to the next iteration population.

An important aspect of this last operator is the concept if *elitism*: in fact, many optimization algorithms keep the best solution found from an iteration to the following one.

The other important elements of the optimization algorithm that should be decided for designing the optimization systems are the algorithm parameters, the number of objective function calls and the optimization variables (see Figure 2.1).

The *algorithm parameters* are all the coefficients that tune the functioning of the operators. In EAs, they are very important because they highly affect the convergence capabilities of the algorithm. Among them, the most important one is the *population size*, *i.e.* the number of individuals that survive at every iteration.

The population size tunes the trade-off between *exploration* (the capability of the algorithm to introduce in the population new information: it corresponds to the capability of inspecting a large part of the search space) and the *exploitation* (the capability of the algorithm to use the available information to improve the candidate solutions).

The algorithms works with the *optimization variables* that in general are different from the design variables of the problem. This difference is

very important for avoiding the need of scaling all the algorithm parameters accordingly to the search space size. Moreover, this gives some degrees of freedom in the codification of the problem: more details on this are provided in the next Section.

The optimization variables can be real, integer or binary. The most known EA, the Genetic Algorithm, has been firstly implemented with binary optimization variables. By means of a proper translation between the optimization variables and the design variables it is possible to solve real or integer problems also with binary algorithms, even if it can be inefficient.

There are many specific binary or real algorithms, even if in most of the cases for an algorithm both the two implementation have been tested (Genetic Algorithms is a good example of this, even if also Particle Swarm Optimization, native for real value, has been implemented and used with binary variables).

Few algorithms are specifically designed for integer variables: in most of the cases they are specific implementations of Genetic Algorithms in which the operators are specifically designed for the optimization problem for creating always feasible solution.

2.4 Problem codification

The problem codification is the selection of the design variables and their relations with the optimization variables.

The problem codification is one of the most important aspects in the optimization system design: it affects the non-linearities of the function, the number of optimization variables, their type, and the number of constraints.

Within the problem codification framework there is the variable decodification procedure, *i.e.* the translation from optimization to design variables.

Distinguishing the two kind of variables is important for several reason. Firstly, in this way it is possible to solve problem with a specific type of variables (integer or real, for example) with an algorithm that works with another type (mainly binary or, in some cases, real).

In this way, it is possible to solve integer problems with algorithm that are not specifically designed for them, increasing the flexibility of Evolutionary Optimization.

Another important aspect of the decodification procedure is the differentiation between the objective function search space $\mathbb{S}$ and the optimization variable domain $\mathbb{D}$. In fact, for real variable algorithms, $\mathbb{S}$ is usually the range $[0, 1]$. This makes the algorithm parameters valid for many optimization problems.

2.5 Box domain condition

This function is in charge to keep the optimization variables the search space $\mathbb{S}$, mainly taking into account the lower and upper limits for the design variables. For what concern the other constraints, they are often and successfully managed within the cost definition with a penalty approach.

There are several approaches to the solutions that are outside the box boundary domain. The first one considers the boundary as an impenetrable wall. If one or more components of the candidate solution exceed the limit, they are curtailed in the search space $\mathbb{S}$ and all the other components are not altered. This kind of feasibility boundary is very useful if the optimal solution can be close to the search space limits; On the other hand, the exploration is reduced. An exemplification of this boundary condition is shown in Figure 2.3(a).

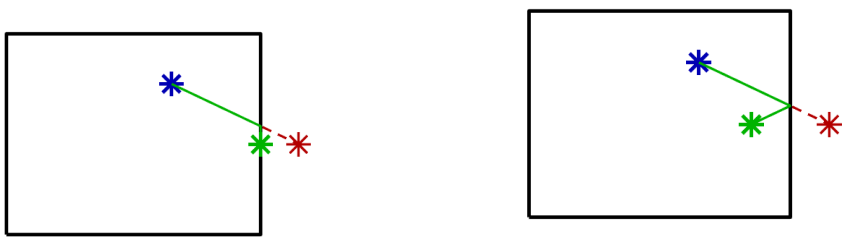
The mathematical formulation of this boundary condition is the following:

$$\tilde{x}_i = \begin{cases} L_i, & x_i < L_i \\ U_i, & x_i > U_i \\ x_i, & \text{otherwise} \end{cases} \tag{2.4}$$

where $\tilde{x}_i$ is the i-th component of the modified candidate solution, x_i is the candidate solution, U_i is the upper bound for the i-th component and L_i is the lower bound.

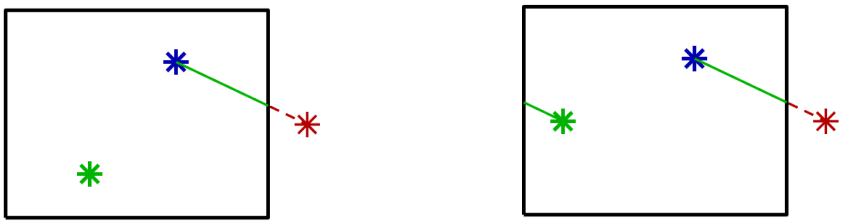
Another approach is to model the boundary as an elastic bound. If one or more components of the candidate solution exceed the bound, they are reflected inside the domain accordingly to the following rule:

$$\tilde{x}_i = \begin{cases} L_i + |L_i - x_i|, & x_i < L_i \\ U_i - |U_i - x_i|, & x_i > U_i \\ x_i, & \text{otherwise} \end{cases} \tag{2.5}$$



(a) Rigid wall. (b) Elastic wall.

Figure 2.3: Example of feasibility check: rigid and elastic wall conditions.



(a) Eliminating wall. (b) Closed search space.

Figure 2.4: Example of feasibility check: eliminating wall and closed search space.

In this condition, the exploration is slightly increased; however, the capability of finding the best on the search space limits is drastically reduced. This boundary condition is schematized in Figure 2.3(b).

Another possibility is to eliminate the solutions that goes outside the search domain, then a new random solution is created in the domain. In this case the exploration is drastically increased, even if the convergence can be worsened for the algorithms that works with trajectories, like the Particle Swarm Optimization. In fact, this boundary condition completely destroys the trajectory. Its mathematical formulation is:

$$\tilde{x}_i = \begin{cases} r, & x_i < L_i \\ r, & x_i > U_i \\ x_i, & \text{otherwise} \end{cases} \tag{2.6}$$

where r is a random value inside the search domain.

Figure 2.4(a) shows this condition.

Then, it is possible to define the search domain as it is a closed surface, and the boundary can be written in the following way:

$$\tilde{x}_i = \begin{cases} U_i - (L_i - x_i), & x_i < L_i \\ L_i + (x_i - U_i), & x_i > U_i \\ x_i, & \text{otherwise} \end{cases} \tag{2.7}$$

This condition (shown in Figure 2.4(b)) is rarely used, but it can improve the optimization if the design variables refer to periodic elements (angles for example).

Finally, there is the possibility to implement the transparent wall: in this condition the solution that exceed the boundary is accepted but the cost value is highly penalized.

2.6 Surrogate models

Surrogate models are part of the problem simulators in the objective function evaluation. It is interesting to analyse them and their application to Optimization Systems because they can affect the convergence of the optimizer.

The aim of the application of them in optimization is to reduce the computational time of the optimization problem evaluation when simpler models are not available or not enough accurate [38].

In these conditions it is possible to refer to the original optimization problem as real cost function, that differs from the cost function obtained with the surrogate model.

Surrogate models are based on a set of samples evaluated on the real optimization problems: these are used for creating a mathematical function that should approximate as well as possible the original cost function.

Given an m-dimensional problem, it is possible to define as *target function* the real relation between input and output (objective function), that can be found by a time expensive simulator or to experiments:

$$y = F(\mathbf{x}) \tag{2.8}$$

where $\mathbf{x} \in \mathbb{R}^m$ is the input containing, for example, a set of values for the design variables, $y \in \mathbb{R}$ is the target value, assumed to be scalar, and $F : \mathbb{R}^m \to \mathbb{R}$ is the target objective function.

In order to train the surrogate model it is required to have a set of input values, that can be sampled from the problem domain[1]:

$$\underline{\mathbf{x}}_S = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \dots \\ \mathbf{x}_n \end{bmatrix} \tag{2.9}$$

with $\underline{\mathbf{x}}_S \in \mathbb{R}^{n \times m}$.

To each of these points, the target value can be obtained with the target function:

$$\underline{y}_S = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix} = \begin{bmatrix} F(\mathbf{x}_1) \\ F(\mathbf{x}_2) \\ \dots \\ F(\mathbf{x}_n) \end{bmatrix} \tag{2.10}$$

This can be written shortly as:

$$\underline{y}_S = F(\underline{\mathbf{x}}_S) \tag{2.11}$$

[1] The bold definition of the variables refers to point in m-dimensional spaces, while the underline symbol represent a vector of values.

where $\underline{y}_S \in \mathbb{R}^{n\times 1}$.

The sample dataset is formed by all the couples

$$(\mathbf{x}_i, y_i) \quad \forall i = 1...n \tag{2.12}$$

The surrogate model is a function $\hat{F} : \mathbb{R}^m \to \mathbb{R}$ that reproduces the behaviour of F at least in the sampling points and that is able to give an acceptable value for any $x_j \notin \underline{\mathbf{x}}_S$.

Several surrogate models are available in literature, like the Response Surface Model (RSM), the Kriging, and the Artificial Neural Networks [81].

The Response Surface Methodology is based on a polynomial approximation of the target function. Typically, the second order polynomial is selected, because it is a good compromise between the computational cost and the reliability [81].

The Ordinary Kriging has the great advantage that it is the only method that returns not only an estimation value, but also an indicator of the quality of the estimation [81].

Neural Networks are global approximation techniques and, thus, can be used to reproduce functions [82]. In the basic application they do not returns any indicator of the quality of the results, but using ensemble methods it is possible to extract them [83].

The selection of the surrogate model takes into account, the number of sample required to have a proper training, the capability of the model to provide also a confidence level of the estimation, and the capability of the model to fit highly non-linear function [84].

In this thesis, the Ordinary Kriging has been investigated and adopted: in fact, the training of this model is relatively fast and, even with few training points, it gives reasonable results. Moreover, the confidence level of the estimation can be used to improve the sampling strategy.

2.6.1 Ordinary Kriging

The *Ordinary Kriging*, or Gaussian Process approximation, is a surrogate model firstly developed in the environment of geostatic literature [85].

It is a function estimator based on a weighted sum. It incorporates a covariance function in the estimation of the prediction value [86].

Given the target function F, the sample dataset formed by the couples $(\underline{\mathbf{x}}_S, \underline{y}_S)$, and the value of the prediction point $\mathbf{x}_0$, a generic weighted sum method can be formalized as [86]:

$$y_0 = \sum_{i=1}^{n} w_i y_i = \underline{w}^T \underline{y}_S \tag{2.13}$$

The specific feature of the kriging model is that it uses not only the location of the sample points but also the *similarity* between these points, represented by a covariance structure [86]. The specific covariance structure present in the ordinary kriging is composed by the covariance matrix, $\underline{\mathbf{x}}_S$, and the covariance vector, $\underline{c}_0$.

Given a predetermined covariance function $C : \mathbb{R} \to \mathbb{R}$, it is possible to express the covariance matrix of a vector of sampling points $\underline{\mathbf{x}}_S$ [86]:

$$\underline{\underline{C}}_1 = \begin{bmatrix} C_{11} & C_{12} & ... & C_{1n} \\ C_{21} & C_{22} & ... & C_{2n} \\ ... & ... & ... & ... \\ C_{n1} & C_{n2} & ... & C_{nn} \end{bmatrix} \tag{2.14}$$

where $C_{ij} = Cov(\mathbf{x}_i, \mathbf{x}_j) = C(|\mathbf{x}_i - \mathbf{x}_j|)$. The covariance matrix is symmetric, in fact:

$$|\mathbf{x}_i - \mathbf{x}_j| = |\mathbf{x}_j - \mathbf{x}_i| \tag{2.15}$$

Moreover, all the diagonal values C_{ii} are null:

$$C_{ii} = Cov(\mathbf{x}_i, \mathbf{x}_i) = C(|\mathbf{x}_i - \mathbf{x}_i|) = C(0) \tag{2.16}$$

With a similar approach, it is also possible to define the covariance vector between the sample points and the prediction point [86]:

$$\underline{c}_0 = \begin{bmatrix} C_{10} \\ C_{20} \\ ... \\ C_{n0} \end{bmatrix} \tag{2.17}$$

where $C_{i0} = Cov(\mathbf{x}_i, \mathbf{x}_0)$ and the covariance function is the same of before.

These two covariance structures allow to evaluate the weights $\underline{w}$ required by the Kriging [86]:

$$\begin{bmatrix} \underline{w} \\ \lambda \end{bmatrix} = \begin{bmatrix} \underline{\underline{C}}_1 & \underline{1} \\ \underline{1}^T & 0 \end{bmatrix}^{-1} \begin{bmatrix} \underline{c}_0 \\ 1 \end{bmatrix} \tag{2.18}$$

where λ is the Lagrange multiplier and $\underline{1}$ is a vector in $\mathbb{R}^n$ containing all 1s.

Inverting the partitioned matrix [87], it is possible to find the explicit value of the Kriging weights and of the Lagrange multiplier:

$$\underline{w} = \underline{\underline{C}}_1^{-1}\underline{c}_0 - \frac{\underline{\underline{C}}_1^{-1}\underline{1} \cdot \underline{1}^T\underline{\underline{C}}_1^{-1}\underline{c}_0}{\underline{1}^T\underline{\underline{C}}_1^{-1}\underline{1}} + \frac{\underline{\underline{C}}_1^{-1}\underline{1}}{\underline{1}^T\underline{\underline{C}}_1^{-1}\underline{1}} \tag{2.19}$$

$$\lambda = \frac{\underline{1}^T\underline{\underline{C}}_1^{-1}\underline{c}_0}{\underline{1}^T\underline{\underline{C}}_1^{-1}\underline{1}} + \frac{1}{\underline{1}^T\underline{\underline{C}}_1^{-1}\underline{1}} \tag{2.20}$$

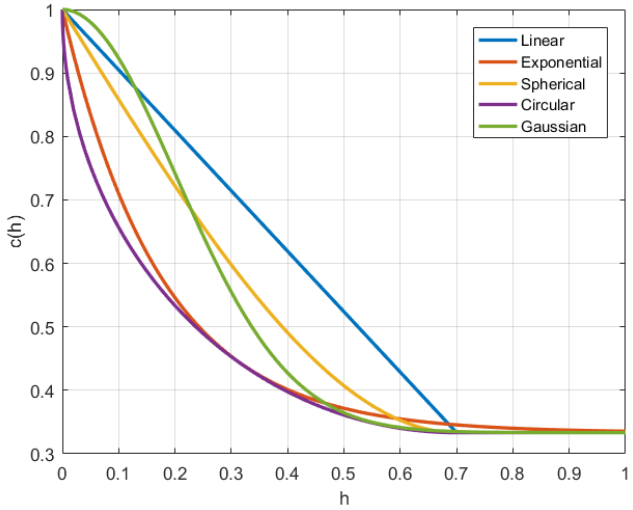


Figure 2.5: Covariance functions

Thus, the prediction value y_0 can be easily computed using Eq. 2.13, while the prediction variance can be calculated as:

$$\sigma_0^2 = \sigma^2 - \left[\begin{array}{cc} \underline{w} & \lambda \end{array} \right] \left[\begin{array}{c} \underline{c}_0 \\ 1 \end{array} \right] \tag{2.21}$$

where σ^2 is the estimated variance of the covariance function.

There are many different covariance functions, that impact on both the prediction values and variance. All of them contain three parameters:

- the nugget effect, ν_{sv}, that represent an estimation of the error at the sampling points;
- the range value, r_{sv}, that represents the distance over which the points are not autocorrelated;
- the still value, s_{sv}, that is the limit value of the variogram when the distance is greater that the range.

Figure 2.5 shows some covariance functions (for their definition, see [88]); for all of them the range parameter has been set to 0.7.

It is possible to see that all the functions, for values greater than 0.7 have a similar behaviour. The function that most differ from the other is the exponential function because it has a smoother transient towards the asymptotic value. Among the selected functions, it is possible to see that the gaussian one is the only with a sublinear behaviour for $h \to 0$.

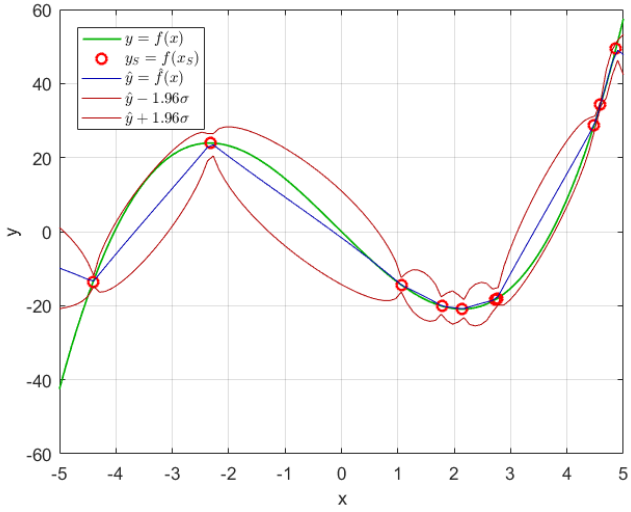


Figure 2.6: Target function, sampling points and estimated function and 95 % confidence interval.

Figure 2.6 shows an application of Ordinary Kriging to a simple 1-D target function:

$$F(x) = x^3 + 0.3x^2 - 15x \tag{2.22}$$

The sampling points are represented with red circles in the Figure. The green line is the target function, while the blue one is the estimated function. The red lines represent the values of the confidence range at 95%.

The estimated function fits exactly the target function in the sampling points: in these, also the confidence range goes to zero.

It is possible to see that the estimated function is completely unreliable outside the sampled range (in this case, this is clear in the left part of the search space).

2.6.2 Surrogate models in optimization

As can be easily argued, a critical aspect of the surrogate models use is the number of samples required for the training: the higher this number, the higher the accuracy, but with a growing computational load, due to sampling and training the model [85, 89].

Several approaches can be used to combine the optimization algorithm with the surrogate models. It is important to take into consideration that also the problem formulation can be important in the possibility to apply surrogate models in the optimization process [90].

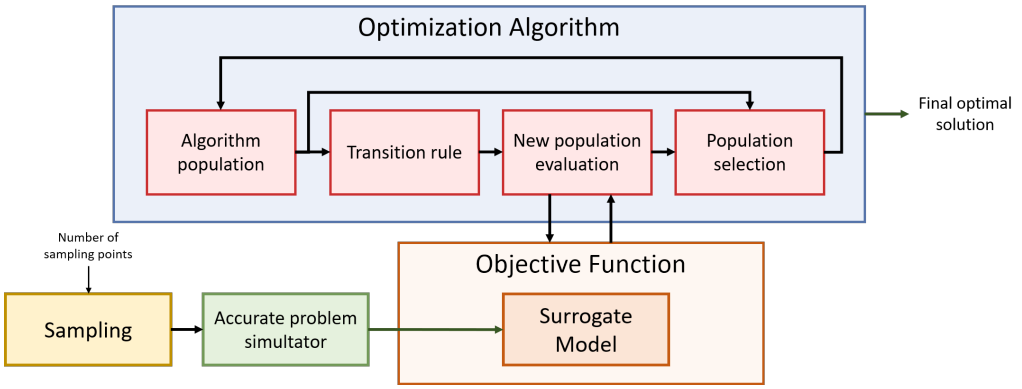


Figure 2.7: The surrogate model is created using a predefined number of sampling points and then it is used for the optimization process.

A basic approach is shown in Figure 2.7: a set of sampling points are evaluated on the accurate problem simulator (it can be also a real experimental setup) and with these points a surrogate model is then trained. At each iteration of the optimization algorithm, only the surrogate model is used to evaluate the objective function.

This approach has the great disadvantage: the model accuracy is not checked during the optimization process. In order to overcome this problem, the model can be updated during the optimization process.

In [91] an iterative process is implemented: for instance, an initial set of sampling points is used to compute a first surrogate model. This model is used during a first optimization process and then the optimal solution obtained is evaluated on the accurate simulator and it is added to the training set of the surrogate model (see Figure 2.8). This process is repeated several times.

This process has a high accuracy but it requires longer time: it can be very effective while using local optimization techniques, but with Evolutionary Optimization Algorithms its effectiveness reduces drastically. In fact, most of the algorithm exploration capability is lost due to the lower surrogate model accuracy far from the previously localized minima.

In [85] the authors propose a combined method with two surrogate models: a global one that is used to reproduce the trend of the function and it is used by the global optimizer to update its population, and a local model, associated with a local optimizer, is used to better exploit the information around the candidate best solution. In this scheme, there is only a sampling operation (that is the most costly in terms of computational time): the global model is constructed using all the sampling points, while the local model uses only a subset of these points. There are two feedback lines from

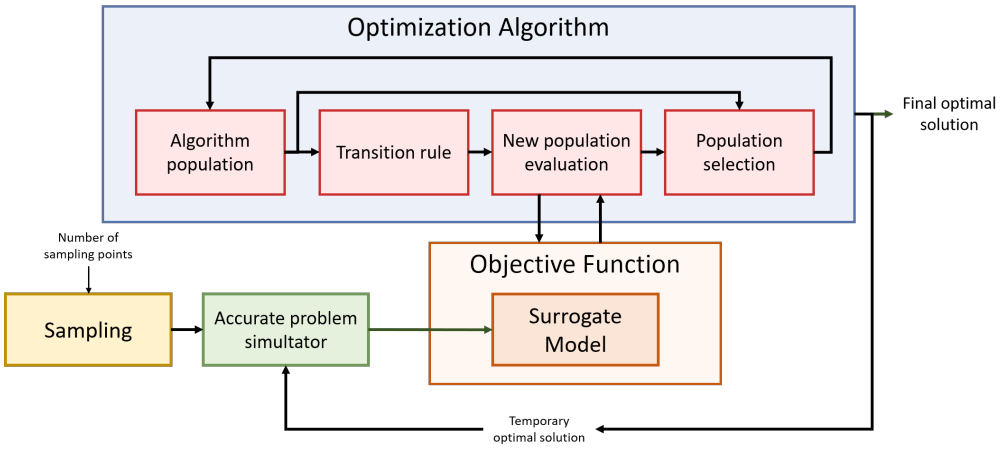


Figure 2.8: The surrogate model is created using a predefined number of sampling points and then updated iteratively using the optimal solutions found by the optimizer.

the optimization process to the sampling: at each iteration, both the information coming from the optimal solution of the local search and part of the population are used to sample new points. The flow chart of this method is shown in Figure 2.9.

This last system is interesting because the process of updating the surrogate model is done during the optimization process, but it has the main defect of increasing the computational time related to the creation of the two models and of running two optimization processes. Moreover, it is limited due to the presence of the local optimizer that, in many cases, has more requirements on the type of cost function.

An effective way for combining surrogate models with optimization algorithms is an integrated approach in which the model is updated during the optimization process. Moreover, it is possible to think to modifications of the algorithms to better exploit the surrogate model features [37]. For example, in [92] the update of the global and personal best parameters of the PSO has been customized for taking into consideration the confidence level.

Figure 2.10 shows an example of combined interaction between the optimizer and the surrogate model. This system is based on two connections between the optimization algorithm and the objective function environment: in the first relation, the population of the algorithm is used for selecting the sampling points for the creation or the update of the surrogate model. The second one is the evaluation of the population on the updated surrogate model.

This process is repeated in the iterations; thus the surrogate model is

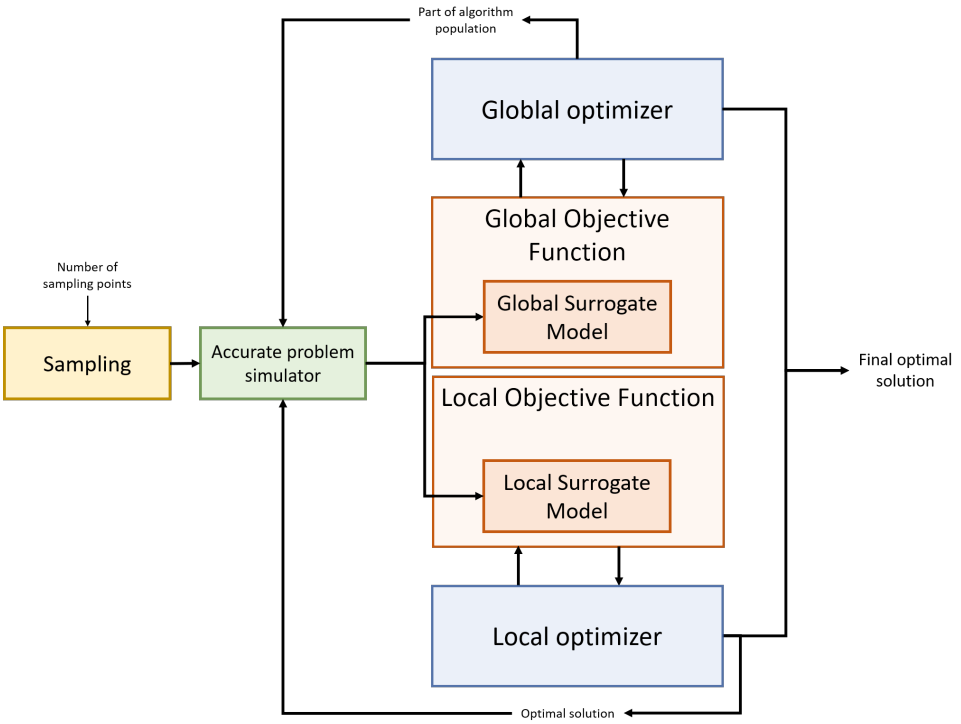


Figure 2.9: Local and global optimization algorithms are used in combination with two surrogate models.

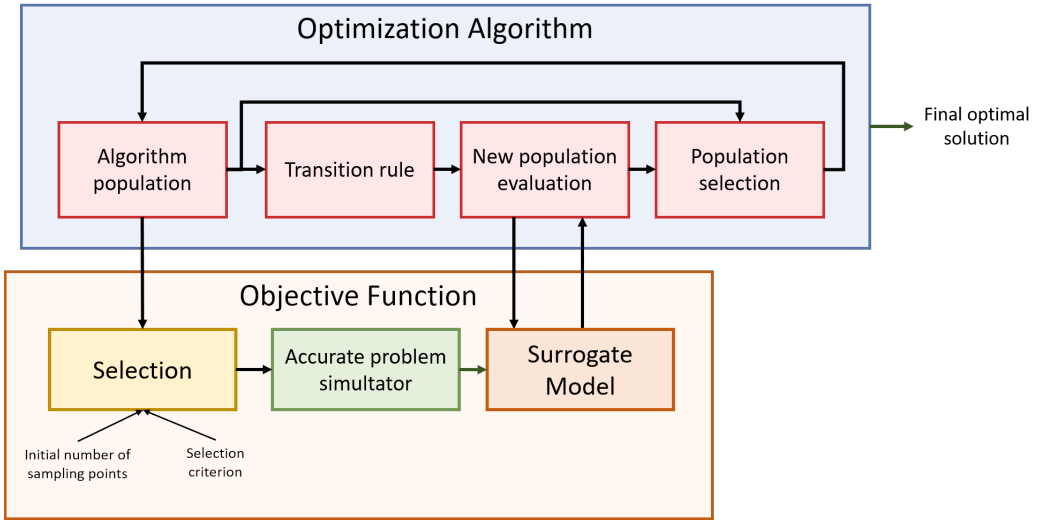


Figure 2.10: Scheme of the surrogate model optimization with high interaction between optimizer and surrogate model.

updated at every iteration: it keeps its accuracy in the exploration, but it can be also used for exploring new areas of the cost function.

Two fundamental aspects have to be set in this scheme: the initial number of training points and the selection criterion during the iterations. For this, it is possible to exploit the variance given by the Ordinary Kriging.

As rule of thumb, the selection should prefer solutions that possibly can be the optimum of the function; solution with high variance can be selected; and finally, enough exploration should be kept for having a proper definition of the entire function.

Chapter 3

Social Network Optimization

Social Network Optimization (SNO) is an Evolutionary Optimization Algorithm (EOA) developed during the research project of this PhD program: this algorithm has proven its performance in many problems of different nature.

Several papers have been published in international journals and conferences, based on this algorithm and its applications: for instance, in [93] has been applied to microwave circuit design, and a preliminary comparison between SNO and other EAs is provided. In [71] the algorithm has been applied to low-frequency electromagnetic optimization, in particular it has been compared with GA and PSO on the TEAM25 problem and then it is applied on the optimization of a Tubular Permanent Magnet Linear Generator. More details on these applications are provided in this thesis. The algorithm has been applied in [70] for parameter selection for power production forecasting with a physical model. Finally, in [94] the algorithm has been applied to Wireless Sensor Network optimization, and on this application a comparison with other EAs is provided.

Among the conference papers on SNO, one of the most relevant is [68] because it has been presented of the most important IEEE conference about Evolutionary Computation.

The structure of the algorithm makes it a very flexible tool: in fact, its features are ready for multi objective optimization, for an effective combination with surrogate models, and for many other advanced applications.

The algorithm has been implemented both for real-variable problems (*i.e.* problems in which the design variables are codified as real numbers) and for binary problems (*i.e.* problems in which the design variables can be codified using only 0 and 1).

In this chapter a detailed and comprehensive analysis of this algorithm is provided: the chapter starts with a description of Online Social Networks, the

inspiration of the algorithm: this analysis is aimed to show the main feature of social networks and the methods commonly adopted for their modelling. Then the algorithm will be described in detail: the first description shows the operators and the data structures of SNO; while the real value and the binary implementations are detailed in the two last sections of this chapter.

3.1 Introduction to Online Social Networks

This Section is aimed to provide a brief overview on Online Social Network. This is basically a review that contains some important concepts of social websites that have been then implemented in the optimization algorithm. This introduction is not designed to describe completely and accurately all the models for Online Social Networks, that requires too much space and it is out of the scope of this thesis.

This Section contains firstly a general description, and then it is focused on the mathematical models adopted for emulating the evolutions of social networks. These are very important because have been used for designing the operators of SNO.

3.1.1 Description of Social Networks

Social Networks are websites for networking and content sharing [95]. Generally speaking, they are structures composed by entities that are linked together. Depending on specific type of social network, the social entity can be an individual (*e.g.* in Facebook or Twitter), corporations (*e.g.* in LinkedIn or TripAdvisor), collective social units, or organizations. The type of connection depends on the Social Network and can by friendship, common interests, beliefs or financial exchange [96].

The most important role of a social network is the diffusion of ideas, opinions and behaviours. There are different models that are used to describes how ideas spread in Social Networks: two of the most common are the *simple contagion* and the *complex contagion*. These models are based on the studies of disease diffusion [97].

In the first model it is assumed that a single contact between two people is able to completely transmit the behaviour. As consequence of this, the ideas spread more rapidly in networks; the number of redundant connections required to diffuse a specific behaviour is reduced. In the second model it is assumed that a person requires multiple contacts to modify its behaviour. In this second case, ideas spread better in networks with more redundant connections [98].

The interconnections between people required by the ideas diffusion mechanism can have different natures and, thus, different specific features. In particular, each individual has some preferred and stronger connections with a limited amount of other social network members [96]. These preferred interaction channels are the basic features that drives the creation of groups. Groups are fundamental structure of the entire society and, thus, this is also valid for Social Networks [99].

In Social Networks groups can have different features. The three most important types of groups are the *singletons*, the *giant component* and the *middle region*. The analysis of groups is often developed by market analyser for identifying the key individuals that can influence a large amount of the network [100].

The singletons are zero-connection nodes: they do not participate actively in the social network and they are a limit case of groups. On the other hand, the giant components are large or very large groups that are connected each other by paths in the social network: they are very important for ideas diffusion and for marketing because the ideas spreads in the giant components at high speed due to the high number of connections. Finally, the middle region is the remaining one: it consists in various isolated communities that do not interact with the rest of the network. Generally, they are characterized by the presence of the so-called *stars*: a single charismatic individual linked to other users that have few connections with the rest of the network [101].

Another important feature of Online Social Network regards the actors that appears in the information exchange. In fact, in common interpersonal relations, the interaction happens by a direct communication between people, while in a Social Network it is mediated by the Online Community. The exchange of information between the website and people is reciprocal and, very important, the source of the information is cited only in terms of credibility of the information [102].

This last aspect is fundamental in any networks and, more specifically, in social networks: in fact, it concerns the topic of the *trust* and of *trust networks*. Trust is the basic feature in a recommender system where it is present a set of users and a set of items. The most important goal of a recommender system is to estimate the rate value given by a user on a specific item, non already rated, using the data of known rating of the same user and of other users [103]. More details on trust and trust networks are provided in Section 3.1.3.

Dealing with a Social Network means considering three aspects of the information. These are the content (*what*), the social dimension (*who*), and the temporal dimension (*when*). These three aspects are fundamental to understand the correlation among news and the spread of ideas [104].

Due to the fact that the process of ideas diffusion in Social Network is less mediated, it is possible to see the formation of *echos chambers*, facilitated by the process of increased polarization in the population [104].

3.1.2 Modelling of Social Networks

Modelling of Social Networks is a very complex aspect, also due to the dynamic evolution of the websites. There are some basic features that are common to all the social networks. Many models of Social Networks are extracted from epidemic models, in which the ideas are considered as an infection [105].

Firstly, we can consider a Social Network as a set of users in the domain of a social media [96]:

$$\mathbb{S} = \{u_1, u_2, ..., u_m\} \tag{3.1}$$

where $\mathbb{S}$ is the Social Network with m members and u_u is a generic member of it. Each user can be described as a set of further simple information.

It is possible to define several connection types between people that can be modelled in different ways. These can be classified accordingly to two criteria. The first one is the reciprocity of the connection: some connections are reciprocal (as the friendship on Facebook) and other not (like the relation follower - followed on Instagram). The second one is the importance of the relation: some of them are stronger and, thus, ideas can spread in an easier way.

The basic modelling of groups is based on the concept of related list: a set of users $\mathbb{L}_u \subseteq \mathbb{S}$ is generally connected to a person $u \in \mathbb{S}$ and $u \notin \mathbb{L}_u$; if any $v \in \mathbb{L}_u$ is connected with u:

$$\mathbb{L}_u = \{v | v \in \mathbb{S} \wedge E_{uv} = 1\} \tag{3.2}$$

where E_{uv} represents a connection between u and v. This very general definition does not imply the reciprocity of the connection.

Each connection in the friend list can be weighted in order to take into account, by means of an appropriate measurements value, the strength of the connection. It is important to notice that the weights of the connections can be asymmetric: in fact, it is possible to find a set of two users u_1 and u_2 such that the connection between u_1 and u_2 is strong while the opposite connection is weak [96].

This model of Social Network can be represented as a directed graph, where the nodes are individuals and the edges indicate social relationships [105]. Figure 3.1 shows an example of small social network (m = 7) represented as a direct unweighted graph.

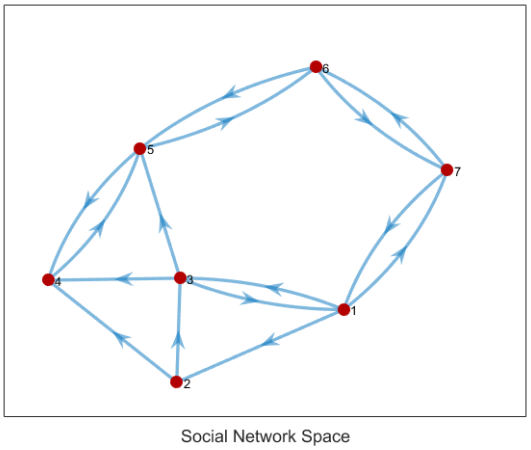


Figure 3.1: Modelling of a small social network as a direct graph.

For understanding the evolution of Online Social Network, it is important to analyse the formation and the evolution of groups; in particular, it is possible to study the features that influence a person to join a specific group, the growth of the group itself and how groups can change during time.

The group evolution can be modelled defining and evaluating the probability that an individual u joins a group $\mathbb{G}$ as function of the number k of connected people that are already in the group:

$$p_{u,k} = \mathcal{P}(u \in \mathbb{G}(t+1) | u \notin \mathbb{G}(t) \wedge k = ||\mathbb{G}(t) \cap \mathbb{L}_u||) \tag{3.3}$$

where t represents a discrete time instant.

The analysis of Social Networks shows that the probability $p_{u,k}$ has a s-shaped behaviour with the number of friends k. In fact, for small numbers of k having one more friend in the community has a strong effect, while for larger k this effect tends to zero, even if the curves continues to increase [99].

For what concern the classification of groups in singletons, middle region and giant components, an analysis of social networks performed in [101] shows that the number of singletons is reduced almost linearly during time, the middle region has a constant size and the giant component grows.

It is possible to develop more complex models of the relations present in Social Networks. For example, in [104] the authors propose to use three overlapped networks: in homogeneous networks, where the users can be modelled as nodes of the same type, the three proposed layers are the friendship network, the diffusion network, and the credibility network.

In this model, friendship networks represent the social connections between users, diffusion network the idea propagation paths, and the credibility

network creates the supporting or the opposition interaction between users.

Diffusion networks represent a model of the ideas spread among the Social Network. This process can be represented with a quite simple threshold model, in which each individual has an activation function that measures the effect of the idea contamination from its neighbours [105]. The general assumptions of this model are [106]:

- the process is *progressive*, an active node will be active forever;
- the threshold values are random, that takes into account the lack of knowledge about users' personality in the model;
- the activation functions are monotone increasing, that is, a node becomes more infected when a larger set of its neighbours are infected.

The ideas diffusion in Social Network leads to a very important phenomenon: the *echo chambers*. It is a specific information path that amplify an idea: in particular the source of information receives back their idea and they are further convinced by this echo. The echo path should be enough long to make the information different enough to be unrecognizable by the initial source.

Echo chambers can be modelled using the probability density function of users' polarization for a specific idea [107]. Each user can be neutral, polarized or partially polarized. This is represented by a polarization level $\pi_u \in [0, 1]$ for each user u, such that $\pi_u = 0$ means that the user is completely neutral and $\pi_u = 1$ means that the user is completely polarized.

The following one is an example of a simple sigmoid polarization function:

$$f_\pi(x) = \frac{\xi_\pi(x) - \xi_\pi(0)}{\lim_{x\to+\infty}(\xi_\pi(x)) - \xi_\pi(0)} \tag{3.4}$$

where

$$\xi_\pi(x) = \frac{1}{1 + e^{-\eta_u \cdot (x-\theta_u)}} \tag{3.5}$$

where θ_u is the threshold level of the user u and η_u is the polarization attitude of the same user. This definition of the polarization function allows to tune the parameters θ_u and η_u for modelling the differences between individuals; moreover, the polarization function is always between 0 and 1.

For the case of $\theta_u = 0.5$ and $\eta_u = 8$ the polarization function is shown in Figure 3.2.

The polarization of a user evolves in time accordingly to the polarization of their friends. Calling l_u the number of friends of the user u it is possible

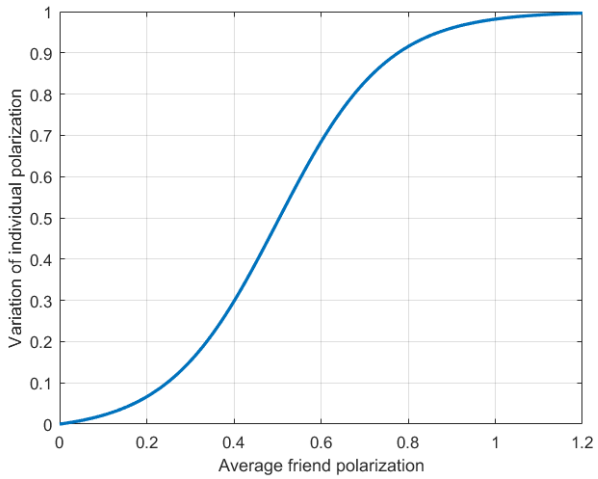


Figure 3.2: Polarization function for the user u with $\theta_u = 0.5$ and $\eta_u = 8$.

to model the evolution of polarization as:

$$\pi_u(t+1) = f_\pi \left(\pi_u(t) + \frac{1}{|\mathbb{L}_u|} \sum_{v \in \mathbb{L}_u} \pi_v(t) \right) \tag{3.6}$$

where f_π is the polarization function and represents the non-linear behaviour of humans' polarization.

There are two important aspects that can affect the ideas sharing: the visibility and the effect on content popularity. The first one is critical because depends a lot on the specific rules decided by the social network sites. For example, in Flickr and in Facebook, posts that have received many *likes* are more visible. Moreover, also the type of content can affect its visibility [95].

3.1.3 Trust networks

Trust is a fundamental aspect in online relation because the communication channel is limited. All the non-verbal part of the communication is eliminated and also the verbal part is highly mediated. In online marketing, this aspect is even more important because the consumer hasn't the opportunity to try the product [108]. The online marketing is just an example of how important is trust in computer mediated processes.

The definition of trust is not unique. In [109] the trust is interpreted as *reliability trust*, defined as follows:

> trust is the subjective probability by which an individual, A, expects that another individual, B, performs a given action on which

its welfare depends.

In [110], the authors note that this definition is not perfect, due to the fact that it is possible to have high trust levels without entering in a situation of dependence on that person [108].

Another definition of trust is reported in [111]:

> Trust is the extent to which one party is willing to depend on something or somebody in a given situation with a feeling of relative security, even though negative consequences are possible.

In this definition, there are present some aspects, like the reliability of the trusted part, the utility and the risk attitude [108].

There are two kind of trusts: the first one is the *referral trust*; the second one is the *functional trust*. The second one is related to a specific activity, while the first one is related to the capability of the trusted person to give a good *recommendation* [112].

In some specific occasions, trust can be transitive: this means that it is possible to create an indirect trust between two people that have never been in contact. In this process, some trusted people are required to give a recommendation. In this case, it is possible to define a trust network [113]. The model of trust transitivity is based on a threshold activation function.

In trust networks there are some specific trust paths. In general, these paths end with a functional trust (usually the first) [112]. In the case of Social Network, it is possible to have a trust network without a specific functional scope. This leads to an higher possibility of creation of echoes chambers.

3.2 Social Network Optimization description

The basic ideas of Online Social Network that inspire SNO have been analysed above. These features have been included in the algorithm with a certain amount of simplification in order to keep low the computational time required to run the algorithm.

In this Section the algorithm is described: firstly, the basic data structures contained in it, and then the implemented operators. The implementation of some operators is affected by the problem type, in particular if it is real-valued or binary: the specific differences are analysed in two separate sections.

3.2.1 Social Network Optimization data structures

The basic data structure of SNO is the social network. It is the virtual space in which the interactions take place and, thus, in which people exchange

ideas and opinions. It contains the two basic elements: the users, that is the population of the algorithm, and the posts, that are the structures that drive the interaction between users:

$$\mathbb{S} = \{\{u_u\}, \{p_s\}\} \tag{3.7}$$

In real Social Networks users write posts to express their opinion and read posts of other users. This iteration is the basic evolution mechanism of the population in SNO and it is described later.

The population size of the algorithm is the number of users in the Social Networks.

All the users interacting on the Social Network share their *opinions* on some topics. Each user is also characterized by a list of friends and a reputation value for all the other users. These two structures are used for driving the interaction between groups.

The user can be represented in mathematical terms as:

$$u_u = \{\mathbf{o}_u, \mathbb{L}_u, \{r_{uv}\}\} \tag{3.8}$$

where $\mathbf{o}_u$ is a vector containing the set of all the user opinions, $\mathbb{L}_u$ is the friend list of the user u, and r_{uv} is the reputation from the user u to the user v.

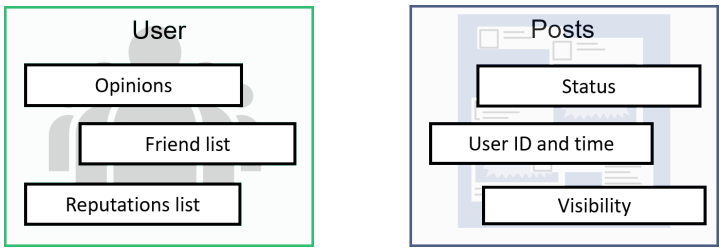
The posts are what the users write to express their opinions. The main element of a post is the *status*, $\mathbf{s}_s$, that is the content in which the user expresses its opinion. In order to take into account the three aspects of the news in Social Networks, the *what*, the *who*, and the *when* [104], in addition to the status itself there are three information associated: the first one is the name (the ID) of the user that have posted it (u_s), the time in which it is posted t_u and a visibility value (v_s):

$$p_s = \{\mathbf{s}_s, u_s, t_s, v_s\} \tag{3.9}$$

The status is the optimization variables that is mapped to the design variables of the optimization problem. Figure 3.3 summarize the composition of the data structures of SNO.

The interaction takes place on two different kind of networks: the first one is a friend network, represented by the fiend list of each user, and the second is a trust network represented by a trust matrix.

These two networks are significantly different. The friend network is symmetric, the connections are particularly strong, and the evolution of this network depends on events in the real word. On the other hand, the trust network is not symmetric, *i.e.* trust is not reciprocal, the connections are



(a) User composition (b) Post composition

Figure 3.3: Composition of the two main data structures of SNO.

weaker that the friendship connections, and the evolution of this network depends only on online relations.

The data structure described above evolves in time thanks to the algorithm operators described in the following.

3.2.2 Social Network Operators

The algorithm operators are designed to evolve the population: in particular, they regulate the personal growth, the creation of a post from the user's opinions, and the evolution of the friend and trust networks.

The most important operator is the one devoted to the evolution of the user, in particular the creation of the personal growth as function of the time history of the growth itself and of the contagion of the ideas.

The operator implemented emulates the assumption of a *complex contagion* [98], that guarantees a better tradeoff between exploration of the domain and the exploitation of the acquired knowledge.

In the most general case, this operator can be represented in the following way:

$$\mathbf{c}_u(t) = \mathcal{I}\left(\mathbf{c}_u(t-1), \mathbf{c}_u(t-2), ..., \mathbf{c}_u(t-\Delta t_{max}), \mathbf{a}_u(t)\right) \tag{3.10}$$

where $\mathcal{I}$ is the function that models how the ideas are spread in Social Networks. The vector $\mathbf{a}_u(t)$ is the attraction point, and represents the impact of the idea contagion on the user u.

The specific $\mathcal{I}$ function depends on the type of problem, real-value or binary. In both these implementations it has been chosen to set $\Delta t_{max} = 1$ to accelerate the algorithm convergence. Figure 3.4 schematize the growing process. The green blocks are part of the user data structure, the red ones are the operators and the purple one is an additional data created in each iteration.

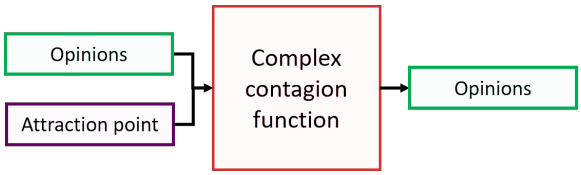


Figure 3.4: Schematization of the personal evolution in SNO.

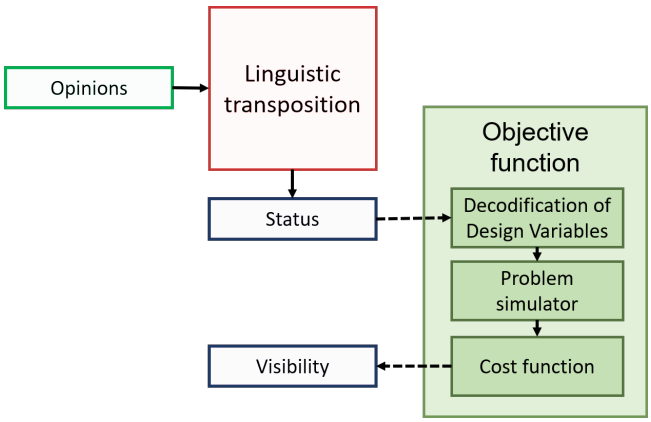


Figure 3.5: Schematization of the basic interaction between SNO and the optimization problem.

The status is created starting from the users' opinions. The operator used to create it is called *linguistic transposition*, because it reflects the difficulties in expressing a specific idea. The linguistic transposition operator Λ depends on the linguistic error $\boldsymbol{\lambda}$ that changes with time and differs from an individual to another:

$$\mathbf{s}_u(t) = \Lambda(\mathbf{o}_u(t), \boldsymbol{\lambda}_u(t)) \tag{3.11}$$

The implementation of the linguistic transposition operator Λ depends on the type of design variable and is defined in the following sections. The status is a time function because the opinions evolve, and the linguistic transposition varies. This is important because it reduces the stagnation of the algorithm: in fact, even if the opinions do not vary with the algorithm iterations, the status is every iteration different thanks to the linguistic transposition.

The status is then mapped to the design variables of the optimization problem. The performance of the system with the configuration specified by the status are calculated and mapped back to a single performance value that is the post visibility.

The process is described in Figure 3.5, where the opinions are the ones obtained at the end of the process of Figure 3.4, the blue elements are part of the post data structure and the orange block is the optimization problem.

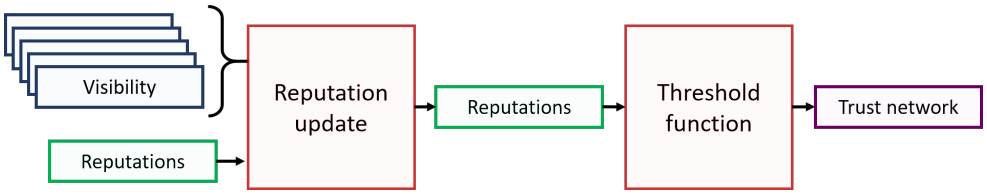


Figure 3.6: Schematization of the process of reputation update and creation of trust network.

The lines connecting the post and the optimization problem are dashed to remember that they represent a mapping operation.

The visibility values of the entire population are used to update the reputations. This operation transfers global information among the entire population. They are firstly normalized for eliminating biasing due to the fitness function creation:

$$\tilde{v}_s = \frac{v_s - \min_i v_i}{\max_i v_i - \min_i v_i + \varepsilon} \tag{3.12}$$

where ε is a small value chosen to avoid problems when all the visibility values are equal.

It is important to notice that this normalization is affected by the definition of the visibility starting from the problem performance parameters: localized high values of visibility reduce the trust variability and thus the dynamic evolution of the trust network. This effect is not highly detrimental for the algorithm because the evolution of all the other structures is independent.

Then the trust values are modified accordingly to the normalized visibilities:

$$\Delta r_{ij} = \tilde{v}_i - \tilde{v}_j \tag{3.13}$$

The trust network is created by means of a threshold function. It is possible to notice that the evolution of the trust values is asymmetric, *i.e.* $\Delta r_{ij} \neq \Delta r_{ji}$ and thus the trust network is asymmetric. The entire process of trust update and trust network creation is schematized in Figure 3.6. Only the trust values are used in the following iterations, while the trust network is created *ex novo* each time.

The evolution of the friend network is driven by different operators than the trust network. The creation of new friendship relations depends on the number of common friends, while the elimination of one of these relation on a uniform distribution with probability p_d. This probability has been modelled as a constant value and it is considered user-independent. The specific "social attitude" of the single user is neglected.

The probability that two users $u \in \mathbb{P}$ and $v \in \mathbb{P} | v \neq u \wedge v \notin \mathbb{L}_u$ becomes

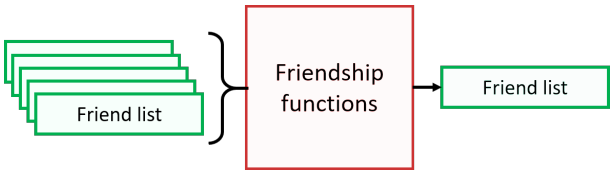


Figure 3.7: Schematization of the process of evolution of friend network.

friend is function of the number of common friends:

$$p_f = f_f\left(||\mathbb{L}_u \cap \mathbb{L}_v||\right) \tag{3.14}$$

The function f_f is a sigmoid function, in the algorithm the following one has been implemented:

$$f_f(x) = \arctan\left(\frac{x^2}{\pi}\right) \tag{3.15}$$

Figure 3.7 shows the schematization of the process of evolution of friend lists. The friendship function block contains both the creation and the disruption of friend connections.

Among the friends and the trusted connections, some opinions are selected as sources of influence for creating the attraction point $\mathbf{a}_u(t)$. The influencing information are selected among the available posts, accordingly to the visibility value (rank-based selection). The two influencing ideas are merged adopting a uniform crossover operator: each element of the vector is taken from one of the two influencing ideas accordingly to a random variable.

The entire process of selection of the influencers and creation of the attraction point is shown in Figure 3.8.

The last operator that should be described is the selection of the posts shown by the Social Network. This process is very important because gives a first reduction of the available information and allows the ideas to contaminate the social network for longer time: in fact, it is possible that good posts are visible for long time, increasing the exploitation capabilities of the algorithm. This operation makes the algorithm ready for multi objective implementation.

The selection of the visible posts is done accordingly to the visibility values. During the implementation phase of the algorithm, several possibilities have been taken into account: random selections, elitism mechanism, random with elitism, elitism with some low-level solutions, and others. Each operator affects both the exploration and the exploitation capabilities of the algorithm.

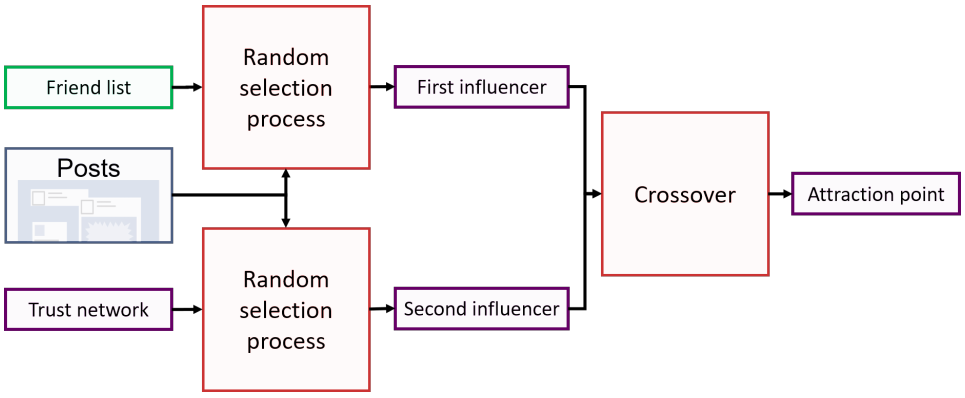


Figure 3.8: Schematization of the process of selection of the influencers and creation of the attracting point.

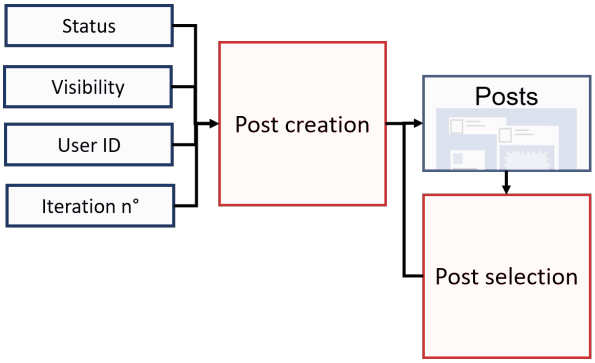


Figure 3.9: Schematization of the process of evolution of the post visible in the Social Network.

At the end of the preliminary tests, the pure elitism strategy has been adopted: in fact, the linguistic transposition gives to the algorithm enough exploration and with random selection the convergence process becomes too slow.

The evolution of the stored posts is the following: after the creation of a new status and the evaluation of the visibility, the new post is included in the batch of visible posts. After the selection of the influencers, the best posts are kept in memory and made available for the following iteration.

The entire process of post evolution is shown in Figure 3.9

Figure 3.10 shows the entire overview of the algorithm operators and their interactions with the data structures.

Figure 3.11 shows the flow chart of SNO, in which the iterative nature of the algorithm is clear. In the flow chart, in all the logical blocks the operators applied are highlighted.

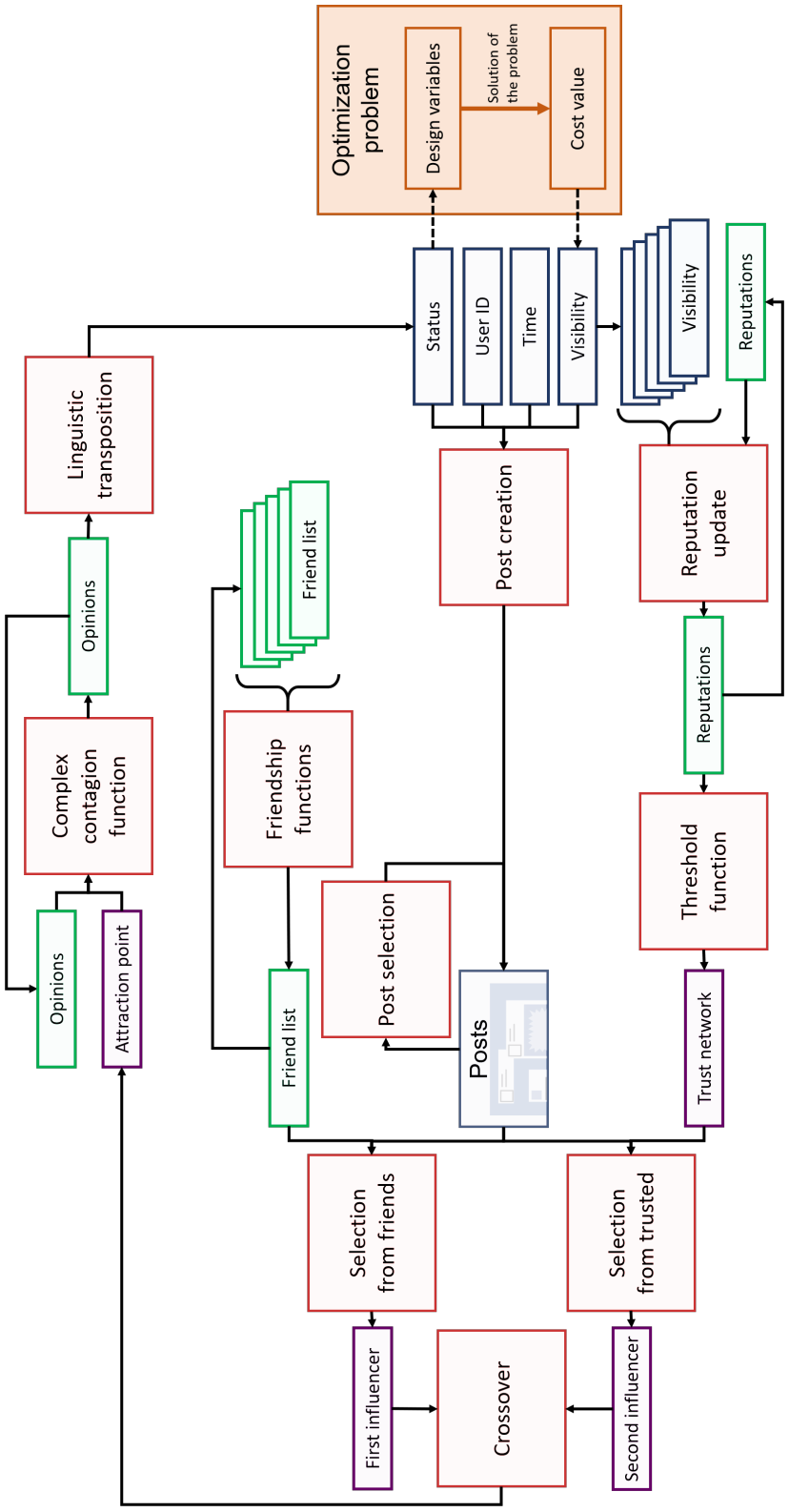


Figure 3.10: Overview on all the operators of SNO.

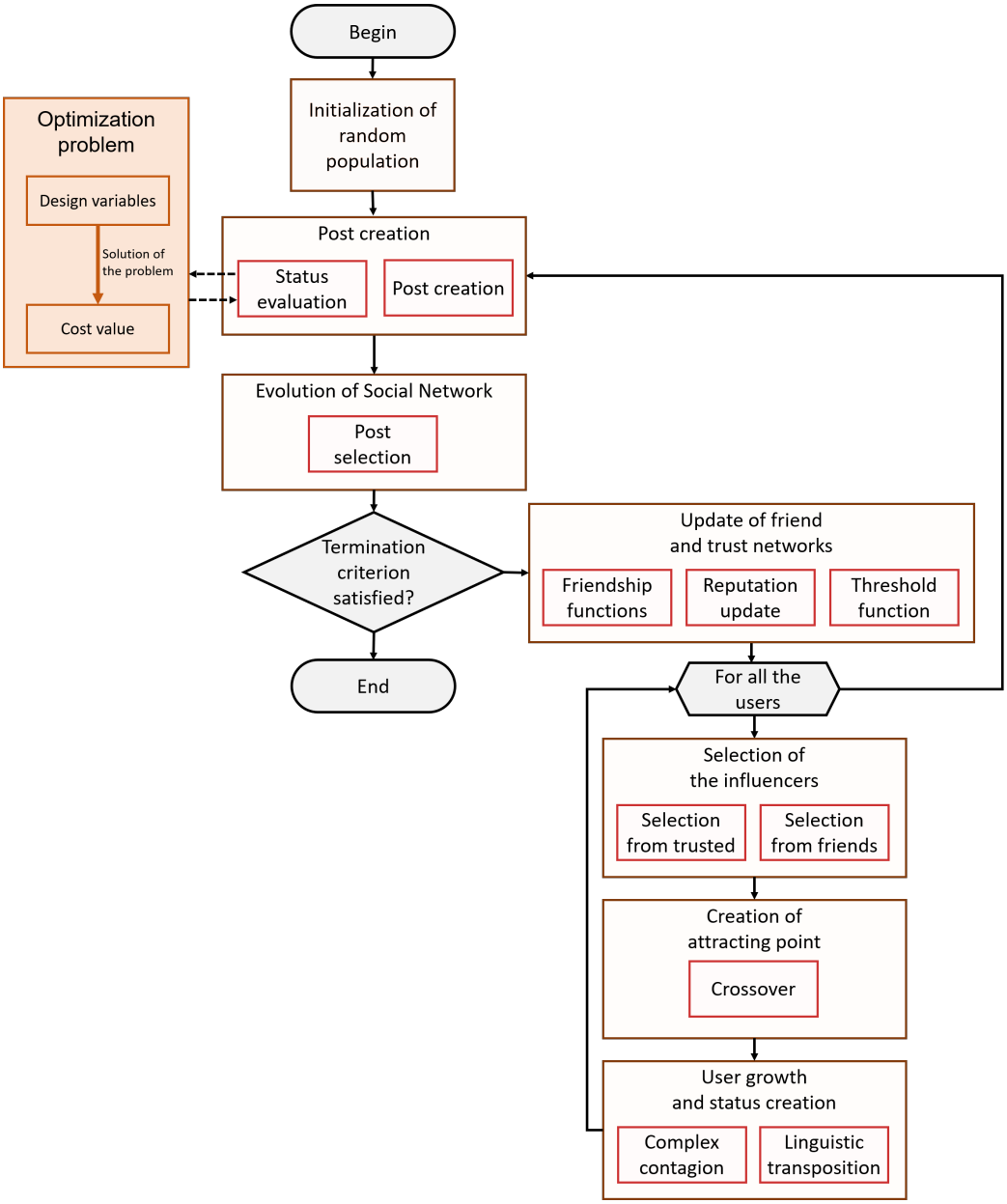


Figure 3.11: Flow chart of SNO, in all the logical blocks the operators used are highlighted.

3.3 Real-value implementation of SNO

The general principle of Social Network Optimization can be applied in different ways according to the type of optimization problem, *i.e.* if the optimization variables can assume real or binary values. The optimization variables are related to the design variables by means of the process of mapping described before (see Figure 3.5).

In this Section, the real-value definition of the complex contagion formula and of the linguistic transposition is firstly described. Then, an analytic analysis on the complex contagion operator is presented. Finally, the effect on performance of all algorithm parameters is analysed empirically.

In a real-value problem it is usually assumed that the optimization variables can range between 0 and 1. This assumption avoid scaling problem of the algorithm operators, but does not limit the algorithm generality because the mapping process can associate the optimization range to any range in $\mathbb{R}$.

Thus, it is possible to fix the box domain for both the opinions and the statuses:

$$o_{u,i} \in [0,1] \quad \forall i \tag{3.16}$$

$$s_{u,i} \in [0,1] \quad \forall i \tag{3.17}$$

The complex contagion operator can be expressed by the following equation:

$$\mathbf{o}_u(t+1) = \mathbf{o}_u(t) + \alpha[\mathbf{o}_u(t) - \mathbf{o}_u(t-1)] + \beta[\mathbf{a}_u(t) - \mathbf{o}_u(t)] \tag{3.18}$$

Another operator that has not completely defined before because it is affected by the problem type, is value of the linguistic error $\boldsymbol{\lambda}_u(t)$ (see Equation 3.11. In real-value SNO it is implemented in the following way:

$$\lambda_{u,i} = \mu_a \cdot r_N \cdot r_i \tag{3.19}$$

where r_N is a random variable extracted from a normal distribution $\mathcal{N}(0,1)$, r_i is a random binary variable that assume value 1 with probability μ_r and μ_a is a algorithm parameter.

The linguistic transposition operator Λ is defined in the following way:

$$\Lambda(\mathbf{o}_u(t), \boldsymbol{\lambda}_u(t)) = \mathbf{o}_u(t) + \boldsymbol{\lambda}_u(t) \tag{3.20}$$

For understanding the converging behaviour of the algorithm and for having a better knowledge in the selection of the algorithm parameters, it is important to analyse the complex contagion operator implemented in SNO.

3.3.1 Analysis of the complex contagion operator

It is possible to make a study of the complex contagion operator, as it has been done in [114] for Particle Swarm Optimization. The results here reported have been partially published in [69].

By examining equation 3.18, it is clear that the evolution of each optimization variable is independent from the others. This has two consequences. Firstly, the working principles of Social Network Optimization are not affected by the size of the problem under investigation, but only on the type of the objective function. Secondly, the behaviour and the stability of the algorithm can be analysed considering only one variable.

Considering only one optimization variable means rewriting equation 3.18 in a scalar way:

$$o_u(t+1) = o_u(t) + \alpha[o_u(t) - o_u(t-1)] + \beta[a_u(t) - o_u(t)] \tag{3.21}$$

where the term $a_u(t)$, that depends on the specific optimization function has been considered constant [114].

In order to analyse this equation, it is possible to split it in two parts introducing a new variable $c(t)$:

$$c_u(t+1) = o_u(t+1) - o_u(t) \tag{3.22}$$

Expressing all the values at time $t+1$ as a function only of the time t:

$$o_u(t+1) = o_u(t) + \alpha c_u(t) + \beta\left[a_u - o_u(t)\right] \tag{3.23}$$

$$c_u(t+1) = \alpha c_u(t) + \beta\left[a_u - o_u(t)\right] \tag{3.24}$$

The term βa_u represents the steady state condition of this equation and it will be called p in the following.

It is possible to rewrite the two equations in matrix form:

$$\begin{bmatrix} o_u(t+1) \\ c_u(t+1) \end{bmatrix} = \begin{bmatrix} 1-\beta & \alpha \\ -\beta & \alpha \end{bmatrix} \cdot \begin{bmatrix} o_u(t) \\ c_u(t) \end{bmatrix} + \begin{bmatrix} p \\ p \end{bmatrix} \tag{3.25}$$

The behaviour of this model is driven by the matrix eigenvalues:

$$\lambda_{1,2} = \frac{-(\beta - \alpha - 1) \pm \sqrt{(\beta - \alpha - 1)^2 - 4\alpha}}{2} \tag{3.26}$$

Stability analysis

Analysing the stability is an important study for the understanding of the optimization capability of the algorithm itself [115]. It is not strictly required that at each iteration the behaviour is stable, but over long run the stability is required for the most of the step to guarantee that, if the attracting point is the global optimal solution, the algorithm is able to reach that solution.

The stability of the system is function of the eigenvalues, in particular it is required that:

$$\max\left(|\lambda_1|, |\lambda_2|\right) < 1 \tag{3.27}$$

This analysis is performed distinguishing the case in which the eigenvalues are complex conjugate and the case in which they are real numbers.

Complex eigenvalues: the eigenvalues are complex if

$$(\beta - \alpha - 1)^2 - 4\alpha < 0 \tag{3.28}$$

The eigenvalues are complex conjugate, thus their modules are equal:

$$|\lambda_1| = |\lambda_2| = \sqrt{Re(\lambda_{1/2})^2 + Im(\lambda_{1/2})^2} \tag{3.29}$$

Using the definition of the eigenvalues of Equation 3.26 the stability condition is:

$$\frac{[-(\beta - \alpha - 1)]^2 + [4\alpha - (\beta - \alpha - 1)^2]}{4} < 1 \tag{3.30}$$

Doing some maths, it is possible to obtain the stability condition for complex eigenvalues:

$$\begin{cases} (\beta - \alpha - 1)^2 - 4\alpha < 0 \\ \alpha < 1 \end{cases} \tag{3.31}$$

In this system, the first equation represents the condition for having complex eigenvalues, while the second is the real stability condition. This system is still valid also for time varying coefficients in the attraction equation [116]. Figure 3.12 shows the graphical representation of the system behaviour: the yellow part represents the stability condition and the blue part the condition for having complex eigenvalues. The intersection is the effective stability condition.

Real eigenvalues: this condition is given by:

$$(\beta - \alpha - 1)^2 - 4\alpha > 0 \tag{3.32}$$

The stability condition is the same seen before:

$$\max\left(|\lambda_1|, |\lambda_2|\right) < 1 \tag{3.33}$$

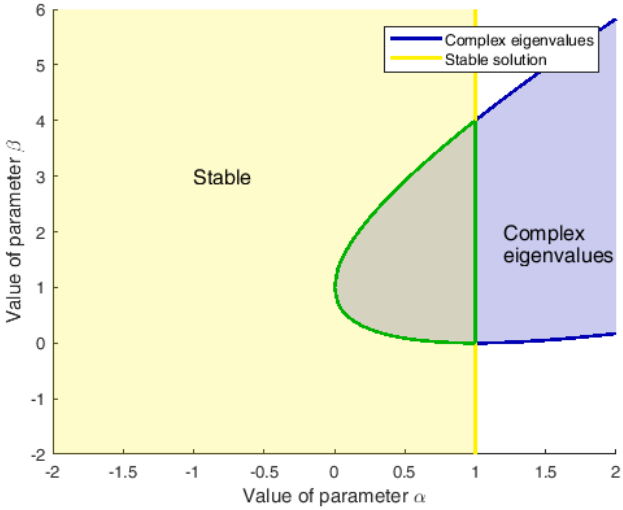


Figure 3.12: Stability condition for complex eigenvalues: the yellow part represents the stability condition and the blue part the condition for having complex eigenvalues. The intersection is the effective stability condition.

This case is slightly more complicated because it is necessary to evaluate the eigenvalue with the largest module. Thus, it is firstly analysed the stability condition when the first eigenvalue has a larger module and then the stability condition for the second eigenvalue.

The condition for having the module of the first eigenvalue larger than the second one is:

$$\left|\frac{-(\beta-\alpha-1)+\sqrt{(\beta-\alpha-1)^2-4\alpha}}{2}\right| > \left|\frac{-(\beta-\alpha-1)-\sqrt{(\beta-\alpha-1)^2-4\alpha}}{2}\right| \tag{3.34}$$

This inequality is satisfied when:

$$\beta-\alpha-1<0 \tag{3.35}$$

The stability condition for the first eigenvalue is the following:

$$\left|\frac{-(\beta-\alpha-1)+\sqrt{(\beta-\alpha-1)^2-4\alpha}}{2}\right| < 1 \tag{3.36}$$

Doing some maths, it is possible to find the solution of the inequality:

$$\begin{cases} \alpha<1 \\ \beta>0 \end{cases} \tag{3.37}$$

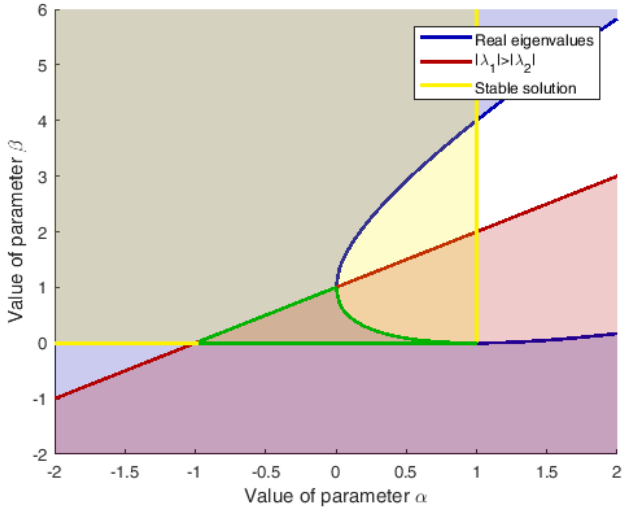


Figure 3.13: Stability condition for real eigenvalues, when the first is greater in module: the blue area represents the conditions in which the eigenvalues are real, the red is when the first eigenvalue has a greater module and the yellow area the stability condition. The intersection is the solution of the system.

Thus, the complete system for representing this stability condition is the following:

$$\begin{cases} (\beta - \alpha - 1)^2 - 4\alpha > 0 \\ \beta - \alpha - 1 < 0 \\ \alpha < 1 \\ \beta > 0 \end{cases} \tag{3.38}$$

where the first inequality express the condition for having real eigenvalues, the second one is for having $|\lambda_1| > |\lambda_2|$ and the last two conditions are related to the stability of the system.

Figure 3.13 shows a graphical representation of the stability system in this condition. The blue area represents the conditions in which the eigenvalues are real, the red is when the first eigenvalue has a greater module and the yellow area the stability condition. The solution of the system is the area surrounded by the green line.

Then, it is necessary to analyse the case with real eigenvalues when the module of the second one is grater than the first. The equation expressing this last condition is:

$$\beta - \alpha - 1 > 0 \tag{3.39}$$

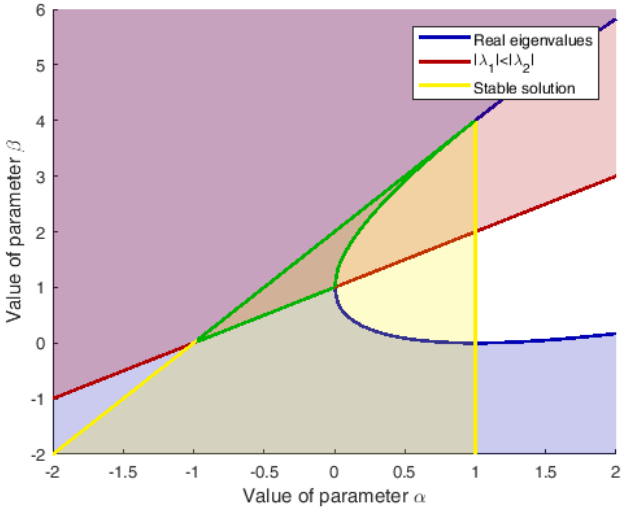


Figure 3.14: Stability condition for real eigenvalues, when the second one is greater in module: the blue area represents the conditions in which the eigenvalues are real, the red is when the second eigenvalue has a greater module and the yellow area the stability condition. The solution of the system is the area surrounded by the green line.

In this case, the stability condition is the following:

$$\left|\frac{-(\beta-\alpha-1)-\sqrt{(\beta-\alpha-1)^2-4\alpha}}{2}\right|<1 \tag{3.40}$$

Solving the inequality, it is obtained the following condition:

$$\begin{cases} \alpha<1 \\ 2\alpha-\beta+2>0 \end{cases} \tag{3.41}$$

That means that the final system for this condition is:

$$\begin{cases} (\beta-\alpha-1)^2-4\alpha>0 \\ \beta-\alpha-1>0 \\ \alpha<1 \\ 2\alpha-\beta+2>0 \end{cases} \tag{3.42}$$

Figure 3.14 shows the conditions in this case. The blue area represents the conditions in which the eigenvalues are real, the red is when the second eigenvalue has a greater module and the yellow area the stability condition. The solution of the system is the area surrounded by the green line.

The overall stability of the system is the union of the three calculated solutions. It is possible to represent it graphically, as in Figure 3.15. The parameters should be inside the coloured area to have a stable behaviour of the population of the algorithm.

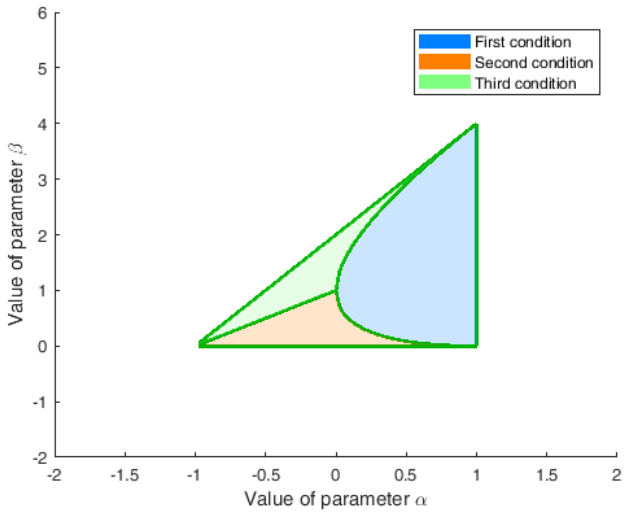


Figure 3.15: Overall stability of the operation, union of the three stability conditions.

In the next section, the attraction operator will be analysed for what concern the type of trajectories of the Social Network users in the space of the solutions.

Trajectory analysis

From the attraction equation 3.18 written with the matrix form of Equation 3.25, it is possible to find the type of trajectories in the plane of the solutions. This analysis is important to understand the impact of the operators on the exploration and exploitation capabilities of the algorithm.

It is possible to find four type of solutions according to two criteria: oscillations and zigzagging. This means that the four types of trajectories are oscillating, non-oscillating, zigzagging and zigzagging with oscillations.

These trajectories are driven by two elements [114]:

- trajectories are oscillating if the eigenvalues are complex values;
- trajectories are zigzagging when the real part of at least one of the eigenvalues is positive.

The condition for the oscillation has been already analysed before, when the condition for having complex eigenvalues has been analysed:

$$(\beta - \alpha - 1)^2 - 4\alpha < 0 \tag{3.43}$$

The oscillating behaviour is shown in Figure 3.16. The blue part represents the oscillations, while the light blue is the non-oscillating region. The grey triangle is the stability region, that is independent from the behaviour of the population: in fact, it is possible to have an oscillating or static (*i.e.* non oscillating) convergence and an oscillating or static divergence.

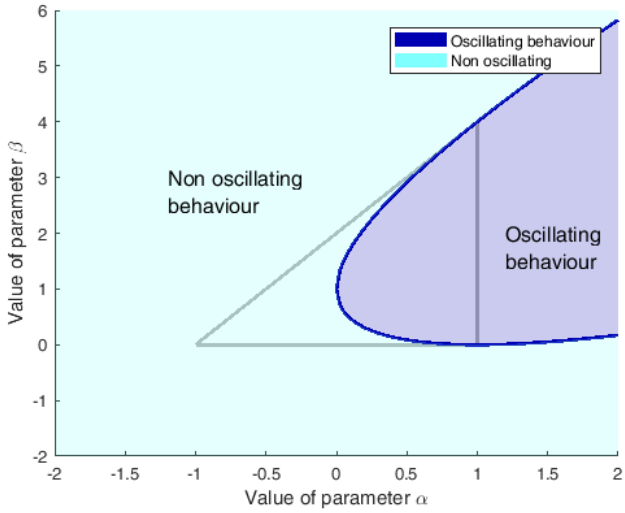


Figure 3.16: Behaviour of the population of SNO: the blue part represents the oscillations, while the light blue is the non-oscillating region. The grey triangle is the stability region.

The second possible trajectory type criterion is the zigzagging. Zigzagging appears when at least one of the eigenvalues has a real negative part. This condition can be analysed in two cases: the first case is when the eigenvalues are complex, the other one is when the eigenvalues are both real. This mathematical consideration reflects the fact that oscillatory behaviour and zigzagging can be combined.

When the eigenvalues are complex, the condition on zigzagging is expressed as:

$$-(\beta - \alpha - 1) < 0 \tag{3.44}$$

So, the first zigzagging condition:

$$\begin{cases} (\beta - \alpha - 1)^2 - 4\alpha < 0 \\ \alpha - \beta + 1 < 0 \end{cases} \tag{3.45}$$

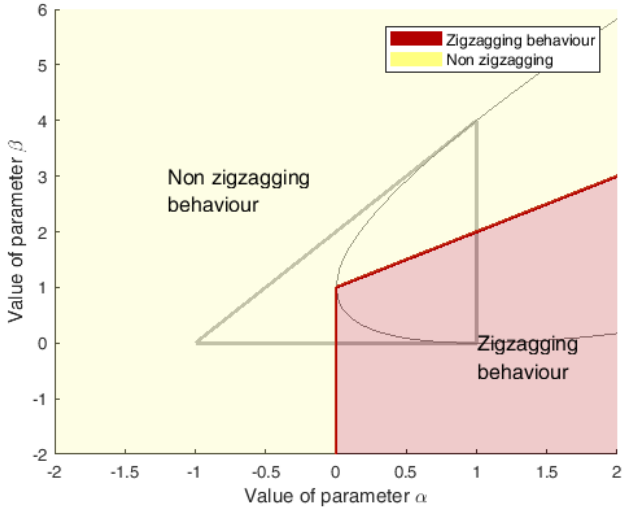


Figure 3.17: Behaviour of the population of SNO: the red part represents the zigzagging, while the yellow is the non-zigzagging region. The grey triangle is the stability region.

On the other hand, when eigenvalues are real, it is possible to analyse the smaller one:

$$-(\beta-\alpha-1)-\sqrt{(\beta-\alpha-1)^2-4\alpha}<0 \tag{3.46}$$

Performing some calculations, the second zigzagging condition is obtained:

$$\begin{cases} (\beta-\alpha-1)^2-4\alpha>0 \\ \alpha>0 \end{cases} \tag{3.47}$$

Figure 3.17 shows the condition of zigzagging behaviour: the red part represents the zigzagging, while the yellow is the non-zigzagging region. The grey triangle is the stability region.

Finally, it is possible to combine the information of oscillations and zigzagging to have the complete four behaviours of the population. Figure 3.18 shows it: the four behaviours are identified by the colours (grey for static, blue for oscillating, yellow for zigzagging and green for both zigzagging and oscillating), while the stability area is the highlighted triangle in the centre of the domain.

3.3.2 Parametric Analysis

In the previous Section, a theoretical analysis of the attraction formula has been done: the goal of that analysis is to give a better understanding on the

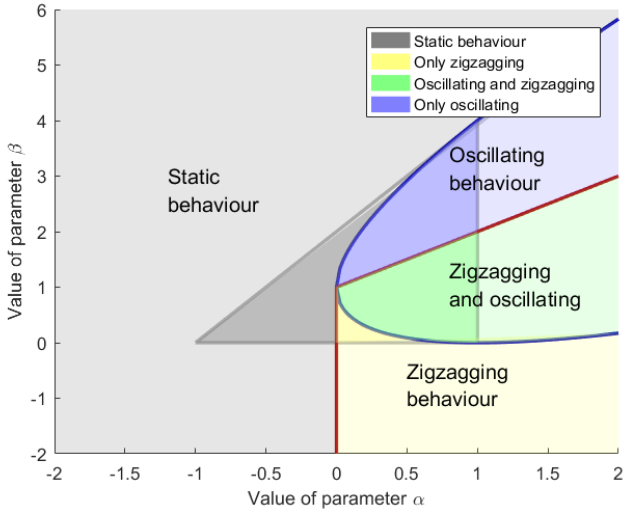


Figure 3.18: Complete trajectory behaviour as function of the parameters α and β. The four possible behaviours are highlighted with colours, the stability area has stronger colours.

meaning of the parameters. Nonetheless, the final choice of the free algorithm parameters should be done with a parametric analysis [76].

The parametric analysis is performed in order to define the "optimal" value of the algorithm parameters that drive the most important operators. In particular, in this analysis three operators are investigated: the complex contagion function, the growing function and the linguistic transposition. These are the most important operators because they drive the trade-off between exploration and exploitation.

In addition to the parameters of these operators, also the impact of the population size is investigated. Table 3.1 shows all the parameters tested, their range value and the number of samples in the range. The parameters are grouped because the parametric analysis has been done with two varying parameters per time. In this way their cross-influence is investigated, obtaining as output a cost surface.

The optimal value of the parameters is related with the specific objective function that is optimized. In this parametric analysis, four objective functions have been used to explore a set of different features. They are:

- Ackley function, it is a multimodal function that presents many small local minima on a general trend with concave shape. This function generally requires a good amount of exploitation in the optimization

Parameter name	Symbol	Min value	Max value	Samples
Number of design variables	N	5	100	20
Population size	m	5	250	50
Influence inertia	α	-2	2	150
Influence attraction	β	-2	6	150
Linguistic error rate	μ_r	0	1	100
Linguistic error amplitude	μ_a	0	1	100

Table 3.1: Parameter analyzed, range and number of samples.

algorithm.

$$f(\mathbf{x}) = 20 + e^1 - 20 \cdot e^{-0.2\sqrt{\sum (x_i - x_0)^2/N}} - e^{\sum \cos(2\pi(x_i - x_0)/N} \tag{3.48}$$

where N is the length of $\mathbf{x}$ and $x_0 = -7$. This function is defined in the domain:

$$-15 \leq x_i \leq 15 \tag{3.49}$$

- Griewank function, that is characterized by a trend as a quadratic function with many local minima.

$$f(\mathbf{x}) = 1 + \sum (x_i - x_0)^2/4000 - \prod \cos \frac{x_i - x_0}{\sqrt{i}} \tag{3.50}$$

where $x_0 = 150$. Its domain is:

$$-600 \leq x_i \leq 600 \tag{3.51}$$

- Schwefel-226 function, that is an odd function characterized by two important minima far one from the other.

$$f(\mathbf{x}) = 418.9829 \cdot N + \sum x_i \cdot \sin \sqrt{|x_i|} \tag{3.52}$$

where N is the length of $\mathbf{x}$. Its domain is:

$$-512 \leq x_i \leq 512 \tag{3.53}$$

- Multidimensional Sinc function (Caridinal Sine, Sinc-N). This is a multimodal function with very small local minima. Generally, it requires a well-tuned trade-off between the exploration required to find the global minimum attractor and exploitation to converge quickly toward the minimum.

$$f(\mathbf{x}) = 1 - \prod |\sin[\pi \cdot (x_i - x_0)]| \tag{3.54}$$

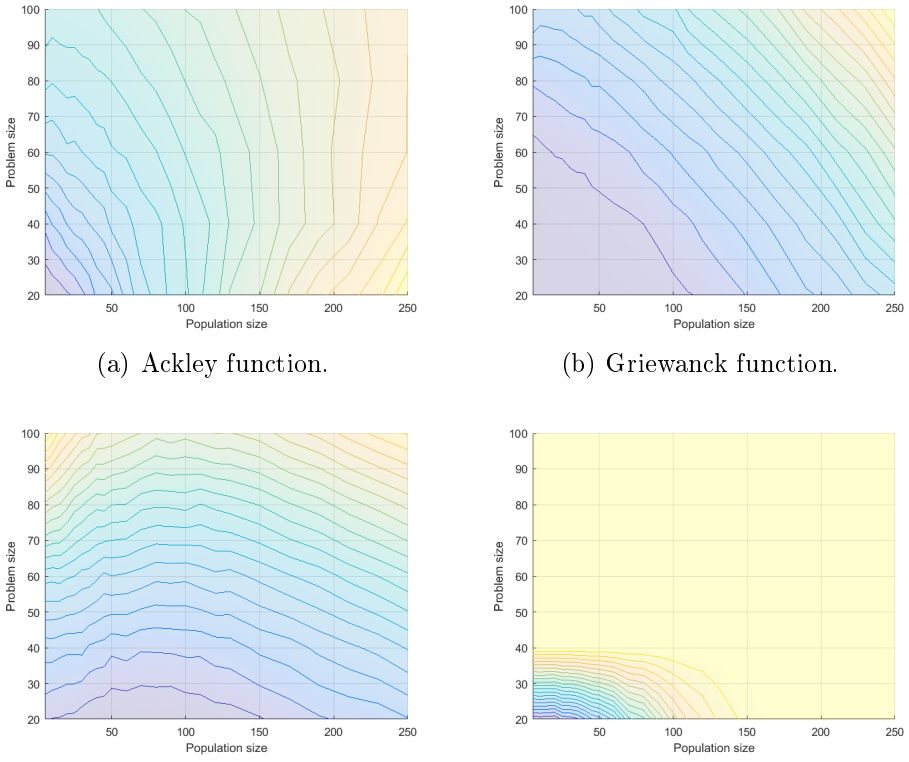
where $x_0 = 3$. Its domain is:

$$0 \leq x_i \leq 10 \tag{3.55}$$

For having a statistical reliability and for reducing the impact of the intrinsic stochastic behaviour of the algorithm, 50 independent trials have been performed for each parameter combination.

In the second and third tests, the number of design variables has been fixed to 20 and the number of objective function calls to 5,000, while for the first test the number of design variables is an analysed parameter and the objective function calls have been modified linearly, fixing 5,000 with 20 design variables.

The first tests have been done on the optimal population size changing the problem size. The results obtained are shown in Figure 3.19.



(a) Ackley function.

(b) Griewanck function.

(c) Schwefel 226 function.

(d) Sinc-N function.

Figure 3.19: Parametric analysis on the population size as function of the number of design variables of the problem.

The SinN function is not interesting because the algorithm reaches the same value for the most of the test done.

It is possible to see two different behaviours for Ackley/Griewank and Schwefel-226 functions: in the first two, the optimal population size is always

very low because it increase the exploitation, while for the third function the optimal value is in the range 50-100 individuals: a large population, in fact, increases the exploration capability of the algorithm and the amount of information introduced in the first initialization.

For what concerns the trend of modification of the population size with the problem size, for all the three functions the best population size grows increasing the problem size.

Figures 3.20,3.21, 3.22, and 3.23 show the results of the parametric analysis on the complex contagion operator parameters, α and β. In these figures, in the background the interaction formula analysis results are shown, in particular the green triangle is the stability area and the grey lines shows the divisions between different population behaviour. The blue area is the best result, while red ones are high cost point.

These results are very interesting: it is possible to see, especially in Figure 3.22, that for $\alpha < -1$ or $\beta < 0$ the algorithm reaches the worst results. Another important aspect is that the convergence area overcomes sometime the stability limit: this is due to the fact that the analysed formula has some less features with respect to the real one.

There is a limit in the convergence that is not highlighted by the previously proposed analysis: it is due to the limitations of the search domain. In fact, the population oscillates with very high amplitude, and, thus, they exit the search domain. In this condition, the feasibility check changes the algorithm behaviour avoiding the convergence.

The converging area is common for Ackley, Griewank, and Sinc-N functions (Figures 3.20,3.21, and 3.23) while it is slightly different for Schwefel function (Fig. 3.22). This shows the great differences between these functions.

Analysing the convergence of Sinc-N function, it is possible to notice that there are two converging zones: a first one that is in the zigzagging and oscillating part, that ensures good exploration for the overshooting, and a second one in the static behaviour: in this case the exploration is guaranteed by the very low inertia of the system that makes the movement of the population very fast. It is possible to say that in this last case the behaviour of SNO becomes more similar to a GA.

The best point of convergence of Schwefel-226 (Figure 3.22)is an interesting point because it guarantees also good results for the other functions, even if they are not optimal.

From this analysis two possible set of parameters have been extracted. The first one is the optimal point for Ackley and Griewanck functions, while the second one is the optimum for Schwefel-226 function. In the following parametric analysis both these combinations have been tested. The final

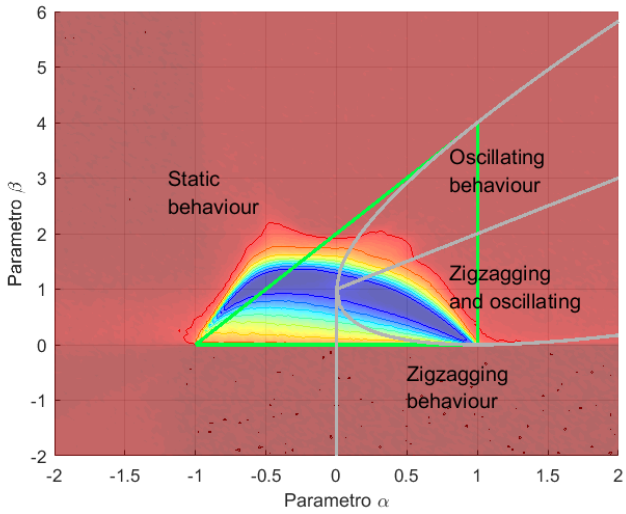


Figure 3.20: Parametric analysis on the complex contagion operator parameters, α and β. In the background the results of the analysis are reported. Results for Ackley function.

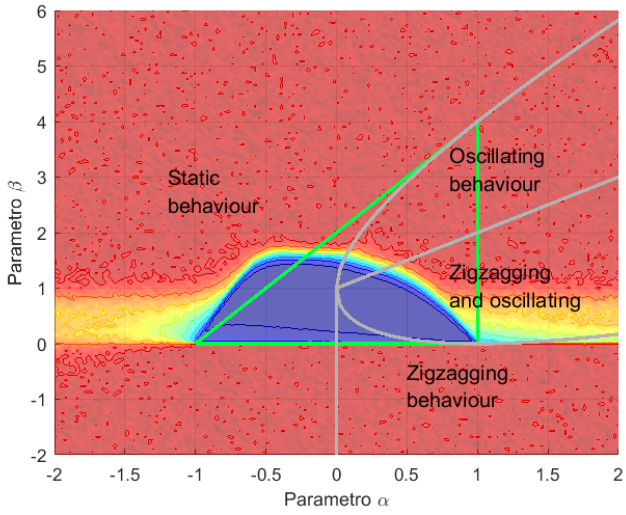


Figure 3.21: Parametric analysis on the complex contagion operator parameters, h and ω_0. In the background the results of the analysis are reported. Results for Griewanck function.

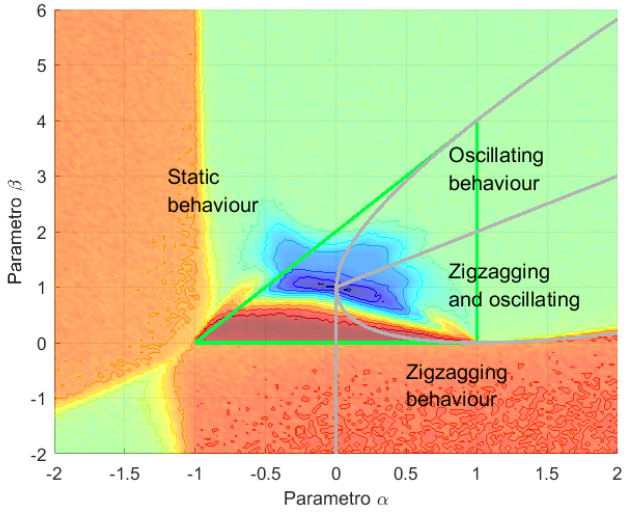


Figure 3.22: Parametric analysis on the complex contagion operator parameters, h and ω_0. In the background the results of the analysis are reported. Results for Schwefel-226 function.

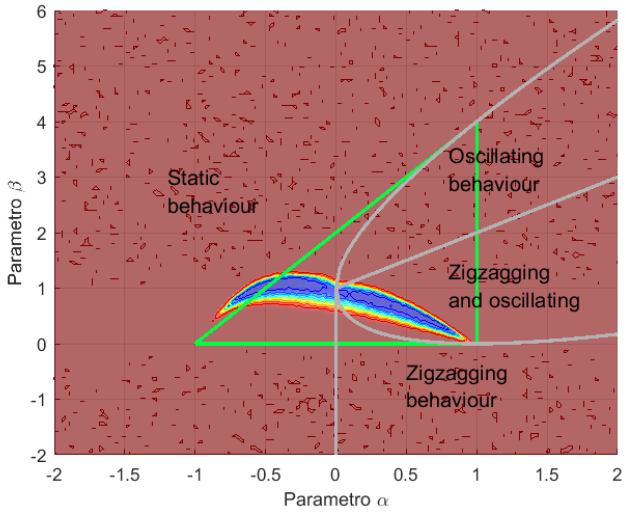
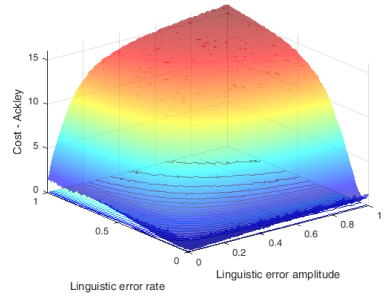


Figure 3.23: Parametric analysis on the complex contagion operator parameters, h and ω_0. In the background the results of the analysis are reported. Results for Sinc-N function.

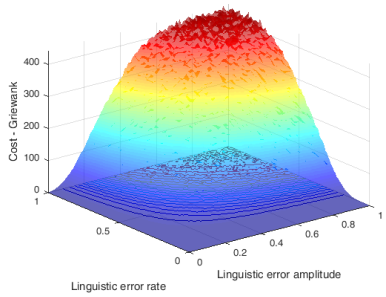
selection is discussed in the next chapter, with the comparison among algorithms.

The second operator analysed is the linguistic transposition. This has two parameters, the linguistic error rate and amplitude. These are coupled, thus they have been analysed together. The combination of these two parameters tunes the equivalent of the mutation in SNO. They are the main driver for the introduction of completely new information in the algorithm population.

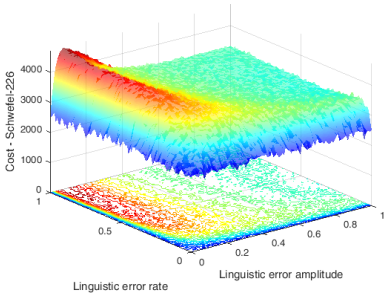
The results with the first set of α and β are shown in Figure 3.24. As highlighted also for the previous parametric analysis, the Schwefel-226 has a very different behaviour with respect to the other functions.



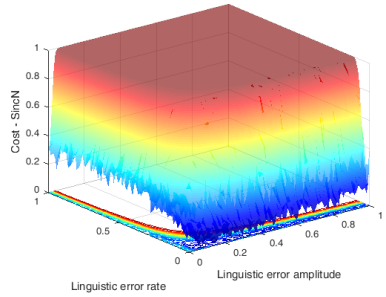
(a) Results on Ackley function.



(b) Results on Griewank function.



(c) Results on Sinc-N function.



(d) Results on Schwefel-226 function.

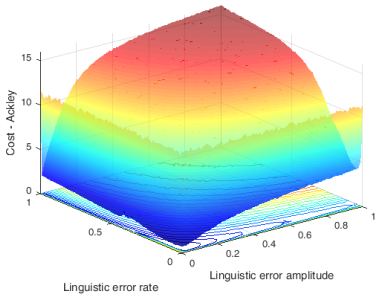
Figure 3.24: Parametric analysis on the linguistic transposition parameters, error rate and error amplitude.

For all the functions, the values obtained for very high mutation rates represent the results of a random search in the domain.

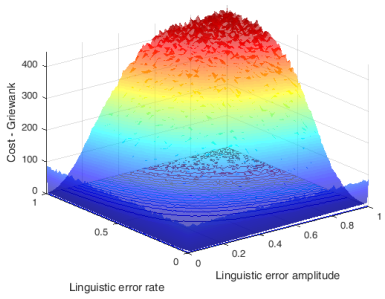
For Ackley and Sinc-N function (Figures 3.24(a) and 3.24(c)), the optimal results are obtained for low values of linguistic error rate and amplitude, but for null values the results are non optimal. Moreover, in these functions, the results decay quite quickly when the mutation is increased too much. This behaviour is more evident for Sinc-N function.

On the other hand, for Griewanck (Fig. 3.24(b)) function the algorithm works very well also for null mutation. This is due to the concave average shape of the function. Here, the decay of the performance increasing the mutation is lower than for the other functions.

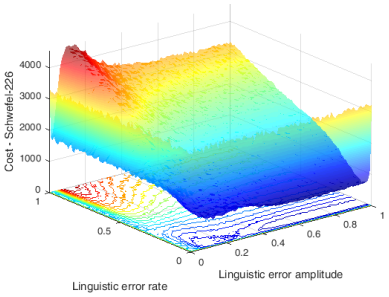
The results for Schwefel-226 function (Fig. 3.24(d)) are completely different. The algorithm works better for very low or null mutation, even if this is not relevant because the reached value is much higher: this shows that the algorithm is not able to reach convergence. For this function, it is very interesting the peak reached for linguistic error amplitude around 0.1. This is due to the peculiar shape of the Shcwefel-226 function: in fact, this function has a local minimum on the boundary of the domain, thus a diverging results as the one obtained with a large linguistic error amplitude gives better results than a simple random search localized in a central region of the domain.



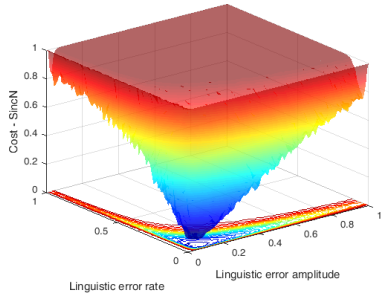
(a) Results on Ackley function.



(b) Results on Griewank function.



(c) Results on Sinc-N function.



(d) Results on Schwefel-226 function.

Figure 3.25: Parametric analysis on the linguistic transposition parameters, error rate and error amplitude.

Figure 3.25 shows the results of the parametric analysis on the linguistic transposition using the second set of parameters α and β.

As expected, the optimal results for Ackley and Griewank functions (Figures 3.25(b) and 3.25(a)) are worsened, while for Schwefel-226 function (Fig-

ure 3.25(c)) the improvement is very important: in fact the optimal value is reduced from 1870 of the previous test to 436.

For Sinc-N function (Figure 3.25(d)) the optimal result is the same as before, but the optimal zone is shifted. The parameter values are more similar to Ackley and Griewank functions, even if the optimal area is smaller.

3.4 SNO performance and comparison with other algorithms

In this Section, the performance of SNO on some standard benchmarks for Evolutionary Optimization are analysed and compared with the results of a set of other algorithms.

3.4.1 Genetic Algorithm

The Genetic Algorithm (GA) is the most popular among Evolutionary Optimization algorithms. His first implementation is suited for binary problems [78]. Then, many other implementations have been introduced; in particular, many implementations for real-valued problems have been developed [59].

The Genetic Algorithm is based on three operators: the selection, the crossover and the mutation [117].

The selection operator is devoted to extract a certain number of individuals from the population. Several criteria can be used: the random selection can be used for increasing the population diversity; the roulette wheel selection can be used to select individuals with probability proportional to the fitness. The individuals can be also selected proportionally to their rank in the population (rank-based selection).

An important selection operator is the *stud selection* that at every iteration extract the best individual of the population. The use of this operator often is explicitly indicated in the name of the algorithm: the resulting algorithm is the *stud-genetic algorithm* (SGA) [20].

The second operator is the crossover, that is devoted to creating the new population recombining the existing information in the population. Several variations of this operator have been tested in literature: from the single point crossover, to the uniform or arithmetic crossover for the real-value problems. Many other crossover operators have been implemented for solving some specific problems [76].

Finally, the last operator of GA is the mutation: this is devoted to introducing new information in the population with some random modifications

of the population. Also in this case, many possible rules can be used [118].

3.4.2 Differential Evolution

The Differential Evolution (DE) is an algorithm that is not biologically inspired. It has been introduced in [119] for solving continuous optimization problems with the aim to handle non-differentiable, nonlinear and multimodal cost functions. Moreover, it has been designed to be easy for the user (robustness of the choice of the parameters) and suitable for parallelization [119].

The algorithm has been applied to a wide range of problems obtaining very good results [120].

The algorithm is based on the *vector-based mutation*: this operator has the objective to create the new population starting from the existing one [121].

As first step, two individuals of the population are selected (all the selection possibilities can be used) and the difference vector is calculated. Then a third individual is selected, and it is moved in the search space by a quantity that is proportional to the difference vector [121]:

$$\mathbf{x}_i(t+1) = \mathbf{x}_{r_1}(t) + F \cdot (\mathbf{x}_{r_2}(t) - \mathbf{x}_{r_3}(t)) \tag{3.56}$$

For improving the potential diversity of the population, crossover or mutation operators can be applied [121].

3.4.3 Biogeography Based Optimization

The Biogeography Based Optimization (BBO) is a biologically inspired algorithm recently developed [13]. This algorithm is a variation of the GA in which the crossover operator has been modified for having an higher convergence rate.

This modification often leads to an early convergence in many multimodal problems: for solving this problem a modification has been introduced in [122] modifying the selection operator and introducing the *cataclysm* when the population stagnates. In this thesis, this modification has been used and it is referred as mBBO.

Another modification has been here tested for solving the problem of the early convergence: in this case the mutation operator has been modified and its impact has been increased. This second modification is called nBBO.

3.4.4 Particle Swarm Optimization

The Particle Swarm Optimization (PSO) is another well-known population-based evolutionary algorithm implemented for real-value problems [123].

This algorithm has been widely studied and applied [124]. His performance is highly dependent on the specific selection of the parameters and, with respect to GA, it is characterized by an higher convergence rate that in some cases leads to a premature stagnation in local minima [125].

In this algorithm, each particle (individual of the population) moves in the search space according to three criteria: an inertia, an attraction to the global best position found by the population and the attraction towards his personal best.

Also for this algorithm, several variations have been implemented for reducing the early convergence: in particular, the population is often clustered for having another attraction term, the group best [126].

Some other variations have been tested introducing repulsion terms: these are very useful for avoiding early stagnation, but it can bring the algorithm to instability [64].

3.4.5 Random Search and Random Walk

These two algorithms are not always considered Evolutionary Algorithms because the operators that creates the new population is completely random.

The random search is a completely random algorithm in which the point are selected independently in the search space [76].

On the other hand, the random walk is a point-based algorithm in which, starting from a random position, the candidate solution is moved with random direction and step size in the search space. In this algorithm, an elitism is considered: in fact the point position is updated only if the new fitness value is better than the old one [76].

3.4.6 Point-based algorithms

In the comparison, also three typical point-based methods have been tested: the Quasi-Newton algorithm (QN) [127], the Simplex algorithm (SPX) [128], and the Steepest Descend algorithm (SD) [129].

These algorithms are usually employed for local search due to their high sensitivity to the initial point of the search. For managing this problem, at each independent trials the starting guess is selected randomly.

3.4.7 Numerical resuls

All the numerical results are based on a set of standard mathematical benchmarks [130]. These functions have different features in order to analyse the behaviour of the algorithms in different conditions. Details of these functions can be found in the Appendix A.

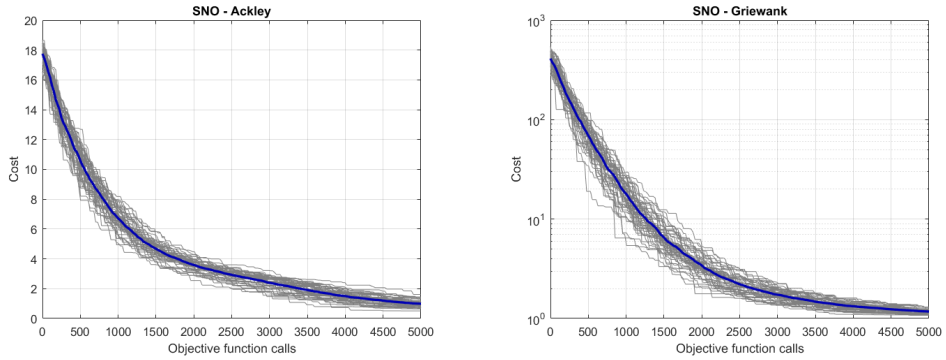
Many of these functions are multi-modal, with different local minima size and location in the search domain; two functions that emulates the penalty definitions; and other are non derivable or non-continuous.

Adopting this set of functions, Social Network Optimization has been firstly assessed and then compared with the other algorithms.

Social Network Optimization

The behaviour of SNO in different conditions can be analysed observing the convergence curves on the different mathematical benchmark functions. For each objective function, 50 independent trials have been performed with termination criterion 5,000 function calls.

Figure 3.26(a) shows the convergence on Ackley function, and Figure 3.26(b) on Griewank function: here, the very small standard deviation of the results obtained by SNO can be appreciated.



(a) Ackley function. (b) Griewank function.

Figure 3.26: Convergence curves of the 50 independent trials of SNO. Each grey line is the convergence of one single trial, while the blue thick line is the average convergence.

On the two Penalty functions (reported in Figures 3.27(a) and 3.27(b)) it is possible to clearly see the two different convergence rates: the first part of the convergence is related to the penalty area of the function, while the

second part is the one devoted to the finding of the minimum within the non-penalized area.

Figure 3.27(c) shows the convergence on the Quartic function: here the behaviour of the algorithm is very regular due to the simple shape of the function. The convergence on the Rastrigin function (Figure 3.27(d)) is quite slow, but the algorithm is never blocked in one of the many local minima of this function.

Figure 3.27(e) shows the convergence curves on the Rosenbrock function: here, there are two groups of convergence that correspond to the two almost flat parts of the function.

The convergence on Schwefel-12 function (Figure 3.27(f)) is quite low, even if it is positive that the algorithm does not stack on the local minima, but the convergence is going on for all the optimization time.

Figure 3.28(a) shows the convergence on the Schwefel-221 function, while Figure 3.28(b) Schwefel-222 function: here, the similarity of the trials is impressive, showing the robustness of SNO.

Figure 3.29(a) shows the convergence on Schwefel-226 function: here the initial very fast convergence is followed by a much slower rate in the last part of the optimization time.

On Sinc function, shown in Figure 3.29(b) , it is possible to see that the convergence of the single trials are often blocked in the local minima of the function.

In the Sinc-N function (Figure 3.29(c)), all the independent trials are able to begin the convergence toward the global minimum: this is an impressive result because this function is characterized by a very large almost flat area.

Finally Figures 3.29(d) and 3.29(e) show the convergence on the Sphere and the Step functions.

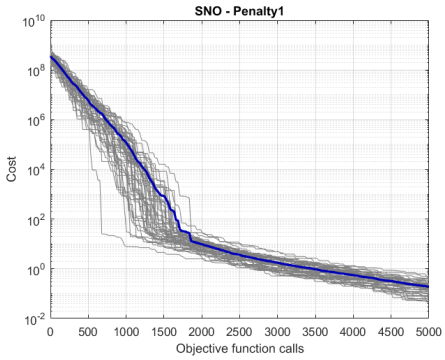
Analysing these convergence curves, it is possible to see the general good behaviour of SNO, that is able to reach very good results in many functions.

In the following, these results are compared with the ones of the other optimization algorithms.

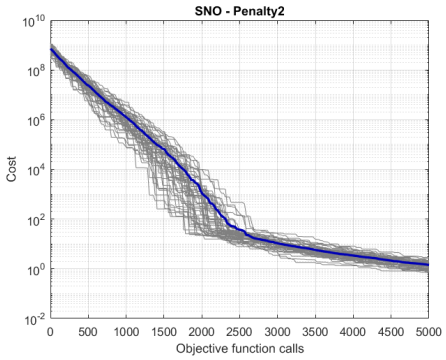
Comparison

Two groups of comparison have been done: the first one is among Evolutionary Algorithms, while the second one is between SNO, the random algorithms and the point-based algorithm.

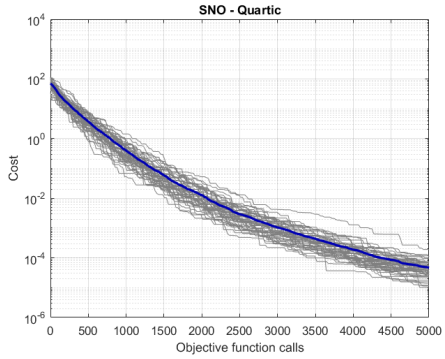
For each algorithm, the termination criterion has been set to be 5,000 objective function calls. For having a statistical reliability 50 independent trials have been done. The objective functions are all defined with 20 design variables.



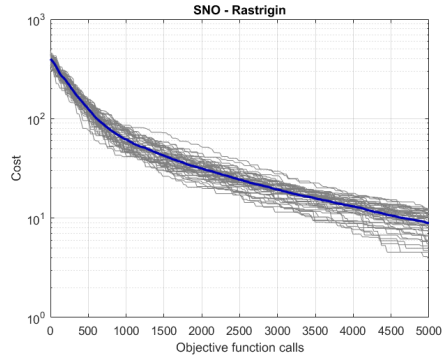
(a) Penalty1 function.



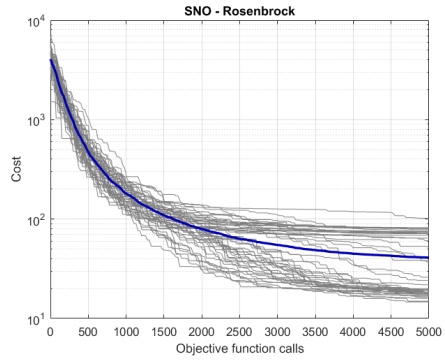
(b) Penalty2 function.



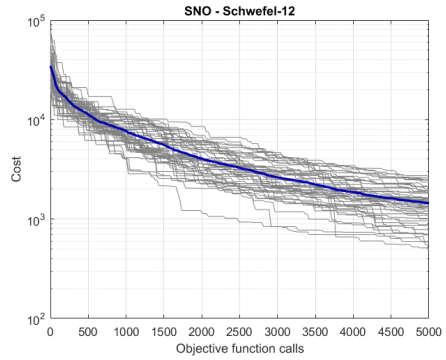
(c) Quartic function.



(d) Rastrigin function.

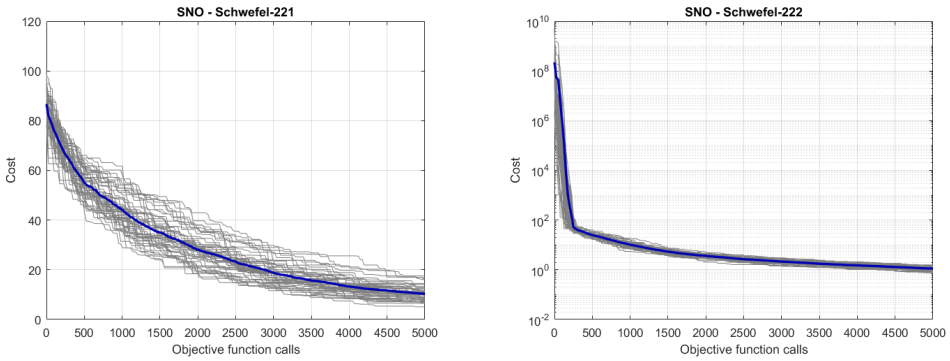


(e) Rosenbrock function.



(f) Schwefel-12 function.

Figure 3.27: Convergence curves of the 50 independent trials of SNO. Each grey line is the convergence of one single trial, while the blue thick line is the average convergence.



(a) Schwefel-221 function.

(b) Schwefel-222 function.

Figure 3.28: Convergence curves of the 50 independent trials of SNO. Each grey line is the convergence of one single trial, while the blue thick line is the average convergence.

For what concern the point-based algorithms, the initial guess has been extracted randomly from the search domain at each trial: for multimodal function, the final result is highly affected by the initial guess.

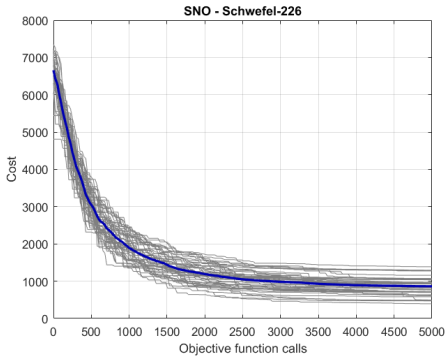
For the comparison, the number of objective function calls can be considered as the most interesting termination criterion: in fact, each optimization algorithm has a different computational cost due to its internal operators, but the most of the total computational time is due to the evaluation of the cost function.

For what concern the computational complexity of the algorithm, it can be calculated as function of the population size N, the number of iterations I, and the number of design variables of the problem (M). It is important to notice that the specific selection of the operators of the algorithms highly impact the computational complexity of the algorithms. For more details on the single operators, see [76].

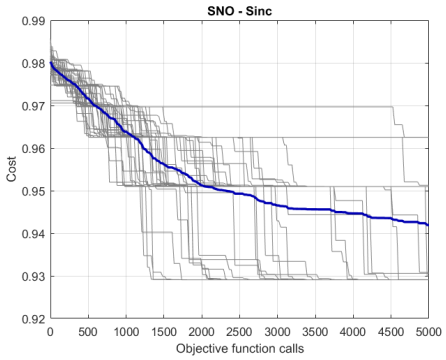
Table 3.2 contains the computational complexity of the evolutionary algorithms implemented: the first column represent the computational complexity as function of I, N, and M, while the second column contains the same information as function of the number of the objective function calls, remembering that:

$$C = N \cdot I \tag{3.57}$$

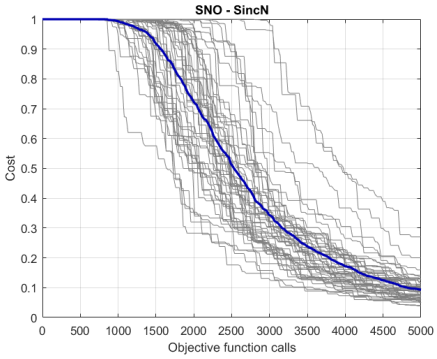
Tables 3.3 and 3.4 shows the average value obtained by all the algorithm. For each function (row) the best result is indicated in bold, while for each algorithm (column) the number of wins and a score value have been calculated.



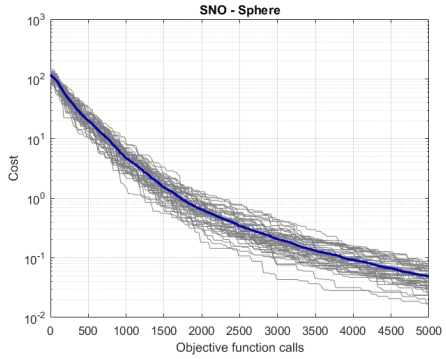
(a) Schwefel-226 function.



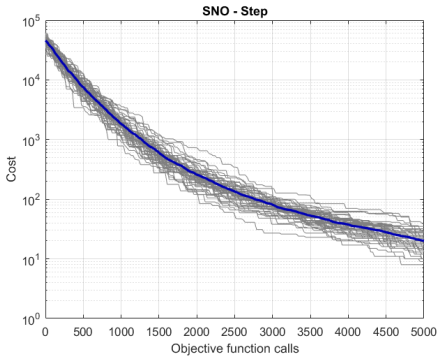
(b) Sinc function.



(c) Sinc-N function.



(d) Sphere function.



(e) Step function.

Figure 3.29: Convergence curves of the 50 independent trials of SNO. Each grey line is the convergence of one single trial, while the blue thick line is the average convergence.

Algorithm	Computational complexity	
DE	$I \cdot N$	C
GA	$I \cdot N$	C
mBBO	$I \cdot N \cdot M$	$C \cdot M$
nBBO	$I \cdot N \cdot M$	$C \cdot M$
PSO	$I \cdot N$	C
SGA	$I \cdot N \cdot \log N$	$C \cdot \log N$
SNO	$I \cdot N \cdot \log M$	$C \cdot \log M$

Table 3.2: Computational complexity of the EAs as function of the number of iterations I, the population size N, the number of design variable M and the number of objective function calls C.

The score used is calculated according to the following formula:

$$S_a = \frac{r_a - \min(r)}{\max(r) - \min(r)} \tag{3.58}$$

where S_a is the score for the a-th algorithm; r_a is the result of the a-th algorithm, $\min(r)$ is the best result achieved by one of the algorithms and $\max(r)$ the worst one.

This score takes into account the fact that the good feature that an algorithm should have is the capability of having a competitive performance on a large set of objective functions; in fact, for a general application of EAs, this feature is more important than the capability of being the best in a narrow set of functions.

Analysing Table 3.3, it is possible to notice that the DE is the algorithm that is able to achieve the maximum number of wins; however, the DE performance are generally low on the other functions. For this reason, the final score is higher than the one of SNO.

In fact, SNO is very competitive on a large set of objective function, with very good results also when it is not the best algorithm among all.

Also the nBBO has a good score, due to the fact that its performance are quite good in all the function, even if never the best: this can be explained by the high mutation in this algorithm that makes it very robust to local minima.

GA and PSO have intermediate results: it is interesting to notice that GA is able to be the best algorithm on two objective function, in particular on Schwefel-12 that is a very hard function.

Finally, the score of mBBO and SGA are quite high, due to the very low performance scored in several functions (on Penalty 2 the SGA is very far from the global minimum).

	DE	GA	mBBO	nBBO	PSO	SGA	SNO
Ackley	**0.02**	2.4	2.84	1.73	3	0.98	0.99
Griewank	**0.24**	1.63	2.1	1.15	1.27	1.07	1.17
Penalty1	**0.02**	0.56	1.18	24.9	5.32	108.21	0.19
Penalty2	**0.23**	2.73	10.96	301.41	11.23	5684.38	1.43
Quartic	$\mathbf{7}\cdot10^{-8}$	$18\cdot10^{-5}$	$12\cdot10^{-4}$	$6\cdot10^{-4}$	$16\cdot10^{-5}$	$3\cdot10^{-3}$	$5\cdot10^{-5}$
Rastrigin	40.63	71.59	15.3	34.47	143.11	11.35	**8.91**
Rosenbrock	**17.38**	23.17	66.59	30.97	19.36	71.81	40.96
Schwefel-226	1581.49	2809.33	**257.25**	766.72	3452.93	1057.17	863.5
Schwefel-12	4123.59	**264.29**	3757.51	900.5	287.08	3988.65	1443.91
Schwefel-222	**0.03**	2.8	2.82	1.22	5.01	0.15	1.12
Schwefel-221	6.22	**4.52**	28.7	11.95	13.29	38.47	10.36
Sinc	**0.93**	0.97	0.97	0.98	0.94	0.98	**0.93**
Sinc-N	0.69	0.73	0.45	0.42	1	0.73	**0.09**
Sphere	$\mathbf{7}\cdot10^{-5}$	0.18	0.35	0.04	0.07	0.03	0.05
Step	**0.18**	67.62	130.48	20.98	33.44	29.36	20.1
Num. wins	**9**	2	1	0	0	0	3
Score	0.17	0.41	0.58	0.29	0.45	0.6	**0.16**

Table 3.3: Comparison between SNO and other EAs. All the functions have 20 design variables, the termination criterion is set 5,000 objective function calls, and the reported value is the average of 50 independent trials.

	RS	RW	QN	SPX	SD	SNO
Ackley	15.27	8.98	17.72	17.75	17.71	**0.99**
Griewank	218.96	39.69	$\mathbf{10}\cdot10^{-3}$	156.78	0.02	1.17
Penalty1	$7\cdot10^{7}$	$19\cdot10^{4}$	$12\cdot10^{7}$	$31\cdot10^{7}$	$13\cdot10^{7}$	**0.19**
Penalty2	$18\cdot10^{7}$	$23\cdot10^{5}$	$16\cdot10^{7}$	$7\cdot10^{7}$	$17\cdot10^{7}$	**1.43**
Quartic	15.24	0.65	$\mathbf{3}\cdot10^{-6}$	45.23	$5\cdot10^{-6}$	$5\cdot10^{-5}$
Rastrigin	268.62	116.56	337.88	364.06	355.74	**8.91**
Rosenbrock	1638.52	238.38	66.27	1141.72	122.28	**40.96**
Schwefel-226	5242.55	5160	3922.16	4293.32	3688.32	**863.5**
Schwefel-12	$11\cdot10^{3}$	3781.69	**61.1**	$18\cdot10^{3}$	7236.99	1443.91
Schwefel-222	1279.3	$16\cdot10^{3}$	$15\cdot10^{11}$	92.86	$8\cdot10^{11}$	**1.12**
Schwefel-221	64.28	32.82	44.82	63.38	**3.58**	10.36
Sinc	0.97	0.95	0.96	0.98	0.94	**0.92**
Sinc-N	1	1	1	1	1	**0.09**
Sphere	61.47	11.64	$11\cdot10^{-13}$	35.22	$\mathbf{11}\cdot10^{-14}$	0.05
Step	$23\cdot10^{3}$	4532.28	$8\cdot10^{4}$	$11\cdot10^{3}$	$8\cdot10^{4}$	**20.1**
Num. wins	0	0	3	0	2	10
Score	0.73	0.3	0.55	0.75	0.49	0.01

Table 3.4: Comparison between SNO and other algorithms. All the functions have 20 design variables, the termination criterion is set 5,000 objective function calls, and the reported value is the average of 50 independent trials.

In Table 3.4 the comparison between SNO and the random algorithms and the point-based algorithms is provided. Also in this case the number of wins and the score are reported.

It is possible to see that SNO outperforms all the other algorithms especially on multimodal functions. In fact, on single modal functions (like Quartic, Rosenbrock and Sphere), the point-based algorithms are the best.

An interesting case is that that Quasi-Newton algorithm and Steepest Descend are able to achieve very good results also on Griewanck function: this function is multimodal, but the oscillations have a very high frequency; the step size of these algorithms is able to jump the local minima.

Comparing the Random Search with the Random Walk, the results of this last algorithm are always better: this is the effect of the elitism that is present in it.

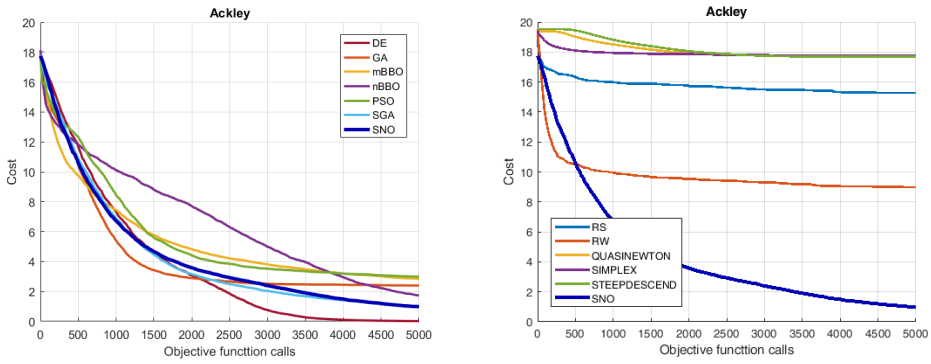


Figure 3.30: Comparison of the convergence curves on the Ackley function.

Figure 3.30 shows the comparison of the convergence curves of all the algorithms. The figure is divided in two parts: the left one is for EAs, while the right side for the other algorithms. For making clearer the comparison, the convergence curve of SNO is reported twice.

Comparing SNO and DE, it is possible to see that the initial convergence is similar but when the convergence of SNO slows down, the one of DE continues.

It is interesting the very fast convergence of the random walk: however, this convergence lead to a local minima and the algorithm is not able to reach good results.

Figure 3.31 shows the comparison on the Griewank function. The behaviour of DE is interesting because it reaches the final results of most the algorithm at the middle of the time and then it is able to make another convergence step. In this function the initial convergence of GA is very fast,

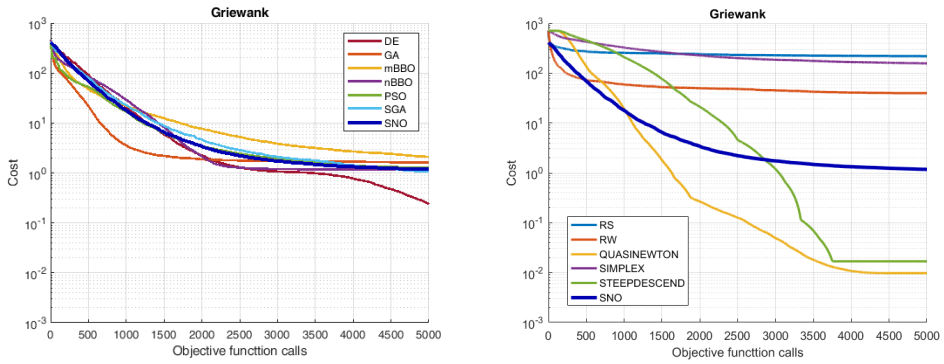


Figure 3.31: Comparison of the convergence curves on the Griewanck function.

reaching a very good result in 1/4 of the available time.

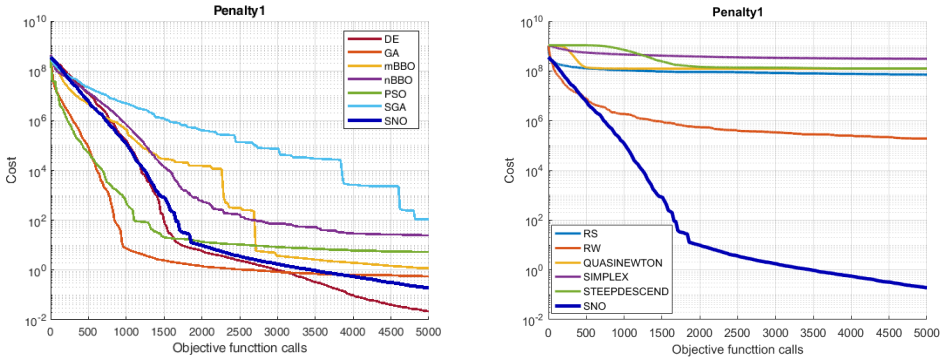


Figure 3.32: Comparison of the convergence curves on the Penalty1 function.

Figures 3.32 and 3.33 show the convergence curves on respectively Penalty 1 and Penalty 2 functions. In both the functions it is possible to see again the very fast convergence of GA; the point-based algorithms have performance that are comparable or lower than random search due to the multimodal shape of the function.

The Quartic function (Figure 3.34) is a single-model function, thus the derivative-based algorithms are very effective in the optimization. It is interesting to see that DE is better to outperform the derivative algorithms also in this function.

In Rastrigin function (Fig. 3.35) SNO is the best algorithm: the shape of the function with the very large local minima is able to drastically slow down the convergence of DE. In this function also the SGA achieve very good results.

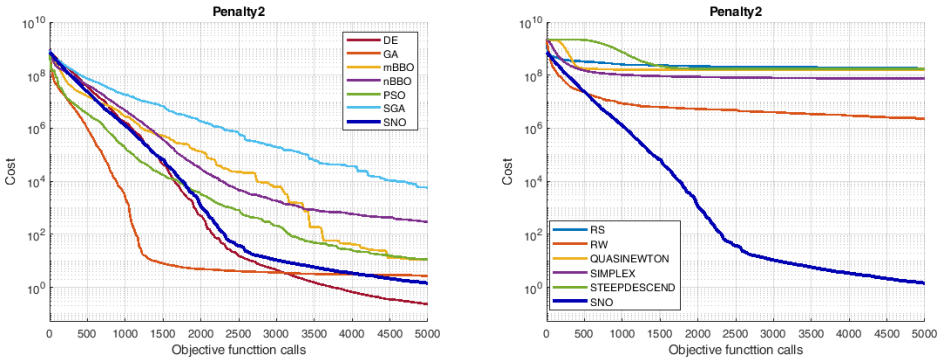


Figure 3.33: Comparison of the convergence curves on the Penalty2 function.

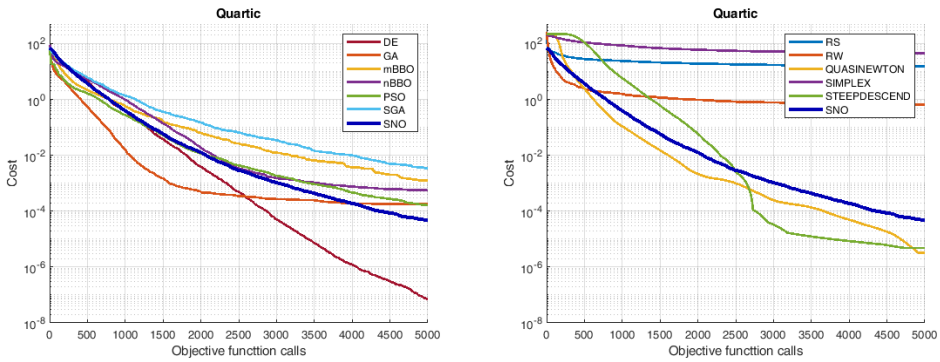


Figure 3.34: Comparison of the convergence curves on the Quartic function.

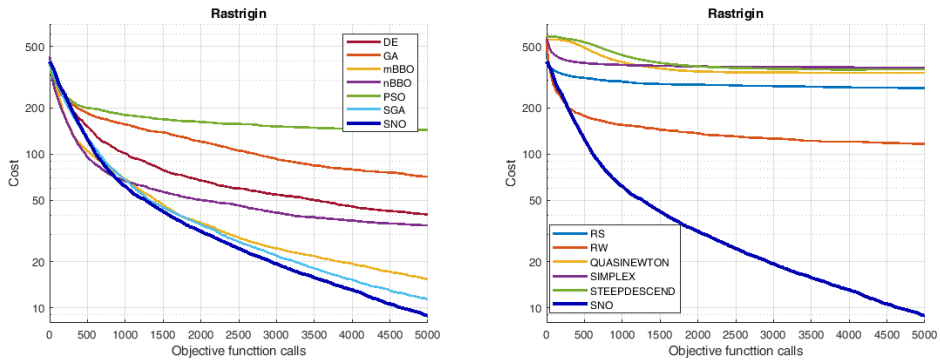


Figure 3.35: Comparison of the convergence curves on the Rastrigin function.

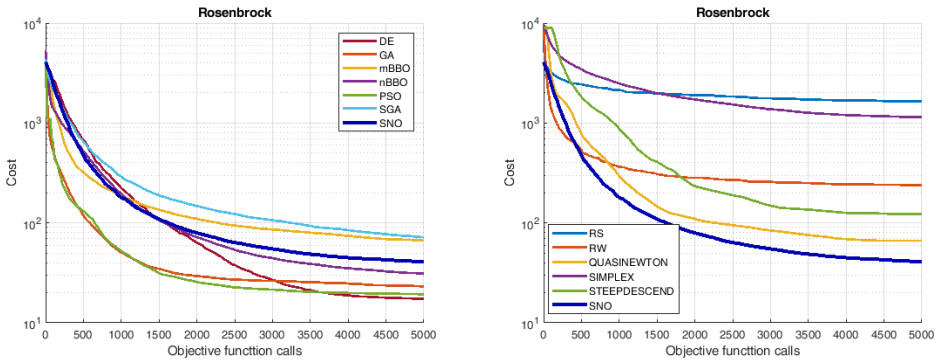


Figure 3.36: Comparison of the convergence curves on the Rosenbrock function.

The Rosenbrock function is a single-modal function, thus the derivative-based algorithms are able to perform well (Figure 3.36). It is interesting the very good behaviour on this function of both GA and PSO.

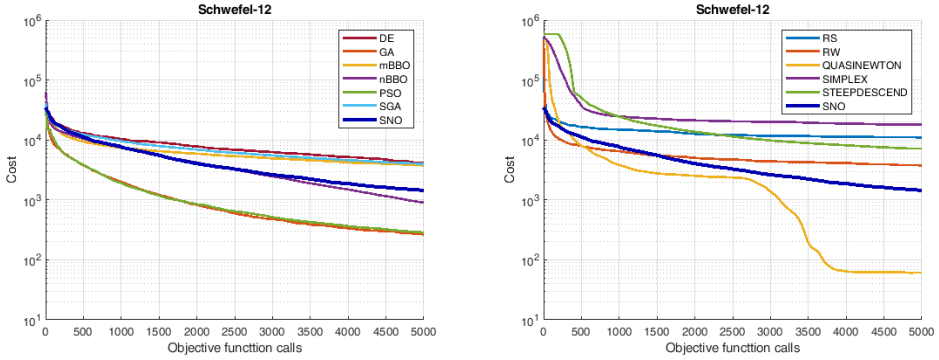


Figure 3.37: Comparison of the convergence curves on the Schwefel-12 function.

Figure 3.37 shows the convergence on Schwefel-12 function. Two interesting behaviours can be notices: firstly, the very good performance of GA and PSO that are able to find a very good minimum (it is a local on: the minimum value of the function is 0, while the best value reached by the two algorithms is higher than 250). Secondly, is peculiar the convergence of Quasi Newton algorithm that outperforms all the EAs (his final value is 61).

The results on the Schwefel-221 function (Figure 3.38) show a good behaviour of the SD algorithm that is able to outperform SNO and to have results very competitive with respect to all the EAs. Among them, the SGA shows a very slow convergence, while the standard GA achieve the best result.

Figure 3.39 shows the convergence on Schwefel 222 function. Here is

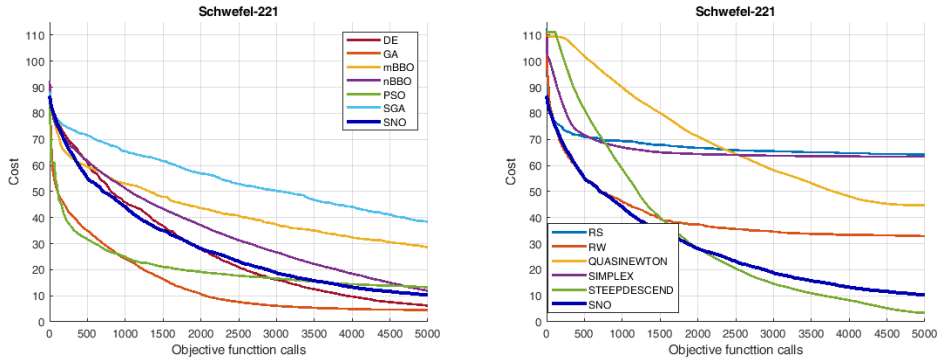


Figure 3.38: Comparison of the convergence curves on the Schwefel-221 function.

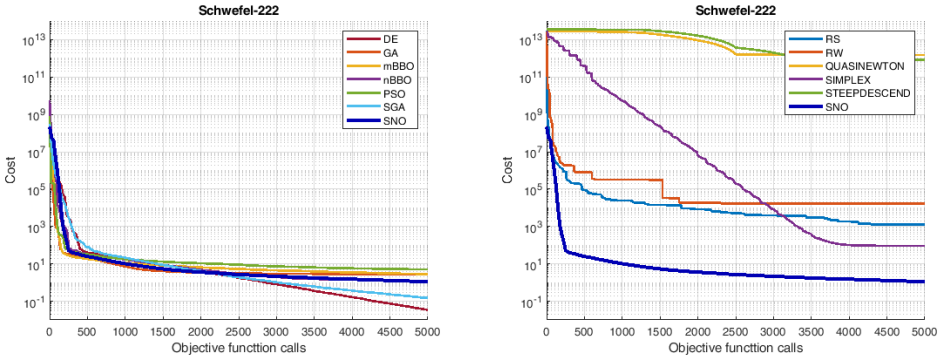


Figure 3.39: Comparison of the convergence curves on the Schwefel-222 function.

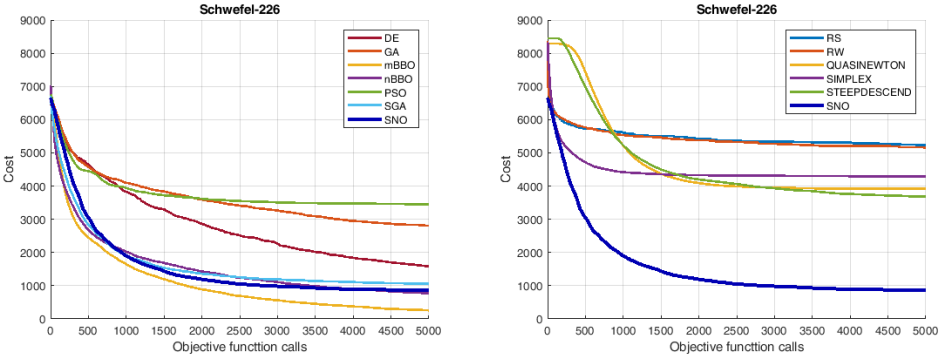


Figure 3.40: Comparison of the convergence curves on the Schwefel-226 function.

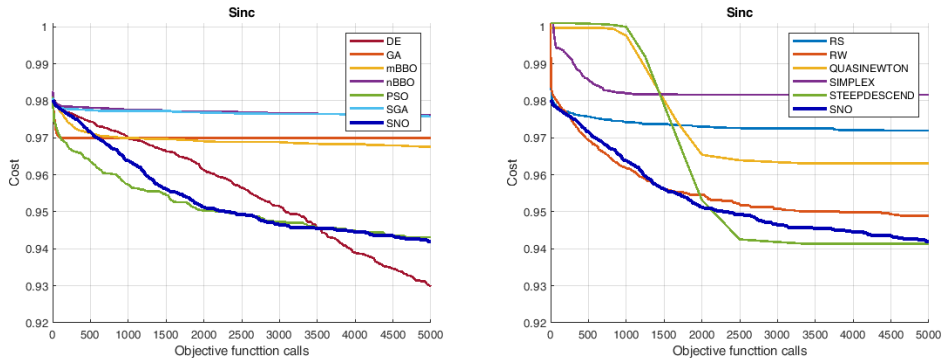


Figure 3.41: Comparison of the convergence curves on the Sinc function.

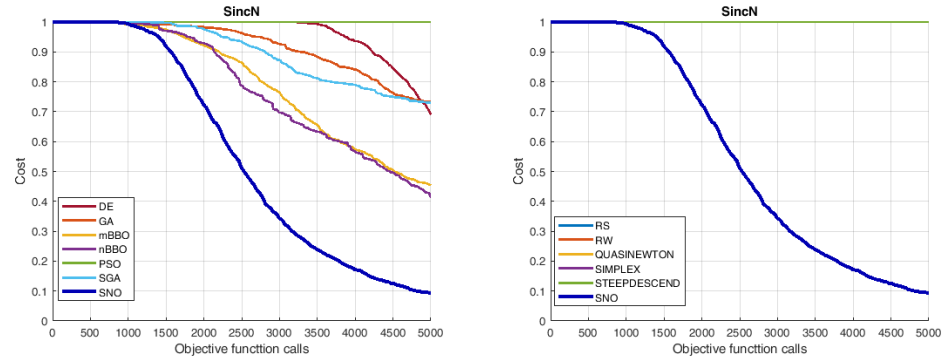


Figure 3.42: Comparison of the convergence curves on the Sinc-N function.

very interesting the behaviour of the random algorithms (both RS and RW are able to achieve good results, even if they are much worse than the ones of the EAs) and of the Simplex Algorithm, that is the most performative point-based algorithm.

In the Schwefel-226 function (Figure 3.40) SNO, SGA and nBBO shows a very similar behaviour, and the nBBO is slightly better than them.

The results of the Sinc function (Figure 3.41) are not significant at all because none of the algorithms is able to have a convergence toward the real minimum (0 cost value), but all of them are blocked in a very high local minimum.

In the Sinc-N function (Figure 3.42), all the EAs but PSO are able to start the convergence toward the global minimum. From other tests on this function, it has been noticed that, if an algorithm starts the convergence, it is able to reach the optimum point. The other algorithms are completely blocked in the small local minima.

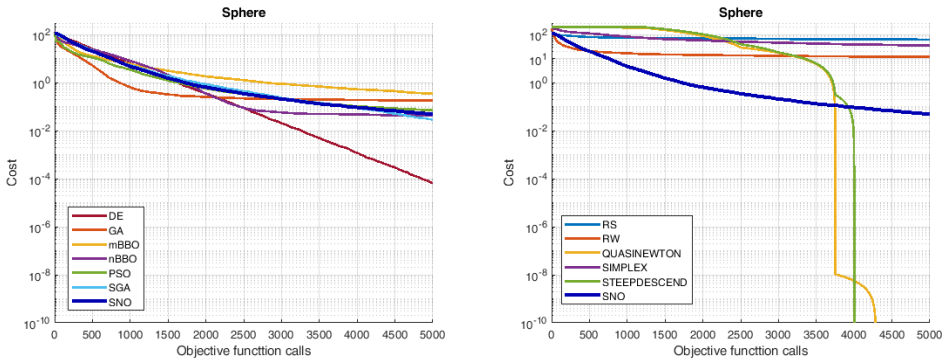


Figure 3.43: Comparison of the convergence curves on the Sphere function.

The Sphere function is a single-modal function, thus the derivative-based algorithms are able to reach the minimum (Figure 3.43). The final value they reach is comparable with the numerical error of the evaluation of the function.

The behaviour of DE is very interesting because its convergence rate is very high and very stable.

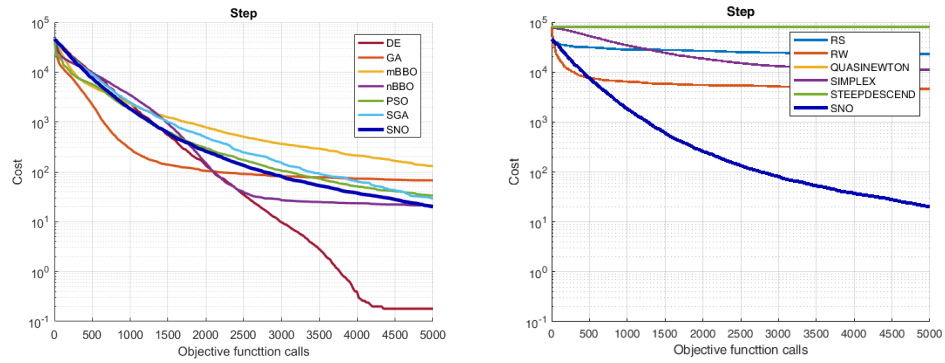


Figure 3.44: Comparison of the convergence curves on the Step function.

Finally, Figure 3.44 shows the results on the Step function. This function is characterized by a set of flat area, thus the derivative-based algorithms are not able to begin the convergence.

For what concern the EAs, it is interesting the convergence of the nBBO that at the beginning of the convergence period behaves like the DE.

The preliminary results presented in this chapter validate the performance of SNO and show its global reliability.

Chapter 4

Beam Scanning Reflectarray

Reflectarrays (RAs) are antennas structures originally aimed to improve directivity. They consist of a low profile planar array of printed radiating elements illuminated by a primary feed source [131].

Usually these antennas are characterized by a flat reflector: this solution reduces the production costs and the antenna volume (important aspect especially in aerospace applications). Moreover, with respect to parabolic reflectors, it is possible to have more customizable solutions, such as conformal reflectors [132].

The planar reflector consists of several patches with different geometrical parameters, that affects their reflection proprieties, such as the reflection phase shift and the attenuation in the module of the field. A proper selection of the geometrical parameters and electromagnetic response of all the patches can be used for obtaining the desired antenna performance [133]. The improved design methods for antenna and the improved computational capabilities enlarged the applicability of RAs.

The reflectarray design consists in two phases: the design and assessment of the single patch, and then the design of the entire reflector [133].

The introduction of new applications of RAs makes crucial the application of optimization algorithms: in fact, the traditional deterministic techniques (like the design of a pencil-beam antenna) are no more sufficient. Moreover, the design and optimization of a RA is an highly dimensional and multimodal problem, thus the use of Evolutionary Algorithms becomes very important [134].

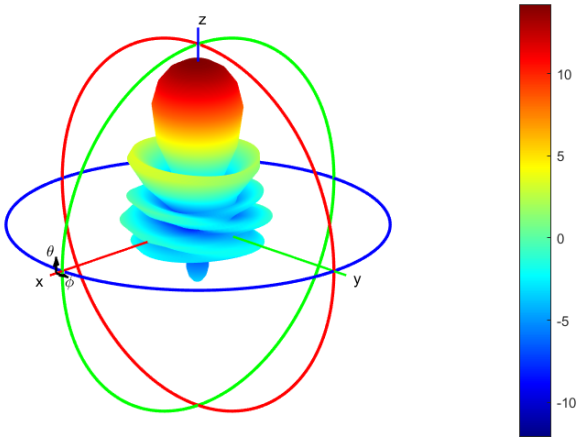


Figure 4.1: Example of radiation pattern of a helix antenna. The origin of the reference system is the position of the antenna, while the colours are representative of magnitude of the radiated field in dB.

4.1 Introduction to antenna

Antennas are system devoted to carry electrical signals by means of radiation. With respect to transmission lines, antennas do not require any guiding lines: in fact, the transmission of information is performed by means of free-space electromagnetic waves [25].

It is possible to divide antennas into four classes: electrically small antennas, resonant antennas, broadband antennas, and aperture antennas. This last type is characterized by a high gain, that increases with frequency, and moderate bandwidth. Reflectarrays belong to this last class of antennas [25].

4.1.1 Radiation pattern and antenna performance parameters

The *radiation pattern* of an antenna is the graphical representation of the radiation proprieties of the antenna itself, in particular the variation with respect to the observation angles θ and ϕ. Figure 4.1 shows a radiation pattern obtained with the antenna tool of Matlab for a helix antenna. The origin of the reference system is the position of the antenna, while the colours are representative of magnitude of the radiated field in decibels (dB).

The radiation pattern, thus, is representative of the electric field received in a sphere with a fixed radius r. Due to the fact that the radius is arbitrary, the plot is often normalized with respect to the maximum field value. The radiation pattern function $F(\theta, \phi)$ is a complex-valued function, and thus it

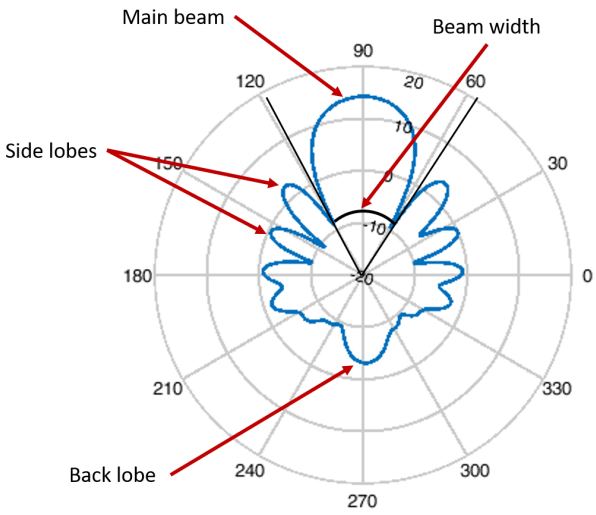


Figure 4.2: Proprieties of a radiation: main beam, side and back lobes and beamwidth.

is represented its module, very often in decibels (dB) [25].

There are some important radiation pattern proprieties. It is composed by several *lobes*: the *main beam* is the lobe containing the direction of maximum radiation. All the other lobes are *side lobes*, if they are oriented in a different direction with respect to the main beam, or *back lobes* if they are in the half-space opposite to the main beam [25].

A measure of how well the power is concentrated in the main beam is the *side lobe level* (SLL), *i.e.* the ratio between the pattern value of the maximum side lobe with respect to the pattern value in the main beam [25].

Another important feature of the radiation pattern is the beamwidth, *i.e.* angle between the first two nulls around the radiation pattern.

Figure 4.2 shows the main radiation pattern proprieties that are used in the following for characterizing the antenna performance.

4.1.2 Introduction to radiation analysis

The radiation pattern evaluation is a key aspect in aperture antenna design: it should be calculated with high accuracy for having a reliable forecasting of the actual radiation of the real antenna.

One of the most accurate technique is the solution via Finite Element Methods. This system, called also *full wave*, has the disadvantage that for

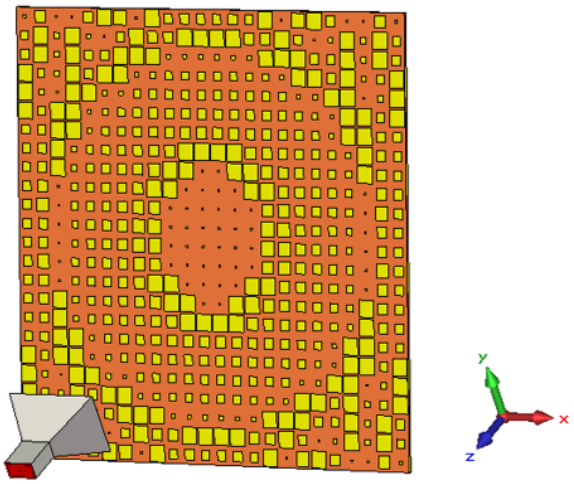


Figure 4.3: Schematic view on the antenna system: both the offset feeder and the reflector are depicted.

medium-large reflectarrays the calculation time becomes an issue: it can last for several hours, thus it is not feasible for iterative design processes [135].

Other techniques have been implemented for reducing the time required by the computation of the radiation pattern.

One of the simplest methods is the *array theory*, that can be used for calculating the radiation pattern, as used in [136] and in [137]. This method is very fast, even if the performance is not very accurate for reflectors in which the distribution of the patches is not very regular.

Another method that can be used is the aperture field method [135]: the calculation of the radiation pattern with this method is slightly slower, but the accuracy is higher. Moreover, in this method it is possible to take into account more information, like the different reflection behaviour of the patch as function of the incidence angle of the field.

The selected method, that will be shortly described in the following of this Section, is the aperture field. An analysis of the accuracy of it is provided in Section 4.4.3.

4.1.3 Aperture Field Method

Here the aperture field method is briefly described. For a more detailed analysis of this method it is possible to refer to [138].

The geometry considered is composed by a feeder and a planar reflector (see Figure 4.3). Each element of the planar reflector is identified by its position assuming a reference system located in the plane of the reflector

and centred with respect to the antenna:

$$\vec{r}_{mn} = \left\{ \begin{array}{c} x_{mn} \\ y_{mn} \\ 0 \end{array} \right\} \tag{4.1}$$

The distance between each patch and the centre of the array is:

$$r_{mn} = \sqrt{x_{mn}^2 + y_{mn}^2} \tag{4.2}$$

The feeder location is identified with spherical coordinates: in this way, the radial coordinate d_f is representative of the amount of energy radiated outside the reflector, and the tilting angles represent the position of the offset feeder that is oriented towards the centre of the array.

Due to the fact that generally the feeder is tilted only with the θ coordinate, its vector position is:

$$\vec{r}_f = \left\{ \begin{array}{c} -d_f \sin\theta_f \\ 0 \\ d_f \cos\theta_f \end{array} \right\} \tag{4.3}$$

Thus, the distance between the feed and each patch is:

$$r_{fmn} = \sqrt{x_{mn}^2 + y_{mn}^2 + z_f^2} \tag{4.4}$$

The two important elements that are used for the evaluation of the radiation pattern are the radiated field from the feeder and the reflection properties of each patch that is characterized by the two parameters q_E and q_H.

In order to evaluate the field received by the reflector, it is required to evaluate the angles ϕ_{F_mn} and $\theta_{F,mn}$ of each patch, seen from the feeder:

$$\phi_{F,mn} = \arccos\left(\frac{x_{mn} + d_f \sin\theta_f}{\sqrt{(x_{mn} + d_f \sin\theta_f)^2 + y_{mn}^2}} \right) \tag{4.5}$$

$$\theta_{F,mn} = \arccos\left(\frac{d_f^2 + |\vec{r}_{mn} - \vec{r}_f|^2 - r_{mn}^2}{2d_f \sqrt{(x_{mn} + d_f \sin\theta_f)^2 + y_{mn}^2}} \right) \tag{4.6}$$

The field received by each patch, expressed in spherical coordinates in the feeder reference system, is:

$$\mathbf{E}^F = \left(\begin{array}{c} E_\theta^F \\ E_\phi^F \end{array} \right) = \left(\begin{array}{l} j\dfrac{k_0}{2\pi r_{fmn}} e^{-jk_0 r_{fmn}} \cdot \cos^{q_{fe}}\theta_{F,mn} \cdot \cos\phi_{F,mn} \\ -j\dfrac{k_0}{2\pi r_{fmn}} e^{-jk_0 r_{fmn}} \cdot \cos^{q_{fh}}\theta_{F,mn} \sin\phi_{F,mn} \end{array} \right) \tag{4.7}$$

where q_{fe} and q_{fh} are the two parameters that characterize a feed horn.

Then, this field projected in Cartesian coordinates of the feeder reference system is:

$$\mathbf{E}^F = \begin{pmatrix} E_x^F \\ E_y^F \\ E_z^F \end{pmatrix} = \begin{pmatrix} \cos\theta_{mn}\cos\phi_{mn}E_\theta^F - \sin\phi_{mn}E_\phi^F \\ \cos\theta_{mn}\sin\phi_{mn}E_\theta^F + \cos\phi_{mn}E_\phi^F \\ -\sin\theta_{mn}E_\theta^F \end{pmatrix} \tag{4.8}$$

Finally, it is possible to calculate the field in the reference system of the reflector:

$$\mathbf{E}_{mn}^R = \begin{pmatrix} E_{mn,x}^R \\ E_{mn,y}^R \end{pmatrix} = \begin{pmatrix} \cos\theta_f E_x^F - \sin\theta_f E_z^F \\ -E_y^F \end{pmatrix} \tag{4.9}$$

The field calculated is the one that each patch receives as input from the feeder. Then it is possible to calculate the reflected field. For doing it, the geometrical characteristics of the patch (design variable of the antenna design problem) should be considered, since they affect the reflection coefficient.

For a given patch length L_{mn}, it is possible to calculate the reflection proprieties (amplitude S_{MN} and phase ϕ_{mn}). Thus, the reflected field from each patch is:

$$\mathbf{a}_{mn} = \begin{pmatrix} E_{mn,x}^R \cdot S_{mn}e^{j\phi_{mn}} \\ E_{mn,y}^R \cdot S_{mn}e^{j\phi_{mn}} \end{pmatrix} \tag{4.10}$$

The combination of the radiated fields of all the patches is:

$$\mathbf{E}^R(\theta,\phi) = \begin{pmatrix} E_x^R \\ E_y^R \end{pmatrix} = \begin{pmatrix} \sum_{m=1}^{N_x}\sum_{n=1}^{N_y} a_{mn,x}\cdot e^{jk_0(u\cdot x_{nm}+v\cdot y_{nm})} \\ \sum_{m=1}^{N_x}\sum_{n=1}^{N_y} a_{mn,y}\cdot e^{jk_0(u\cdot x_{nm}+v\cdot y_{nm})} \end{pmatrix} \tag{4.11}$$

Finally, this field is rotated in the θ,ϕ reference system:

$$\mathbf{E}(\theta,\phi) = \begin{pmatrix} \frac{-jk_0r_{ff}}{2\pi r_{ff}}(E_x^R\cos\phi + E_y^R\sin\phi) \\ \frac{-jk_0r_{ff}}{2\pi r_{ff}}(-E_x^R\cos\theta\sin\phi + E_y^R\cos\theta\cos\phi) \end{pmatrix} \tag{4.12}$$

The radiation pattern of an antenna is the module of the radiated field and generally it is expressed in decibels.

4.2 Beam Scanning Reflectarray

An important aspect that has been recently investigated in literature is the possibility of having a scanning capability (the possibility to direct the main bean in different angles) with reflectarrays.

This capability is often achieved with mechanical systems, but it is generally slower and more expensive with respect to system like active planar arrays.

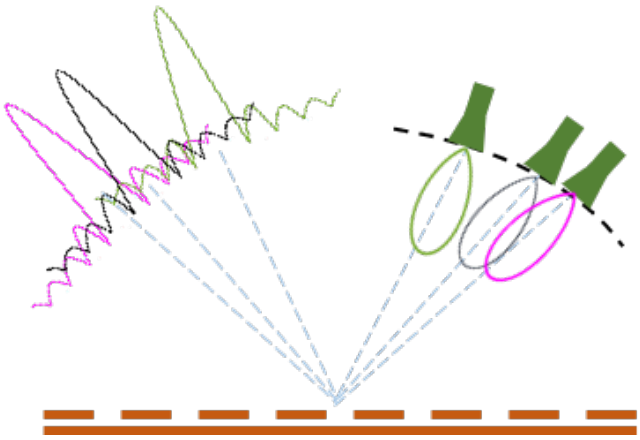


Figure 4.4: Side view of a scanning beam reflectarray: the feed moves along a circular arc, and changing its position also the direction of maximum radiation varies.

Some tests have been performed in literature for creating an electrical scanning changing the reflection proprieties of the patches with a bias electrical circuit [139]. This type of design is very flexible because the variation of the reflection phase of the patches can be adjusted changing the biasing voltage, but it requires quite complex feeding circuit, especially in large reflectors.

For simplifying the biasing circuit, some authors tested PIN diode patches that can have only two reflection states [140]. In this case the feeding circuit is easier, but the reflection performance that can be achieved are lower.

Another method for beam scanning is steering the feeder and having a fixed reflector designed for having good reflection proprieties with different scan angles. This system, shown in Figure 4.4, is much simpler because it does not require a complex biasing system for the reflector, but the radiation pattern worsen quickly when the scan angle increase.

There are no optimal deterministic solutions for this problem, thus it has been here solved with Evolutionary Optimizer.

4.2.1 Antenna description

The antenna system is composed by a feed (a horn antenna) and a planar reflector. The surface of the RA is divided in a proper number of square unit cell, with size lower than or equal to $\lambda_0/2$, where λ_0 is the wavelength computed at the design frequency f_0.

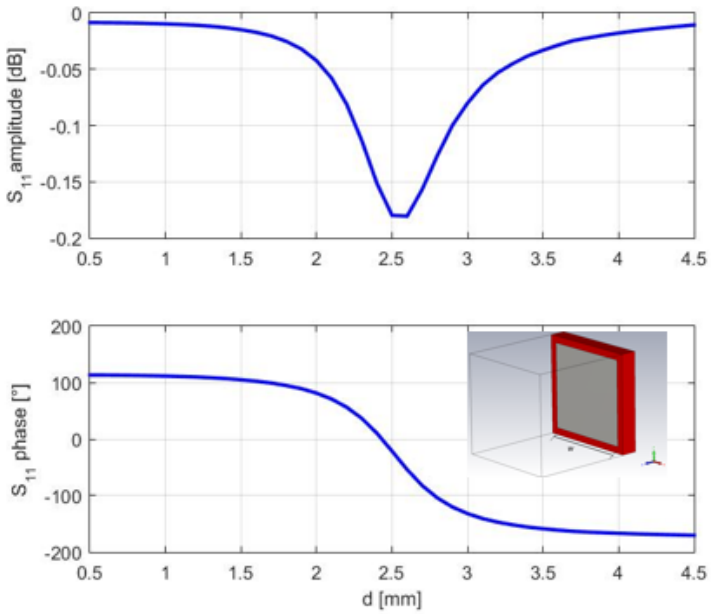


Figure 4.5: Variation of the phase (top) and amplitude (bottom) of the reflection coefficient provide by each unit cell, as a function of the size d of the square patch. Inset: sketch of the unit cell.

Here, the considered reflector is made of a dielectric substrate with relative dielectric constant $\mu_r = 2.55$ and height $h = 0.8mm$. The feed is a standard horn, located at a distance $F = 10.8\lambda_0$ from the centre of the reflectarray, and it can move along an arc, covering the angular range that corresponds to have a beam scanning between -40°and +40°.

Each unit cell includes a square patch (see inset in Figure 4.5, whose size d is used to control the phase and the amplitude of the reflection coefficient S_{11} provided by the cell itself. Their variation with d is plotted in Figure 4.5. S_{11} is computed with the well-known full-wave commercial simulator CST Microwave Studio, carrying on a full-wave simulation of the unit-cell embedded in a periodic structure and for normal incidence.

The feed has a radiation pattern that can be approximated with a cosine one with $q_e = q_h = 7.7$:

$$f(\theta, \phi) = \cos^{q_e}(\theta) \cos?q_h(\phi) \tag{4.13}$$

The analysed antenna is composed by 24 × 24 patches, and thus the reflector size is $12\lambda_0$.

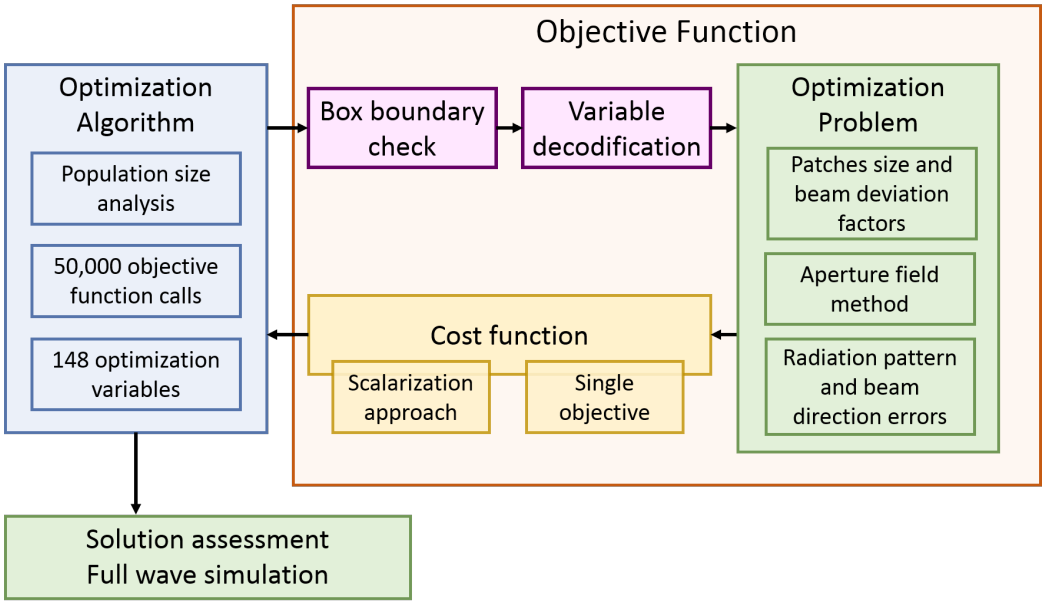


Figure 4.6: Optimization scheme for the reflectarray: for constraints on the computational time the full wave simulation cannot be inserted in the optimization loop.

4.3 Optimization procedure

The optimization procedure is designed starting from the scheme presented in the Optimization System chapter, and it has been adapted for facing an important constraint of this specific problem: the full wave analysis (the accurate simulator of antennas) requires many hours for each simulation, thus it is not possible to use it inside the optimization loop. Moreover, the optimization process is characterized by 148 design variables and it is almost impossible to train an effective surrogate model.

Due to this reason, the calculation of the radiation pattern during the optimization process is done by means of the aperture field method. This has been tested with several cases presented in different international conferences and the results obtained were quite reliable [67, 73].

Figure 4.6 shows the optimization scheme. The algorithm, during its run, exploits the radiation pattern calculated by the aperture field method, that is then used to calculate the cost value. Only at the end of the optimization process the full wave simulation is run and the obtained results are validated.

The optimization loop should be tested in order to properly select the algorithm parameters and the objective function. Moreover, the aperture field method should be assessed, and its performance compared with the full

wave simulator.

4.4 Designing the process

The first element of the optimization process that should be designed is the free variables.

The antenna behaviour is characterized by the length of the patches and by the so-called *beam deviation factor*. The symmetries of the system are completely exploited for reducing the number of design variables.

Each of these design variables is obtained from a corresponding optimization variable with a proper scaling. The algorithm has a very high flexibility in creating new patterns with the patches.

Here, the number of design variables is 148. In fact, the antenna has 576 patches, but the application of the symmetries reduces this number to 144. In addition to these variables, for each scan angle the beam deviation factor is introduced, *i.e.* the angular difference between the feeder angle from the vertical line and the main beam direction [141].

The calculation of the radiation pattern takes 1.73s on Intel Core i7 for the 24×24 antenna, that corresponds to 23.8h for the entire optimization.

For a proper design of the optimization process, several tests have been done: firstly, a basic analysis on the cost function is done for taking into account properly all the performance parameters of the system. Secondly, an analysis on the feasibility function is proposed for understanding its impact on the optimization procedure. Thirdly, the aperture field method is assessed: in particular, two different implementations of the same method are compared with the full wave analysis. Finally, an analysis on SNO parameters has been performed.

4.4.1 Cost function definition

The first analysis that has been done regards the cost definition. In fact, the problem is intrinsically multi-objective and it is faced by means of the scalarization method.

For each of the scanning angles, the antenna requirements are the maximum main beam half width, the maximum SLL as function of the angle θ, and the direction of the main beam.

The first two requirements can be imposed with a mask, *i.e.* the maximum level that the optimal radiation pattern can have for every θ and ϕ.

From the mask definition and the third requirement, several cost functions can be identified:

- The integral of the radiation pattern exceeding the mask:

$$c_1(\mathbf{d}) = \iint err(\theta, \phi) d\theta d\phi \tag{4.14}$$

where err is the error, defined by means of the Heavyside function H:

$$err(\theta, \phi) = 10^3 \cdot [E(\theta, \phi) - M(\theta, \phi)] \cdot H\left[E(\theta, \phi) - M(\theta, \phi)\right] \tag{4.15}$$

- Maximum error between the mask and the radiation pattern:

$$c_2(\mathbf{d}) = \max_{\theta,\phi}(err(\theta, \phi)) \tag{4.16}$$

- The scan angle error:

$$c_3(\mathbf{d}) = \Delta\theta_s = \left(\frac{\theta_s - \theta_{s,a}}{\pi} \cdot 180\right)^2 \tag{4.17}$$

where θ_s is the desired scan angle and $\theta_{s,a}$ is the actual one. This cost function is very focused and it can be used for improving the others.

The costs c_1 and c_3 can be scaled for making the results more readable. The scaling procedure does not affect the optimization procedure.

The integral error between the mask and the radiation patter is a very common cost value, but it cannot sense precisely the scan angle error and detect clearly the presence of narrow high peaks in the radiation pattern. Due to this reason, it is convenient to combine this cost value with the other two cost functions, for having a numerical value of the performance that is more coherent with the desired system behaviour. These three cost definitions can be combined with the scalarization method usually implemented for multi-objective problem [142].

This application can be considered multiobjective at two levels: in fact, a first level can be identified for each scanning angle. In this level the three performance values previously identified can be combined in a single cost value that represents the effectiveness at that precise scan angle.

The second level is the compression of the four data of the scan angles into the final cost that should be returned to the optimizer. This second level is also important because the type of coordinates used in the radiation pattern evaluation makes the optimizer more sensible to errors concentrated in the central region of the radiation pattern. This results in the fact that the optimization of low scan angles is usually faster than the optimization of larger scan angles.

Case number	λ_1	λ_2	λ_3	λ_4
1	1	1	1	1
2	1	1.25	1.25	1.5
3	1	1.25	1.5	2
4	1	1.25	2	4
5	1	1.25	2	10
6	1	1.25	2	15

Table 4.1: Value of the Lagrange multipliers for the six scalarization tests performed.

At the first level of scalarization, only the scan angle error and the radiation pattern integral error have been used: in fact, from early tests, it has been shown that the maximum error can worsen the optimization because it can give misleading values when the error is very close to the main beam.

Thus, the single beam cost can be calculated with the following equation:

$$c_s(\mathbf{d}) = c_1(\mathbf{d}) + \lambda c_3(\mathbf{d}) \tag{4.18}$$

where λ is the lagrangian multiplier.

Seven different values of the lagrange multiplier λ have been tested and the results are shown in Figure 4.7. The Figure shows the average value of 8 independent trials (central dot) and the bars represent the standard deviation. The red curve is the angular error, while the blue one is the radiation pattern error.

It is possible to see that the best results are obtained for the scan angle $\theta_s = 20°$. This is due to the fact that the central scan angles are easier to te obtained.

From Figure 4.7 it is possible to select the best value that minimizes the error and the standard deviation. The selected value is 10^2.

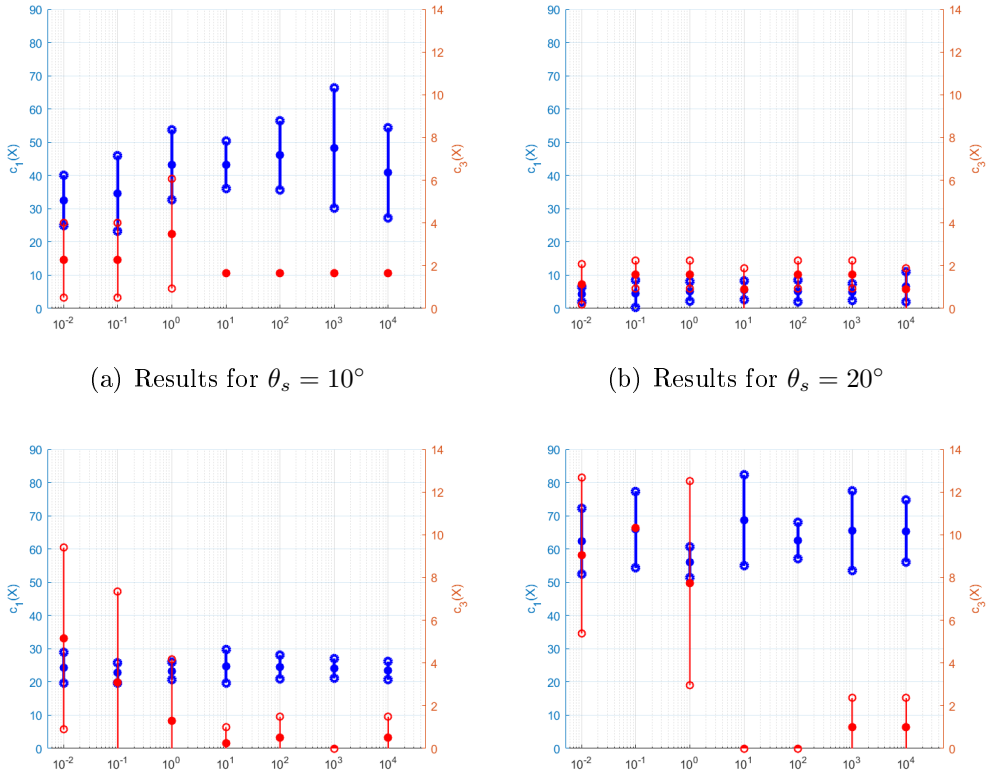
A second analysis has been performed on the scalarization from the four costs of the single scan angle to the final optimization cost.

The final cost is:

$$C = \lambda_1 c_{10} + \lambda_2 c_{20} + \lambda_3 c_{30} + \lambda_4 c_{40} \tag{4.19}$$

where λ_i are four Lagrange multiplier and c_{10}, c_{20}, c_{30} and c_{40} are the cost of the four scan angles.

Six tests have been performed with the values of the Lagrange multipliers summarized in Table 4.1. The importance of the higher scan angles has been increased their directivity of them is usually lower.



(a) Results for $\theta_s = 10°$

(b) Results for $\theta_s = 20°$

(c) Results for $\theta_s = 30°$

(d) Results for $\theta_s = 40°$

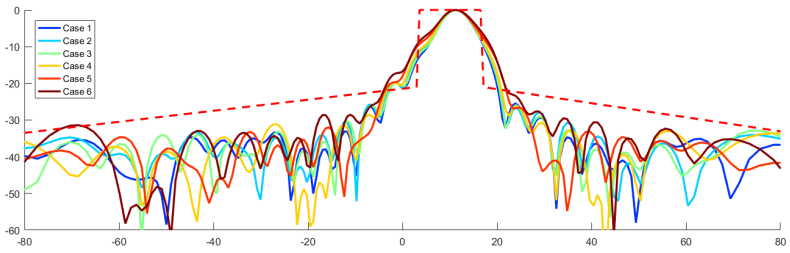
Figure 4.7: Analysis of scalarization factor λ for the four scan angles.

Figure 4.8 shows the radiation pattern of the four scan angles in the six cases. It is possible to make a graphical analysis of the radiation patterns for understanding which set of λs is the best one.

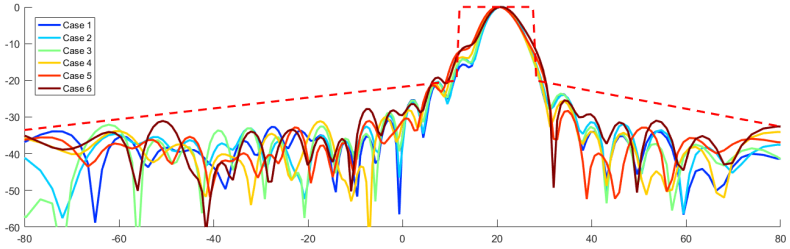
Analysing the results for $\theta_s = 40°$, it is possible to notice that the first four cases have a very big main beam. An opposite trend can be notice for $\theta_s = 20°$.

The sixth case is very detrimental for the low scan angles, much less effective than the case # 5 that, especially for $\theta_s = 10°$ has performance comparable to the other cases.

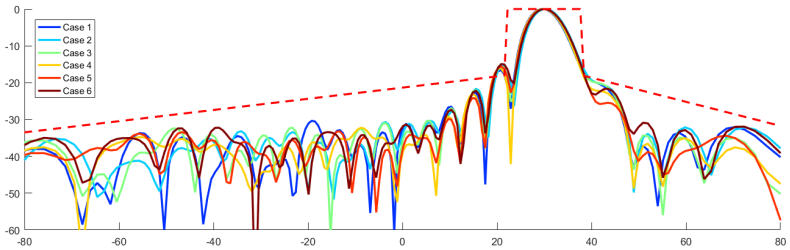
Due to the fact that in this application it is required to have similar performance for all the scan angles, the set of Lagrange multipliers of the 5-th case can be considered the best choice.



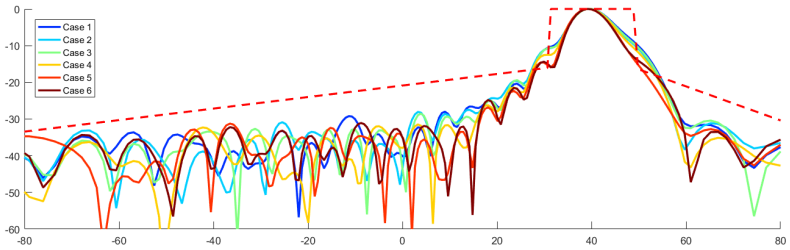
(a) Results for $\theta_s = 10°$



(b) Results for $\theta_s = 20°$



(c) Results for $\theta_s = 30°$



(d) Results for $\theta_s = 40°$

Figure 4.8: Radiation patterns for the four scan angles in the six tested cases.

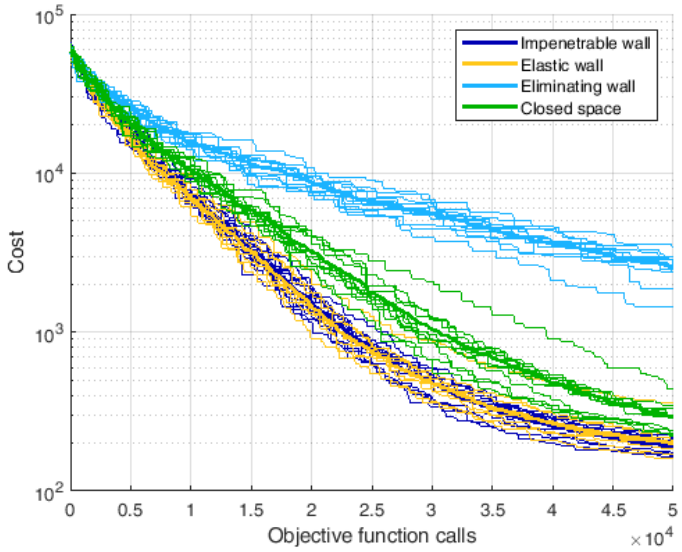


Figure 4.9: Comparison among the different convergence curves obtained with the four box boundary conditions. The curves are in semilogarithmic scale.

4.4.2 Box boundary conditions

The box boundary condition has been analysed because it can change the convergence proprieties of the algorithm. In particular, in the antenna problem they can highly influence the convergence due to the fact that a physical propriety of the patches, the reflection angle, is characterized by a periodic behaviour.

In particular, the four different functions seen in Chapter 2 (impenetrable wall, elastic wall, eliminating wall and close search space) have been here tested on the antenna problem. Several independent trials have been done with 50,000 objective function calls. Figure 4.9 shows the convergence curves with the four tested feasibility functions.

Here, it is possible to notice that the eliminating wall has a much worst convergence since the early iterations: this type con condition highly affects the final solution. In fact, the best solution obtained with this condition have no patches with length equal to the minimum or the maximum: in this case, the feasibility condition is reducing the number of solutions that can be easily reached by the optimizer. Moreover, this problem requires a good amount of exploitation, especially in the central part of the optimization time.

The closed search space has an intermediate behaviour, but the convergence is slower than the two best conditions, even if is the domain that correspond more to the physical behaviour of the variables.

Box condition	Mean	Standard deviation	Best result
Impenetrable wall	**191.41**	**19.75**	164.14
Elastic wall	201.08	52.76	**159.1**
Eliminating wall	2566.66	527.95	1452.67
Closed space	291.09	62.12	219.96

Table 4.2: Comparison of the results with the four feasibility function tested. The results are obtained with 12 independent trials and with 50,000 objective function calls.

This behaviour can be understood analysing the *idea-diffusion* operator: in fact, the continuous oscillation of the individuals from one side to the other of the domain creates an alternate attraction on the other individuals resulting in a slower convergence.

The other two conditions are characterized by a very similar convergence. Similar considerations can be drawn analysing the numerical results that are shown in Table 4.2.

4.4.3 Assessing the model

Another important test that should be conducted is the assessment of the reliability of the aperture field method with respect to the full wave simulation.

For doing so, an antenna geometry has been selected and it has been simulated with three methods.

In the first simulation, the standard aperture field method has been used, without considering the incidence angle. This is the method used in the previous tests. As said before, it is simple, but it has shown a good accuracy with respect to the full wave also in other optimization problems in literature.

The second method implements the aperture field method too, but in this case the incidence angle has been considered. In this case, the reflection curves of the patch are function of two variables: the size of the element and the angle between the patch and the feed.

Finally, the third method is the full wave analysis with CST Studio. This is the more accurate analysis because it takes into account the blocking effect of the feed and all the coupling effects due to the different size of adjacent patches.

Figures 4.10 and 4.11 shows the behaviour of the patch as function of the length and of the incidence angle θ.

Analysing the angles, a second resonance (that can be recognized by a drop in the module and a jump in the phase) appears for high incidence

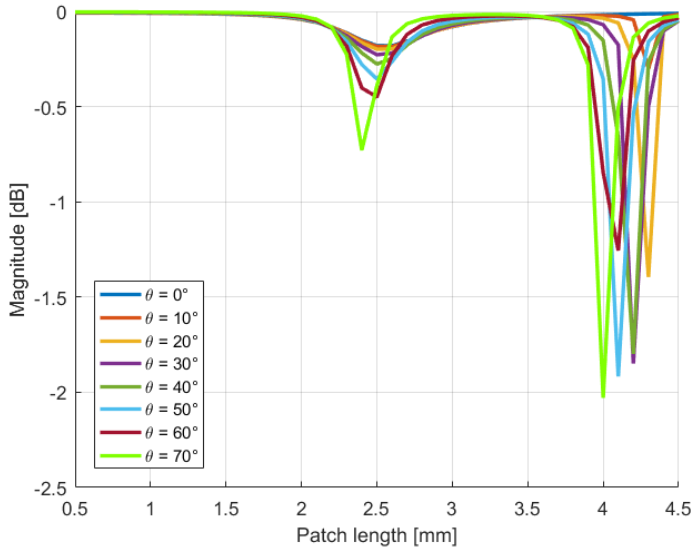


Figure 4.10: Patch characterization of the reflection module as function of the incidence angle θ (in dB).

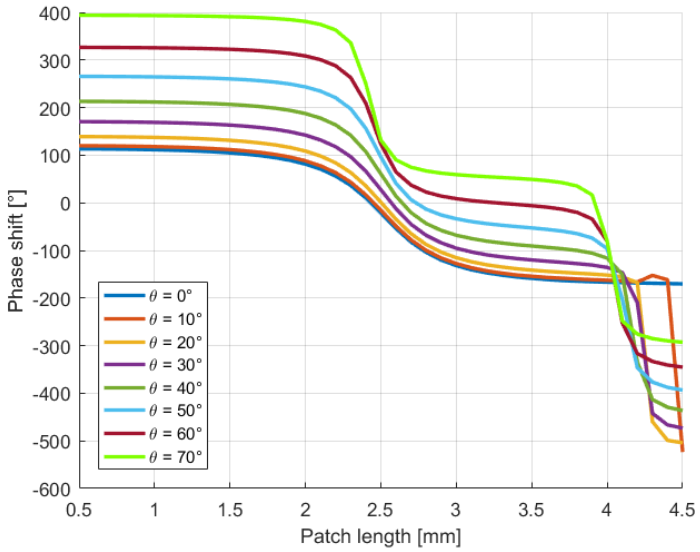


Figure 4.11: Patch characterization of the reflection phase as function of the incidence angle θ (in degrees).

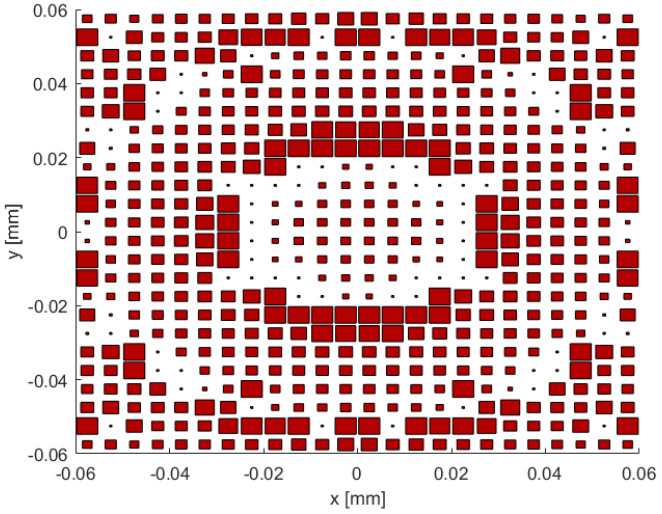


Figure 4.12: Geometry used for analysing the differences between the models.

angles when the patch becomes larger.

The finally adopted geometry is shown in Figure 4.12. The patch size in the figure is proportional to the real patch size.

The first comparison has been done analysing the different reflection proprieties of each patch. It is important to consider that, while for the single-entry model the reflection characteristics of the patches are the same for all the scan angle, with the double-entries model they vary because the incidence angle is different.

Figures 4.13 and 4.14 shows the difference of reflection losses and phase between the two models. The differences of the losses (Fig. 4.13) are quite low and concentrated in few patches, while the differences in phase (Fig. 4.14) are much more consistent. As it was expected, the greater differences are concentrated on the patches with higher size, especially the ones on the boundaries.

Figure 4.15 shows the main cuts of the radiation patterns on the E plane: the green one has been calculated with the FEM model, *i.e.* it can be considered the most accurate. The blue line is the radiation pattern of the double-entries model and the red one the single-entry model.

There are significative differences between the two simplified models, and they present bot an error with respect to the FEM result.

Generally speaking, the radiation pattern of the double-entries model is less reliable with respect to the full wave simulation, while the single-entry model is generally more accurate, showing that the assumption done in that

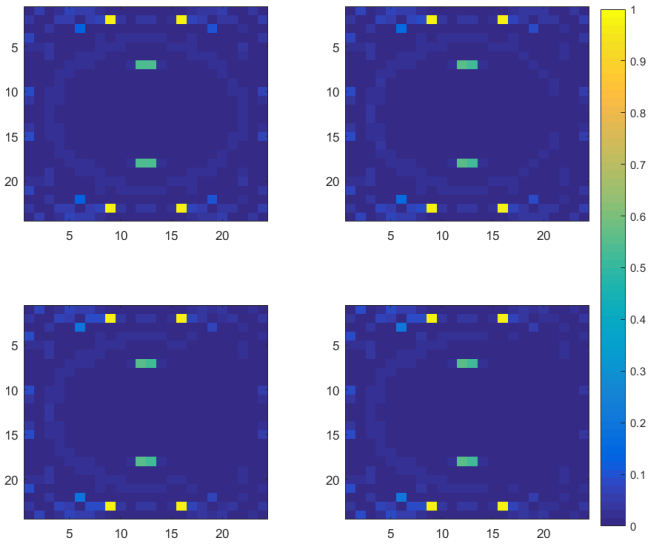


Figure 4.13: Differences of the reflection losses of each patch among the two simulation models.

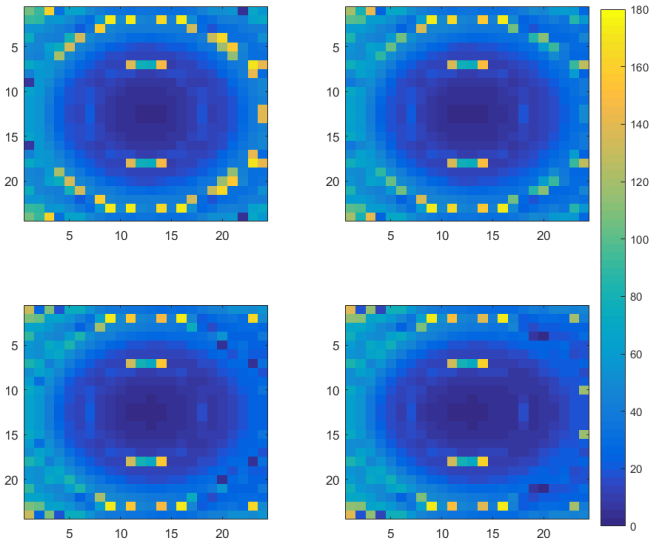


Figure 4.14: Differences of the reflection phase of each patch among the two simulation models.

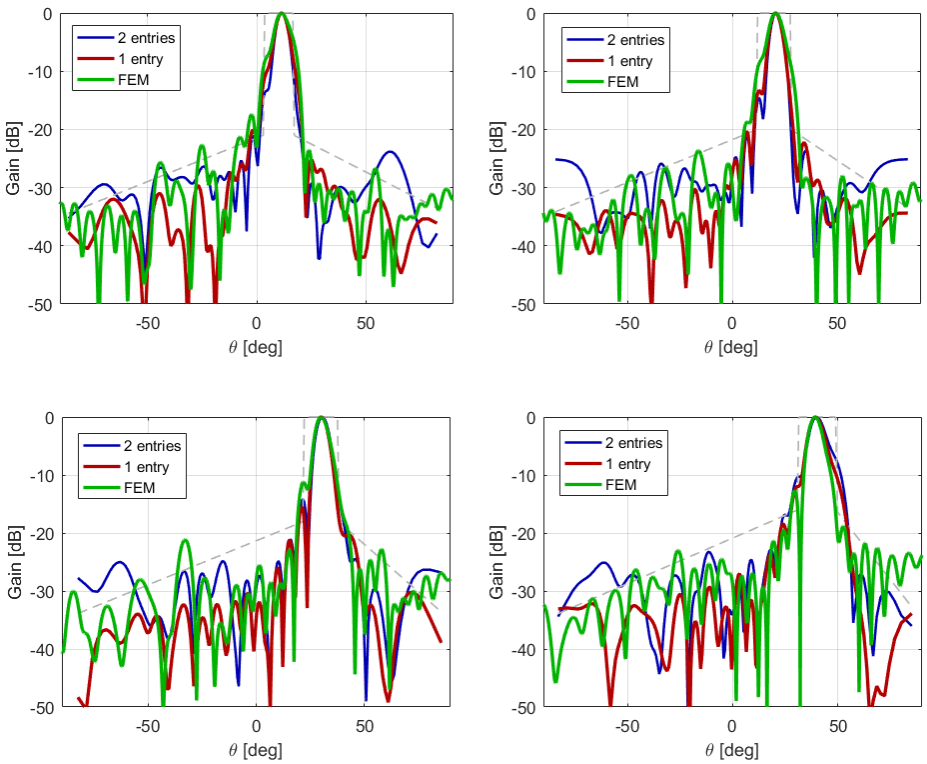


Figure 4.15: Comparison between the radiation patterns obtained with the three models.

model (plane wave) is good.

4.4.4 Algorithm parameters selection

Finally, the last aspect that should be considered in the design of the optimization process, is the proper selection of the algorithm parameters that are able to guarantee a good optimization convergence.

The performed test are devoted to identify the best set of algorithm parameters, analysing in particular the number of iterations (that is directly connected through the objective function calls with the population size), two combinations of attraction parameters (ω_0 and h) and three combinations of linguistic transposition parameters (error rate and error amplitude). Table 4.3 shows the test performed and the combination of parameters for each test. The number of objective function calls have been set to 50,000 in all the proposed tests.

Test	Iterations	Attraction parameters	Linguistic parameters
T1	200	0.8 & 0.3	0.015 & 0.1
T2	500	0.8 & 0.3	0.015 & 0.1
T3	1000	0.8 & 0.3	0.015 & 0.1
T4	2000	0.8 & 0.3	0.015 & 0.1
T5	200	1 & 0.5	0.05 & 0.1
T6	500	1 & 0.5	0.05 & 0.1
T7	1000	1 & 0.5	0.05 & 0.1
T8	2000	1 & 0.5	0.05 & 0.1
T9	200	1 & 0.5	0.04 & 0.4
T10	500	1 & 0.5	0.04 & 0.4
T11	1000	1 & 0.5	0.04 & 0.4
T12	2000	1 & 0.5	0.04 & 0.4

Table 4.3: Test performed for identifying the best algorithm parameters.

The combination of parameters has been determined on some preliminary tests on Schwefel-224 function, that has shown the optimization behaviour more similar to the specific antenna problem.

Firstly, the set of parameters of T1, T2, T3, and T4 have been used in the optimization. In all of them, the main operator parameters are the same, but the population size is changed. Table 4.4 shows the results of these tests, and Figure 4.16 shows the convergence curves.

From Table 4.4, it is possible to notice that a very critical aspect in these optimization is the generally high standard deviation of the results, which may be due to preliminary stagnation of the algorithm in local minima. The result with 500 iterations is the best case for both the mean results and the standard deviation, while with 1000 iterations the best results is optimal.

Test	Mean	Standard deviation	Best result
T1	565.24	186.86	317.64
T2	**200.54**	**29.67**	154.14
T3	226.83	126.59	**131.01**
T4	812.93	405.16	239.14

Table 4.4: Results of the tests on SNO: average final value, standard deviation, best results - Tests T1, T2, T3, and T4.

An analysis of these results can be done observing the convergence curves of Figure 4.16, remembering that the graph is plotted in semi-logarithmic scale. As it is expected, the number of iterations drives the initial convergence rate of the algorithm.

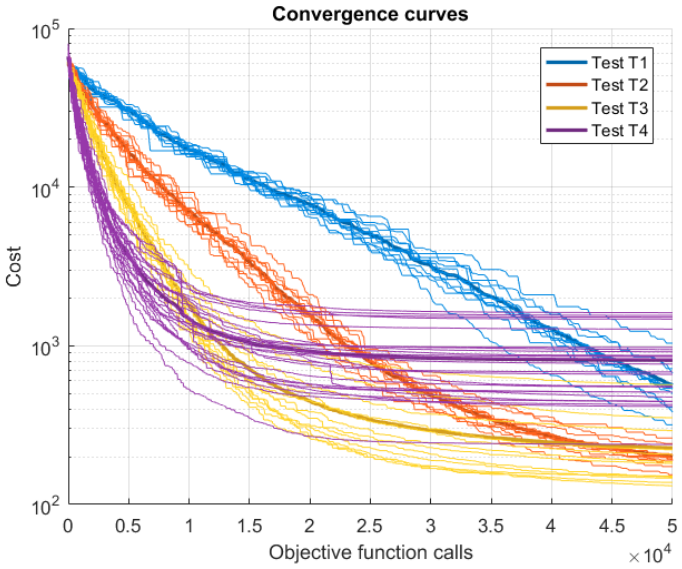


Figure 4.16: Results of the tests on SNO: convergence curves - Tests T1, T2, T3, and T4. The four colours represent the four tests, the thinner lines the convergence of each trial and the thick lines the average convergence.

The test T1 shows a convergence rate that is still high at the end of the optimization process, demonstrating that the algorithm has not reached the minimum value. On the other hand, in the test T4, with high number of iteration and so low population size, all the solutions stop the convergence around the half of the optimization time: this is probably due to a combination of low exploration and low information in the initial population.

In the other two tests (T2 and T3) the average value is approximately the same, but the test T3 shows a faster initial convergence that leads for

many solutions to an early convergence in local minima. Instead, the test T2 has a lower initial convergence rate but all the solutions show the same convergence profile. At the end of the optimization the convergence is almost reached.

Tests T5-T8 are characterized by a new set of attraction parameters that, in the early tests, shows a good behaviour in terms of capability of the particle to jump from a minimum to another. The linguistic error rate has been slightly increased because these jumps reduce the exploration of completely new regions of the solution space.

Table 4.5 shows the results of these tests, and Figure 4.17 shows the convergence curves. Analysing the numerical results, it is possible to notice that the standard deviation is here much more correlated with the population size. As seen also before, the intermediate population sizes gives the best results.

Test	Mean	Standard deviation	Best result
T5	325.51	**22.01**	293.28
T6	**235.89**	32.19	190.63
T7	297.98	172.52	**169.14**
T8	1042.78	746.17	295.47

Table 4.5: Results of the tests on SNO: average final value, standard deviation, best results - Tests T5, T6, T7, and T8.

Figure 4.17 shows that the convergence rate has been increased also with less iteration and the test T5 shows a generally very good behaviour. On the other hand, the behaviour with 2000 iterations (test T8) shows a huge dispersion of the results.

The tests T6 and T7 have a very similar behaviour. The much higher standard deviation of test T7 is due to couple of solutions that reaches a local minimum in the early stages of the optimization.

The tests T9-T12 has been done increasing the linguistic error in order to improve the exploration capabilities of SNO with the new set of parameters. Table 4.6 shows the results of these tests, and Figure 4.18 shows the convergence curves.

As can be noticed in Table 4.6, the convergence of the trial with 1000 iterations (T11) is significantly worse than the others, while with the other combination of parameters the standard deviation is very low. In general, with higher mutation rate the convergence is worsened and the final cost value is higher.

Figure 4.18 shows the convergence of this last four tests: the convergence

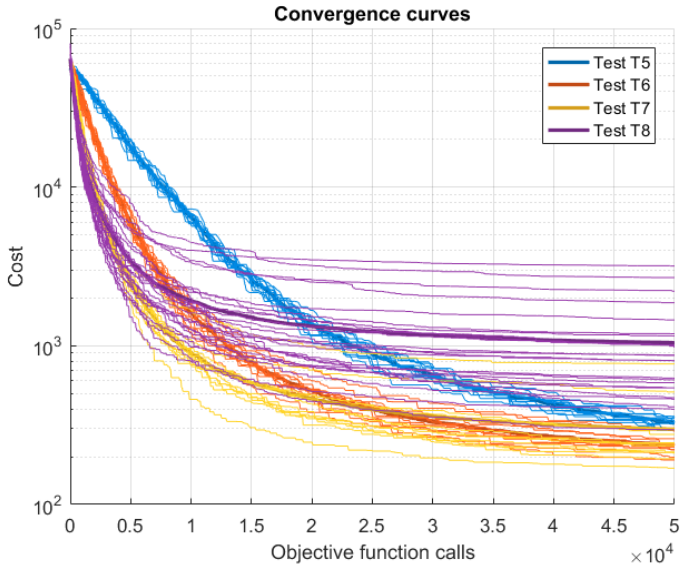


Figure 4.17: Results of the tests on SNO: convergence curves - Tests T5, T6, T7, and T8.

Test	Mean	Standard deviation	Best result
T9	593.19	72.69	479.46
T10	**383.67**	**48.12**	336.64
T11	1009.22	123.11	772.56
T12	778.51	322.69	**324.36**

Table 4.6: Results of the tests on SNO: average final value, standard deviation, best results - Tests T9, T10, T11, and T12.

is generally faster for all the tests. The trial with 2000 iterations shows a very early convergence on a local minimum.

Generally speaking, this last group of tests is drastically less effective with respect to the others.

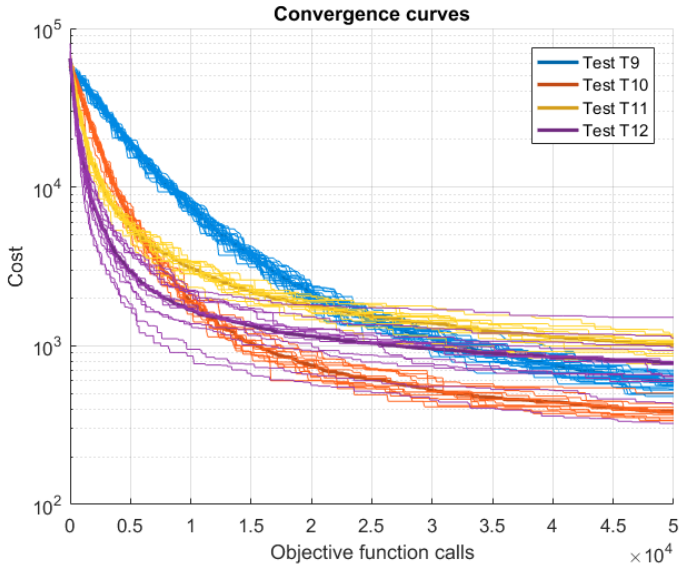


Figure 4.18: Results of the tests on SNO: convergence curves - Tests T9, T10, T11, and T12.

4.5 Social Network Optimization results

After having analysed the results obtained with several set of parameters of SNO, it is possible to focus the attention on the best set of parameters.

In this case the parameters of test T2 have been selected for running the algorithm. The choice has been driven by the very low average result and also by the low standard deviation. The parameters of test T3 have been discarded because, even if the best result is very well performing, the standard deviation is too high.

The convergence curves of 36 independent trials are shown in Figure 4.19. Each line is a single trial, while the blue thick line is the average convergence. The results show a very good convergence and reliability of SNO.

The geometry of the best solution is shown in Figure 4.20. It is interesting to see the regularity of the pattern: this is a good feature because the plane wave model (the one used in the optimization) is more reliable.

Less regular is the geometry on the corners: this is very common because these patches have lower impact on the final radiation pattern due to the much lower intensity of incident field that they receive from the feeder.

The radiation patterns obtained by the Aperture Field method are shows in Figure 4.21 In this Figure, it is possible to see that most of the radiation pattern exceeding the mask is concentrated in the main beam, especially for high scan angles.

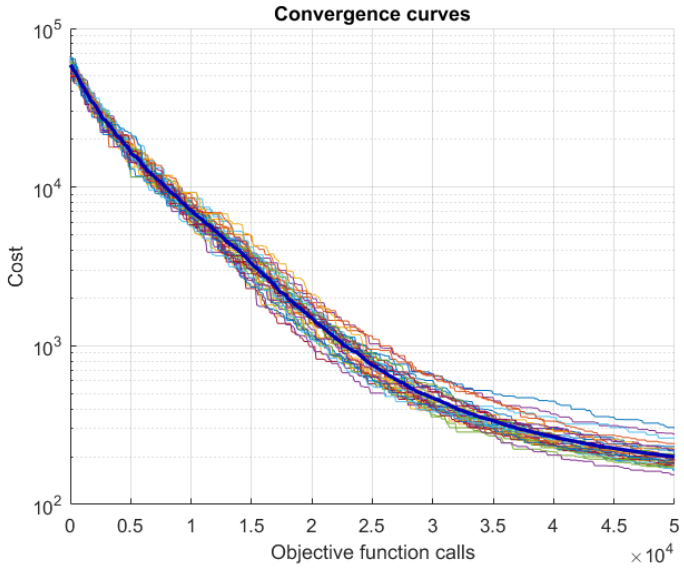


Figure 4.19: Convergence curves of SNO in 36 independent trials. The blue thick line is the average convergence.

Finally, Figures 4.22 and 4.23 validates this results by comparing the radiation pattern computed with the full wave analysis and the aperture field methods respectively.

The bandwidth of this kind of antenna is low, as for all the reflectarray: in fact, the gain computed with a full-wave simulation is reduced of 2dB with 1GHz of frequency shift.

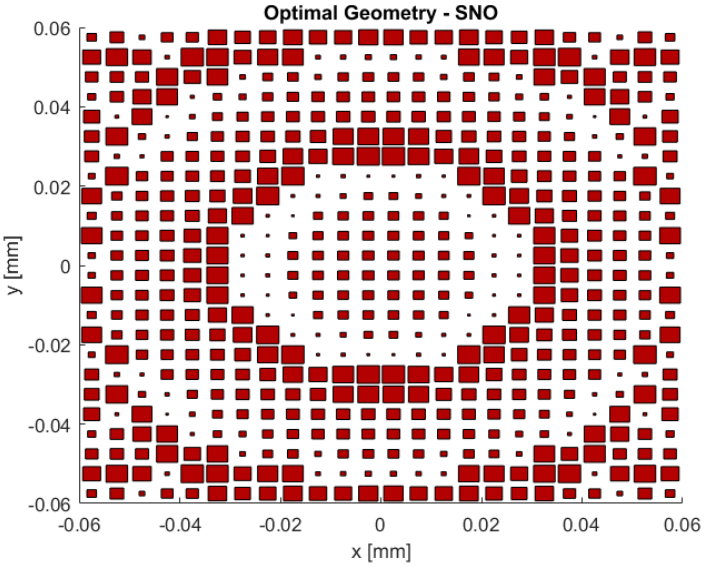


Figure 4.20: Optimal geometry obtained by SNO.

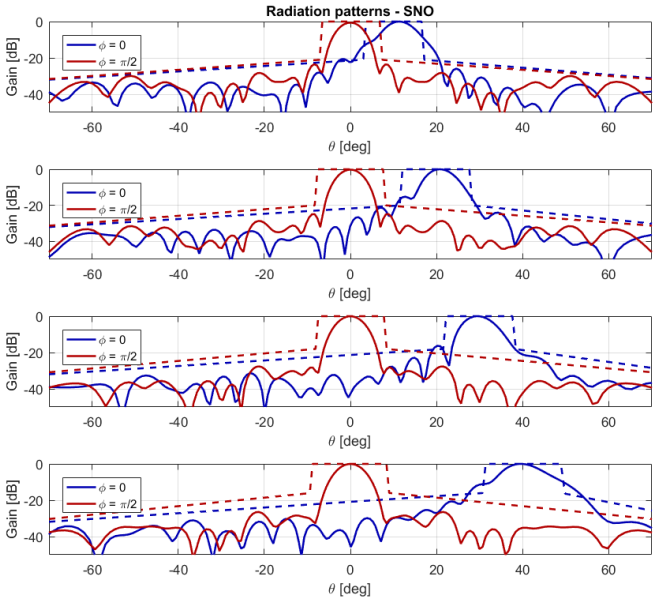


Figure 4.21: Radiation patterns in E- and H- plane for all the four scanning angles.

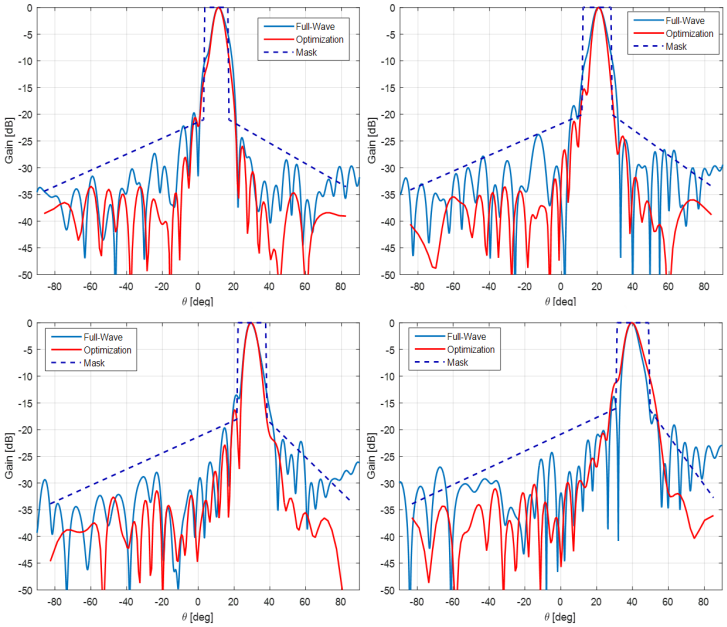


Figure 4.22: Results of the full wave simulation on the E-plane.

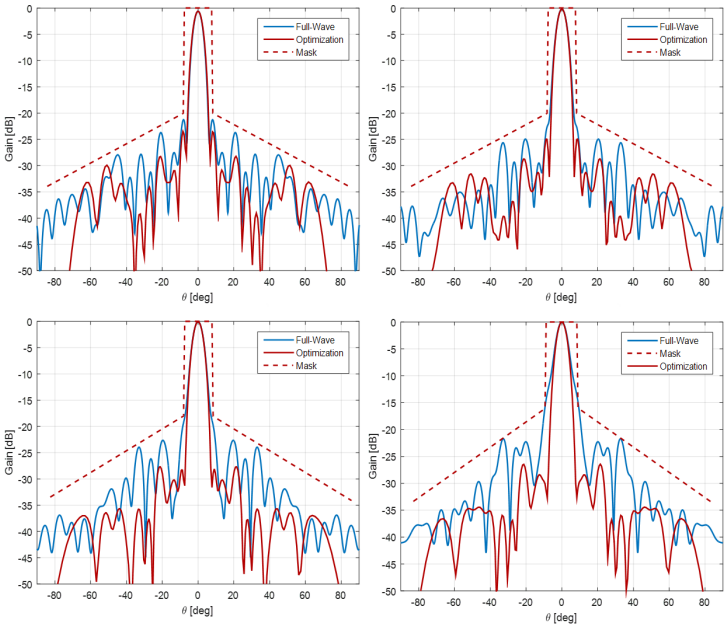


Figure 4.23: Results of the full wave simulation on the H-plane.

4.6 Comparison with other algorithms

The performance of different algorithms has been then compared. The adopted algorithms are the traditional algorithms (GA, PSO, SGA), the DE, and the two modifications of BBO (mBBO and nBBO).

For all the algorithms the termination criterion has been set 50,000 objective function calls. The population size and the value of the algorithm parameters has been set accordingly to a parametric analysis on the Schwefel-224 function. For all the algorithms, 24 independent trials have been performed.

Algorithm	Mean	Standard deviation	Best result
PSO	26823.4	6314.18	13603.59
GA	6124.97	1329.65	4224.91
SGA	873.38	1778.9	234.97
DE	196.57	67.49	**125.13**
nBBO	402.39	106.84	269.46
mBBO	539.64	106.72	335.32
SNO	**195.95**	**27.65**	154.14

Table 4.7: Comparison between SNO and other optimization algorithms.

The results are shown in Table 4.7. It is possible to notice that the results of GA and PSO are much worse than the other algorithms. The SGA, nBBO and mBBO have comparable results, but their optimal results are less competitive than DE and SNO.

The best results are achieved by DE, followed by SNO. The difference between these algorithms is very low, but the standard deviation of SNO is much lower. meaning better reliability and stability of the obtained results.

Figures 4.24 and 4.25 show the comparison among the convergence curves of the algorithms. In the first figure the more traditional algorithms have been compared with SNO, while in the second the most performing algorithms have been analysed.

From Figure 4.24 it is possible to notice that PSO and GA have a very small initial convergence and, then, they stuck in local minima. On the other hand, SGA has a higher exploitation capability, with a very fast initial convergence. Even if this convergence rate drastically drops at the half of the optimization period, the solutions are still converging.

Analysing deeply the curves of SGA, it is possible to notice that the average value is biased by a single trial that perform more than an order of magnitude worse than the others meaning lower reliability of this algorithm.

Figure 4.25 shows the comparison among DE, nBBO, mBBO, and SNO. The results of all these algorithms are very good. For what concern DE, the

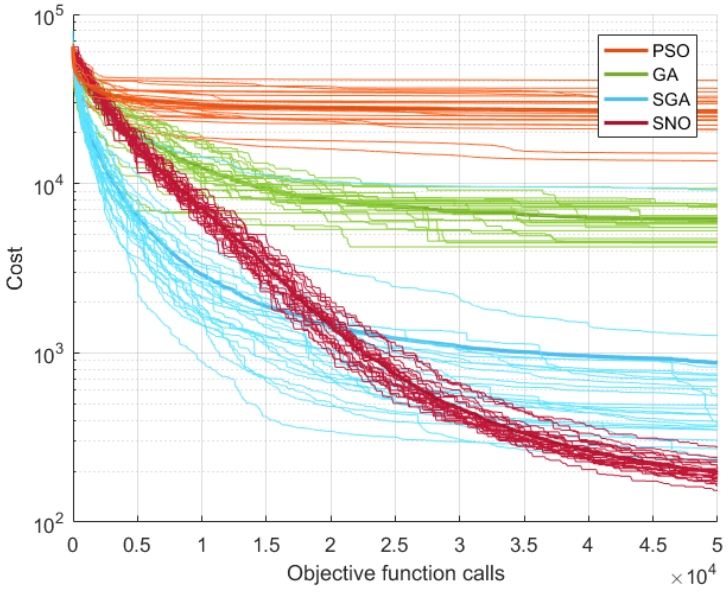


Figure 4.24: Comparison among algorithms: convergence curves - 1. GA, PSO, SGA, and SNO.

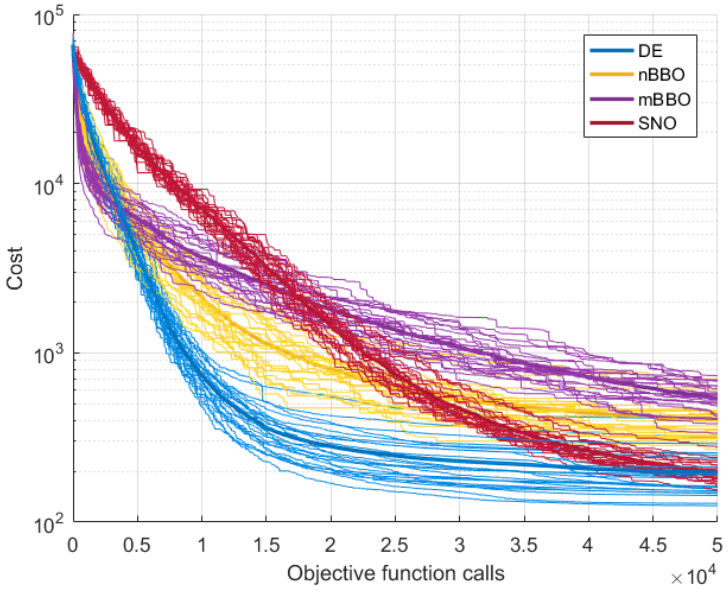


Figure 4.25: Comparison among algorithms: convergence curves - 2. DE, nBBO, mBBO, and SNO.

initial convergence is much faster than for the other algorithms, especially SNO that becomes competitive after the half of the optimization period.

Therefore, it is clear that SNO, in this specific electromagnetic problem, is able to reach the same performance of DE but its reduced standard deviation makes is reliable for this computationally intensive problem.

Chapter 5

TEAM 25 Problem

The TEAM 25 problem is one of a set of benchmarks introduced for testing the electromagnetic analysis models [1]. Among these problems, the TEAM 25 is an optimization problem.

The system is a die press with an electromagnet for orientating the magnetic powder, that creates an anisotropic permanent magnet. The aim of this problem is to obtain the shape of the die molds for having a specific magnetic flux inside the powder.

It consists in the optimization of the induction magnetic field inside a die press. It has been widely used in literature for making a comparison among optimization algorithms [143, 144]. It is a quite small problem that can be solved in reasonable time, thus it is well suited to testing the performance of the algorithm.

The problem presents only 4 design variables, but it has been proven in [145] that the cost function has a noisy shape, creating issues in deterministic algorithms. This application has been already published in [71] as preliminary test and as assessment of SNO compared with GA and PSO.

The problem should be solved with low-frequency FEM software. One of the most accurate of them is Comsol Multiphysics. The computational time required to run a single simulation is relatively high: in fact, each solution of the problem requires, as average, five minutes: this means that for a simple optimization with 2,500 objective function calls the total required time is almost two hours.

In this chapter, the optimization of this problem is faced in three different ways: firstly, a simpler, faster but less accurate software is used (FEMM4.2). With this, many tests on the algorithm and on the search space have been done.

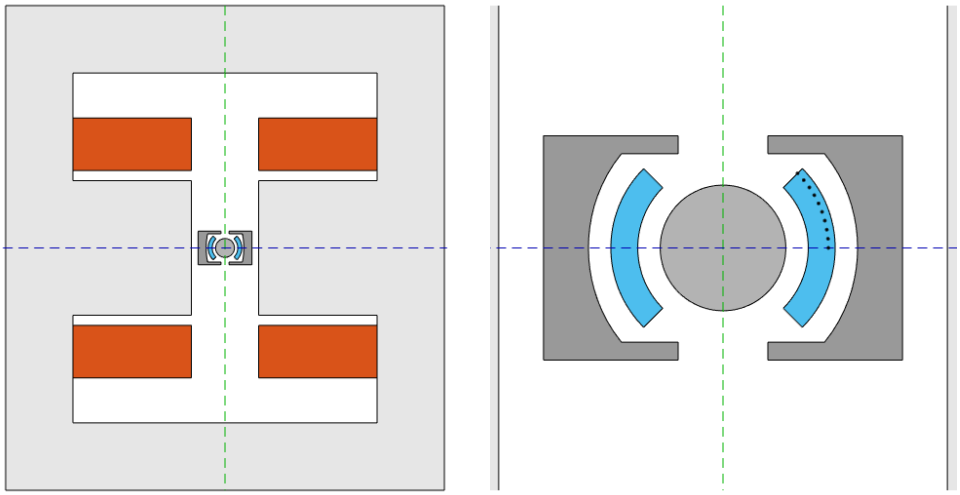
[1] `https://www.compumag.org/wp/team/`, visited 15/06/2019

Secondly, the problem has been faced directly with Comsol, in order to conduct a comparison among different algorithms has been performed.

Thirdly, using this accurate FEM simulator, a surrogate model bases on the Ordinary Kriging has been exploited in order to reduce the computational load without reducing the accuracy of results.

5.1 Problem description

The system, shown in Figure 5.1, is composed by an iron core, a part of the electromagnet, made of iron (light grey in the picture); two die molds devoted to properly create the radial magnetic flux distribution, made of iron (darker greys in the pictures); a cavity, in which the magnetic powder is inserted (light blue in the picture); and two copper coils for creating the magnetic field (orange in the picture). The white parts of the picture are filled with air.



(a) Complete geometry. (b) Zoom of the die mold.

Figure 5.1: Geometry of TEAM25 problem.

The iron is characterized by its B-H curve, shown in Figure 5.2. The dots represent the points specified in the problem settings; the interpolation as well as the extrapolation has been done linearly.

The copper has electric conductivity equal to $\sigma = 5.998 \cdot 10^7\ S/m$. The magnetic permittivity of the air is $\mu_r = 1.049$. Each coil is composed by 17479 turns and the flowing current is $0.2433A$.

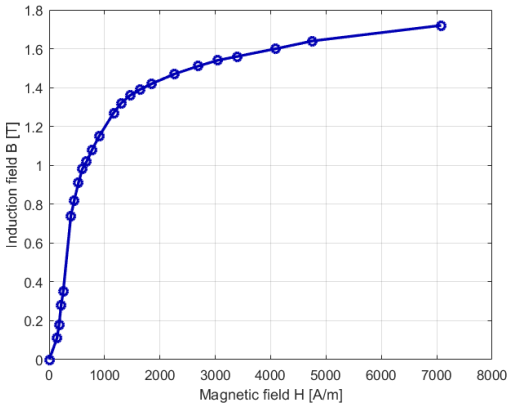


Figure 5.2: Iron B-H curve.

As described in [54], the model is assumed as two-dimensional. The symmetries of the system (dashed lines in Figure 5.1) can be exploited to reduce the computational effort. The boundary conditions of the systems are a magnetic insulation condition on the external boundary and on the horizontal symmetry line (blue dashed line in Figure 5.1, Dirichlet boundary condition), and a perfect magnetic conductor on the vertical symmetry line (green dashed line in Figure 5.1, Neumann boundary condition).

The design variables, accordingly to the definition of the TEAM problem 25, are four dimensions of the die press: L_1, L_2, L_3, and L_4. They are depicted in Figure 5.3.

The objective is to obtain a specific magnetic induction field (B_x and B_y) on an integration line placed in the cavity. This line has been approximated with a set of $N = 10$ points, as shown in Figure 5.1(b).

The requested magnetic field (target) is:

$$B_{x0} = 0.35 \cos\theta \tag{5.1}$$

$$B_{y0} = 0.35 \sin\theta \tag{5.2}$$

where θ is the angle depicted in Figure 5.3.

The cost function that has to be minimized is:

$$C = 10^3 \cdot \sum_{i=1}^{N} (|B_{xpi} - B_{x0i}| + |B_{ypi} - B_{y0i}|) \tag{5.3}$$

where B_x and B_y are the components of flux density computed on the sampling points.

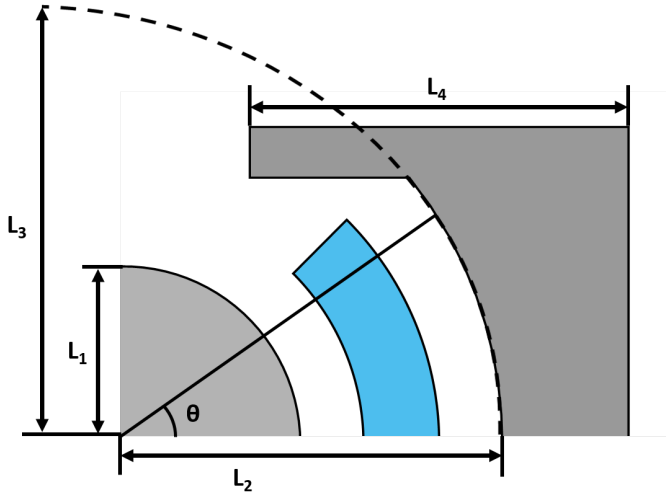


Figure 5.3: Detail of the geometry of the TEAM25 problem with highlighted the design variables.

The search space is defined with a box domain of the design variables:

$$\begin{aligned} 5 \leq L_1 \leq 9.4 \\ 12.6 \leq L_2 \leq 18 \\ 14 \leq L_3 \leq 45 \\ 4 \leq L_4 \leq 19 \end{aligned} \tag{5.4}$$

The optimization variables, indicated as $X_i,\ i = 1...4$, correspond to the design variables normalized in the range $[0, 1]$.

5.2 Optimization with FEMM4.2

The first set of tests have been performed with the free FEM simulator FEMM4.2: this software is much faster than Comsol: in fact, it requires fifteen minutes to perform 2,500 objective function calls, instead of COSMOL that requires two hours.

The optimization scheme is depicted in Figure 5.4: as mentioned, the problem is characterized by 4 optimization variables; 2,500 objective function calls have been used as basic termination criterion.

The only constraint of the problem is the box domain: the wall boundary condition has been used. As seen before, the design variables are the physicals dimensions of the object: each of them corresponds to an optimization variable.

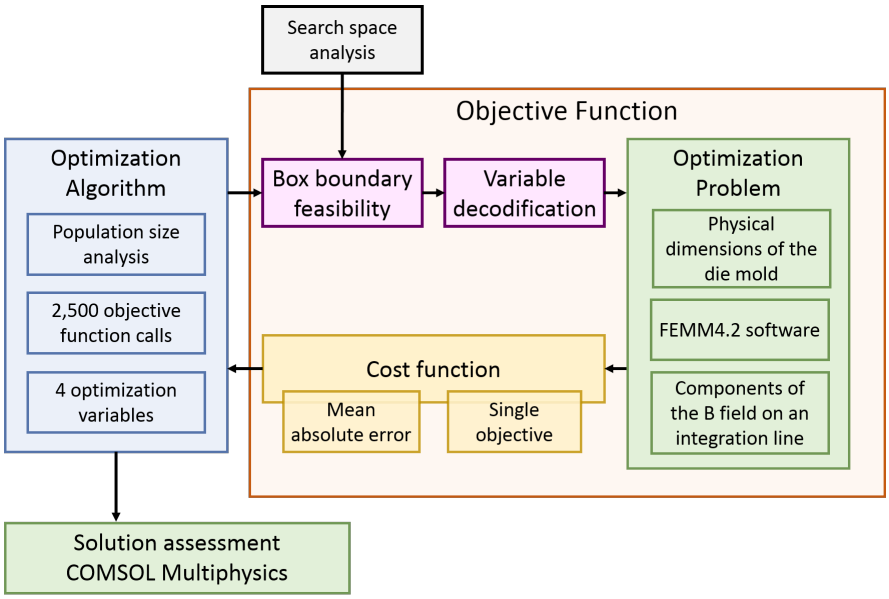


Figure 5.4: Schematization of the optimization system used with FEMM4.2 software.

The FEMM4.2 software is used for calculating the cost value in the optimization loop, while Comsol has been used at the end for assessing the obtained solution.

The problem is single objective and the cost value is the mean absolute error between the x- and y- component of the B field and their reference values.

The first test performed is the sensitivity analysis on the population size of SNO. In fact, as it has been shown in the previous chapter, the population size of an algorithm is a key parameter for having a good convergence.

Secondly, the size of the search space is inspected: firstly, some trials have been done for identifying the best action for improving the convergence of the algorithm, then an adaptive approach has been used.

Thirdly, several EAs have been compared on this problem, and, finally, the solution obtained by SNO is analysed with Comsol.

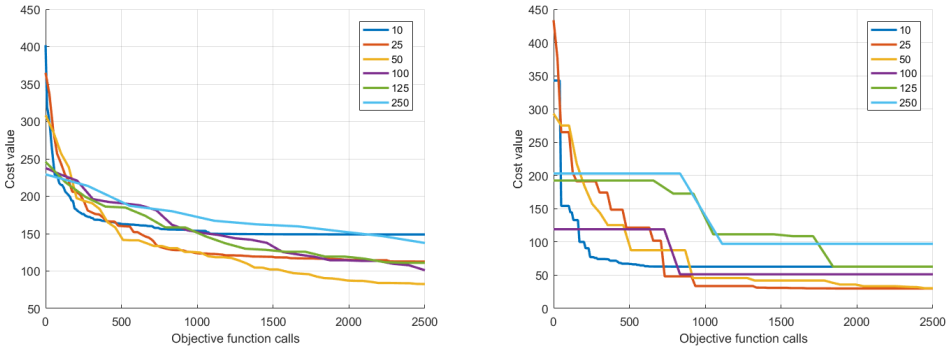
5.2.1 Analysis of SNO population size

For what concerns the analysis of SNO, the population size has been selected as parameter to be inspected due to its importance in the convergence of the algorithm.

Six values of population size have been tested: for each test some indepen-

Algorithm	Mean	Standard deviation	Best result
10	148.91	39.36	62.8
25	112.7	56.05	**29.98**
50	**82.64**	52.47	30.38
100	101.1	42.41	51.39
125	111.01	41.13	62.92
250	137.51	**21.66**	97.21

Table 5.1: Results of SNO with different population size: average value, standard deviation and best results of 12 independent trials



(a) Average convergence. (b) Convergence of best solution.

Figure 5.5: Results of SNO with different population size. Average and optimal convergence of 12 independent trials.

dent trials have been performed with termination criterion 2,500 objective function calls.

The results obtained are summarized in Table 5.1, where the average results, the standard deviation, and the best results are reported.

Figure 5.5 shows the average convergence curves. From these curves, it is possible to see that the convergence with 50 individuals is able to proceed much better than the others. Then, there is a group of solutions (25, 100 and 125 individuals) for which the average results is similar, even if the convergence with 25 individuals is much faster.

Figure 5.5(b) shows the convergence of the best trial. In this graph, it is possible to see that the results with population size 25 and 50 are very similar, even if the first one reaches the best results in less time.

5.2.2 Analysis of the search space

Here, an analysis on the search space has been done starting from the results obtained from the sensitivity analysis on the population.

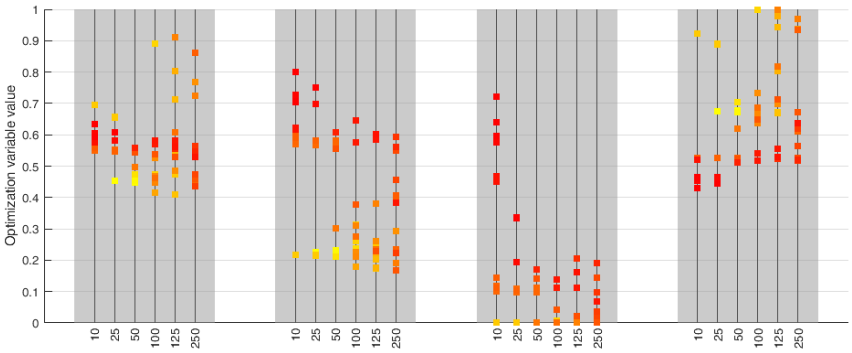


Figure 5.6: Value of the optimization variables of all the solutions found by SNO in the sensitivity analysis on the population. Each grey area corresponds to a design variable (from left to right) and each vertical line is a different population size. The colour is proportional to the fitness (lighter the better).

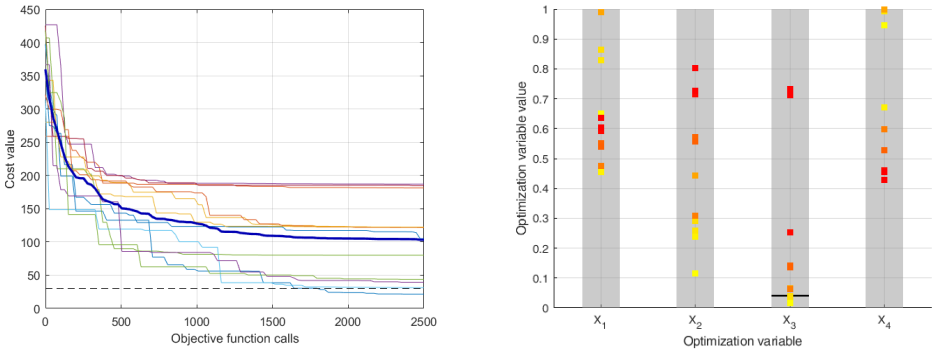
Figure 5.6 shows the value of the optimization variables of all the solutions found by SNO in the sensitivity analysis on the population. Inside each design variable column (grey areas) the results are shifted accordingly to the population size from the left (lower population size) to the right (higher population size). In this way, each variable has 72 dots grouped in columns of 12. The colour of the dots is proportional to the cost value, where the red indicates the higher cost and the yellow the lower cost.

From this plot, is possible to analyse the selected range of the variables: in fact, the optimal solutions are concentrated only in some part of the search space.

In particular, the normalized variables X_1 and X_4, are concentrated in the upper part of the space. For X_4, it is possible to notice that the best solutions are all in the upper part of the domain, while for X_1 the good solutions are more distributed (the two best solutions have X_1 very close to 0.45).

For X_2 and X_3 the good solutions are very concentrated in a narrow part of the search space. For the X_3 variable, they are packed on the boundary of the search space: this can mean that probably relaxing the lower boundary on the design variable L_3 the results can be improved.

A new allowed space has been tested on L_3: in particular, the lower boundary has been enlarged down to $12,6mm$, that is the minimal value



(a) Convergence curves. (b) Optimization variables values.

Figure 5.7: Optimization with the enlarged range for L_3 variable. In the convergence curves, the blue thick line is the average value, while the dashed line is the previous best result. In the design variable plot, the black line is the previous limit on L_3 variable.

that can be accepted due to the physical conformation of the system.

The results of this optimization are shown in Figure 5.7. In the figure the convergence of 12 independent trials and the final values of the optimization variables are shown. In the convergence curves, the black dashed line is the level of the minimal value obtained with the standard range.

Analysing the convergence curves, only one trial has been able to pass the optimal value obtained before, and other three solutions are very close to that limit.

It is possible to see that there are three very different levels of final solutions, that probably correspond to local minima of the cost function.

Figure 5.7 shows also the value of the design variables of the 12 optimal solutions found by this new optimization run. As done before, the colour is proportional to the cost value, where yellow means lower cost. The black line on the X_3 variable represents the old limit of the search space.

Only one of the independent trials has been able to find an optimal solution that overcome the previous limit on L_3, but that solution is the one with the best cost value (the solution that improves the previous best result).

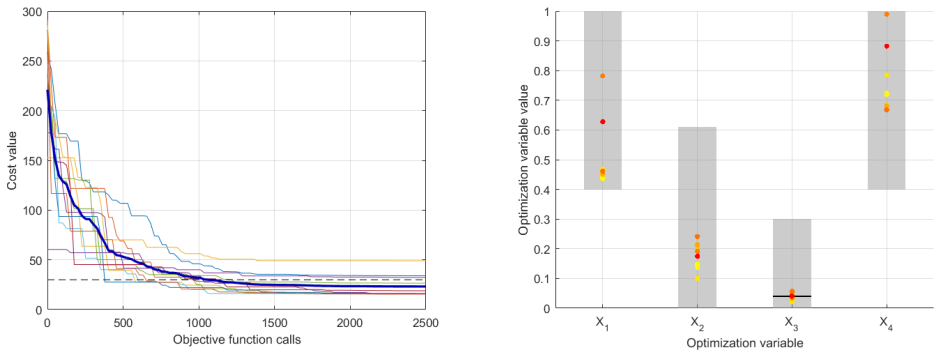
A new test has been then conduced for analysing the impact on the optimization convergence of the search space: in fact, for all the variables the allowed range has been reduced eliminating the parts that in the first tests have not produced good solutions. In particular, the considered ranges are now:

$$\begin{aligned} 6.76 \leq L_1 \leq 9.4 \\ 12.6 \leq L_2 \leq 16 \\ 12.7 \leq L_3 \leq 20 \\ 10 \leq L_4 \leq 19 \end{aligned} \tag{5.5}$$

These new ranges of the optimization variables (X_3 is represented with the new lower boundary tested in the previous run) correspond to the following:

$$\begin{aligned} 0.4 \leq X_1 \leq 1 \\ 0 \leq X_2 \leq 0.61 \\ 0 \leq X_3 \leq 0.3 \\ 0.4 \leq X_4 \leq 1 \end{aligned} \tag{5.6}$$

12 independent trials have been done with these new optimization variable ranges. The convergence curves and the value of the 12 optimal solutions found are shown in Figure 5.8.



(a) Convergence curves. (b) Optimization variables values.

Figure 5.8: Convergence curves and optimization variables with the new smaller ranges for the design variables.

In the convergence curves, the black dashed line represents the best solutions found with the first optimization run. In this case, the optimization convergence is much more stable, showing that the reduction in the design variables range is able to eliminate the local minima that affects badly the first optimization.

In this case, most of the solutions are able to perform better than the best solution found before.

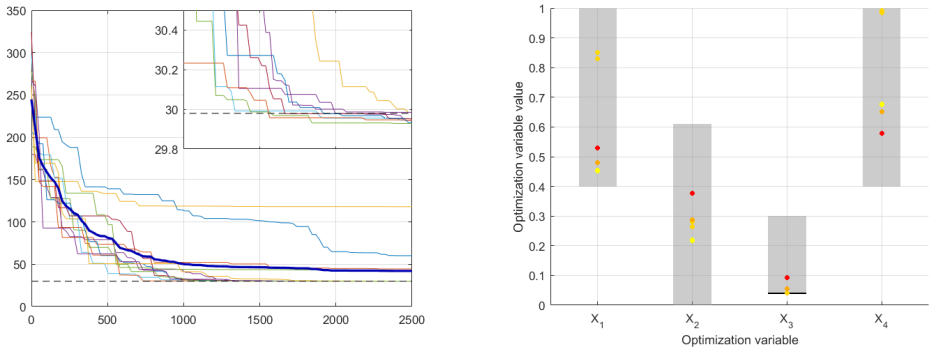
In the design variable plot (Figure 5.8(b)) the new ranges are shown with the grey rectangles. The final values of the optimization variables X_2 and X_3

are much more concentrated with respect to the previous runs. The black line in the X_3 range represent the original limit on this variable.

In this optimization, most of the solutions are outside the first limit on the X_3 optimization variable (black line), but they are not bounded by the new lower level, meaning that probably the minimum of the unconstrained function is now in the search domain. In particular, all the best solutions are outside this limit.

A new optimization test has been performed for understanding the different importance of the reduction of the ranges with respect to lowering the limit on L_3: an optimization with a reduced range on the design variable considering the first lower limit of L_3.

The results are shown in Figure 5.9 where both the convergence curves and the final values of the optimization variables are shown.



(a) Convergence curves.

(b) Optimization variables values.

Figure 5.9: Convergence curves and optimization variables with the new smaller ranges for the design variables.

In Figure 5.9(a) a zoom of the lower part of the convergence curves is shown: here, it is possible to see that several solutions are able to overcome the best result of the base case optimization.

Summarizing the outcome of this analysis:

- Decreasing the lower limit on L_3 gives to the optimizer the possibility to improve the best solution found in the standard test. However, only one of the trials has been able to improve the base case result;

- Reducing the variability range on the optimization variables with the new lower limit on L_3 is the best condition: in fact, in this case also the average convergence is below the result of the base case;

Case	Search domain	L_3 Lower limit	Average	Best	Improvements
1	Original	Original	112.7	29.98	0
2	Original	New	103.30	21.52	8.3%
3	Reduced	New	23.22	15.47	75%
4	Reduced	Original	42.09	29.93	50%

Table 5.2: Comparison between the test performed on the variable ranges: average and best value over 12 independent trials and percentage of trials better than base case (case 1).

- Reducing the variability with the initial lower value of L_3 improves the algorithm capability to find good solutions: in fact, in this case 7 out of 12 solutions outperform the base case.

From this analysis, the best action for improving the optimization capabilities, for this problem, results in reducing the range focusing the search in good area of the search domain.

Table 5.2 shows a numerical comparison between the analysed case: the four cases differ for the search domain (limitation of the search domain with respect to the basic case), and for the lower limit on L_3 (original or lowered).

The comparison is performed analysing the average and the best result over 12 independent trials, and the percentage of trials that are able to outperform the best results of the base case (case 1).

Analysing the obtained result is possible to see that the best solution is mainly influenced by the lower limit on L_3: in fact, the best solution of case 1 and case 4 have approximately the same value. On the other hand, the other two cases are able to find a much better solution.

The reduction of the search domain influences the standard deviation of the solutions: this can be seen by the difference between the best and the average cost, that in the cases 3 and 4 is much lower than for the other two configurations.

This means that the cost function has several important local minima and the reduction of the search domain excludes some of them from the search space.

According to the results obtained in these tests, a new optimization strategy has been tested with an adaptive modification of the optimization variables.

5.2.3 Adaptive range optimization

In these optimizations, a multi-step approach has been applied for exploiting the results obtained before: in particular, the progressive elimination of the local minima can be used for reducing the total optimization time keeping very good results.

The procedure adopted in this Section exploits the multiple independent trials: in fact, the total number of iterations is divided into groups and at the end of each group of iterations the information found by all the trials is exploited for reducing the search domain of the following trial.

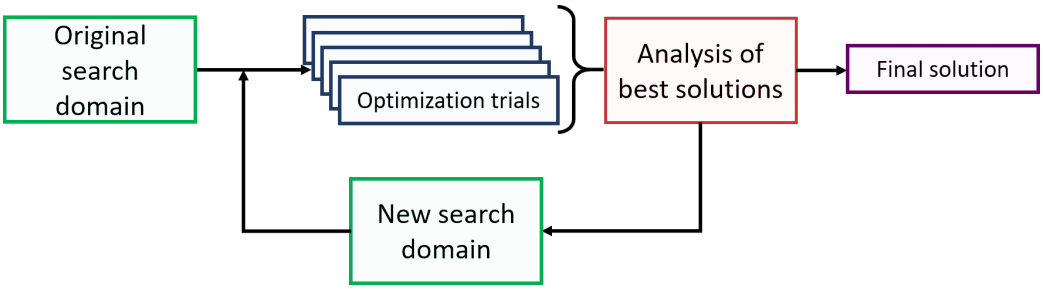


Figure 5.10: Schematization of the adaptive range procedure: the search domain is modified iteratively exploiting the output of the independent trials.

Figure 5.10 shows the schematization of the adaptive range optimization procedure: the first set of independent trials is performed on the original search domain. Then the optimal solutions are analysed and from this information the new search domain is computed.

The computation of the search domain is done in the following way: the lower bound is obtained subtracting the standard deviation of the obtained solutions from the best solution of the trials. Similarly, the upper bound is the sum of the standard deviation and the best solution of the trials.

In the Team 25 Problem optimization, the original search domain is the one with lowered bound of L_3 variable.

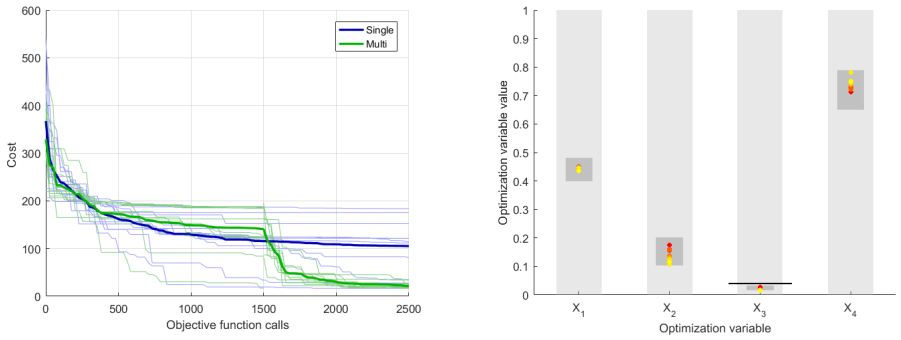
This procedure is aimed to achieve two results: the first one is a reduction of the optimization time, the second one is a reduction of the standard deviation of the solution obtained. This last is important for giving to the designer the possibility to selecting from a set of solution with a similar cost value.

Moreover, the procedure here introduced can be further used for analysing in detail the most important local minima of the function: for this application, not only the best solution of the trials should be used for creating the new search domain but all the solution with different location in the search domain.

It is interesting to notice that, with this procedure, the trials are not completely independent: in fact, from a combination of all the solutions the new search domain is computed.

The first test on this procedure is done using 2,500 objective function calls, *i.e.* the same number of iterations used in all the previous run. The adaptive range optimization objective function calls are divided into two blocks: the first one with 1,500 calls and the second one with 1,000. Figure 5.11 shows the results of this optimization, both in terms of convergence curves and search domain.

Figure 5.11(a) shows the convergence curves of the 12 trials. Two possibilities are compared: in blue there are the results of a standard optimization while in green the results of the adaptive range are depicted. The thick lines are the average convergence, while the thin ones represent the single trials.



(a) Average convergence. (b) Convergence of best solution.

Figure 5.11: Comparison of convergence curves.

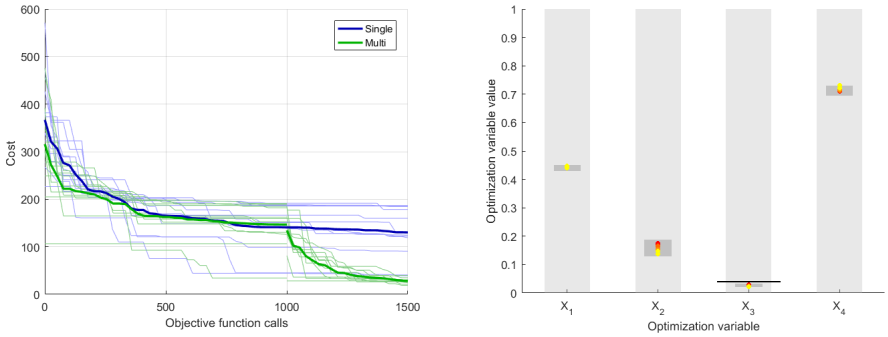
In the green lines there are some discontinuities that represent the restart of the optimization at 1,500 objective function calls.

Figure 5.11(b) shows the search domain of the two runs of the adaptive range optimization: the light grey is the original domain, while the darker one the domain of the second run. The dots are the final solutions, and the colour is proportional to the cost value (the lighter the better). The black line on the L_3 variable is the original lower bound of the search domain.

Analysing these results, it is possible to see that the adaptive range optimization improves drastically the average convergence, while the best solution is approximately the same.

The search domain in the second optimization run is very small, especially for the X_3 optimization variable: this means that the solutions of the first optimization run have value of X_3 very similar to the others.

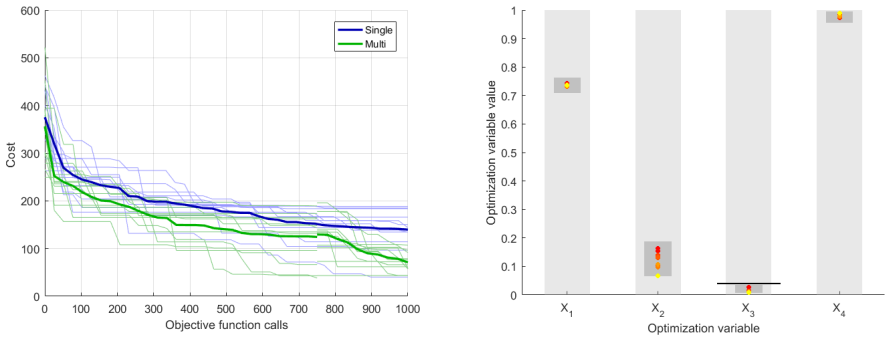
A second test is performed reducing the total number of objective function calls: 1,500 calls have been done, divided in a group of 1000 and another one of 500.



(a) Average convergence. (b) Convergence of best solution.

Figure 5.12: Comparison of convergence curves.

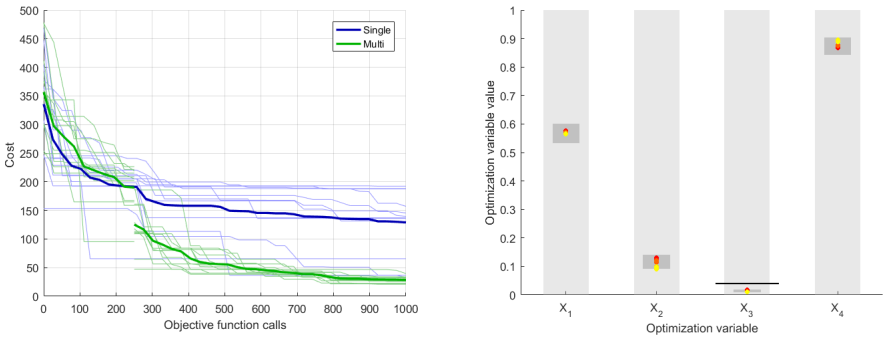
The results are shown in Figure 5.12. In this optimization run, the improvements of the adaptive range regard not only the average result but also the best one. Nonetheless, the capabilities of this technique are not completely exploited.



(a) Average convergence. (b) Convergence of best solution.

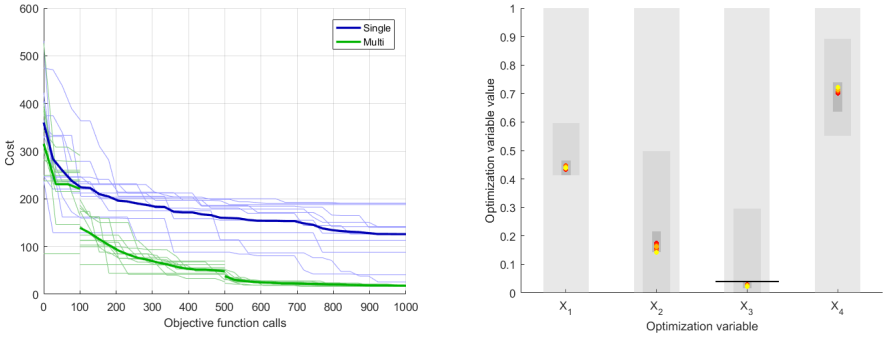
Figure 5.13: Comparison of convergence curves.

For trying to exploit at most the capabilities of the proposed technique, another run has been performed with 1000 total independent trials divided in a run of 750 and a second one of 250. The results of this optimization are shown in Figure 5.13, where it is possible to see that the results are not much better than the standard optimization.



(a) Average convergence. (b) Convergence of best solution.

Figure 5.14: Comparison of convergence curves.



(a) Average convergence. (b) Convergence of best solution.

Figure 5.15: Comparison of convergence curves.

Test	Objective function calls	Average results		Best results	
		New	Reference	New	Reference
1	1500 + 1000	**21.32**	105.12	16.36	16.37
2	1000 + 500	**28.19**	130.15	**18.98**	39.09
3	750 + 250	**71.07**	139.62	43.25	**40.20**
4	250 + 750	**28.54**	128.95	**20.93**	33.08
5	100 + 400 + 500	**17.88**	126.01	**16.25**	25.62

Table 5.3: Numerical comparison among the results obtained in the five different configurations for adaptive search domain.

For this reason, another trial with 1000 calls have been done dividing the calls in a run of 250 and then another one with 750.

In this case, shown in Figure 5.14, the results obtained are much better because EAs are very fast to identify interesting area in the search domain. Thus, the first run is able, at least for few trials, to find good searching areas. Then, the second run exploits this information for finding a very strong optimal solution.

For further exploiting this feature of EAs, a trial with three adaptation of the domain is performed. In this case, 1000 objective function calls have been divided in a first run of 100 calls, the second one with 400 and the third with 500.

Figure 5.15 shows the results in this case. It is possible to see that the first reduction of the domain is much lower with respect to the other cases, thus the exploration is not avoided too much.

All the results obtained in the previous run are summarized in Table 5.3, where it is possible to appreciate the significant improvement of the adaptive search space, especially for the average result.

5.2.4 Comparison between EAs

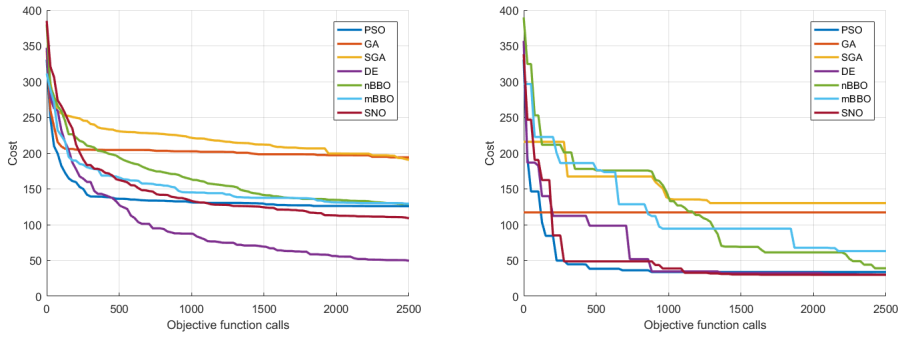
The optimization base case has been used for a comparison among EAs. For this comparison, 12 independent trials have been done as good trade-off between computational time and statistical reliability. The results of the comparison are shown in Figure 5.4, where the mean value, the standard deviation and the best result are reported for all the algorithms.

Algorithm	Mean	Standard deviation	Best result
PSO	125.86	73.77	33.79
GA	194.09	41.8	117.18
SGA	190.54	41.82	130.18
DE	**49.59**	**26.32**	30.27
nBBO	128.81	53.37	39.09
mBBO	129.34	44.77	63.06
SNO	108.93	53.21	**29.98**

Table 5.4: Comparison between SNO and other optimization algorithms.

From the data of the table, it is possible to see that the DE achieves the best mean value significantly better than the other algorithms - and the best standard deviation. For what concerns the optimal solution, the results of PSO, DE, nBBO and SNO are close one to the other, and SNO is the best one.

The same results can be observed from the convergence curves of Figure 5.16.

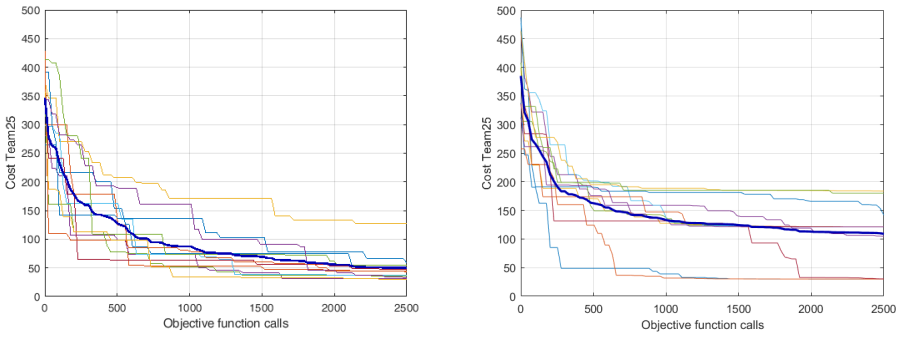


(a) Average convergence. (b) Convergence of best solution.

Figure 5.16: Comparison of convergence curves.

Analysing the convergence curves of the best solution, it is possible to see that both PSO and SNO are able to reach the best result very soon, while DE is slightly slower. However, all these three algorithms are able to reach the convergence before the half of the available time.

Figure 5.17 shows all the convergence of the independent trials of DE and SNO. The convergence of DE is characterized by a set of smaller steps: in this way, all the solutions have a very similar behaviour. On the other hand, SNO is characterized by some solutions that makes a bigger step toward the optimal solution.



(a) DE convergence. (b) SNO convergence.

Figure 5.17: Comparison among the convergence curves of all the trials of DE and SNO. The thicker line is the average convergence.

While here DE seems much more performing with respect to SNO, in the following it will be seen that, with a more accurate FEM simulator, the results of SNO improve.

5.2.5 Analysis of the solution found by

The best solution obtained by SNO in the optimization with FEMM4.2 has been inspected. This solution is the optimal one found in the set of trials used in the comparison between EAs.

The geometry obtained by the optimizer is shown in Figure 5.18, and in Table 5.5 shows the optimal values of the optimization and design variables.

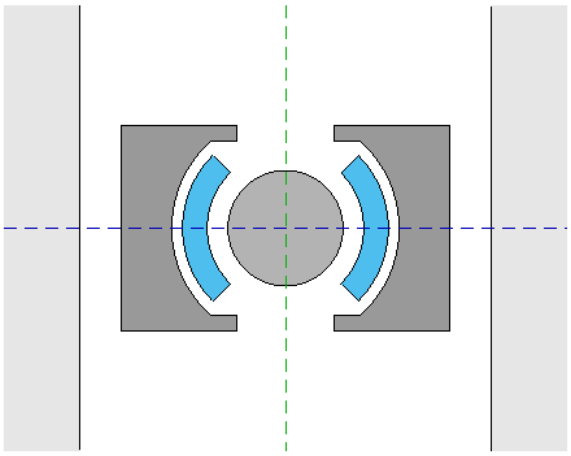


Figure 5.18: Geometry of the optimal solution found by SNO in the optimization with FEMM4.2.

Optimization variable	Value	Design variable	Value
X_1	0.4542	L_1	6.9984
X_2	0.2267	L_2	13.8242
X_3	0.0000	L_3	14.0000
X_4	0.6759	L_4	14.1389

Table 5.5: Optimal values obtained by SNO: design and optimization variables.

The values of X_3 in Table 5.5 shows the behaviour analysed before: in fact the optimal solution is on the boundary of the optimization domain.

This solution has been analysed both with FEMM4.2 and with Comsol for understanding the differences between the two simulators. The results are shown in Figure 5.19 where the values of the B field have been calculated with Comsol (blue line) and with FEMM4.2 (red line).

It is possible to see that the y- component of the induction field is very accurate, while in the x- component the FEMM4.2 solution has a quite large error when evaluated with the other commercial simulator.

This difference can be appreciated also analysing the corresponding cost value: evaluating this solution with FEMM4.2 the cost value is 29.98, while with COSMOL it is 80.94.

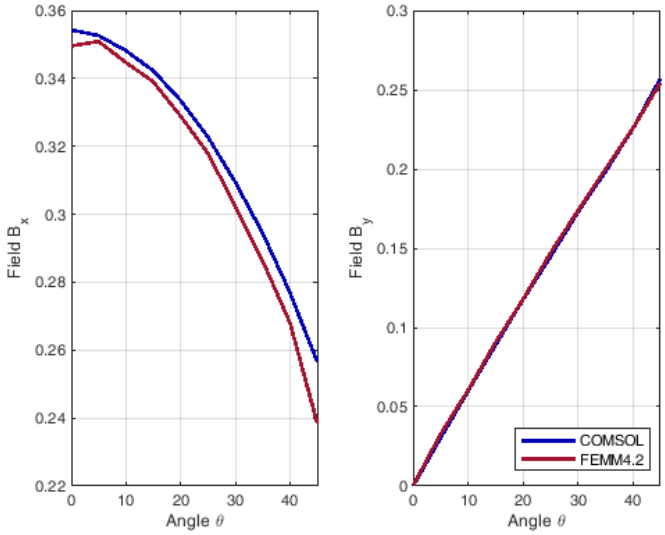


Figure 5.19: Values of the field of the optimal geometry calculated with Comsol (blue line) and FEMM4.2 (red line).

5.3 Optimization with Comsol Multiphysics

The same optimization problem has been them optimized also using Comsol Multiphysics. This is a FEM software that is able to solve many simulation problems. The AC/DC module has been exploited in modelling the TEAM25 geometry and its physical proprieties.

As done for FEM2.4, only one fourth of the geometry is represented and the boundary conditions have been implemented for obtaining the correct results (Dirichlet and Neumann boundary conditions).

All the materials proprieties have been set as done for FEMM4.2 simulator, in particular the non-linear relation B-H of the iron and the magnetic permeability of the magnetic powder in the cavity have been taken into account. The field is produced inserting in the simulator a current in the coils.

For what concern the mesh, free triangular elements have been used in the simulation. The solver is a stationary one, and the results on the integration line are obtained with a post-processing of the results. No additional points are included in the initial geometry because they induce an high distortion of the mesh.

The geometry is created with the LiveLink module of Comsol that makes possible the interaction directly with Matlab, and the same module is used for gathering the field values after the solution of the problem.

The time required by the solution of the problem with Comsol is much higher than the one needed by FEMM4.2: in fact, with this simulator two hours are required to perform a trial with 2,500 objective function calls.

5.3.1 Optimization with SNO

The first optimization of TEAM25 with Comsol have been performed with SNO. 2500 objective function calls have been used as the termination criterion and the population size of the algorithm has been set to 25 individuals.

The convergence curves of 10 independent trials are shown in Figure 5.20, where the thin lines represent each single trial and the blue thick one the average convergence.

The convergence curves of SNO are one very close to the other, showing that the function is quite regular. The largest convergence happens in the first 1000 objective function calls, while in the remaining optimization time is devoted to a fine tuning of the solution.

Comparing this convergence curves with the ones of FEMM4.2 (Figure 5.17(b)) it is possible to see that with Comsol solver the problems seems much more regular: in fact, the convergence is more uniform and all the solutions are one close to the other, while the other convergences are characterized by three attraction points.

On Figure 5.21, it is possible to see that the y- component of the field is able to follow in a very good way the reference value, excepted for a slight error for high values of θ angle: this variation is highly influenced by the edge in the geometry that induces a tiny deformation in the field. The x-component of the field is less accurate, even if the difference is very low.

Finally, Figure 5.22 shows the induction field in the geometry: in this plot, it is clear the effect of the edge in the geometry.

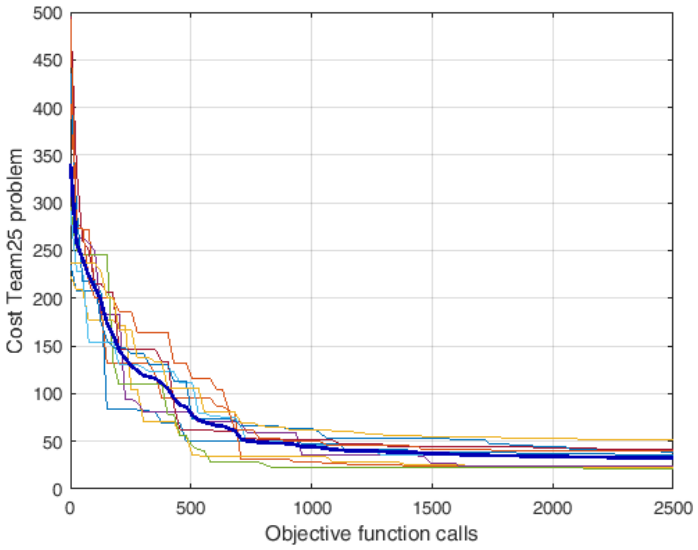


Figure 5.20: Convergence curves of 10 independent trials of Social Network Optimization. The thick line is the average convergence.

Figure 5.23 shows the distribution of the design variables of the 10 final solutions of SNO. As done in the previous plots, the colour is proportional to the cost where lighter points means better solutions.

The grey area is the allowed search space: in this optimization, the original limit on the L_3 variable has been taken into consideration, thus the search space is considered as slightly reduced with respect to the maximum allowed.

This can be seen comparing these results with the ones of Figure 5.7, in which it is possible to see that, at least for variable 1 and 4, there are two identified local minima.

Also in this simulations, the X_3 variable is characterized by good results on the edge of the search space, thus a new simulation with the enlarged search domain has been performed.

Figure 5.24 shows the convergence curves of the new optimization with the enlarged search domain. Each independent trials is drawn with the grey line, the average convergence of these is the blue line, while the red line is the average convergence of the standard optimization (it is the same of Figure 5.20).

From this Figure, it is possible to see that, in this case, the convergence is drastically worst, mainly in the mid of the optimization time. The final average result is worst and the standard deviation is much higher.

Figure 5.25 shows the design variables of the independent trials, shown

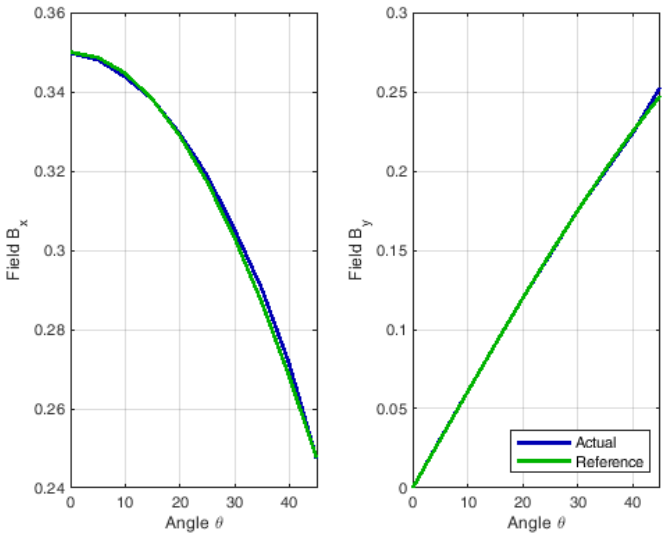


Figure 5.21: Results of the field in x- and y- direction for the best solution achieved by SNO with Comsol. In green the reference value, in blue the actual one.

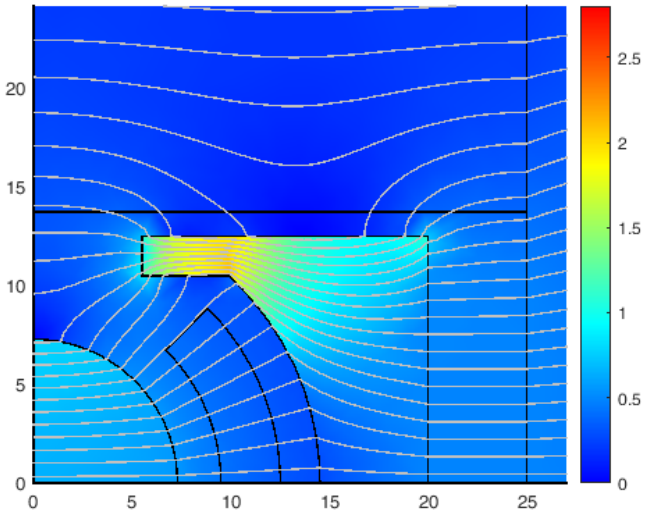


Figure 5.22: Module of the B field in the system computed by Comsol.

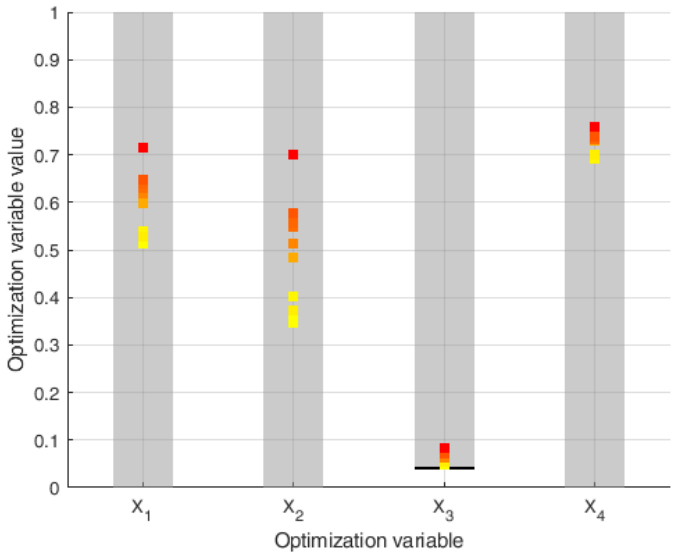


Figure 5.23: Final values of the design variables in the 10 independent trials with Comsol: the colour is proportional to the cost, lighter the better.

with the same method used before.

It is interesting to notice that the optimizer does not overcome the previous limit on the X_3 variable (the black horizontal line): this means that the optimal solutions found are all within the previous search domain. This explains also the results shown in the convergence curves: the search domain is larger, thus the convergence is harder, while no better solutions are in the new search space.

5.3.2 Comparison between EAs

After the analysis on the results obtained by SNO, a new comparison between EAs has been done using Comsol. For all of them, 10 independent trials have been performed using as termination criterion 2,500 objective function calls.

Table 5.6 shows the results of all the optimizers: in particular, the mean value, the standard deviation and the best results are computed on 10 independent trials.

The DE achieve the best mean value with a very low standard deviation. SNO is only slightly worst than DE on the average value, but its optimal result is the best one among all the optimization algorithms.

It is interesting to see that on the mean value only SNO and DE obtain comparable results; on the other hand, analysing the best trial, four algorithms (SNO, DE, PSO, and nBBO) achieve comparable results.

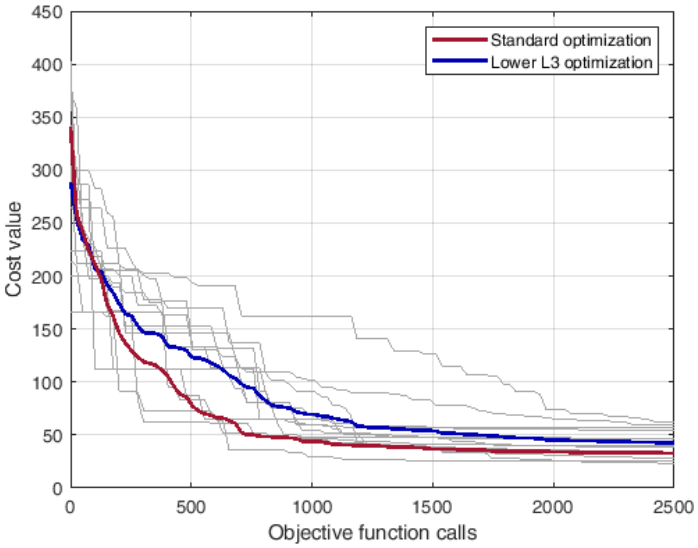


Figure 5.24: Convergence curves of 10 independent trials with the new range for L_3 (grey lines), average convergence (blue line) and average convergence for the standard optimization (red line).

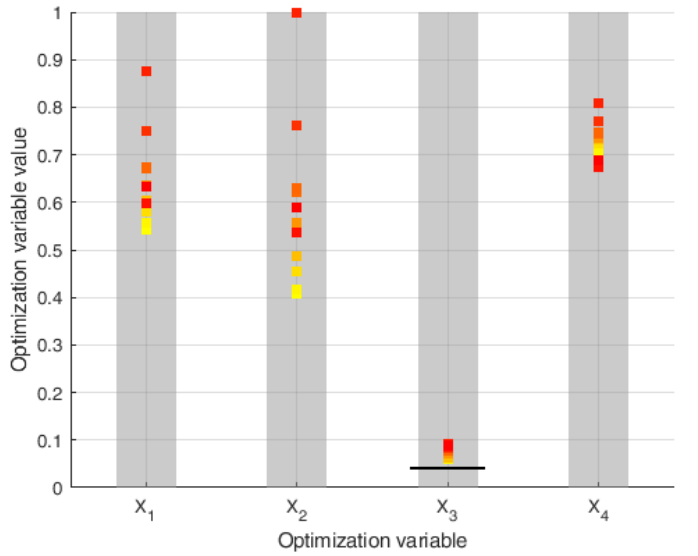
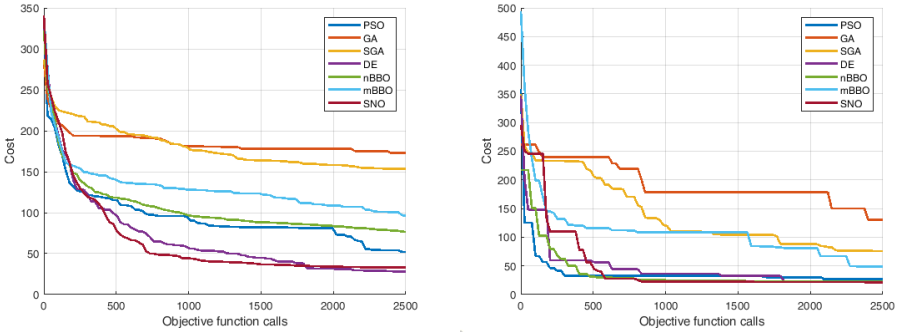


Figure 5.25: Final values of the design variables in the 10 independent trials with the new L_3 range: the colour is proportional to the cost, lighter the better.

Algorithm	Mean	Standard deviation	Best result
PSO	52.03	10.37	26.57
GA	172.93	28.7	130.11
SGA	153.23	42.78	75.76
DE	**28**	**6.06**	22.57
nBBO	76.48	49.56	22.9
mBBO	96.74	43.62	49.06
SNO	32.93	10.35	**21.57**

Table 5.6: Comparison between SNO and other optimization algorithms: mean, standard deviation and best results of 10 independent trials.



(a) Average convergence.

(b) Convergence of best solution.

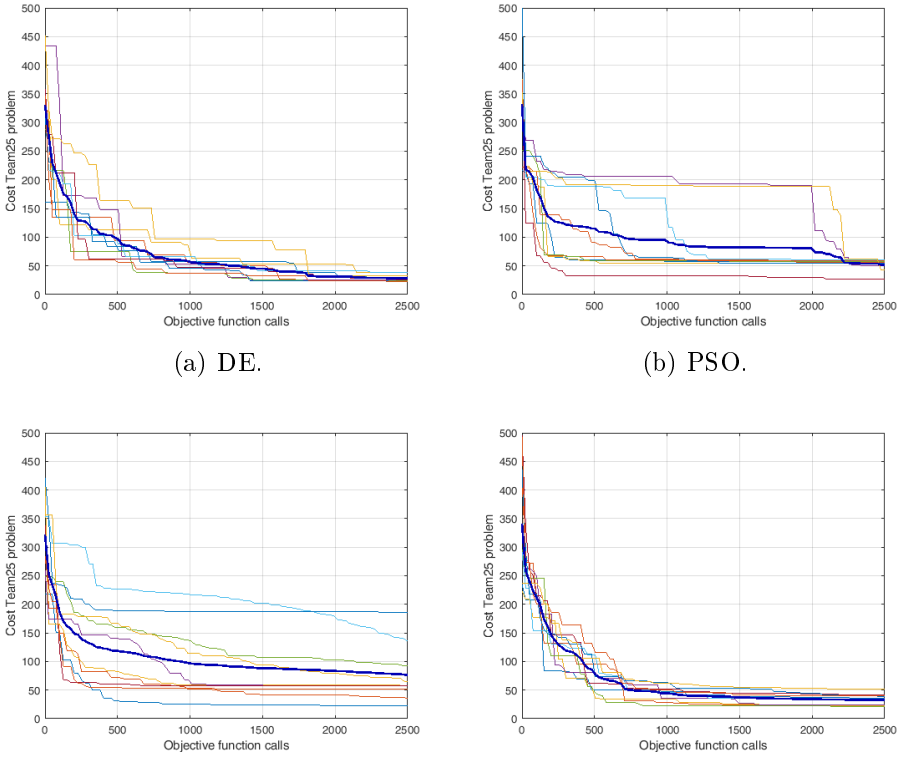
Figure 5.26: Comparison of convergence curves.

Figure 5.26 shows the average convergence curves and the convergence of the best trial for each algorithm. Analysing the average convergence curves, it is possible to see that SNO and DE have a similar behaviour: for a set of iterations SNO is slightly better than DE, and then the condition is inverted at the end of the optimization, thus confirming the issue could be anticipated by the well-known No Free Lunch theorem.

The convergence of the best trial is slightly different: PSO shows the best convergence at the beginning, while SNO is the slowest in the first 500 objective function calls.

Figure 5.27 shows the convergence curves of the first 4 top algorithms: DE, SNO, nBBO, and PSO.

Comparing these convergence curves, SNO and DE show the most uniform behaviour: in fact, PSO is able to achieve a very good solution only with one trial while all the others are concentrated on another local minimum. In the PSO convergence it is possible to notice a local minimum with cost



(a) DE. (b) PSO.

(c) nBBO. (d) SNO.

Figure 5.27: Comparison among the convergence curves of all the trials of DE, PSO, nBBO, and SNO. The thicker line is the average convergence.

around 200 in which two solutions are stacked for most of the convergence time.

The behaviour of nBBO is characterized by a very high standard deviation: the best solution is obtained very soon in the optimization time, while for most of the others the convergence is very slow. In this case there is one solution that is stacked in the local minimum identified by the PSO curves.

Figure 5.28 shows the distribution of the design variables of the final solutions of all the algorithms. Each grey area is a design variable, while the vertical lines are the optimization algorithms. As before, the colour is proportional to the cost value (lighter the better).

Analysing this Figure, the correlation between the variables is clear: in fact, the cost value of a solution is almost driven by the third design variable. Only in a second step, the others are considered. For understanding this, it is possible to compare the values of SNO and SGA: they have solutions with comparable values of X_1, X_2 and X_4, but the cost is significantly higher due

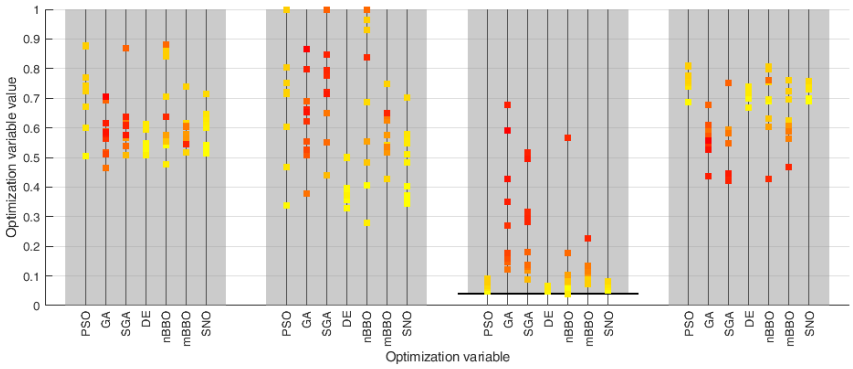


Figure 5.28: Comparison of the final values of the optimization variables for all the algorithms. The design variables are identified by the grey areas. Each vertical column represent an optimization algorithm; the colour is proportional to the cost (the lighter the better).

to the difference on the third design variable. On the other hand, PSO and SNO differs mainly for the other variables and they have similar values on the third design variable. nBBO is the only algorithm that reaches exactly the lower boundary on X_3.

5.3.3 Application of surrogate models

The last test done on this benchmark problem is the application of surrogate models. In particular the Ordinary Kriging [85] has been used due to its simplicity and its capability to return a confidence value.

The method used for the integration between the optimizer and the surrogate models is the one shown in Figure 5.29.

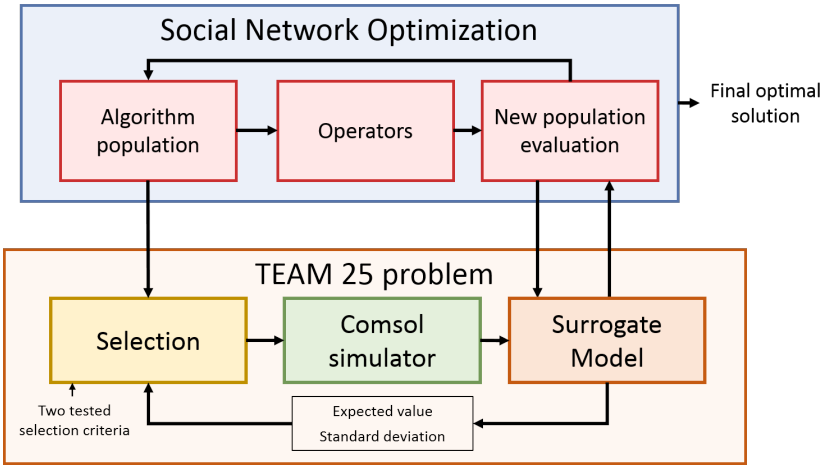


Figure 5.29: Comparison between the convergence curves with different sampling percentage - first update algorithm. The thick lines are the average convergences, while the thin ones are the best trial for each case.

Here, two criteria for updating the sampling points have been tested: in fact, this is the key feature for tuning the trade-off between optimization time and result accuracy. The first criterion is selecting a specific percentage of the solutions, sorted accordingly to the cost value. In the second criterion, the sort is performed using the value minus the standard deviation.

Figure 5.30 shows the convergence curves with the first selection criterion: the thick lines are the average convergence, while the thin ones the best trial out of 12. The results show that, even if the average result is drastically higher, the best trial is comparable with the one with 100% of real objective function calls. This means that this method reduced too much the exploration capabilities of the algorithm.

In order to improve the exploration, also the standard deviation has been taken into account in the sampling process for updating the Ordinary Kriging model: in particular, the best solutions has been considered using, as criterion:

$$\text{sort}(y - s) \tag{5.7}$$

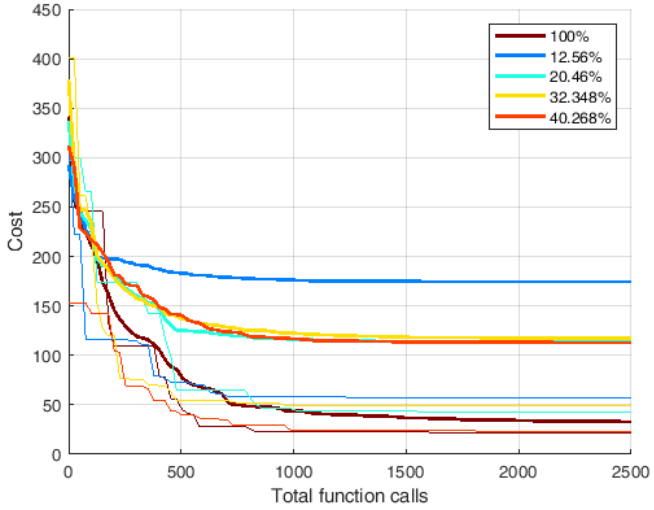


Figure 5.30: Comparison between the convergence curves with different sampling percentage - first update algorithm. The thick lines are the average convergences, while the thin ones are the best trial for each case.

The convergence curves, in this case, are shown in Figure 5.31; also in this figure, both the average result (thick lines) and the best trial (thin lines) are shown.

The average convergence is drastically improved with respect to the previous case, in particular for 32% and 40% of sampling.

The best trial is slightly less performing with respect to the standard optimization, but the time improvement is considerable: with 40% of sampling the required time is the 56% of the original time, while with 32% of sampling the required time is 44%.

The difference between the sampling percentage and the time saving corresponds to the time required for training the surrogate model. This overhead time is less impactive when the computational time for sampling is higher.

Figure 5.32 shows the x- and y- components of the B field on the sampling points, and their comparison with the reference.

It is possible to compare this Figure with Figure 5.21, that shows the field for the solution with 100% of sampling, and with Figure 5.19, that shows the performance of the solution obtained with FEMM4.2.

It is clear that the two solutions obtained with Comsol are comparable and their error is much lower with respect to the solution with FEMM4.2.

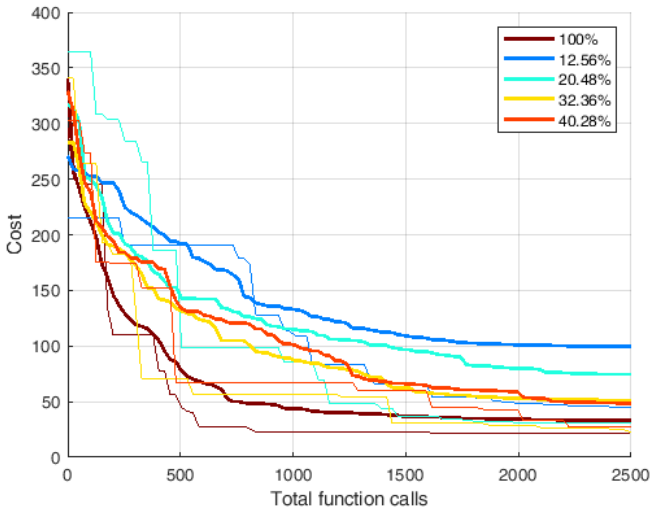


Figure 5.31: Comparison between the convergence curves with different sampling percentage - second update algorithm. The thick lines are the average convergences, while the thin ones are the best trial for each case.

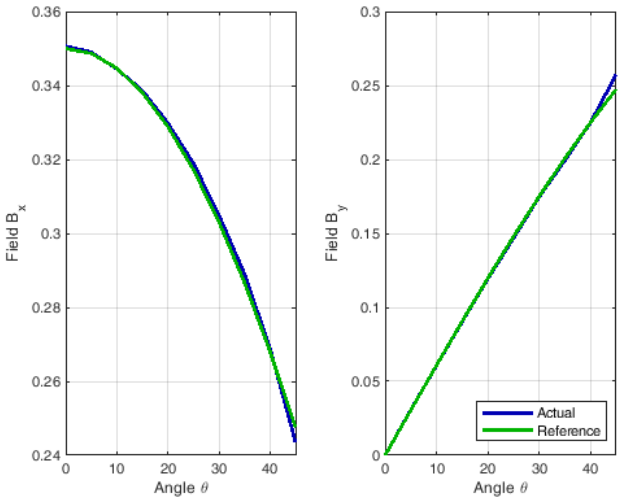


Figure 5.32: Comparison between actual field components and reference for the solution obtained with 40% of sampling.

5.4 Results discussion

In this Chapter the optimization of a low frequency electromagnetic application has been investigated by means of two different FEM software: the simpler and faster FEMM4.2 and the more accurate Comsol.

The lower time required by the first software allows several tests on the optimization process: the optimal population size of Social Network Optimization is analysed and a comparison between different algorithms have been performed.

Moreover, several tests have been done on the search space because the optimal solution is on the boundary of the search space. Here, a new approach with an adaptive range has been proposed and tested successfully.

The optimal solution obtained by FEMM4.2 is very good, but it shows a non satisfactory behaviour when it is evaluated by means of Comsol, so this last software has been used directly in the optimization.

Finally, another approach has been investigated for reducing the computational time, *i.e.* the use of the Ordinary Kriging surrogate model. The combination of Comsol and the surrogate model shows a good trade-off between results accuracy and the computational load requested, thus confirming the validity of the proposed approach.

Part of the analysis here shown has been developed in the optimization process of a Tubular Permanent Magnet Linear Generator, that has been published in [71].

Chapter 6

Conclusions

In this thesis, the design of a novel evolutionary approach for the optimization of complex electromagnetic systems has been studied in most of its components.

The whole optimization system architecture has been based on two pillars: the optimization algorithms and the optimization problem. These two interact by means of two connections: the first one, from the optimizer to the problem, is composed by the box boundary condition management and the design variables mapping, while the second one is composed by the cost function.

All these parts of the optimization system work together and highly affect the final result of the process.

For what concerns the optimization algorithm, the first choice is the type of algorithm that should be used: this selection depends on the problem itself (linear problems, multimodal problems, ...), on the type of optimization variables that are most useful (real valued or binary values), and on the number of objective functions (single-objective, multi-objective, many-objective).

For multimodal problems, a good choice are Evolutionary Optimization Algorithms (EAs) that are able to provide a global search in the entire search domain: they are able to easily face non-linear problems and they are also widely applied for multi-objective application.

For a proper optimization process, the algorithm should be selected and analysed: in fact, in many cases the algorithm operators can be adapted to the specific peculiarities of the problem; in all the cases, these algorithms behaviour depends on some user-defined parameters. The most common of them is the population size.

The selection of the algorithm parameters should take into account the trade-off between two important factors: the capability of the algorithm to explore the search domain and the capability to exploit available information

to find the optimal solution.

In this thesis, Social Network Optimization has been used as reference algorithm. This choice is motivated by two important factors: the development of the algorithm has been performed in this research work, thus the algorithm operators and the parameters effects are well known. Secondly, this algorithm shows a very good behaviour in many applications in which it has been tested.

The algorithm working principles have been deeply investigated and two different approaches have been used for understanding the parameters effect on the algorithm performance: firstly, some analytical models have been used to investigate the population behaviour as a function of the parameters, and then a deep parametric analysis has conducted done with a numerical approach.

The reported results show that the numerical behaviour of the algorithm corresponds within some limits to the analytical models; an important feature has been discovered with this numerical analysis: the two most influencing parameters have their optimal values in the same area for different objective functions. This is a very important peculiarity because it increases the range of this algorithm applications.

Social Network Optimization has been then compared with other optimization algorithms on a set of 15 mathematical benchmarks that are commonly used for assessing evolutionary algorithms. Some of the algorithms used in the comparison are EAs while other are point-based methods. The results of these tests show a very good stability of SNO performance on different problems.

Finally, two different electromagnetic problems have been used for testing the entire optimization scheme design: the design of a beam-scanning reflectarray and the design of an electromagnetic die mold.

In both these applications SNO has been used as the main algorithm for testing different designs of the optimization scheme.

For what concerns the reflectarray problem, the optimization scheme has been analysed in the following aspects:

- The definition of the cost function has been deeply investigated because the antenna has several performance parameters that are included in a single cost value. The scalarization parameters should be properly chosen: they drastically affect the final optimal solution.

- The function that guarantees the box boundary conditions has been investigated: the traditional wall condition shows the best performance, while the closed search space results in a lower convergence rate because it creates useless oscillations in the population.

- The aperture field method used for calculating the radiation pattern has been assessed with respect to the same method but with a more complex patch characterization and with respect to a commercial fill-wave simulator. The results show that the simple aperture field method, in which the patch is defined by only its length, is enough accurate for performing the optimization process.

- Different SNO parameters have been analysed; in particular, the effect of the population size has been considered. The results are that a large population size guarantees a more constant convergence rate, but it requires a huge computational effort. On the other hand, a small population size guarantees a very fast convergence at the beginning that may lead to a premature convergence on local minima. The characterization of these two behaviours is important because it can be used as heuristic technique for defining the optimal population size.

After this analysis of the optimization scheme, the results of SNO have been analysed and compared with the ones of other EAs. From this comparison conduced over 24 independent trials, SNO resulted to have a performance comparable with DE, but with lower mean value and with a much smaller standard deviation, thus confirming it as a reliable optimization approach.

The second problem analysed is a different electromagnetic design problem and it has been firstly faced using a free FEM simulator, performing several tests on the optimization scheme:

- The optimal population size of SNO has been analysed: the results shows the same behaviour seen in the first problem, but with smaller populations.

- The solutions found by SNO in the sensitivity analysis on the population size have been analysed for understanding the search space of the problem. In particular, the optimal solutions are very often on the boundary of one of the design variables. For this reason, a new optimization test has been run with an enlarged search space: the optimal solutions improves, but the problem appears to be more complex because an higher number of solutions are in a local minima.

- With the aim of improving the optimization performance, a different approach has been used for the search domain definition: the domain is modified iteratively at the end of smaller optimization runs. In this way the exploitation capabilities of the algorithm are increased and the performance are improved both in terms of optimal and average solutions found.

Additionally, a commercial FEM software has been used for the same optimization problem: the well-known Comsol Multiphysics that is very accurate, but the computational time required is 8 times higher. With this software a comparison among the EAs has been performed, showing again the very good results of SNO.

Finally, the optimal solution obtained by SNO with FEMM4.2 have been analysed with Comsol: it results that, even if the cost value computed with FEMM is very low, the same solution has a very high error when analysed with the more accurate FEM simulator.

For solving this problem, and for finding a trade-off among the required computational time and the accuracy, a surrogate model has been introduced in the optimization scheme.

Here, the surrogate model is the Ordinary Kriging: it has been selected because its formulation is easy and it can be trained with an analytical approach. Moreover, the output of this model is the expected value and the confidence level: this last value can be very useful for selecting iteratively during the optimization procedure the new sampling point from the algorithm population in order to guarantee the accuracy of the model.

By using the surrogate models approach it has been possible to achieve comparable results with respect to the standard optimization with Comsol in about 30/40% of the required time. The results obtained in this simulation are very accurate because at each iteration the best individual of the population is evaluated with Comsol.

The results obtained in these two electromagnetic applications show that, for achieving highly performing results with EAs, a proper design of the optimization system should be found. This is highly affected by the specific problem, but there are several common features to all of them and the proposed optimization apporach was found to be suitable to properly address these aspects.

Appendix A

Benchmark functions

In this appendix the fitness functions used to test the algorithms will be explained.

These fitness functions are often used in literature [76]. For all of them, it is possible to have the same formulation with an arbitrary number of design variables. In the test performed in the thesis, 20 design variables have been used, so they are function:

$$f : \mathbb{R}^{20} \rightarrow \mathbb{R} \tag{A.1}$$

The pictures of the 2-D version of the functions have been elaborated with a Intel-i7 computer.

Ackley function

The Ackley function (Figure A.1) is a multimodal function characterized by a concave trend with a medium-frequency oscillation on it. Its mathematical formulation is:

$$f(\mathbf{x}) = 20 + e^1 - 20 \cdot e^{-0.2\sqrt{\sum_i (x_i - x_0)^2 / M}} - e^{\sum_i \cos(2\pi(x_i - x_0)/M}} \tag{A.2}$$

where M is the length of $\mathbf{x}$ and it represent the number of design variables. $x_0 = -7$ is the position of the function global minimum.

The Ackley search domain is:

$$-15 \leq x_i \leq 15 \tag{A.3}$$

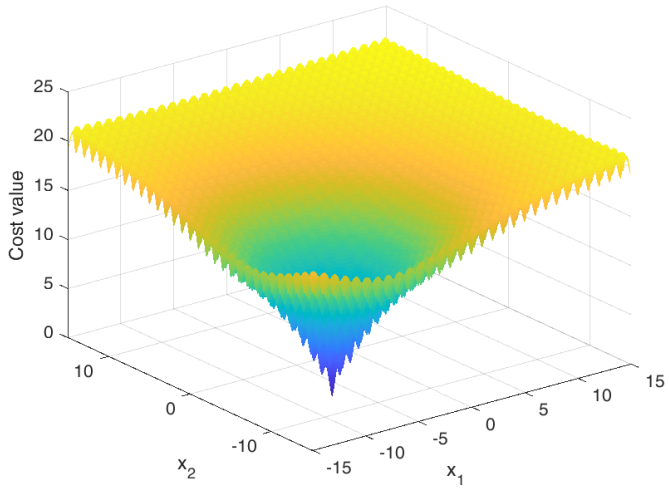


Figure A.1: Ackley function

Griewank function

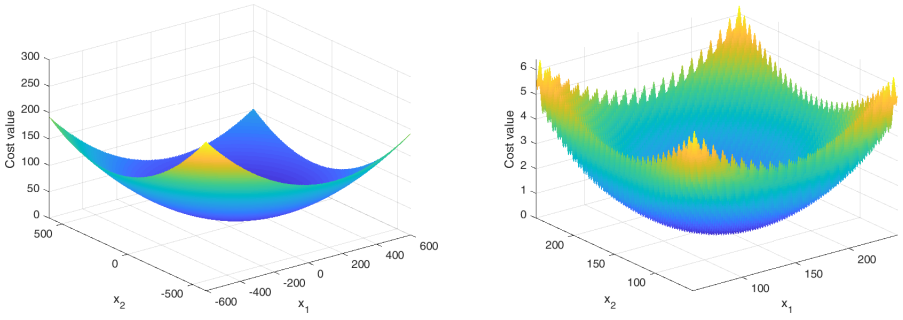
The Griewank function (Figure A.2) is a multimodal benchmark characterized by an high frequency oscillation around a parabolic trend.

It is mathematically defined as:

$$f(\mathbf{x}) = 1 + \sum_{i=1}^{M} (x_i - x_0)^2/4000 - \prod_{i=1}^{M} \cos \frac{x_i - x_0}{\sqrt{i}} \tag{A.4}$$

where $x_0 = 150$ and its domain is:

$$-600 \leq x_i \leq 600 \tag{A.5}$$



(a) Entrie search domain (b) Zoom around the global optimum

Figure A.2: Griewank function

Penalty 1 function

This function, shown in Figure A.3, emulates the behaviour of a penalty definition of a constrained function: the function value drastically grows out of a specific region that emulates the feasible part of the search space. The function is multimodal in all its domain.

This function is characterized by a very high differences in its values in the domain, challenging some selection operators.

The function can be expressed as the sum of two terms, where $g(\mathbf{x})$ it the penalization term:

$$f(\mathbf{x}) = g(\mathbf{x}) + h(\mathbf{x}) \tag{A.6}$$

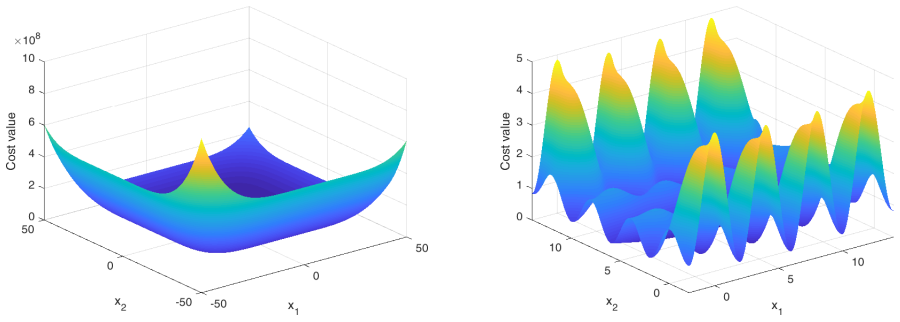
$$g(\mathbf{x}) = \sum_{i=1}^{M} \begin{cases} 100(x_i - x_0 - 10)^4 & (x_i - x_0) > 10 \\ 100(-x_i - x_0 - 10)^4 & (x_i - x_0) < -10 \\ 0 & -10 < (x_i - x_0) < 10 \end{cases} \tag{A.7}$$

$$h(\mathbf{x}) = 10\sin^2\left[\pi\left(1 + \frac{x_1 - x_0 + 1}{4}\right)\right] + \left(1 + \frac{x_M - x_0 + 1}{4} - 1\right)^2 \cdot \frac{\pi}{30} +$$
$$+ \sum_{i=1}^{M-1}\left[1 + \frac{x_i - x_0 + 1}{4} - 1\right]^2 \cdot \left(1 + 10\sin^2\left[\pi\left(1 + \frac{x_{i+1} - x_0 + 1}{4}\right)\right]\right)\frac{\pi}{30}$$

where $x_0 = 7$ its the global function minimum.

Its search domain is:

$$-50 \leq x_i \leq 50 \tag{A.8}$$



(a) Entrie search domain (b) Zoom around the global optimum

Figure A.3: Penalty 1 function

Penalty 2 function

This second penalty function (Figure A.4) have definition similar to Penalty 1, but the penalization term is much more relevant, the feasible region is smaller, and the function non-linearities higher.

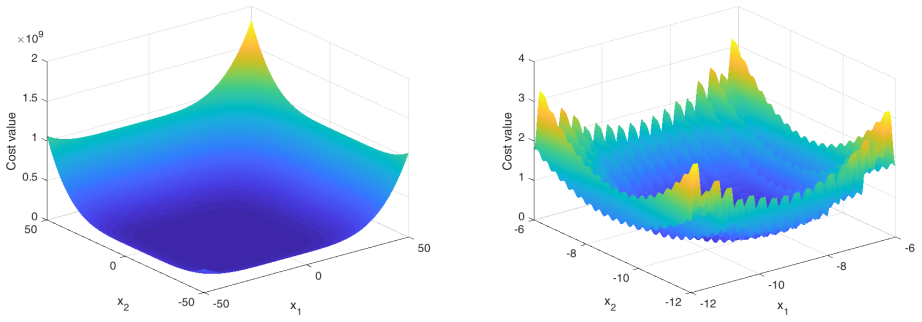
$$f(\mathbf{x}) = g(\mathbf{x}) + h(\mathbf{x}) \tag{A.9}$$

$$g(\mathbf{x}) = \sum_{i=1}^{M} \begin{cases} 100(x_i - x_0 - 5)^4 & (x_i - x_0) > 5 \\ 100(-x_i - x_0 - 5)^4 & (x_i - x_0) < -5 \\ 0 & (x_i - x_0) = 5 \end{cases} \tag{A.10}$$

$$h(\mathbf{x}) = \frac{1}{10}\sin^2\left[3\pi\left(x_1 - x_0\right)\right] + \left(x_M - x_0 - 1\right)^2 \cdot \left(1 + \sin^2\left[2\pi\left(x_M - x_0\right)\right]\right) + \\ + \sum_{i=1}^{M-1}\left(x_i - x_0 - 1\right)^2 \cdot \left[1 + \sin^2\left[3\pi(x_i - x_0)\right]\right]$$

where $x_0 = -10$ is the global minimum. Its domain is:

$$-50 \leq x_i \leq 50 \tag{A.11}$$



(a) Entrie search domain (b) Zoom around the global optimum

Figure A.4: Penalty 2 function

Quartic function

The Quartic function (Figure A.5) is a single-modal function defined as:

$$f(\mathbf{x}) = \sum_{i=1}^{M} i \cdot (x_i - x_0)^4 \tag{A.12}$$

where $x_0 = 0.4$ is the function minimum. The Quartic domain is:

$$-1.28 \leq x_i \leq 1.28 \tag{A.13}$$

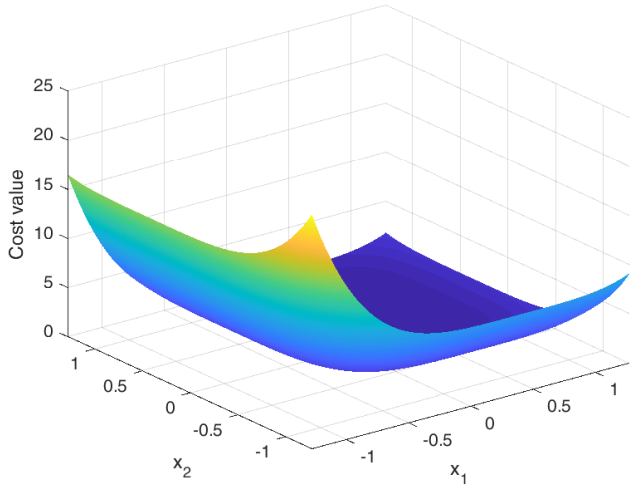


Figure A.5: Quartic function

Rastrigin function

The Rastrigin function (Figure A.6) is a multimodal function with a low-frequency oscillation. It defined as:

$$f(\mathbf{x}) = \sum_{i=1}^{M} (x_i - x_0)^2 - 10\cos\left[2\pi(x_i - x_0)\right] \tag{A.14}$$

where $x_0 = -\pi$ is the global minimum. Its domain is:

$$-5.12 \leq x_i \leq 5.12 \tag{A.15}$$

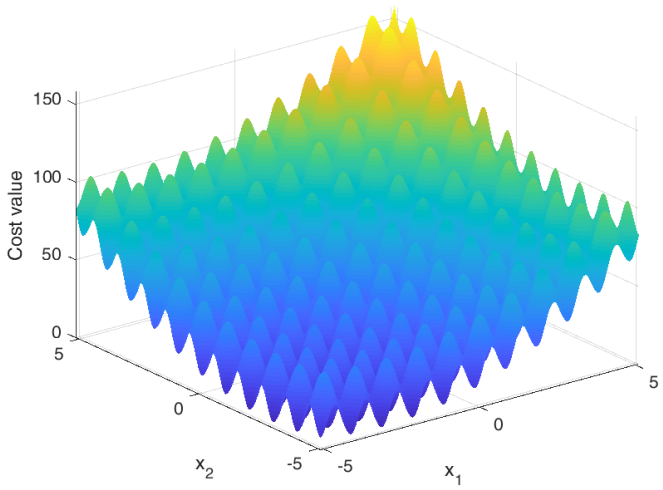


Figure A.6: Rastrigin function

Rosenbrock function

The Rosenbrock function (Figure A.7) is a single-modal function generally adopted for testing derivative-based algorithms for its flat region. It is defined as:

$$f(\mathbf{x}) = \sum_{i=1}^{M-1} (x(i+1) - x_i)^2 + (x_i - 1)^2 \tag{A.16}$$

Its domain is:

$$-2.048 \leq x_i \leq 2.048 \tag{A.17}$$

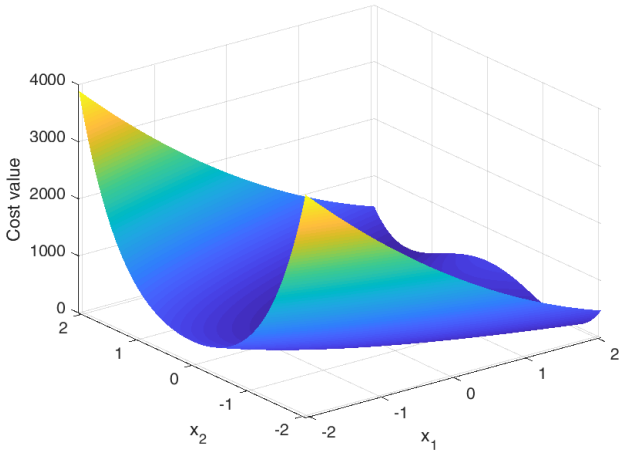


Figure A.7: Rosenbrock function

Schwefel-226 function

The Schwefel-226 (Figure A.8) is a multimodal function with no general trend. The gloabl minimum is close to the search domain boundary and quite far from the other good local minima. It is defined as:

$$f(\mathbf{x}) = 418.9829 \cdot M + \sum_{i=1}^{M} x_i \cdot \sin\sqrt{|x_i|} \tag{A.18}$$

Its domain is:

$$-512 \leq x_i \leq 512 \tag{A.19}$$

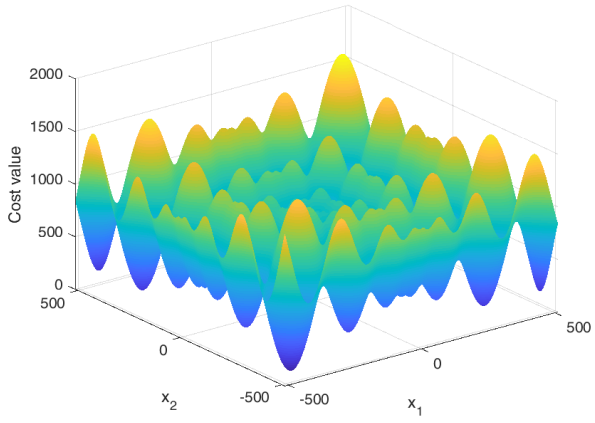


Figure A.8: Schwefel-226 function

Schwefel-12 function

The Schwefel-12 (Figure A.9) is a single modal function defined as:

$$f(\mathbf{x}) = \sum_{i=1}^{M} \sum_{j=1}^{i} x_j - x_0 \tag{A.20}$$

where $x_0 = 10$ is the position of the global minimum. Its domain is:

$$-65.536 \leq x_i \leq 65.536 \tag{A.21}$$

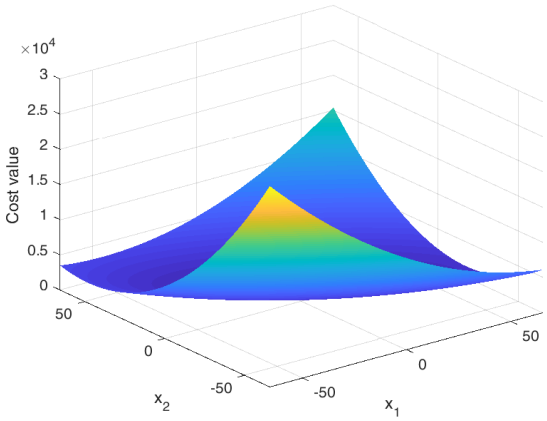


Figure A.9: Schwefel-12 function

Schwefel-222 function

The Schwefel-222 (Figure A.10) is defined as:

$$f(\mathbf{x}) = \sum_{i=1}^{M} |x_i - X_0| + \prod_{i=1}^{M} |x_i - X_0| \tag{A.22}$$

where and $x_0 = 1$.

Its domain is:

$$-10 \leq x_i \leq 10 \tag{A.23}$$

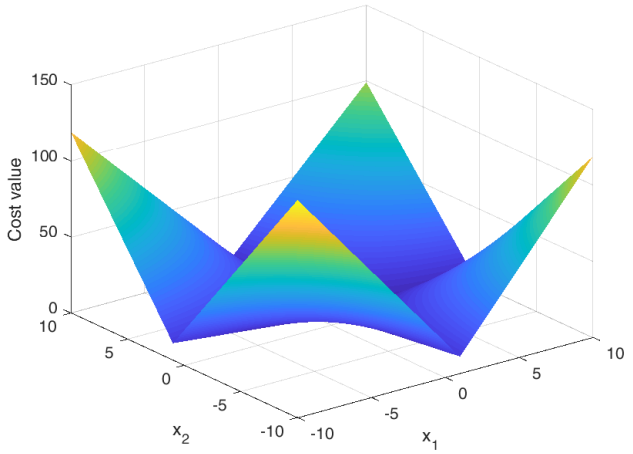


Figure A.10: Schwefel-222 function

Schwefel-222 function

The Schwefel-221 (Figure A.11) is a uni-modal function defined as:

$$f(\mathbf{x}) = \max |x_i - x_0| \tag{A.24}$$

where $x_0 = -20$. Its domain is:

$$-100 \leq x_i \leq 100 \tag{A.25}$$

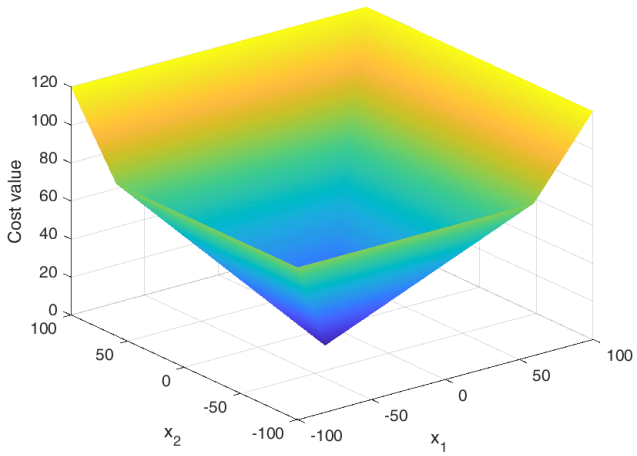


Figure A.11: Schwefel-221 function

SincN function

The SincN function (Figure A.12) is one of the two possible definition of the cardinal sine. It is a multi modal function in which there is a very large flat area and a quite narrow minimum.

Its formulation is:

$$f(\mathbf{x}) = 1 - \prod_{i=1}^{M} |\sin[\pi \cdot (x_i - x_0)]| \tag{A.26}$$

where $x_0 = 3$ is position of the global minimum.

Its search domain is:

$$0 \leq x_i \leq 10 \tag{A.27}$$

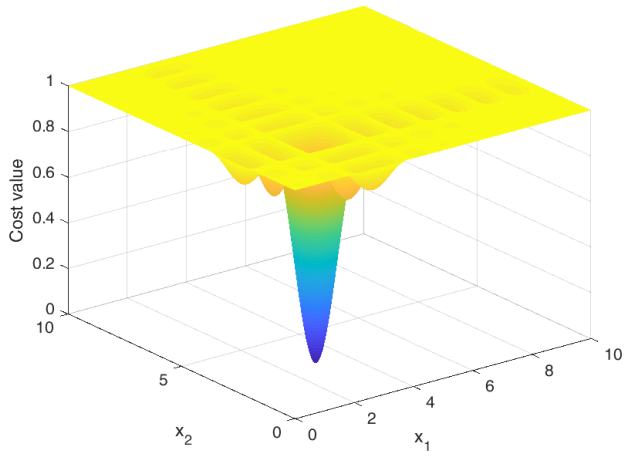


Figure A.12: SincN function

Sinc function

The Sinc function (Figure A.13) is the second definition of the candinal sine. It is characterized by an axial symmetry and a set of important local minima.

Its mathematical formulation is:

$$f(\mathbf{x}) = 1 - \frac{\sin|\mathbf{x} - x_0|}{|\mathbf{x} - x_0|} \tag{A.28}$$

where $x_0 = 5$ and the search domain is:

$$-20 \leq x_i \leq 20 \tag{A.29}$$

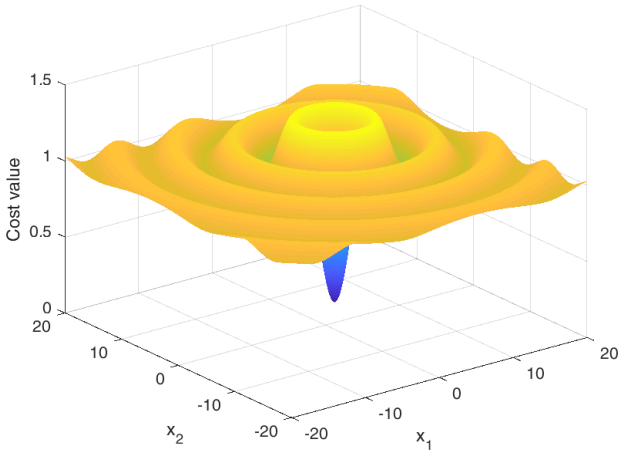


Figure A.13: Sinc function

Sphere function

The Sphere function (Figure A.14) is a convex function defined as:

$$f(\mathbf{x}) = \sum_{i=1}^{M} (x_i - x_0)^2 \tag{A.30}$$

where $x_0 = 1$. Its domain is:

$$-5.12 \leq x_i \leq 5.12 \tag{A.31}$$

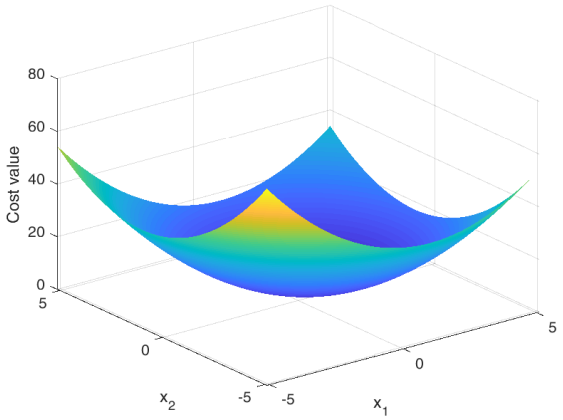


Figure A.14: Sphere function

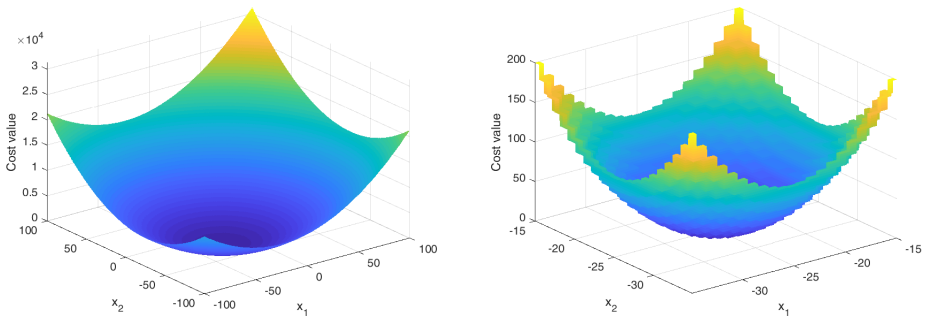
Step function

The Step function (Figure A.15) is a discontinuous function in which the derivative, when it is defined, it is always zero. Its formulation is:

$$f(\mathbf{x}) = \sum_{i=1}^{M} \left(\left\lfloor x_i - x_0 + \frac{1}{2} \right\rfloor \right)^2 \tag{A.32}$$

where $x_0 = -25$. Its domain is:

$$-100 \leq x_i \leq 100 \tag{A.33}$$



(a) Entrie search domain (b) Zoom around the global optimum

Figure A.15: Step function

List of scientific publications of my PhD project

During my PhD research projects, several papers have been published and presented in international journals and conferences.

Journal papers:

- Alessandro Niccolai, Francesco Grimaccia, Marco Mussetta, Riccardo Zich, *Optimal Task Allocation in Wireless Sensor Networks by Means of Social Network Optimization*, Mathematics (2019).

- Emanuele Ogliari, Alessandro Niccolai, Sonia Leva, Riccardo Zich, *Computational intelligence techniques applied to the day ahead PV output power forecast: PHANN, SNO and mixed*, Energies (2018).

- Francesco Grimaccia, Giambattista Gruosso, Marco Mussetta, Alessandro Niccolai, Riccardo E. Zich, *Design of tubular permanent magnet generators for vehicle energy harvesting by means of social network optimization*, IEEE Transactions on Industrial Electronics (2017).

- Francesco Grimaccia, Sonia Leva, Alessandro Niccolai, *PV plant digital mapping for modules' defects detection by unmanned aerial vehicles*, IET Renewable Power Generation (2017).

Book chapters:

- Francesco Grimaccia, Marco Mussetta, Alessandro Niccolai, Riccardo Zich, *Modelling of interaction in Swarm Intelligence focused on Particle Swarm Optimization and Social Networks Optimization*, in Swarm Intelligence: Volume 1: Principles, Current Algorithms and Methods.

- Michele Beccaria, Ho Man Linh, Andrea Massaccesi, Alessandro Niccolai, Nguyen Huu Trung, Nguyen Khac Kiem, Riccardo Zich, Paola

Pirinoli, *Optimization strategies for efficient antenna design*, Springer Verlag, 2019 (Accepted for publication).

Conference papers:

- Alessandro Niccolai, Alessandro Gandelli, Francesco Grimaccia, Riccardo Zich, Sonia Leva, *Overview on Photovoltaic Inspections Procedure by means of Unmanned Aerial Vehicles*, IEEE Milan PowerTech, 2019.

- Alessandro Niccolai, Riccardo Zich, Michele Beccaria, Paola Pirinoli, *SNO Based Optimization for Shaped Beam Reflectarray Antennas*, IEEE European Conference on Antennas and Propagation, 2019.

- Michele Beccaria, Andrea Massaccesi, Paola Pirinoli, Alessandro Niccolai, Riccardo Zich, *SNO and mBBO Optimization Methods for beam scanning Reflectarray Antennas*, IEEE International Symposium on Antennas and Propagation, 2019.

- Francesco Grimaccia, Marco Mussetta, Alessandro Niccolai, Riccardo E. Zich, *Optimal computational distribution of social network optimization in wireless sensor networks*, IEEE Congress on Evolutionary Computation, 2018.

- Alberto Dolara, Sonia Leva, Giampaolo Manzolini, Alessandro Niccolai, Luca Votta, *Impact of Cell Microcracks Size and Spatial Distribution on Output Power of PV Modules*, IEEE International Conference on Environment and Electrical Engineering, 2018.

- Francesco Grimaccia, Sonia Leva, Alessandro Niccolai, Giulio Cantoro, *Assessment of PV Plant Monitoring System by Means of Unmanned Aerial Vehicles*, IEEE International Conference on Environment and Electrical Engineering, 2018.

- Alessandro Niccolai, Alberto Dolara, Francesco Grimaccia, *Analysis of Photovoltaic Five-Parameter Model*, International Conference on Smart Systems and Technologies, 2018.

- Francesco Grimaccia, Marco Mussetta, Alessandro Niccolai, Riccardo E. Zich, *Comparison of Binary Evolutionary Algorithms for Optimization of Thinned Array Antennas*, IEEE Congress on Evolutionary Computation, 2018.

- Alessandro Niccolai, Xiaotian Pan, Paola Pirinoli, Fan Yang, Riccardo E Zich, Shenheng Xu, *Flat beam optimization of 1-bit Reflectarray by means of Social Network Optimization*, IEEE International Symposium on Antennas and Propagation, 2018.

- Michele Beccaria, Paola Pirinoli, Alessandro Niccolai, Riccardo Zich *Application of Social Network Optimization to shaped beam Transmitarray Antennas*, IEEE International Symposium on Antennas and Propagation, 2018.

- Francesco Grimaccia, Marco Mussetta, Alessandro Niccolai, Paola Pirinoli, Riccardo E. Zich, *Effect of introduction of hypotheses in Antenna Optimization: Thinned array test case*, IEEE International Symposium on Antennas and Propagation, 2018.

- Francesco Grimaccia, Marco Mussetta, Alessandro Niccolai, Paola Pirinoli, Riccardo E. Zich, *Leadership-based algorithm for planar array optimization*, IEEE International Symposium on Antennas and Propagation, 2017.

- Francesco Grimaccia, Marco Mussetta, Alessandro Niccolai, Paola Pirinoli, Riccardo E. Zich, *SNO multi-objective implementation for sparse array optimization*, IEEE International Conference on Electromagnetics in Advanced Applications, 2017.

- Francesco Grimaccia, Giambattista Gruosso, Marco Mussetta, Alessandro Niccolai, Riccardo E. Zich, *Optimized linear generator for vehicle energy harvesting by social network optimization algorithm*, IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society, 2017.

Bibliography

[1] D. H. Wolpert, W. G. Macready *et al.*, "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation*, vol. 1, no. 1, pp. 67–82, 1997.

[2] Y. Dhebar and K. Deb, "Design of an adaptive push-repel operator for enhancing convergence in genetic algorithms," in *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2018, pp. 696–703.

[3] Y. Rahmat-Samii and E. Michielssen, "Electromagnetic optimization by genetic algorithms," *Microwave Journal*, vol. 42, no. 11, pp. 232–232, 1999.

[4] H. Ramadan, A. Bendary, and S. Nagy, "Particle swarm optimization algorithm for capacitor allocation problem in distribution systems with wind turbine generators," *International Journal of Electrical Power & Energy Systems*, vol. 84, pp. 143–152, 2017.

[5] P. Virtič, M. Vražić, and G. Papa, "Design of an axial flux permanent magnet synchronous machine using analytical method and evolutionary optimization," *IEEE Transactions on Energy Conversion*, vol. 31, no. 1, pp. 150–158, 2015.

[6] R. Nason, *It's Not Complicated: The Art and Science of Complexity in Business*. University of Toronto Press, 2017.

[7] K. Deb, *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, 2001, vol. 16.

[8] W. Lenwari, M. Sumner, and P. Zanchetta, "The use of genetic algorithms for the design of resonant compensators for active filters," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 8, pp. 2852–2861, Aug 2009.

[9] B. Ufnalski, A. Kaszewski, and L. M. Grzesiak, "Particle swarm optimization of the multioscillatory lqr for a three-phase four-wire voltage-source inverter with an lc output filter," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 1, pp. 484–493, Jan 2015.

[10] D. Y. Li, W. C. Cai, P. Li, Z. J. Jia, H. J. Chen, and Y. D. Song, "Neuroadaptive variable speed control of wind turbine with wind speed estimation," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 12, pp. 7754–7764, Dec 2016.

[11] M. Q. Duong, F. Grimaccia, S. Leva, M. Mussetta, and R. Zich, "Improving lvrt characteristics in variable-speed wind power generation by means of fuzzy logic," in *2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, July 2014, pp. 332–337.

[12] Y. Rahmat-Samii, J. M. Kovitz, and H. Rajagopalan, "Nature-inspired optimization techniques in communication antenna designs," *Proceedings of the IEEE*, vol. 100, no. 7, pp. 2132–2144, 2012.

[13] D. Simon, "Biogeography-based optimization," *IEEE transactions on evolutionary computation*, vol. 12, no. 6, pp. 702–713, 2008.

[14] U. Singh, H. Kumar, and T. S. Kamal, "Design of yagi-uda antenna using biogeography based optimization," *IEEE Transactions on Antennas and Propagation*, vol. 58, no. 10, pp. 3375–3379, 2010.

[15] P. Pirinoli, A. Massaccesi, and M. Beccaria, "Application of the m m c n-bbo algorithms to the optimization of antenna problems," in *Electromagnetics in Advanced Applications (ICEAA), 2017 International Conference on.* IEEE, 2017, pp. 1850–1854.

[16] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2008.

[17] A. M. Sutton, M. Lunacek, and L. D. Whitley, "Differential evolution and non-separability: using selective pressure to focus search," in *Proceedings of the 9th annual conference on Genetic and evolutionary computation.* ACM, 2007, pp. 1428–1435.

[18] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz, "Boa: The bayesian optimization algorithm," in *Proceedings of the 1st Annual Conference on*

Genetic and Evolutionary Computation-Volume 1. Morgan Kaufmann Publishers Inc., 1999, pp. 525–532.

[19] B. V. Ha, P. Pirinoli, R. E. Zich, M. Mussetta, and F. Grimaccia, "Modified bayesian optimization algorithm for sparse linear antenna design," *Progress In Electromagnetics Research B*, vol. 54, pp. 385–405, 2013.

[20] W. Khatib and P. J. Fleming, "The stud ga: a mini revolution?" in *International Conference on Parallel Problem Solving from Nature*. Springer, 1998, pp. 683–691.

[21] S. Selleri, M. Mussetta, P. Pirinoli, R. E. Zich, and L. Matekovits, "Differentiated meta-pso methods for array optimization," *IEEE Transactions on Antennas and Propagation*, vol. 56, no. 1, pp. 67–75, 2008.

[22] Q. Kang and H. He, "A novel discrete particle swarm optimization algorithm for meta-task assignment in heterogeneous computing systems," *Microprocessors and microsystems*, vol. 35, no. 1, pp. 10–17, 2011.

[23] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, vol. 5. IEEE, 1997, pp. 4104–4108.

[24] Z. D. Zaharis and T. V. Yioultsis, "A novel adaptive beamforming technique applied on linear antenna arrays using adaptive mutated boolean pso," *Progress In Electromagnetics Research*, vol. 117, pp. 165–179, 2011.

[25] W. L. Stutzman and G. A. Thiele, *Antenna theory and design*. John Wiley & Sons, 2012.

[26] J. Huang, "Reflectarray antenna," *Encyclopedia of RF and Microwave Engineering*, 2005.

[27] D. Pozar, S. Targonski, and R. Pokuls, "A shaped-beam microstrip patch reflectarray," *IEEE Transactions on Antennas and Propagation*, vol. 47, no. 7, pp. 1167–1173, 1999.

[28] P. Nayeri, F. Yang, and A. Z. Elsherbeni, "Design and experiment of a single-feed quad-beam reflectarray antenna," *IEEE Transactions on Antennas and Propagation*, vol. 60, no. 2, pp. 1166–1171, 2011.

[29] ——, "Beam-scanning reflectarray antennas: A technical overview and state of the art." *IEEE Antennas and Propagation Magazine*, vol. 57, no. 4, pp. 32–47, 2015.

[30] ——, "Bifocal design and aperture phase optimizations of reflectarray antennas for wide-angle beam scanning performance," *IEEE Transactions on Antennas and Propagation*, vol. 61, no. 9, pp. 4588–4597, 2013.

[31] I. Mauser, J. Müller, F. Allerding, and H. Schmeck, "Adaptive building energy management with multiple commodities and flexible evolutionary optimization," *Renewable Energy*, vol. 87, pp. 911–921, 2016.

[32] S. Kurt, H. U. Yildiz, M. Yigit, B. Tavli, and V. C. Gungor, "Packet size optimization in wireless sensor networks for smart grid applications," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 3, pp. 2392–2401, 2016.

[33] B. Zhu, X. Xia, and Z. Wu, "Evolutionary game theoretic demand-side management and control for a class of networked smart grid," *Automatica*, vol. 70, pp. 94–100, 2016.

[34] V. N. Coelho, I. M. Coelho, B. N. Coelho, A. J. Reis, R. Enayatifar, M. J. Souza, and F. G. Guimarães, "A self-adaptive evolutionary fuzzy model for load forecasting problems on smart grid environment," *Applied Energy*, vol. 169, pp. 567–584, 2016.

[35] K. R. Rao, T. Sunderan, and M. R. Adiris, "Performance and design optimization of two model based wave energy permanent magnet linear generators," *Renewable energy*, vol. 101, pp. 196–203, 2017.

[36] P. Rocca, G. Oliveri, and A. Massa, "Differential evolution as applied to electromagnetics," *IEEE Antennas and Propagation Magazine*, vol. 53, no. 1, pp. 38–49, 2011.

[37] M. Salucci, L. Tenuti, G. Gottardi, A. Hannan, and A. Massa, "System-by-design method for efficient linear array miniaturisation through low-complexity isotropic lenses," *Electronics Letters*, vol. 55, no. 8, pp. 433–434, 2019.

[38] G. Oliveri, F. Viani, N. Anselmi, and A. Massa, "Synthesis of multilayer waim coatings for planar-phased arrays within the system-by-design framework," *IEEE Transactions on Antennas and Propagation*, vol. 63, no. 6, pp. 2482–2496, 2015.

[39] B. Liu, S. Koziel, and Q. Zhang, "A multi-fidelity surrogate-model-assisted evolutionary algorithm for computationally expensive optimization problems," *Journal of computational science*, vol. 12, pp. 28–37, 2016.

[40] P. Singh, M. Rossi, I. Couckuyt, D. Deschrijver, H. Rogier, and T. Dhaene, "Constrained multi-objective antenna design optimization using surrogates," *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, vol. 30, no. 6, p. e2248, 2017.

[41] P. K. Roy, S. P. Ghoshal, and S. S. Thakur, "Combined economic and emission dispatch problems using biogeography-based optimization," *Electrical Engineering*, vol. 92, no. 4, pp. 173–184, 2010.

[42] M. Alonso, H. Amaris, and C. Alvarez-Ortega, "Integration of renewable energy sources in smart grids by means of evolutionary optimization algorithms," *Expert Systems with Applications*, vol. 39, no. 5, pp. 5513 – 5522, 2012.

[43] K. B. Lee, H. Myung, and J. H. Kim, "Online multiobjective evolutionary approach for navigation of humanoid robots," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 9, pp. 5586–5597, Sept 2015.

[44] A. A. Menezes and I. V. Kolmanovsky, "Energy and power management in a series hybrid electric vehicle using selective evolutionary generation," in *53rd IEEE Conference on Decision and Control*, Dec 2014, pp. 3310–3315.

[45] T. Marcic, B. Stumberger, and G. Stumberger, "Differential-evolution-based parameter identification of a line-start ipm synchronous motor," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 11, pp. 5921–5929, Nov 2014.

[46] A. Pirisi, M. Mussetta, F. Grimaccia, and R. E. Zich, "Novel speed-bump design and optimization for energy harvesting from traffic," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1983–1991, Dec 2013.

[47] A. C. Zavoianu, E. Lughofer, G. Bramerdorfer, W. Amrhein, and S. Saminger-Platz, "A surrogate-based strategy for multi-objective tolerance analysis in electrical machine design," in *2015 17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, Sept 2015, pp. 195–203.

[48] L. Guo, H. Zhang, M. Galea, J. Li, and C. Gerada, "Multiobjective optimization of a magnetically levitated planar motor with multilayer windings," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 6, pp. 3522–3532, June 2016.

[49] A. Fatemi, D. M. Ionel, N. A. O. Demerdash, and T. W. Nehl, "Optimal design of ipm motors with different cooling systems and winding configurations," *IEEE Transactions on Industry Applications*, vol. 52, no. 4, pp. 3041–3049, July 2016.

[50] Y. Wang, D. M. Ionel, V. Rallabandi, M. Jiang, and S. J. Stretz, "Large-scale optimization of synchronous reluctance machines using ce-fea and differential evolution," *IEEE Transactions on Industry Applications*, vol. 52, no. 6, pp. 4699–4709, Nov 2016.

[51] M. Sreejeth, M. Singh, and P. Kumar, "Particle swarm optimisation in efficiency improvement of vector controlled surface mounted permanent magnet synchronous motor drive," *IET Power Electronics*, vol. 8, no. 5, pp. 760–769, 2015.

[52] M. Calvini, M. Carpita, A. Formentini, and M. Marchesoni, "Pso-based self-commissioning of electrical motor drives," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 2, pp. 768–776, Feb 2015.

[53] A. Canova, F. Freschi, G. Gruosso, and B. Vusini, "Genetic optimisation of radial eddy current couplings," *COMPEL - The International Journal for Computation and Mathematics in Electrical and Electronic Engineering*, vol. 24, no. 3, pp. 767–783, 2005.

[54] A. Canova, G. Gruosso, and M. Repetto, "Magnetic design optimization and objective function approximation," *IEEE Transactions on Magnetics*, vol. 39, no. 5, pp. 2154–2162, Sept 2003.

[55] A. Pirisi, M. Mussetta, G. Gruosso, and R. Zich, "An optimized three phase tpm-lig for marine applications," 2010, pp. 1712–1717.

[56] ——, "Optimization of a linear generator for sea-wave energy conversion by means of a hybrid evolutionary algorithm," 2010.

[57] F. Bizzozero, M. Giassi, G. Gruosso, S. Bozzi, and G. Passoni, "Dynamic model, parameter extraction, and analysis of two topologies of a tubular linear generator for seawave energy production," 2014, pp. 433–438.

[58] H. Huang, A. Hoorfar, and S. Lakhani, "A comparative study of evolutionary programming, genetic algorithms and particle swarm optimization in antenna design," in *Antennas and Propagation Society International Symposium, 2007 IEEE*. IEEE, 2007, pp. 1609–1612.

[59] A. H. Wright *et al.*, "Genetic algorithms for real parameter optimization," *Foundations of genetic algorithms*, vol. 1, pp. 205–218, 1991.

[60] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[61] J. Robinson and Y. Rahmat-Samii, "Particle swarm optimization in electromagnetics," *IEEE transactions on antennas and propagation*, vol. 52, no. 2, pp. 397–407, 2004.

[62] N. Jin and Y. Rahmat-Samii, "Advances in particle swarm optimization for antenna designs: Real-number, binary, single-objective and multiobjective implementations," *IEEE Transactions on Antennas and Propagation*, vol. 55, no. 3, pp. 556–567, 2007.

[63] S.-W. Qu and G.-B. Wu, "Reflectarray antenna design," in *Electromagnetics in Advanced Applications (ICEAA), 2017 International Conference on*. IEEE, 2017, pp. 1839–1841.

[64] M. Ruello, A. Niccolai, F. Grimaccia, M. Mussetta, and R. E. Zich, "Black-hole pso and sno for electromagnetic optimization," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*. IEEE, 2014, pp. 1912–1916.

[65] N. N. Pathak, G. K. Mahanti, S. K. Singh, J. K. Mishra, and A. Chakraborty, "Synthesis of thinned planar circular array antennas using modified particle swarm optimization," *Progress In Electromagnetics Research Letters*, vol. 12, pp. 87–97, 2009.

[66] A. Modiri and K. Kiasaleh, "Modification of real-number and binary pso algorithms for accelerated convergence," *IEEE Transactions on Antennas and Propagation*, vol. 59, no. 1, pp. 214–224, 2011.

[67] A. Niccolai, R. Zich, M. Beccaria, and P. Pirinoli, "Sno based optimization for shaped beam reflectarray antennas," in *2019 13th European Conference on Antennas and Propagation (EuCAP)*. IEEE, 2019, pp. 1–4.

[68] F. Grimaccia, M. Mussetta, A. Niccolai, and R. E. Zich, "Optimal computational distribution of social network optimization in wireless sensor networks," in *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2018, pp. 1–7.

[69] F. Grimaccia, M. Mussetta, A. Niccolai, and R. Zich, "Modelling of interaction in Swarm Intelligence focused on Particle Swarm Optimization and Social Networks Optimization," in *Swarm Intelligence: Principles, Current Algorithms and Methods*, ser. Control, Robotics and Sensors, Y. Tan, Ed. IET, 2018, ch. 19.

[70] E. Ogliari, A. Niccolai, S. Leva, and R. Zich, "Computational intelligence techniques applied to the day ahead pv output power forecast: phann, sno and mixed," *Energies*, vol. 11, no. 6, p. 1487, 2018.

[71] F. Grimaccia, G. Gruosso, M. Mussetta, A. Niccolai, and R. E. Zich, "Design of tubular permanent magnet generators for vehicle energy harvesting by means of social network optimization," *IEEE Transactions on Industrial Electronics*, 2017.

[72] A. Niccolai, F. Grimaccia, M. Mussetta, and R. E. Zich, "Reflectarray optimization by means of sno and pso," in *Antennas and Propagation (APSURSI), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 781–782.

[73] M. Beccaria, P. Pirinoli, A. Niccolai, and R. Zich, "Application of social network optimization to shaped beam transmitarray antennas," in *2018 IEEE International Symposium on Antennas and Propagation & USNC/URSI National Radio Science Meeting*. IEEE, 2018, pp. 2203–2204.

[74] G. B. Dantzig, *Linear programming and extensions*. Princeton university press, 1998.

[75] W. W. Hager and H. Zhang, "A survey of nonlinear conjugate gradient methods," *Pacific journal of Optimization*, vol. 2, no. 1, pp. 35–58, 2006.

[76] D. Simon, *Evolutionary optimization algorithms*. John Wiley & Sons, 2013.

[77] Ş. İ. Birbil, S.-C. Fang, and R.-L. Sheu, "On the convergence of a population-based global optimization algorithm," *Journal of global optimization*, vol. 30, no. 2-3, pp. 301–318, 2004.

[78] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Machine learning*, vol. 3, no. 2, pp. 95–99, 1988.

[79] A. Bertoni and M. Dorigo, "Implicit parallelism in genetic algorithms," *Artificial Intelligence*, vol. 61, no. 2, pp. 307–314, 1993.

[80] M. Dorigo and L. M. Gambardella, "Ant colonies for the travelling salesman problem," *biosystems*, vol. 43, no. 2, pp. 73–81, 1997.

[81] Z.-H. Han and K.-S. Zhang, "Surrogate-based optimization," in *Real-world applications of genetic algorithms*. InTech, 2012.

[82] E. J. Hartman, J. D. Keeler, and J. M. Kowalski, "Layered neural networks with gaussian hidden units as universal approximations," *Neural computation*, vol. 2, no. 2, pp. 210–215, 1990.

[83] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE transactions on pattern analysis and machine intelligence*, vol. 12, no. 10, pp. 993–1001, 1990.

[84] A. I. Forrester and A. J. Keane, "Recent advances in surrogate-based optimization," *Progress in aerospace sciences*, vol. 45, no. 1-3, pp. 50–79, 2009.

[85] Z. Zhou, Y.-S. Ong, P. B. Nair, A. J. Keane, and K. Y. Lum, "Combining global and local surrogate models to accelerate evolutionary optimization," *IEEE Transactions On Systems, Man and Cybernetics-Part C*, vol. 37, no. 1, pp. 66–76, 2007.

[86] T. C. Bailey and A. C. Gatrell, *Interactive spatial data analysis*. Longman Scientific & Technical Essex, 1995, vol. 413.

[87] H. V. Henderson and S. R. Searle, "On deriving the inverse of a sum of matrices," *Siam Review*, vol. 23, no. 1, pp. 53–60, 1981.

[88] R. Dimitrakopoulos and X. Luo, "Spatiotemporal modelling: covariances and ordinary kriging systems," in *Geostatistics for the next century*. Springer, 1994, pp. 88–93.

[89] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. K. Tucker, "Surrogate-based analysis and optimization," *Progress in aerospace sciences*, vol. 41, no. 1, pp. 1–28, 2005.

[90] A. Massa and M. Salucci, "Dealing with complexity in electromagnetics through the system-by-design paradigm-new strategies and applications to the design of airborne radomes," in *2018 IEEE International Symposium on Antennas and Propagation & USNC/URSI National Radio Science Meeting*. IEEE, 2018, pp. 529–530.

[91] J. A. Caballero and I. E. Grossmann, "An algorithm for the use of surrogate models in modular flowsheet optimization," *AIChE journal*, vol. 54, no. 10, pp. 2633–2650, 2008.

[92] M. Salucci, L. Poli, D. Masotti, A. Costanzo, and P. Rocca, "Pso-driven synthesis of realistic time modulated arrays with optimal instantaneous directivity through a system-by-design implementation," in *2018 IEEE International Symposium on Antennas and Propagation & USNC/URSI National Radio Science Meeting*. IEEE, 2018, pp. 1591–1592.

[93] A. Niccolai, F. Grimaccia, M. Mussetta, P. Pirinoli, V. H. Bui, and R. E. Zich, "Social network optimization for microwave circuits design," *Progress In Electromagnetics Research C*, vol. 58, pp. 51–60, 2015.

[94] A. Niccolai, F. Grimaccia, M. Mussetta, and R. Zich, "Optimal task allocation in wireless sensor networks by means of social network optimization," *Mathematics*, vol. 7, no. 4, p. 315, 2019.

[95] J. G. Lee, P. Antoniadis, and K. Salamatian, "Faving reciprocity in content sharing communities: A comparative analysis of flickr and twitter," in *Advances in Social Networks Analysis and Mining (ASONAM), 2010 International Conference on*. IEEE, 2010, pp. 136–143.

[96] J. J. Cameron, C. K.-S. Leung, and S. K. Tanbeer, "Finding strong groups of friends among friends in social networks," in *Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on*. IEEE, 2011, pp. 824–831.

[97] D. Centola, "The spread of behavior in an online social network experiment," *science*, vol. 329, no. 5996, pp. 1194–1197, 2010.

[98] D. Centola and M. Macy, "Complex contagions and the weakness of long ties," *American journal of Sociology*, vol. 113, no. 3, pp. 702–734, 2007.

[99] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan, "Group formation in large social networks: membership, growth, and evolution,"

in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 44–54.

[100] A. S. Acar and M. Polonsky, "Online social networks and insights into marketing communications," *Journal of Internet Commerce*, vol. 6, no. 4, pp. 55–72, 2007.

[101] R. Kumar, J. Novak, and A. Tomkins, "Structure and evolution of online social networks," in *Link mining: models, algorithms, and applications*. Springer, 2010, pp. 337–357.

[102] J. Brown, A. J. Broderick, and N. Lee, "Word of mouth communication within online communities: Conceptualizing the online social network," *Journal of interactive marketing*, vol. 21, no. 3, pp. 2–20, 2007.

[103] M. Jamali and M. Ester, "Using a trust network to improve top-n recommendation," in *Proceedings of the third ACM conference on Recommender systems*. ACM, 2009, pp. 181–188.

[104] K. Shu, H. R. Bernard, and H. Liu, "Studying fake news via network analysis: Detection and mitigation," *arXiv preprint arXiv:1804.10233*, 2018.

[105] E. Mossel and S. Roch, "Submodularity of influence in social networks: From local to global," *SIAM Journal on Computing*, vol. 39, no. 6, pp. 2176–2188, 2010.

[106] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," *Theory of Computing*, vol. 11, no. 4, pp. 105–147, 2015. [Online]. Available: http://www.theoryofcomputing.org/articles/v011a004

[107] W. Quattrociocchi, A. Scala, and C. R. Sunstein, "Echo chambers on facebook," 2016.

[108] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decision support systems*, vol. 43, no. 2, pp. 618–644, 2007.

[109] D. Gambetta *et al.*, "Can we trust trust," *Trust: Making and breaking cooperative relations*, vol. 13, pp. 213–237, 2000.

[110] R. Falcone and C. Castelfranchi, "Social trust: A cognitive approach," in *Trust and deception in virtual societies*. Springer, 2001, pp. 55–90.

[111] D. H. McKnight and N. L. Chervany, "The meanings of trust," 1996.

[112] A. Jøsang, R. Hayward, and S. Pope, "Trust network analysis with subjective logic," in *Proceedings of the 29th Australasian Computer Science Conference-Volume 48*. Australian Computer Society, Inc., 2006, pp. 85–94.

[113] A. Jøsang and T. Bhuiyan, "Optimal trust network analysis with subjective logic," in *Emerging Security Information, Systems and Technologies, 2008. SECURWARE'08. Second International Conference on*. IEEE, 2008, pp. 179–184.

[114] I. C. Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection," *Information processing letters*, vol. 85, no. 6, pp. 317–325, 2003.

[115] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[116] X. Cai, Y. Cui, and Y. Tan, "Predicted modified pso with time-varying accelerator coefficients," *cognition*, vol. 1, no. 1, p. 3, 2009.

[117] L. Davis, "Handbook of genetic algorithms," 1991.

[118] M. Srinivas and L. M. Patnaik, "Genetic algorithms: A survey," *computer*, vol. 27, no. 6, pp. 17–26, 1994.

[119] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, Dec 1997. [Online]. Available: https://doi.org/10.1023/A:1008202821328

[120] K. V. Price, "Differential evolution," in *Handbook of Optimization*. Springer, 2013, pp. 187–214.

[121] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE transactions on evolutionary computation*, vol. 15, no. 1, pp. 4–31, 2010.

[122] M. Mussetta and P. Pirinoli, "M m c n-bbo schemes for electromagnetic problem optimization," in *2013 7th European Conference on Antennas and Propagation (EuCAP)*. IEEE, 2013, pp. 1058–1059.

[123] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*. Ieee, 1995, pp. 39–43.

[124] Y. Shi *et al.*, "Particle swarm optimization: developments, applications and resources," in *evolutionary computation, 2001. Proceedings of the 2001 Congress on*, vol. 1. IEEE, 2001, pp. 81–86.

[125] F. Grimaccia, M. Mussetta, and R. E. Zich, "Genetical swarm optimization: Self-adaptive hybrid evolutionary algorithm for electromagnetics," *IEEE Transactions on Antennas and Propagation*, vol. 55, no. 3, pp. 781–785, 2007.

[126] S. Selleri, M. Mussetta, P. Pirinoli, R. E. Zich, and L. Matekovits, "Some insight over new variations of the particle swarm optimization method," *IEEE Antennas and wireless propagation letters*, vol. 5, pp. 235–238, 2006.

[127] P. Culot, G. Dive, V. H. Nguyen, and J.-M. Ghuysen, "A quasi-newton algorithm for first-order saddle-point location," *Theoretica chimica acta*, vol. 82, no. 3-4, pp. 189–205, 1992.

[128] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence properties of the nelder–mead simplex method in low dimensions," *SIAM Journal on optimization*, vol. 9, no. 1, pp. 112–147, 1998.

[129] J. Fliege and B. F. Svaiter, "Steepest descent methods for multicriteria optimization," *Mathematical Methods of Operations Research*, vol. 51, no. 3, pp. 479–494, 2000.

[130] X. Li, K. Tang, M. N. Omidvar, Z. Yang, K. Qin, and H. China, "Benchmark functions for the cec 2013 special session and competition on large-scale global optimization," *gene*, vol. 7, no. 33, p. 8, 2013.

[131] E. Carrasco, M. Barba, and J. A. Encinar, "Reflectarray element based on aperture-coupled patches with slots and lines of variable length," *IEEE Transactions on Antennas and Propagation*, vol. 55, no. 3, pp. 820–825, 2007.

[132] M. Beccaria, P. Pirinoli, G. Dassano, and M. Orefice, "Design and experimental validation of convex conformal reflectarray antennas," *Electronics Letters*, vol. 52, no. 18, pp. 1511–1512, 2016.

[133] N. Misran, R. Cahill, and V. Fusco, "Design optimisation of ring elements for broadband reflectarray antennas," *IEE Proceedings-Microwaves, Antennas and Propagation*, vol. 150, no. 6, pp. 440–444, 2003.

[134] F. Grimaccia, M. Mussetta, P. Pirinoli, and R. Zich, "Optimization of a reflectarray antenna via hybrid evolutionary algorithms," in *2006 17th International Zurich Symposium on Electromagnetic Compatibility*. IEEE, 2006, pp. 254–257.

[135] P. Nayeri, A. Z. Elsherbeni, and F. Yang, "Radiation analysis approaches for reflectarray antennas [antenna designer's notebook]," *IEEE Antennas and Propagation Magazine*, vol. 55, no. 1, pp. 127–134, 2013.

[136] J. Huang and R. J. Pogorzelski, "A ka-band microstrip reflectarray with elements having variable rotation angles," *IEEE transactions on antennas and propagation*, vol. 46, no. 5, pp. 650–656, 1998.

[137] P. Nayeri, F. Yang, and A. Z. Elsherbeni, "Radiation analysis and characteristics of conformal reflectarray antennas," *International Journal of Antennas and Propagation*, vol. 2012, 2012.

[138] J. Huang and J. A. Encinar, *Reflectarray Antennas*. IEEE, 2007.

[139] M. Riel and J.-J. Laurin, "Design of an electronically beam scanning reflectarray using aperture-coupled elements," *IEEE Transactions on Antennas and Propagation*, vol. 55, no. 5, pp. 1260–1266, 2007.

[140] E. Carrasco, M. Barba, and J. A. Encinar, "X-band reflectarray antenna with switching-beam using pin diodes and gathered elements," *IEEE Transactions on Antennas and Propagation*, vol. 60, no. 12, pp. 5700–5708, 2012.

[141] S. R. Rengarajan, "Scanning and defocusing characteristics of microstrip reflectarrays," *IEEE Antennas and Wireless Propagation Letters*, vol. 9, pp. 163–166, 2010.

[142] K. Deb, "Multi-objective optimization," in *Search methodologies*. Springer, 2014, pp. 403–449.

[143] B. Xia, Z. Ren, K. Choi, and C.-S. Koh, "A novel subregion-based multidimensional optimization of electromagnetic devices assisted by kriging surrogate model," *IEEE Transactions on Magnetics*, vol. 53, no. 6, pp. 1–4, 2017.

[144] P. Karban, P. Kropík, V. Kotlan, and I. Doležel, "Bayes approach to solving team benchmark problems 22 and 25 and its comparison with other optimization techniques," *Applied Mathematics and Computation*, vol. 319, pp. 681–692, 2018.

[145] M. S. Berkani, S. Giurgea, C. Espanet, J.-L. Coulomb, and C. Kieffer, "Study on optimal design based on direct coupling between a fem simulation model and l-bfgs-b algorithm," *IEEE Transactions on Magnetics*, vol. 49, no. 5, pp. 2149–2152, 2013.