Doctoral program in Aerospace Science and Technology

Optimization on industrial problems focussing on multi-player strategies

by

Martí Coma Company

Directors:

Jordi Pons-Prats
Gabriel Bugeda Castelltort

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at
Universitat Politècnica de Catalunya - BarcelonaTech

Department of Physics
Universitat Politècnica de Catalunya
Barcelona
May 2022

Acknowledgments

I would like to express my gratitude to Dr. Jordi Pons i Prats for offering me the opportunity to continue learning through this research. I also thank him and Dr. Gabriel Bugeda Castelltort for all the support, constructive suggestions and their willingness to give their time during all stages of this research. I also want to acknowledge CIMNE for providing the environment and financial support for this research. I also acknowledge the Severo Ochoa Centre of Excellence (2019-2023), which has partially funded this work under the grant CEX2018-000797-S funded by MCIN/AEI/10.13039/501100011033.

I also would like to thank Navdi Tousi and Dr. Josep M. Bergadà for all his support in the Active Flow Control optimization problems, the publication of the articles and for sharing the computational resources of their research team to solve the AFC problems.

Finally, I would like to thank my family for his support and patience during all these years.

Abstract

Evolutionary Algorithms (EA) are useful optimization methods for exploration of the search space, but they usually have slowness problems to exploit and converge to the minimum with accuracy. On the other hand, gradient based methods converge faster to local minimums, although are not so robust (e.g., flat areas and discontinuities can cause problems) and they lack exploration capabilities.

This thesis presents and analyze four versions of a hybrid optimization method trying to combine the virtues of Evolutionary Algorithms (EA) and gradient based algorithms, and to overcome their corresponding drawbacks. The proposed Hybrid Methods enable working with N optimization algorithms (called players), multiple objective functions and design variables, and define them differently for each player. The performance of the Hybrid Methods are compared against a gradient based method, two Genetic Algorithms (GA) and a Particle Swarm Optimization (PSO).

Tests have been conducted with mathematical benchmark problems (synthetic tests designed to specifically test optimization methods) and an engineering application with high demanding computational resources, a Synthetic Jet actuator for Active Flow Control (AFC) over a 2D Selig-Donovan 7003 (SD7003) airfoil at Reynolds number 6×10^4 and a 14 degree angle of attack. The Active Flow control problem has been used in a single optimization problem and in a two objective optimization problem.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Generally speaking, optimization is a mathematical technique for finding the maximum or minimum value of a function, or set of functions, of several variables subject to a set of constraints.

Optimization is a topic of great interest in engineering and other fields, and it is actively researched. Aerospace engineering is traditionally one of the most interested fields in optimization. This is because of the complexity of aircraft design, space systems design and problems associated with air transport. The competitiveness and exigence of the aerospace market is another major factor to invest and research on optimization.

Nowadays, optimization has reached all engineering fields and it is widely used in the industry and research areas. Optimization applications range from improvement and refinement of mechanical designs to path optimization of delivery routes, or production processes. In airspace engineering, optimization is used even in early stages of projects, such as aircraft preliminary designs; and in advances stages of the design process to further refine the final design.

There are different approaches when dealing with optimization problems. From highly specialized algorithms that only address particular problems, to generic optimization methods that do not require information about the problem at hand. The highly specialized algorithms can be very efficient but only work with the optimization problem that they where designed to solve. On the other hand, there are generic algorithms that only require to evaluate the fitness of the proposed design during the optimization problems. In this work Evolutionary Algorithms are considered, as they are used daily by the author as a research engineer at *Centre Internacional de Mètodes Numèrics en Enginyeria*[1] (CIMNE).

In the following sections of this chapter the justification, objective and document layout are presented.

[1] http://www.cimne.com

1.1 Justification

Optimization is one of the main research topics at CIMNE. It is conducted by the Aerospace Engineering Group[2]. The author collaborates with the Aerospace Engineering Group since 2010 when he was member of CIMNE's Industrial Processes Group[3]. Since September 2015 the author became a full time research engineer at the Aerospace Engineering Group and started focusing on optimization. The work conducted by the author in the Aerospace Engineering Group on optimization can be divided in two aspects. First, applying the in-house optimization tool to several industrial projects and research projects in different fields and topics such as:

- Optimization of metal forming processes with finite element method.
- Design of flexible electric circuits for heating purposes.
- Tuned mass damper design to reduce fatigue on offshore wind turbines.
- Active Flow Control using plasma actuators. In this project the optimizer was coupled with a wind tunnel to conduct the optimization through experimental evaluation of the objective function.

among others. During this period at CIMNE, the main problems encountered when working on optimization problems with industrial requirements are listed below:

- Expensive evaluation of objective functions, which leads to high computational costs.
- Complex work-flows involving different software and programming languages.
- Multi-objective requirements which lead to use Evolutionary Algorithms. The low rate of convergence of evolutionary algorithms lead to high computational costs. The investigation on gradient based methods for multi-objective optimization is currently an active field of research.
- Lack of high performance computational resources or already saturated.

The importance to have an optimization tool able to address industrial and research problems of different fields and requirements is remarkable. For this reasons, it would be adequate to have an unified tool containing different methods and optimization approaches so it is not necessary to spend time linking and testing the work-flow with different optimization tools and strategies. In addition, it is not enough to obtain such a tool, but to study and learn when and how to use the different optimization strategies according to the project requirements.

The second aspect of the work at CIMNE has been the maintenance and development of the in-house optimization platform. The current version of the in-house optimization platform is an implementation re-coded by the author from scratch, named RMOPv2. The research group agreed to implement the software from zero to obtain a more robust new version, easy to maintain and suitable to use it as a

[2] http://www.cimne.com/spacehome/2/1167
[3] http://www.cimne.com/spacehome/2/1159

research platform. The old code was not designed to allow the research on multi-player optimization that is proposed in this work. The architecture of the new implementation was designed with this requirement in mind. Another requirement is to facilitate the coupling with other existing software at CIMNE, namely KRATOS Multi-physics platform[4] among others. In addition, the new platform had to allow working in industrial and research projects with high flexibility and using different computation resources. The new platform allows implementing and testing new algorithms with shorter coding and testing times.

During the development of research projects and when dealing with industrial problems at CIMNE, one need that kept rising was to improve the performance of the optimizer in terms of global search capabilities and accuracy. These two aspects usually have to be compromised and is an active line of research at the Aerospace Engineering Group and many other research groups. Hybridization is one of the techniques that allow to improve both aspects of the methods and is the focus of this work.

1.2 Thesis objectives

The main objective of this work is to present the research work developed on optimization focused on multi-objective and multi-player methods. The research will focus on developing and testing new optimization strategies to solve industrial applications applying hybridization techniques with multi-player strategies that use different optimization methods. Industrial applications usually require high computational costs and limited time to perform the optimization.

1.3 Document layout

This document is organized as follows. Chapter *2* presents a review showing the state of the art in hybridization and places this investigation into context. Next, in Chapter *3*, the proposed hybridization are explained in detail. The exposition is complemented with the explanation of the optimization algorithms used in the hybridization. In Chapter *4*, the metrics used to evaluate the performance of the different optimization algorithms tested are explained and the results with the benchmark tests are presented. The mathematical benchmark test problems are presented first, containing single objective and two objectives optimization test problems. The last sections of Chapter *4* present the performance results on a Synthetic Jet device for Active Flow Control optimization problem, also with one and two objective functions. The document closes in Chapter *5* with the conclusions, future work and a list of publications made during the course of the investigation.

[4] http://www.cimne.com/kratos

Chapter 2

State of the art

A review of the more relevant literature on optimization is presented in this chapter. Different methods and approaches are described, and their strong points and flaws are discussed. The objective of this chapter is to put the optimization methods used in this study into context, not to provide a detailed description of all the optimization methods. A section is dedicated to Hybrid Methods which are optimization methods formed by a combination of other optimization algorithms, such as the one developed in this study.

2.1 Optimization definition

A multi-objective optimization problem can be defined as in (2.1) in its minimization form.

$$\begin{cases} \min(f_1(\vec{x}), f_2(\vec{x}), ..., f_k(\vec{x})), \quad k = 1, ..., K \\ \quad \text{Subject to:} \\ g_l(\vec{x}) \geq 0, \quad l = 1, ..., L \\ h_m(\vec{x}) = 0, \quad m = 1, ..., M \\ \quad \text{Where:} \\ \vec{x} \in X, \quad X \text{ feasible set of decision variables.} \\ f_k : X \to \mathbb{R}, \quad k = 1, ..., K \\ g_l : X \to \mathbb{R}, \quad l = 1, ..., L \\ h_m : X \to \mathbb{R}, \quad m = 1, ..., M \end{cases} \tag{2.1}$$

In order to clarify the nomenclature, the following definitions are stated:

Design Variables (DVs) are the set of decision variables, the independent variables of the optimization problem: $\vec{x} \in X$.

Objective Functions (FOs) are the set of fitness functions. The functions that express the aptitude of a set of DVs: $f_1(\vec{x}), f_2(\vec{x}), ..., f_K(\vec{x})$.

Constraints (CNs) are the set of expressions that needs to be satisfied in order to consider the values of the DVs as feasible. It includes both types, inequality g_l and equality h_m expressions.

2.2 Classification of the Optimization methods

Optimization problems are addressed from many different approaches. The selection of the optimization method to solve a problem depends on the complexity, the continuity and the possibility to calculate the derivatives of the objective functions involved (sensitivities). It is also possible to use different algorithms depending on the state of the study as may require different levels of precision. In earlier stages of a study, an optimization algorithm with more exploration capabilities may be more suitable. As a study reaches advanced phases and the detailed design is performed a method with more exploitation capabilities could be a better option. In Figure 2.1 a scheme of a possible classification of different optimization methods is presented. Not all optimization methods are listed here and one could think about other possible classifications. Two main groups are normally used to classify the optimization methods, the Meta-heuristics (stochastic methods) and the Classicals (deterministic methods). Meta-heuristics methods are usually very robust in scenarios without much knowledge about the shape of the objective functions, with possible discontinuities and non-differentiable functions. The introduction of the Pareto-Optimality concept allows to implement multi-objective meta-heuristics methods.

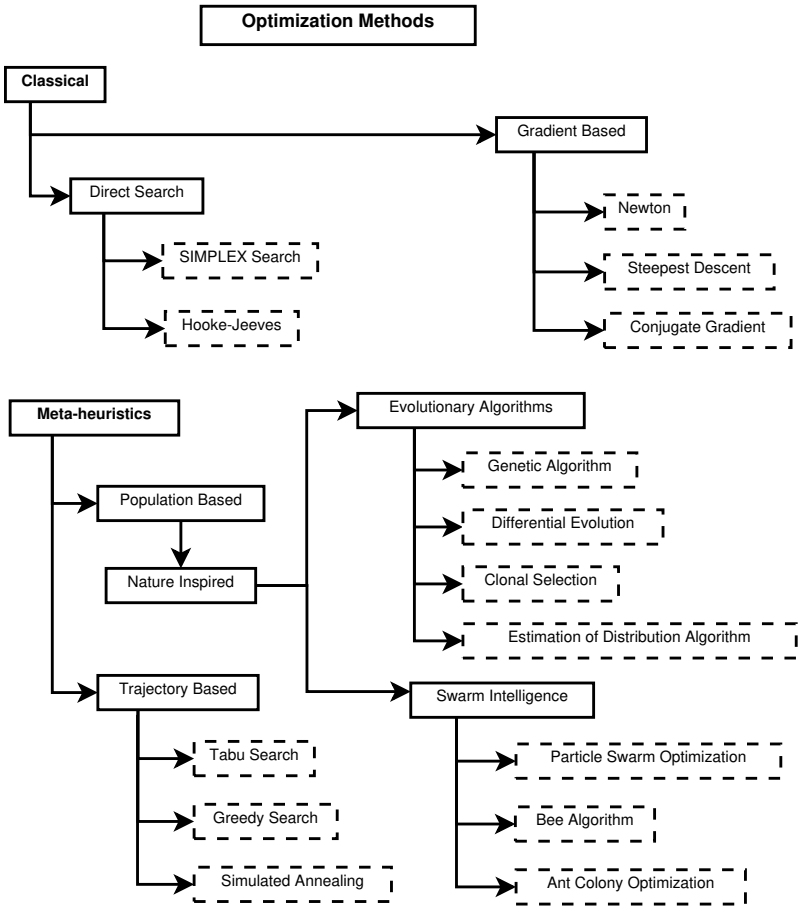


Figure 2.1: Classification of the optimization algorithms.

Meta-heuristics methods usually perform very well in the exploration of the search space but lack exploitation performance. They usually struggle to converge to the exact minimum. On the other hand, Classical methods usually converge faster to the near minimum when appropriate, showing great exploitation capabilities. The main drawback of Classical methods is the lack of exploration capabilities and the limitation of objective functions where are applicable. Classical methods are usually not suitable to optimize problems with non-continuous and/or non-differentiable objective functions. In addition, Classical methods are usually single objective.

A general description of the main methods and techniques is presented in the following sections.

2.2.1 Classical methods

The classical optimization techniques are useful in finding the optimum solutions for functions that are continuous. These methods are analytical and deterministic. These techniques are very efficient but their scope is hardly limited in practical optimization problems. That is because most practical engineering optimization problems are not continuous nor differentiable. Classical methods can be split into two main groups: Direct search and Gradient based.

Direct search

Direct search methods include methods such as the Simplex Search, which is part of the Linear Programming methods on other classifications.

Linear programming is a set of techniques to find the best solution for a problem with a linear objective function (single objective problems), and linear equality and inequalities. One of the most popular methods of linear programming is the SIMPLEX method. The restriction that the functions must be linear makes these techniques of no use in most of the applications of interest in this work. For more details on Linear programming see Luenberger, Ye et al. [1]. Another example of a Direct search method is the Hooke-Jeeves method, also known as Pattern Search, for detailed reference see [2, 3, 4, 5].

Gradient based methods

Gradient-based is another type of method that handle non-linear problems. They use the information about the gradient to select the search direction and, iteratively converge to the optimum solution. These methods can get stuck on local optima and their performance highly depend on the starting solution of the iterative method. These methods are usually efficient, especially when the gradient information can be accurately computed. Different methods have been proposed to estimate the gradient when it can not be computed analytically. Some gradient methods are listed below:

- Newton's method.
- Steepest Descent
- Conjugate gradient method.

A detailed description of these methods, and others, can be found at [6].

Traditionally, these methods are well suited for single objective optimization problems, but can also be used through different strategies, such as multi-player optimization, to address multi-objective optimization problems. This approach will be researched further because of its potential increase in performance in high demanding applications.

The Conjugate Gradient has been used intensively in this study. A more detailed explanation of the used method is explained in Section *2.3*.

2.2.2 Meta-heuristics methods

Meta-heuristics methods are those which use random or pseudo-random tools to generate new variables along with successive iterations. They are capable of performing multi-objective optimization. The use of the Pareto Optimality Criteria is accepted as a relevant tool for the multi-objective optimization and does not impose restrictions on the shape of the objective function. Meta-heuristics methods can handle multiple restrictions of any kind as well. This robustness is one of the most interesting aspects of these methods. Usually, they do not require information about the

gradient, but sometimes its performance can be improved using such information. The counterpart is that they usually need more iterations to converge on problems where other methods can be used. There are two main groups of Meta-heuristics methods, the trajectory based and the population based.

These methods usually perform global optimization efficiently but they lack precision in some cases. Depending on the application it may be needed or interesting to refine the results obtained by a Meta-heuristic optimization method with other methods, such as gradient based methods to increase the precision of the optima. For example, Gudla and Ganguli [7] proposed a hybrid genetic-conjugate gradient to perform achieve such functionality in an automated way. They proposed to run a Genetic Algorithm for n iterations and then start a Conjugate Gradient to improve the solution, all integrated in a single algorithm.

Trajectory based

Trajectory based optimization methods are based on single solutions. The algorithms start with a random solution or from previous knowledge. The initial solution is updated according to information about its neighborhood. These methods do not require information about the gradient, only information about the objective function evaluation around the current point. The comparison of the different individuals is usually performed only between the current point and one neighbor at a time, then the algorithm decides if the neighbor is accepted or not. If the neighbor is accepted, the process is repeated from the new accepted point. They usually include probabilistic methods to avoid getting trapped in local minimums. Some examples are:

- Tabu Search (see [8, 9, 10]).
- Greedy Search (see [11]).
- Simulated Annealing (see [12, 13, 14, 15]).

Population based, nature inspired

Population based methods work with a set of individuals, the set is called a population. The whole population is computed before generating a new set of individuals, the offspring. Nature inspired methods are the most used population based methods. There are two main groups, the Evolutionary Algorithms, and the Swarm Intelligence.

Evolutionary algorithms

- Genetic Algorithms (GA) (see [16, 17, 18, 19]).
- Differential Evolution (see [20, 21]).
- Clonal Evolution (see [22, 23]).
- Estimation of Distribution Algorithm (EDA). There are variations of the EDA algorithm that take into account information of an estimated gradient, see [24].

There are variations of EDA that take into account information about an estimated gradient [24].

Swarm Intelligence

- Particle Swarm Optimization (PSO) (see [25]).
- Ant Colony Optimization (see [26, 27]).
- Bee Algorithm (see [28]).

In the following sections the algorithms used in this study are described with more detail. The optimization algorithms used are a Conjugate Gradient, a Genetic Algorithm and a Particle Swarm Optimization.

2.3 Conjugate Gradient

The Conjugate Gradient is a gradient based optimization method for single objective problems, see [29]. It is a deterministic method as there are no probability variables in its algorithm. The method starts with an initial solution to the problem, the initial point. The method computes the gradient of the objective function at the initial point. In most engineering applications the derivatives of the function can not be obtained analytically, and the gradient has to be approximated, for example, by finite differences which increases the number of function objective evaluations needed to converge.

Once the gradient at the starting point is known a search direction is determined, there are different proposes to do so. For example, the ones proposed by Fletcher and Reeves [30] (also see [31]) and Polak and Ribiere [32]. After determining the search direction a line search is conducted, which consists in finding the minimum in that direction. To do so the starting point is modified by adding the step size to each design variable. The step size could be held constant but usually, it is more efficient to change the step size using the available information of the objective function and its derivatives. There are different methods to calculate the optimal step size for example the Golden Section Search proposed by Kiefer [33] and the Brent Method proposed by Brent [34].

After the minimum of the line search is found the process is repeated from the minimum point. The gradient is computed again and if it is not zero the search direction is determined and the line search is performed until a local minimum is found, where the gradient is zero.

2.4 Genetic algorithm

The Genetic Algorithm is a population based method inspired in the evolution of the species. The algorithm starts generating a random population, although if some individuals are already known can be used as the starting population. After the population is computed (i.e., the objective functions of all individuals are evaluated)

the operators of the method are applied to generate a new population, called the offspring. There are typically three operators:

- Selection operator.
- Crossover operator.
- Mutation operator.

The selection operator determines which individuals are candidates to become parents of the offspring, which means they will participate in the crossover operator. Selection operators usually combine probabilistic variables and ranking methods prioritizing the fittest individuals of the population. The crossover operator combines individuals, typically in pairs, to generate another pair of new individuals. The crossover operator also includes a probabilistic process to determine which variables are combined between the pair of individuals. The mutation operator introduces random changes in the individual's design variables to increase the exploration capabilities and avoid getting trapped in local minimums. The Genetic Algorithm implementation used in this study is based on the implementation of NSGAII by Deb et al. [18]. The operators used at the NSGAII implementation are:

- Selection operator: $\mu + \lambda$ & Crowded-Comparison Operator [18].
- Crossover operator: Simulated Binary Crossover [17].
- Mutation operator: Polynomial Mutation [19].

2.5 Particle Swarm Optimization

The Particle Swarm Optimization is another population based and nature inspired optimization method proposed by Kennedy and Eberhart [25]. The method mimics the move of certain species, for example, the bees when searching for pollen. A swarm of particles (candidate solutions) moves around the search space exploring other areas. The movement of the particles is influenced by the best solution achieved so far and the best solution inside the current swarm. This is achieved by updating the position of each particle in the swarm adding a velocity computed for each particle which depends on the relative position of the mentioned best solutions concerning the particle and in random variations. For a detailed explanation and formulation of the method see [35, 36, 37, 38, 39].

For the two objective problems a Multi-Objective Particle Swarm Optimization method based on the work of Coello and Lechuga [40] has been used.

2.6 Hybrid Methods

Three hybridization strategies are identified from the literature; namely, hybridization through operators, a combination of methods, and the definition of multi-population methods. Although sometimes two of the strategies can be combined, one can find references for the three of them, and even some combinations.

A good starting reference is Cheng, Gen and Tsujimura [41]. Based on Genetic Algorithms, this paper is describing the hybridization of genetic operators. It provides a list of several crossover and mutation operators. It describes the combination of global and local methods as well (Giffler and Thompson method, bottleneck shifting heuristic, beam search). This clear example shows that, sometimes, a local search method could be understood as a new operator, instead of a combination of two methods. The paper is mainly focusing on job scheduling problems, but the described operators can be applied to any type of optimization problem. Similarly, Wang et al. [42] is defining a new selection operator, a new mutation operator, and local search operators. Just focusing on crossover, Weare, Burke and Elliman [43] is defining a hybrid crossover. Its main application is to operational research, with specific application to the definition of scheduling and timetables, but again, the applicability is not limited to that type of problem. In some cases, the hybridization of the operators can benefit from a good knowledge about the problem to solve. Valls, Ballestin and Quintanilla [44] is describing such hybridization, which modifies the crossover operator according to the problem. The Peak Crossover operation does not use randomly selected parts to combine as standard crossover does, to adapt the variability of the design variables to what the problem requires. The paper is also presenting an addition operator, to the standard genetic ones, called Double justification operator. Ho et al. [45] is also presenting a hybrid GA through operator hybridization. The selection operator is improved with local search operations, which are strongly dependent on the problem to be solved.

This first strategy to hybridize through operators can be overlapped with the combination of methods depending on how the methods are used. For example, Shahidi et al. [46] defined a self-adaptive Memetic Algorithm using a Conjugate Gradient as a local hill climb operator where each individual of the Genetic Algorithm population can be improved between the crossover and the selection operators. Another work taking a similar approach to hybridize is proposed by Bautista [47]. In its thesis, three variations of a hybrid method are presented. They combine a Genetic Algorithm (NSGAII) with a Feasible Sequential Quadratic Programming (FSQP) to refine the solutions.

Looking to the second strategy; the combination of two methods, Kelly Jr and Davis [48] presents a hybrid algorithm for classification; combining a genetic algorithm with k-nearest neighbors classification algorithm. It is a good example that hybridization is not limited to optimization methods, since genetic algorithms are not only used for optimization. The fact is that Genetic Algorithms are always good candidates to be hybridized.

Jih and Hsu [49] is introducing a hybrid Genetic Algorithm for Vehicle routing applications. Two algorithms are used. Dynamic Programming is producing a first approach to the solution, and Genetic Algorithms are taking the partial results from Dynamic Programming as the starting point. It also defines several crossover and mutation operators, as part of the hybridization work. To get a good overview of the potential when combining two different methods, El-Mihoub et al. [50] is presenting the results of combining two methods; the first one is Genetic Algorithm, and the second one is another search and optimization technique. The Genetic Algorithm is

used as the exploration tool, while the other method is used as a local search tool. The assessed search methods are Lamarckian learning, Baldwinian Learning, and a hybrid Lamarckian-Baldwinian model. The communication describes the issues about how to benefit from the local search information, and how to find the best balance between global and local search (exploration and exploitation); including the division of exploration and exploitation time. It states that some combinations can produce inefficiencies in the search and premature convergence, which is an undesired effect. A last examples is Kulcke and Lorenz [51] which focus on architecture applications. It combines Genetic Algorithms (Interactive GA, more specifically) and Gradient information to enable the creation of the geometrical features to be used in a new design. Hassanat et al. [52] define and test several mutation operations. Finally, they propose two new mutation operators. The first one creates a selection algorithm to operate with those selected mutation operators at the same time, while the second selects one of those mutation operators to be used along with the analysis. Pan et al. [53] is another example that combines two methods, a Self-Adaptive Genetic Algorithm, and a Conjugate Gradient. They propose a further optimization of the best individuals of the Genetic Algorithm population (it seems that the selection operator copies the best individuals directly to the next population). After the genetic operators are performed the best individuals are improved with the Conjugate Gradient and the worst ones use an immigrate method (not specified in the article) to update.

Regarding the third strategy, the one defining multiple populations working in parallel and sharing genetic information, Ho et al. [45] defines a hybrid Genetic Algorithm with multiple populations applied to Vehicle routing problems. The hybridization strategy is a combination of hybridization of operators and defining multiple populations since one of the populations, the GA one, is enhanced using the Iterated Swarp Procedure (ISP) operator, and the second population combines the Clarke and Wright saving method and Nearest Neighbor Heuristic (NNH) to enhance the GA operators. Briefly, each population uses a different strategy to enhance the genetic information. Another strategy is the one used by Berger and Barkaoui [54] who defines two populations, one of them managing the objective functions and the second one the restrictions of the problem. After each iteration, the genetic information is shared to assess how good are the candidates which improve the objective functions and also fulfill the restrictions, and vice versa. Lee et al. [55] propose a hybrid Genetic Algorithms based on the Nash Games. Each population, called player in this case, is taking care of a part of the problem. An additional population is dealing with the overall problem, so the Pareto Criteria can be easily applied. It is worth mentioning the paper by Vargas et al. [56] who analyzes a multiple population framework (they called sub-populations), and the cooperation between two or more populations/optimization algorithms. They analyze the state of the art and describe the difference between single-population (panmictic or unstructured algorithms) and multi-population or sub-population (structured). Vargas' paper presents the implementation of a structured algorithm using GDE3, and the Multi-Objective Novelty Algorithm. A comparison is done between several hybridization approaches; namely Island model, stepping Stone model, neighborhood model, where the exchanged genetic information belongs to the whole set of populations, adjacent populations or

exchanges from individuals to adjacent individuals, respectively.

Neri, Cotta and Moscato [57] present a book with a compendium of Memetic Algorithms implementations and examples of optimization problems solved with them. The book offers a systematic review of the main aspects when designing and using Memetic Algorithms which include the balancing of local and global search, the selection of the correct operators depending on the problem characteristics among others. Another book about hybridization is presented by Cantu-Paz [58]. This book explains parallel Genetic Algorithms in detail. Parallel Genetic Algorithms are a form of hybridization, as they use more than one instance of a Genetic Algorithm which share information to achieve better solutions and a higher rate of convergence. The book also addresses the problem of configuration and tuning of such algorithms as these aspects are of great importance with multi-population and parallel algorithms. Another example in the same line of work is presented by Periaux, Gonzalez and Lee [59] which introduces Game Theory concepts (Nash and Stackelberg Games) in the hybrid methods and present its application to aeronautics and UAV design. Talbi [60] presents an article which classifies and describes the main aspects of designing parallel multi-objective Evolutionary Algorithms. It addresses the design and implementation aspects of parallel Genetic Algorithms, and explains the differences between the different computational environments (Shared memory, distributed memory grid computing) and how they can affect the design of the hybridization to take the most advantage of the system, for example, asynchronous and synchronous designs are discussed.

2.7 Summary

A review of the state of the art on hybridization and specifically in hybridization using Evolutionary Algorithms with gradient based methods has been conducted. There are two general approaches on this type of hybridization. The first one is to treat the gradient based method as a local search operator usually referred as Memetic Algorithms. The second approach is to use a two stage algorithm, with the first stage being the exploration phase conducted by the EA and a second exploitation stage conducted by the gradient based method.

In this work a multiplayer strategy is proposed, where the Evolutionary Algorithm and the gradient based method of the hybridization are treated as equals and exchange information to cooperate during the whole search process.

Chapter 3

Proposed methodology; Hybrid Method

This chapter describes the proposed new Hybrid methods. As described in Chapter *2*, several approaches have been developed so far. The current proposal is based on a multi-population multi-optimization approach based on Lee et al. [61], which defines three populations. Each population deals with a different set of objective functions and design variables, and hybridizes through the use of Nash Games and Pareto optimality concept. For a problem with two objective functions Lee et al. [61] limited their proposal to three populations or players, all of them using Genetic Algorithms. The main player, called the Global player, is in charge of the optimization of the global problem which uses two objective functions and all design variables. The remaining two players take care of a restricted optimization problem by fixing the values of certain design variables and only considering one of the two objective functions, respectively. The Hybrid Method shares information between these three players trying to achieve a higher rate of convergence. The Hybrid Methods proposed here extend the capabilities to combine several optimization methods among the players, like Gradient Based ones or Particle Swarm Optimization. The current implementation of the code also allows to easily configure a Hybrid Method with N players although it has not been tested.

Multi-player optimization, the third hybridization strategy explained above, consists in creating a new optimization algorithm combining more than one optimization algorithm (which are called players). The different players run their own optimization algorithms and exchange information between them. In a multi-player optimization method there are different aspects to define and create new methods, the main ones are:

- The type of optimization method of each player, which overlaps with the second strategy in the above classification.
- The way and criteria in which the information is exchanged between players.
- The design variables and objective functions that each player works with, as it is possible to make a player optimize a subspace of the problem. Some players

could work with a subset of the search space or work with a subset of the objective functions.

- How much run time is allocated to each player.

Some examples of multi-player strategies are listed below:

- Nash games [62].
- Hybrid games [63, 61].
- Stackelberg games [59].
- Hierarchical optimization [64].

In this study, the main line of research is to perform multi-player optimization with players using different optimization algorithms which is a mix of the second and third hybridization strategies. The main difference between the proposed hybridization and the works of Shahidi et al. [46] and Pan et al. [53] is that in this study the Conjugate Gradient is treated as a player, not as a refinement operator. The Conjugate Gradient player can continue its optimization process independently from the Evolutionary Algorithm in the hybridization and the sharing of information is handled with a Migration Strategy that considers all players equally.

As outlined in the following sections, the general algorithm and its implementation in *C++* code are very flexible. Details of the implementation are explained in Section *3.6*. The new proposed Hybrid Method combines two or more players (i.e., optimization algorithms) into a single optimization strategy.

The main idea of hybridization is to combine different optimization algorithms into a single optimization strategy. In this way, the resulting hybridized method incorporates the main advantages of each optimization algorithm and overcomes the individual drawbacks.

The Hybrid Method algorithm proposed here and its implementation can reproduce the behavior of Lee et al. [61] using its configuration. The implementation and general algorithm have been outlined to allow flexibility and as a platform for testing different variants of the method. The main configurable aspects are:

- Number of players.
- Optimization algorithm of each player.
- Number of design variables and range for each player.
- Number of objective functions for each player.
- The solver that uses each player. This can be useful to perform a hierarchical optimization, among others as explained in Chapter *3.2*.

The algorithm of the Hybrid Method proposed in this work is divided in three main components. Those are:

- *General Algorithm*: It contains the initialization of the players, the main optimization loop, and the post-process of the optimization. This component

is fixed and does not depend on the different variants proposed the Hybrid Method.

- *Migration Epoch algorithm*: It is the function that defines the exchange of information between the different players. It defines which individuals are migrated between players, under which circumstances, etc.
- *Immigrate methods*: The Immigrate method is a function that has to be defined for each type of player. The way each optimization algorithm used as a player can incorporate and use an individual highly depends on the internal algorithm of each type of player. This function defines how each type of player incorporates the individuals that immigrate.

The following sections describe in detail each component of the Hybrid Method.

3.1 General Algorithm

The General Algorithm defines the global execution of the Hybrid Method. It manages the initialization, main loop, and post-process of the Hybrid Method optimization. The General Algorithm of the Hybrid Method is summarized in Algorithm *1*. This algorithm has been implemented in *C++* using Object Oriented Programming paradigm, thus the nomenclature of the algorithm imitates the structure of the code.

The players of the algorithm are stored in a vector named *players*. The first step of the algorithm is to initialize all the players (*player-Initialize*). In this step, the configuration files are read and all the optimization algorithms are initialized.

After the initialization of all players, the algorithm enters the main optimization loop. The loop is repeated until a stop criterion is met. There are four available termination criteria. They are based on the maximum execution time, the number of evaluations of the objective functions, convergence criteria and achieved value of the objective functions.

Algorithm 1: General Algorithm of the Hybrid Method

```
players: vector with all players;
foreach player ∈ players do
    player-Initialize;
while not stop criteria is met do
    foreach player ∈ players do
        player-Generate;
        player-Compute;
        MigrationEpoch(players[i]);
foreach player ∈ players do
    player-PostProcess;
PostProcess;
```

Inside the optimization or main loop, the Hybrid Method interlaces the execution

of its internal players. Each player runs one iteration of its optimization algorithm, then a Migration Epoch occurs before the other player runs one iteration of its own algorithm. One iteration of the player is defined as generating new individuals (*player-Generate*) and computing them (*player-Compute*). For example, in a Genetic Algorithm, one iteration consists of the execution of the selection, mutation, and crossover operators to generate a new offspring and then evaluate the population. In a Gradient Based method, the iteration could consist of computing the gradient and performing a line search, computing all the individuals.

Once the termination criteria are met, the algorithm escapes the main loop and executes the post-process for each player. As each player is a full optimization algorithm it is useful to post-process them one by one (*player-PostProcess*). Finally, the post-process of the Hybrid Method is conducted (*PostProcess*).

3.2 Migration Epoch

The Migration Epoch implementation is what defines the main functionality of the Hybrid Method. It is the mechanism that allows the exchange of information between players. The information exchanged by the players is a selection of individuals which contains its design variables and objective functions. In the Algorithm *1* the Migration Epoch process is named *MigrationEpoch*. The definition of this process is what defines most of the hybrid algorithms.

By changing the *MigrationEpoch* implementation it is possible to define different hybrid strategies. It defines the criteria of which individuals migrate between players, how often they migrate, between which players, etc. By changing the implementation of this function and the number of players it is possible to obtain different hybrid algorithms such as the proposed by Lee et al. [61] for a Nash Game hybridization with a global Pareto Player or to obtain a hierarchical optimization using different layers of precision as proposed by Lee et al. [64].

To obtain a hybridized Nash Game with a global Pareto Player as proposed by Lee et al. [61] three players of Genetic Algorithm are needed. The first one is configured with all the design variables of the problem and with the two objective functions. The two other players only optimize a subset of design variables, which are split between the two players. Each Nash player optimizes one of the two objective functions and the information is exchanged between the three players. This setup is very interesting for applications where there is a natural division between the design variables and the objective functions. For example in multidisciplinary problems. In that case, the Pareto Player can control the full optimization and each player can optimize one discipline of the problem. For instance, in an optimization problem dealing with aeroelasticity, the natural division of the design variables is to assign the ones that affect the most the aerodynamic problem (i.e., the external shape) to one player, and assign the design variables which have more impact in the structural problem (i.e., the thickness and materials of the structure) to the other player. This division brings another opportunity for improvement, as each player may not have to perform a full evaluation of the problem to obtain the objective function which

it is working, thus reducing the computational cost of the evaluation.

Another example of the flexibility achieved with this algorithm is the possibility to conduct a hierarchical optimization (see [64]) by only changing the Migration Epoch algorithm. A hierarchical optimization consists of defining different layers of precision. At the bottom layers, the objective functions are evaluated with less precision than at the top layers. The information is usually shared from the bottom layers to the above ones. To define a hierarchical optimization it is needed one player for each layer. The algorithm of each player could be the same or different depending on the level of accuracy of the layer. It could be interesting to use an optimization method that explores the full search space in the layer that works with the smaller precision, for example, an Evolutionary Algorithm, and an optimization method that converges faster but may not be able to explore the full search space, such as a Gradient Based method, for the layers that work with more precision and computational cost. If a Gradient Based method can not be used it could be interesting to use population based methods tuned to perform the exploitation at the top layers, for example reducing the mutation probability in a Genetic Algorithm or introducing more elitism.

3.3 Immigrate methods

For the full comprehension of the hybrid algorithm, it is important to specify the mechanism with which each player uses the information from the other optimization algorithms. This process is defined for each player and its implementation depends on the type of optimization algorithm, and it affects the general hybrid algorithm. It has one input parameter, the individual that has been selected to immigrate to this player. The function is called *Immigrate* and is responsible to incorporate the individual into the player. This function is called inside the Migration Epoch procedure.

In this study the optimization algorithms used as players are:

- Genetic Algorithm.
- Particle Swarm Optimization.
- Conjugate Gradient.

The treatment is different depending on the nature of the optimization algorithm. The population based methods are treated differently than the non population based methods. In the following sections the *Immigrate* functions used for the Genetic Algorithm, the Particle Swarm Optimization, and the Conjugate Gradient are detailed.

3.3.1 Population based

Population based algorithms contain an evolutionary population. The *Immigrate* process of the population based methods substitutes the design variables of the last individual of its internal population with the immigrated one. This introduces the

information about the immigrated individual into the population. In the next iteration, the design variables of this individual will be used to generate the new offspring. Both used population based methods, the Genetic Algorithm (Evolutionary Algorithm) and the Particle Swarm Optimization (Swarm Intelligence) use the same strategy for the *Immigrate* function.

The Genetic Algorithm and the Particle Swarm Optimization players always incorporate the individuals coming from other players into the population, even if it is the same as in any previous global iteration. The stochastic nature of the Evolutionary Algorithms and Swarm Intelligence can benefit from maintaining the best individual in the population at each iteration. There is a probability that the best individual is not selected in the genetic operators, and to keep the best individual in the population, keeping its genetic information, can help to converge in that region. One could think about problems with elitism, but the method can only force one individual to remain in the population, the absolute best so far. If this happens for too long, most probably the optimization has converged, and in case it is not converged, the algorithm should still be capable to explore other regions (for example with the mutation operator in the Genetic Algorithm) and the stochastic nature of the population based methods.

3.3.2 Non-Population based methods

The non-population based methods, such as gradient based methods have to handle the incoming individuals differently. As these methods do not have an internal population the individual can not be incorporated into it.

The *Immigrate* function of the Conjugate Gradient sets the incoming individual as the starting point of its algorithm and restarts the step size to its default value. In the next iteration the algorithm will compute the gradient at this point, then compute the search direction and perform the line search in that direction.

In order not to repeat computations, the Conjugate Gradient player only performs a new iteration when the immigrant individual is different than in the previous global iteration.

3.4 Tested Hybrid Methods

The new proposed Hybrid Method shares information between the players, in a multi-directional way, to overcome the main drawbacks of the composing optimization algorithms used alone. It tries to achieve a fast rate of convergence and avoid getting stuck at local minima. It can also be interpreted as if each method is a pre-conditioner of the other, and the pre-conditioner is applied between iterations.

The Hybrid Methods presented in this study have been tested with the combination of three different optimization algorithms which are a Genetic Algorithm, a Particle Swarm Optimization, and a Conjugate Gradient. Different tests have been performed with two and three players, depending on the number of objective functions

of the test problem. In all cases, the first player has been an Evolutionary Algorithm (Genetic Algorithm or Particle Swarm Optimization), see [38, 65, 25]. The main goal of this player is to explore the full search space. For that purpose, the Evolutionary Algorithms are suitable and widely used, as they are robust methods that do not get stuck at local minimums.

The other players are intended to perform the exploitation of the promising regions found by the first one. One additional player is added for each objective function of the problem which is responsible for the exploitation of the local search space. The selected optimization method for the exploitation is the Conjugate Gradient [29]. For the test cases with a single objective function, two players are used, an Evolutionary Algorithm and a Conjugate Gradient both working with all the design variables and the same objective function. For the test cases with two objective functions, three players are used. The first one is with the Evolutionary Algorithm working with both objective functions. The other two players are Conjugate Gradients and each player works with one of the objective functions.

As emphasized in previous sections, for each player, the internal operation and operators are applied before the *MigrationEpoch*.

For the Genetic Algorithm player, the genetic operators of selection, crossover, and mutation are performed inside the *Generate* process, which yields a new population, known as the offspring. After the population is computed the *MigrationEpoch* process is called and, after that, the Conjugate Gradient runs an iteration of its algorithm starting with the migrated individual. The process repeats until the stop criteria is met.

Two different versions of the new Migration Epoch have been tested, with the two mentioned Evolutionary Algorithms as the first players. The general algorithm of both versions is that explained above and schematized in Algorithm *1*. The *Immigrate* functions which are the ones explained at Section *3.3.1* and Section *3.3.2* for each player. The Migration Epoch procedure is what distinguishes the two slightly different versions of the studied Hybrid Method. The algorithm of the Migration Epoch procedures, for the two versions of the Migration Epoch, are presented in Section *3.4.1* and Section *3.4.2*. The two versions are named *H0* and *H1*, respectively.

The configuration of the optimization algorithms is detailed in Section *3.5*. The same configuration is used when used as players and when optimizing with the method alone. The configuration of the Hybrid Method is explained in more detail in Section *3.5*.

3.4.1 Migration Epoch *H0*

In this section the version *H0* of the Migration Epoch is detailed. The algorithm is summarized in Algorithm *2*. As explained before the first player is an Evolutionary Algorithm. The players are stored in a vector, so the first player is *player[0]*. The next players, *player[i]* for $i > 0$, are the Conjugate Gradient players, one for each objective function.

The algorithm of the Migration Epoch *H0* starts evaluating an *if/else* condition, see Algorithm 2. This first condition establishes when an exchange of information between players (i.e., migration of individuals) occurs without comparing the values of the objective functions of the players. That means that there is the possibility to migrate some individuals without taking into account if the individual to migrate is better or not than the ones already in the player of destination. If the condition of the first *if* is met, an exchange of information is performed through the *Immigrate* function of the players. The condition establishes that at the first iteration ($iter == 0$) the migration always will occur. Another condition is set with an *or* dependence, which establishes that if the last player that has been evaluated is the first one (*players[0]*) there will be a migration too. When one of these two conditions is met the best individual of the first player is migrated to the Conjugate Gradient players, which will be used as a starting point for its optimization process.

If the first conditional is not met, a second *if* condition is evaluated taking into account the values of the objective functions achieved so far. The inner *if* condition compares the objective function of the last evaluated player (*this-player*) against the objective function of the first player, which in this case is an Evolutionary Algorithm. The condition *this-player-Bestfits* < *players[0]-Bestfits* is clear in a single-objective optimization case, as it is a direct comparison between values of the objective function. On the other hand, for a multi-objective optimization case, the comparison is not direct, as different objective functions cannot be compared between them. In the case of a multi-objective function, the algorithm tracks which objective function is shared between each player and conducts the comparison accordingly. If the objective function of *this-player* is better (lower) than the first player this individual is migrated to the rest of players. The algorithm and implementation are extended to N players and M number of objective functions and track this automatically according to its configuration. For clarity, in the case of two objective functions, the first objective function of the first player (GA) is compared against the objective function of the second player (CG). The second function of the first player (GA) is compared with the objective function of the third player (CG). The implemented Conjugate Gradient is single objective, in any case, a player with a Conjugate Gradient optimizes two objective functions.

If the objective function of *this-individual* is not better, the inner *else* part of the algorithm is conducted. In the function *GetMinBestfits* a search for the best individual (*Bestfit*) among all players is performed, and the individual is migrated to

this-player, the last player that was executed.

Algorithm 2: Migration Epoch version *H0*.

After computing a full iteration of a player the MigrationEpoch(this-player) function does:

```
if iter = 0 or this-player = players[0] then
    foreach player ∈ players do
        player-Immigrate(this-player-Bestfits);
else
    if this-player-Bestfits < players[0]-Bestfits then
        foreach player ∈ players do
            player-Immigrate(this-player-Bestfits);
    else
        Bestfit = GetMinBestfits;
        this-player-Immigrate(Bestfit);
```

Following the General Algorithm explained in Section *3.1*, the Hybrid Method initially runs the Evolutionary Algorithm player. After the first iteration, the best individual found by the Evolutionary Algorithm is transferred to the Conjugate Gradient player (first *if* of the Algorithm *2*, because of the condition: *iter == 0*) which yields to calling the *Immigrate* function before the first iteration of the Conjugate Gradient players. The *player-Bestfits* is the individual with the best objective function found so far for the last player executed. It is updated every time a better individual is found and each player has its own. The Conjugate Gradient player uses this individual as the starting point of its internal algorithm, as defined in the *Immigrate* function, for its own iteration. The iteration of the Conjugate Gradient consists of computing the gradient at the location of the starting point and performing a line search. After the iteration of the Conjugate Gradient players, another Migration Epoch occurs. If the best individual found by the Conjugate Gradients outperforms the best individual found by the Genetic Algorithm, then the best individual of the Conjugate Gradient is sent to the Genetic Algorithm.

3.4.2 Migration Epoch *H1*

The version *H1* of the Migration Epoch is a slight modification of the *H0*. The modified algorithm is detailed in Algorithm *3*. The only difference between both is that the condition of the first *if* of the algorithm has been changed. As explained above, this condition controls the exchange of information between players without taking into account if the objective functions have improved. The condition *this-player == players[0]* has been removed, thus the migration of individuals without the comparison of the objective functions only takes place in the first iteration. The

rest of the algorithm remains the same.

Algorithm 3: Migration Epoch version 1.

After computing an iteration of a player the MigrationEpoch(this-player) function does:

```
if iter = 0 then
    foreach player ∈ players do
        player-Inmigrate(this-player-Bestfits);
else
    if this-player-Bestfits < players[0]-Bestfits then
        foreach player ∈ players do
            player-Inmigrate(this-player-Bestfits);
    else
        Bestfit = GetMinBestfits;
        this-player-Inmigrate(Bestfit);
```

In the version *H0* of the algorithm, due to the first conditional (*this-player == players[0]*) each time that the Evolutionary Algorithm performs an iteration the rest of the players, the Conjugate Gradient players, are transferred a new individual from the Evolutionary Algorithm, without taking into account if this individual is better than the one that the Conjugate Gradient is currently working. This does that the Conjugate Gradient interrupts its search even if it was going in a good direction and may change completely the region of the search space for the next step of the Conjugate Gradient. This has the possible effect that the Conjugate Gradient can only perform one iteration of the method and then has to restart from the new individual. The Conjugate Gradient computes the gradient at the starting point, then computes the search direction and makes a line search. Once the minimum in that direction is found, one iteration is finished. In the next iteration, it recomputes the gradient again from the last point and repeats the process. This behavior may not be possible with the condition of the Migration Epoch *H0*.

With the elimination of this condition, the version *H1* of the algorithm, the Conjugate Gradient is allowed to continue its iterations as it would do running alone, and only change the location if of the search if another region is more promising. As discussed in the Chapter *4* the performance of the two versions of the Hybrid Method depends on the application case.

3.5 Configuration of the optimization algorithms and Hybrid Method

In this section, the configuration of the different optimization algorithms used in this study is presented. The configuration of the optimization algorithms, when used as players inside the Hybrid Method, is the same as when running alone.

The General Algorithm and *Immigrate* function have been maintained the same through all tests. Four different combinations have been tested combining the Migration Epoch version and the optimization algorithm used by each player. In all cases, the first player is a population based algorithm. The additional players are Conjugate Gradients and there is one for each objective function. In Table 3.1 the nomenclature for the different combinations of Migration Epoch versions and Evolutionary Algorithms is presented.

Table 3.1: Nomenclature of the Hybrid Methods versions.

Name	Migration Epoch	Exploration (1st player)	Exploitation One player for each FO
H0-LG	*H0*	LAMU	GRAD
H1-LG	*H1*	LAMU	GRAD
H0-PG	*H0*	MOPSO	GRAD
H1-PG	*H1*	MOPSO	GRAD

The LAMU method is a modified version of the NSGAII algorithm, see Section *3.5.3* for more details.

3.5.1 Hybrid Method single objective configuration

In this section the configuration of the tested Hybrid Methods for the single objective optimization cases is detailed. As explained above, the Hybrid Method for the single objective optimization case uses 2 players. The first one is a population based algorithm. As explained in Table 3.1, a Genetic Algorithm and a Particle Swarm Optimization have been tested. Both players optimize the full problem, so they optimize the same objective function and work with all the design variables of the test problem. The General Algorithm of the Hybrid Methods for the single objective configuration is detailed in Figure 3.1. It is the particularization of the General Algorithm explained in Section *3.1*.

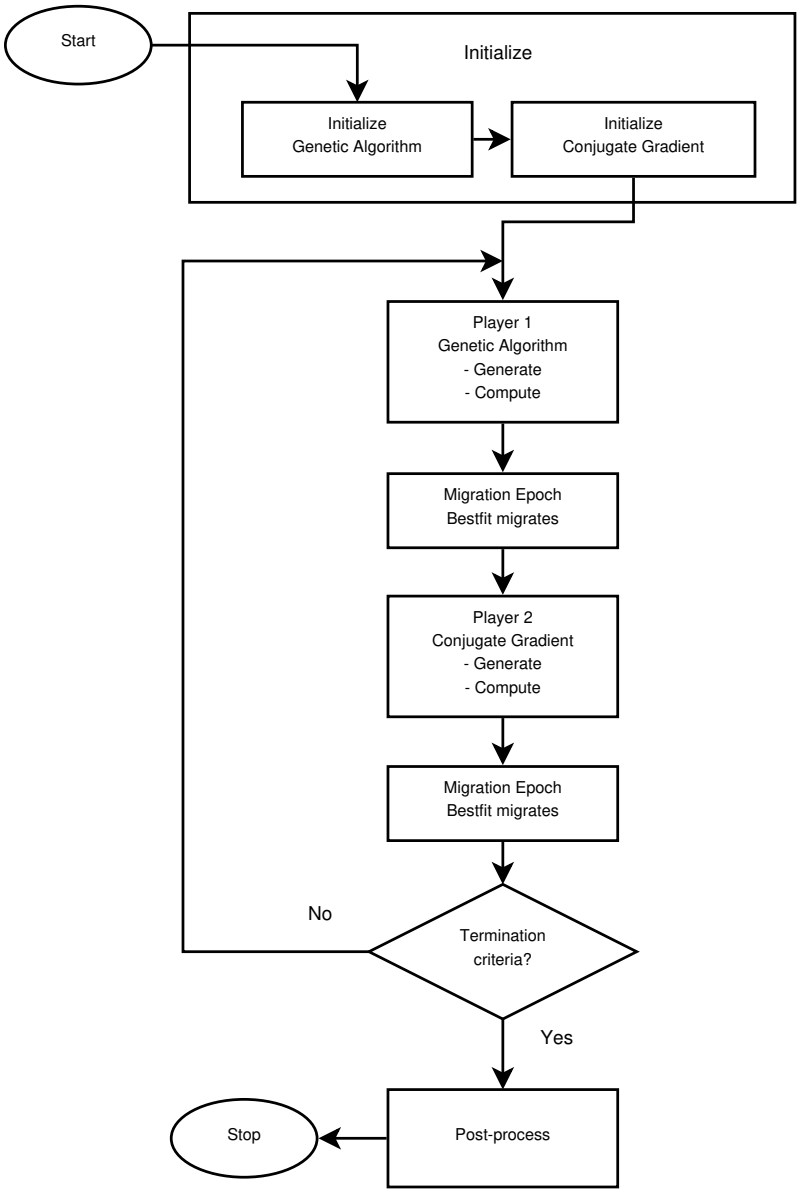


Figure 3.1: General Algorithm of the tested Hybrid Method with 1 objective function.

The first step is to initialize both players. After the initialization, the optimization loop is started. Inside the optimization loop, the first player (i.e., the population based algorithm) is run for one iteration. After the computation of its population is finished, a Migration Epoch occurs. In the Migration Epoch the best individual is transferred to the Conjugate Gradient player. The Conjugate Gradient player computes the gradient around the transferred individual and performs the line search, which completes an iteration of the Conjugate Gradient player. Then, another Emigration Epoch is conducted, and the best individual found by the Conjugate Gradient can migrate to the first player, according to the Migration Epoch implementation as explained in Section *3.4.1* and Section *3.4.2*. This sequence is repeated until the termination criteria is met. Finally, the post process of the optimization

is performed.

3.5.2 Hybrid Method two objective configuration

The configuration of the Hybrid Methods for the two objective test cases has a total of three players. The first player is a population based optimization algorithm that optimizes the full problem. It optimizes the two objective functions and uses all the design variables. The other two players are Conjugate Gradients. Each Conjugate Gradient deals with one of the objective functions and all the design variables. The particularization of the General Algorithm to this configuration is schematized in Figure 3.2.

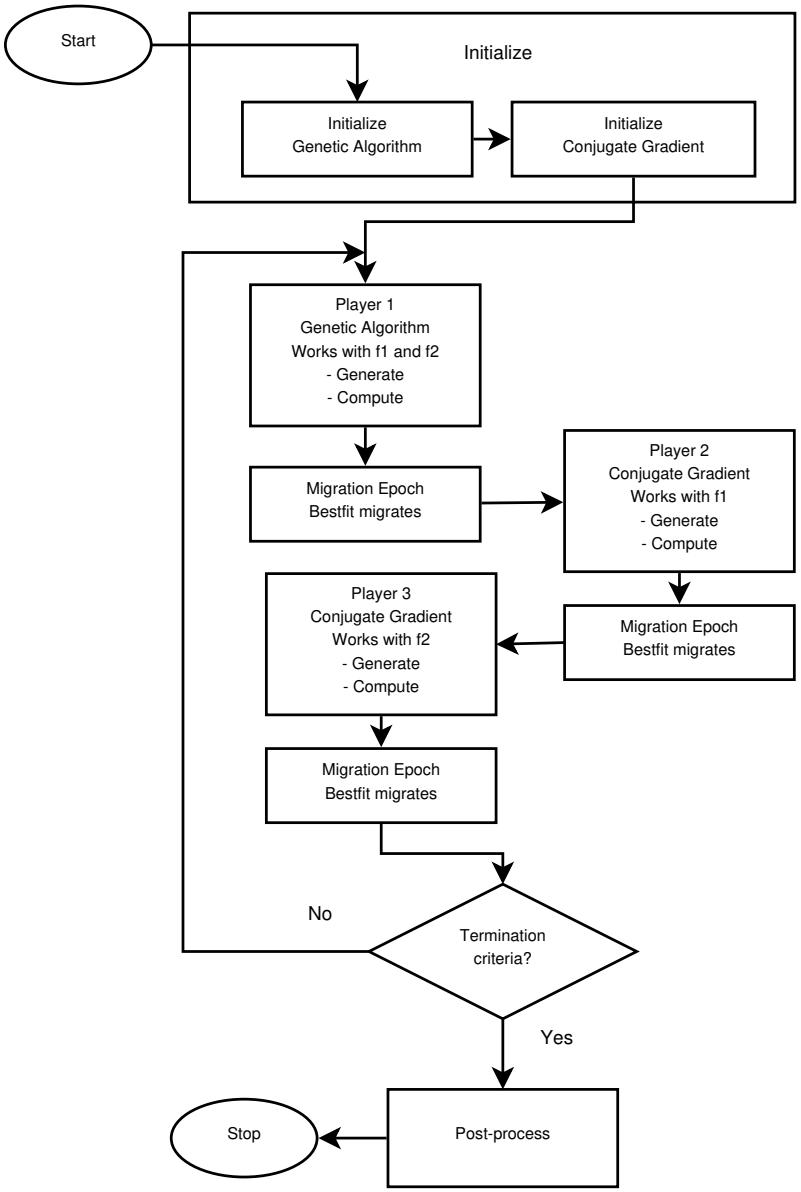


Figure 3.2: General Algorithm of the tested Hybrid Method with 1 objective function.

The workflow of the algorithm is the same as the explained for the one objective function case, but adding the evaluation of the third player and another Migration Epoch before closing the main loop. The comparison of the objective functions is performed taking into account which player optimizes which objective function. It is not possible to compare different optimization functions between them. The first objective function of the first player is compared with the objective function of the second player and the second objective function of the first player is compared with the objective function of the third player.

3.5.3 Genetic Algorithm configuration

The Genetic Algorithm is configurable in the type of operators for the crossover, mutation, and selection operators as explained in Section *2.4*. In addition, the operators themselves can have configurable parameters. The configuration of the Genetic Algorithm is specified in Table 3.2.

Table 3.2: Configuration of the Genetic Algorithm.

Parameter	Value/Type
Selection operator	$\mu + \lambda$ & Crowded-Comparison Operator [18]
Crossover operator	Simulated Binary Crossover [17]
Mutation operator	Polynomial Mutation [19]
Probability of crossover	0.9
Probability of mutation	0.1
Population size	20

This configuration is common in both Genetic algorithms used in this study. The algorithm of NSGAII has not been modified, the only modification made in its code is to add a routine to save a file with all computed individuals during the optimization, which allows obtaining the full Pareto front and the convergence.

As detailed in Table 3.1, the Hybrid Methods use a modified version of the NSGAII algorithm, which is called LAMU, and takes its name from the $\mu + \lambda$ strategy of the selection operator. The modifications increase the flexibility to pair it with an external evaluation tool, parallelize the computation of the population and communicate with a High Throughput Computing environment, see HTCondor [66], among other minor features. A more relevant modification is that it tracks the individuals that are already computed and are transferred to the offspring population without mutation or crossover being applied. With this information, the program avoids repeating the calculation thus it does not count as an evaluation. This feature is really valuable in optimization cases with high computational costs.

3.5.4 Particle Swarm Optimization configuration

The Particle Swarm Optimization has fewer configuration options than the Genetic Algorithm. The implementation used here has only three configurable values, the inertia weight and the two acceleration constants which are usually set to the same value, as explained in Section *2.5*. The configuration of the Particle Swarm Optimization is specified in Table 3.3.

Table 3.3: Configuration of the Particle Swarm Optimization.

Parameter	Value
Inertia weight (ω)	0.4
Acceleration constants (c_1 and c_2)	1.0
Population size	20

3.5.5 Conjugate Gradient configuration

The Conjugate Gradient algorithm is explained in Section *2.3*. The configuration of the Conjugate Gradient is specified in Table 3.4.

Table 3.4: Configuration of the Conjugate Gradient algorithm.

Parameter	Value/Type
Search Direction Method	Fletcher-Reeves [31]
Optimal Step Size Method	Golden Section [34]
Epsilon for numerical diferentiation	1.0×10^{-6}
First step size	1.0×10^{-3}
Optimal step size tolerance	1.0×10^{-3}

In figures and tables of Chapter *4* the Conjugate Gradient method is abbreviated as GRAD.

3.6 Implementation notes

The implementation of the Hybrid Method is coded in *C++* and it follows the Object Oriented Programming paradigm. An abstract class named *OptimizationAlgorithm* serves as a base class for the implementation of each optimization method. The Genetic Algorithm, the Particle Swarm Optimization, and the Conjugate Gradient are coded in classes that derive from the *OptimizationAlgorithm* class.

The main methods of the class *OptimizationAlgorithm* are those outlined in Algorithm *1*. Those are the *Initialize*, *Generate*, *Compute* and *PostProcess* methods. The *Immigrate* method also belongs to the *OptimizationAlgorithm* and has to be

defined for each derived optimization method in order to enable the possibility to use such algorithm as a player of a Hybrid Method.

This architecture allows the flexibility to create different instances of each optimization method and store them in a vector of *OptimizationAlgorithm* pointers. This vector is named *players*. The last piece of the main code is a function that performs the calls detailed in the Algorithm *1*.

The implementation has more code and functionality to read configuration files, write files with results, post-process results, restart optimizations, etc. which are not relevant to the main optimization algorithm discussed in this work.

3.7 Summary

In this chapter the proposed new Hybrid Method has been explained in detail. The three main components of the Hybrid Methods are the General Algorithm, the *MigrationEpoch* algorithm and the *Immigrate* method of each player. The General Algorithm is the same for all the versions studied of the Hybrid Methods. Two versions of the *MigrationEpoch* algorithm have been defined (*H0* and *H1*). Four versions of the Hybrid Method have been defined which are combination of the two versions of the Migration Epoch and the optimization algorithm used by the exploration player, which uses a Genetic Algorithm or a Particle Swarm Optimization, see Table 3.1.

Chapter 4

Test cases and applications

In this section, the performance of the presented hybrid optimization methodology to some relevant benchmark problems and real-word applications is compared with other existing optimization methods. The benchmark problems are mainly mathematical functions with specific particularities of interest. They are considered as reference benchmark test cases for several researchers. The criteria to present these benchmark problems is the number of objective functions; leading to two categories; namely single or multi-objective problems.

Regarding the real-world application, an aeronautical engineering problem that the author had the opportunity to work with is described.

All optimization algorithms were allowed to run with the same amount of evaluations of the objective functions, although some graphics are cut at different numbers of evaluations to show the detail needed to evaluate the results. Some runs stopped without using all available evaluations of the objective functions, this happens for two reasons. The first one is that the algorithm is not able to improve although it keeps trying and exploring the search space. The second reason is that the algorithm gets trapped in a local minimum and is not able to generate new individuals. This happens frequently with the Conjugate Gradient. In some cases, it also has been observed with the Particle Swarm Optimization where the algorithm gets trapped in some sort of elitism and is not able to generate an offspring different from the previous population.

4.1 Performance measures

The evaluation and comparison of the performance of different optimization methods in single-optimization problems are relatively straightforward. The convergence of the method can be used as a metric. The convergence is the relation between the number of evaluations of the objective function performed by the optimization method with the best value of the objective function achieved so far.

The other aspect to take into account is the robustness of the method. Metaheuristics methods work with probability variables which makes them behave dif-

ferently depending on the seed. For this reason, a single run of the method is not representative of the method's performance, as this single run could be in the best or worst extreme of its behavior and not be representative. To consider this in the evaluation of the performance multiple runs have to be conducted and the variability is taken into account in the analysis. In the deterministic methods, even if there are no probability variables, the solution depends on the starting point of the method. For functions with different local minimums, this is relevant as the algorithm may converge to a local minimum and not to a global minimum.

In multi-objective problems, the evaluation of the performance is more difficult as the solution of each optimization method contains multiple individuals. As stated by Deb et al. [67], in multi-objective optimization there are two goals. The first one is to converge to the Pareto optimal solution, which means to get as close as possible to the region of the optimal solutions in a multi-dimensional space. The second one is to achieve a spread solution across the whole Pareto Front and not only converge to some regions.

To take into account both goals a single metric is not adequate. Different metrics have been proposed, for example see Zitzler [68], Fonseca and Fleming [69] and Deb [16]. In this study, the metric used to evaluate the convergence is the convergence metric proposed by Deb [16] which is detailed in Section *4.1.1*. The metric used to evaluate the spread of the solution is also proposed by Deb [16] and is detailed in Section *4.1.2*.

The convergence of the solution is evaluated with the number of iterations of the optimization method or with the number of evaluations of the objective functions. In this study and in all test cases the convergence is evaluated considering the number of objective functions evaluations. There are two reasons for that decision. The first one is that the optimization algorithms evaluated in this study are very different, in how they operate internally and the number of evaluations they perform at each iteration, thus it is not fair to compare them with the number of iterations. The other reason is that in cases with high computational costs the bottleneck of the process is the evaluation of the objective functions and not the cost of the optimization algorithm internal operations.

4.1.1 Distance metric

The distance metric Υ gives a measure of the convergence in a multi objective optimization problem. A set of the Pareto optimal solutions are required to be known. This requirement limits the applicability of this metric to problems where the exact solution is known.

The metric is calculated following this procedure:

1. Obtain a set of points of the Pareto optimal front.
2. For each individual of the solution obtained by the optimization algorithm: Compute the minimum Euclidean distance between this point and the rest of points of the Pareto optimal front.

3. The average of the distances of the step 2 is the value of the Υ metric.

To take into account the robustness of each optimization method multiple runs have to be conducted. The average and standard deviation of the results gives the information to evaluate the robustness of the method.

Υ takes a value of zero when the solution has converged to the exact reference Pareto optimal set. Even if the solution is on the Pareto optimal space, but its points are between the reference ones, the metric does not evaluate to 0. For more detail on the distance metric refer to [67].

4.1.2 Spread metric

The spread metric Δ gives a measure on how uniformly distributed are the solutions across the Pareto front. The equation of the metric is defined as

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} |d_i - \overline{d}|}{d_f + d_l + (N-1)\overline{d}} \tag{4.1}$$

where d_f and d_l are the Euclidean distances between the solutions at the extremes of the obtained non-dominated solution and the extremes of the Pareto optimal set. The extreme solutions are the best individuals when ranked by a single objective function inside a set of non-dominated solutions. d_l is the distance between the individual with the best f_1 value in the non-dominated set of solutions and the best f_1 value of the Pareto optimal set (the exact solution). d_l is the equivalent distance considering f_2. d_i are the distances between consecutive individuals of the non-dominated solutions. $\overline{d}$ is the average distance between the points in the solution. And finally N is the number of individuals in the non-dominated solution.

Δ takes a value of zero for the most widely and uniform distribution as it measures the non-uniformity of the solution. The metric increases its value when the distance between two adjacent non-dominated solutions is different than the mean value of the distances. For more detail on the spread metric refer to [67].

4.2 Single-objective mathematical test cases

A set of single-objective benchmark problems have been selected from the literature in order to compare the performance of the proposed methodologies through a systematic analysis. The reader will identify from simple functions with several design variables to complex ones with few design variables, and some intermediate cases. The selection has been done to cover a broad spectrum of possibilities, although the factor the author acknowledges the limitation in the number of the selected functions in comparison to the large amount of available ones in the literature.

The Genetic Algorithms, the Particle Swarm Optimization and the Hybrid Methods have been run twelve independent times to take into account the robustness of each

method. The Conjugate Gradient has been run 20 times, one starting at each point of the first population of the first genetic Algorithm run to take into account the high dependence on the initial solution.

4.2.1 Ackley function

The Ackley function was proposed in 1987 by Ackley [70]. It is a non-convex function well-known as an optimization benchmark. The corresponding benchmark test presents a single objective problem but, on the other hand, the number of design variable can be set up from 1 to d. The interest of the function remains on the complex shape of the function itself. The shape has a valley on the center, with the global minimum, and a lot of additional local minimums around it. The objective function of the problem is defined as

$$\begin{gathered} f(\mathbf{x}) = -a \exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right) + a + \exp(1) \\ a = 20 \\ b = 0.2 \\ c = 2\pi \end{gathered} \tag{4.2}$$

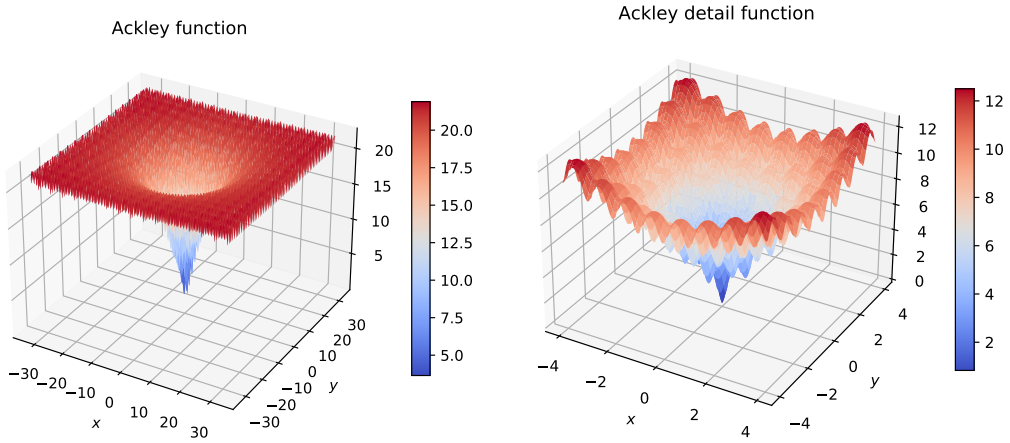
The Ackley function is usually evaluated in the range

$$x_i \in [-32.768, 32.768], \text{for all } i = 1, ..., d$$

and the global minimum is located at

$$f(\mathbf{x}^*) = 0, \text{ at } \mathbf{x}^* = (0, ..., 0).$$

Figure 4.1 shows the shape of the function with 2 design variables as it has been used in the tests and a detail around the global minimum.

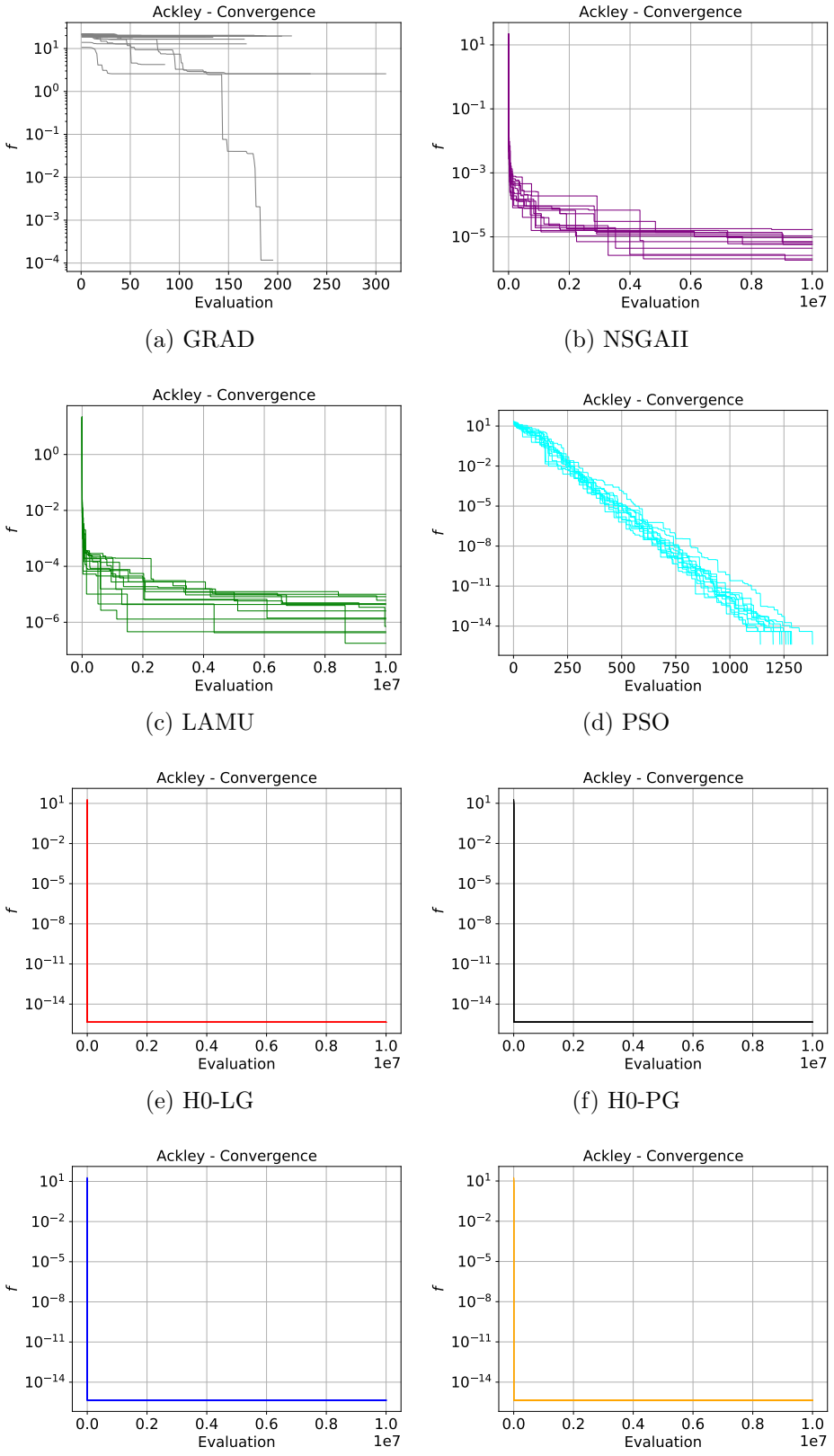


(a) (b)

Figure 4.1: (a) Ackley function and (b) zoomed view of the minimum location.

Figure 4.2 presents the convergence for the Ackley function of all independent runs grouped by the optimization method. Of the 20 runs made with the Conjugate Gradient only one really converged near the global minimum achieving a value of the objective function of $f = 10^{-4}$. This result could have been anticipated as the function has a lot of local minimums and the Conjugate Gradient does not have the mechanism to escape from them. The NSGAII algorithm achieved a similar value in all runs, and even improved it ($f = 10^{-5}$) in some of them. The downside of the NSGAII algorithm is the high number of evaluation needed to achieve this result. The LAMU Genetic Algorithm achieved slightly better results than the NSGAII, probably because of the tracking of already computed individuals. The Particle Swarm Optimization is the non-hybrid method that better performs as it achieved the exact solution ($f = 10^{-14}$) in around 1250 objective function evaluations in all runs demonstrating a great accuracy and robustness.

The Hybrid Methods performed extremely well reaching the exact solution ($f = 10^{-14}$) after about 25 objective function evaluations. The Hybrid Method could take perfectly take advantage of the strong points of each player. Both Evolutionary Algorithms found a sufficiently close point to the global minimum in the very first generation. The Conjugate Gradient could reach the global optimum in the first iteration. In this test case there is no discussion that the Hybrid Methods over performed the others, both in accuracy and robustness.



(a) GRAD (b) NSGAII

(c) LAMU (d) PSO

(e) H0-LG (f) H0-PG

(g) H1-LG (h) H1-PG

Figure 4.2: Convergence of each solver for all runs of the Ackley function.

Figure 4.3 presents the convergence of all runs and solvers in the same graphic. Figure 4.4 shows the mean value of the convergence for each method and the Table 4.1 shows the mean and standard deviation of the results.

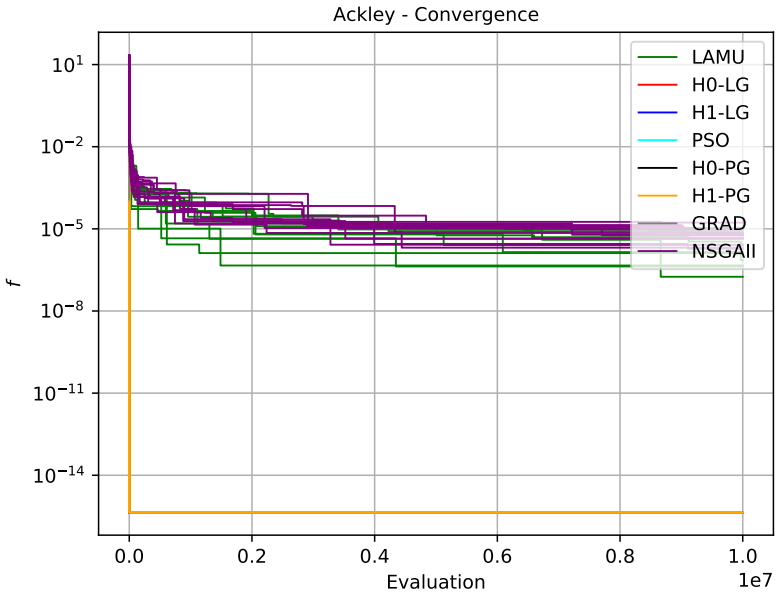


Figure 4.3: Ackley function convergence.

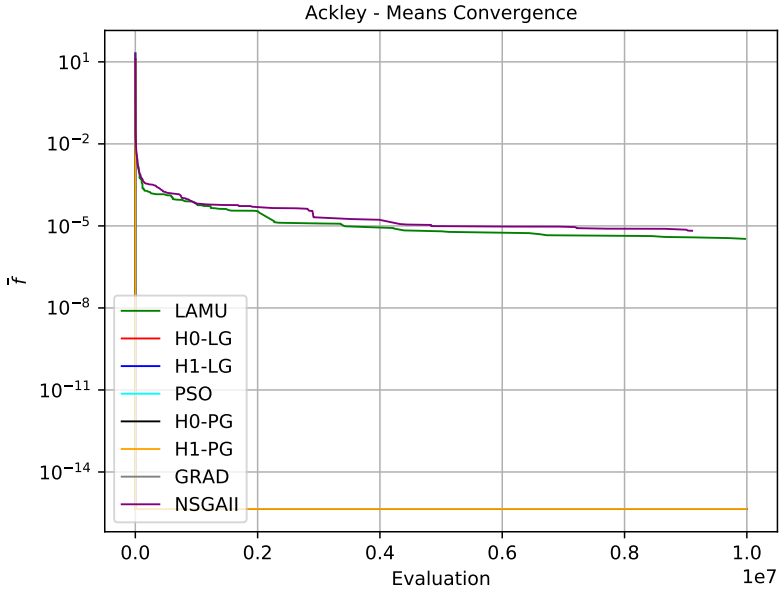


Figure 4.4: Ackley function means convergence.

Method	$\overline{f}_{\text{best}}$	$\sigma_{f_{\text{best}}}$
LAMU	3.361×10^{-6}	1.099×10^{-11}
H0-LG	4.441×10^{-16}	0.000
H1-LG	4.441×10^{-16}	0.000
PSO	1.628×10^{-15}	5.355×10^{-30}
H0-PG	4.441×10^{-16}	0.000
H1-PG	4.441×10^{-16}	0.000
GRAD	1.562×10^{1}	4.959×10^{1}
NSGAII	6.644×10^{-6}	1.783×10^{-11}

Table 4.1: Mean and standard deviation of the best result comparison for the ACKLEY problem.

4.2.2 Levy function

The Levy function [71] has different valleys in one direction but has a slight gradient in the other. This presents a challenge for both evolutionary and gradient based algorithms. The function can be used with d variables and in this study two have been used. Figure 4.5 shows the shape of the function. The objective function of the problem is defined as

$$\begin{aligned} f(\mathbf{x}) = \sin^2(\pi\omega_1) + \sum_{i=1}^{d-1}(\omega_i - 1)^2 \left[1 + 10\sin^2(\pi\omega_i + 1)\right] + \\ + (\omega_d - 1)^2 \left[1 + \sin^2(2\pi\omega_d)\right] \\ \omega_i = 1 + \frac{x_i - 1}{4} \text{for all } i = 1, ..., d \end{aligned} \tag{4.3}$$

The Levy function is usually evaluated in the range

$$x_i \in [-10, 10], \text{for all } i = 1, ..., d$$

and the global minimum is located at

$$f(\mathbf{x}^*) = 0, \text{ at } \mathbf{x}^* = (1, ..., 1).$$

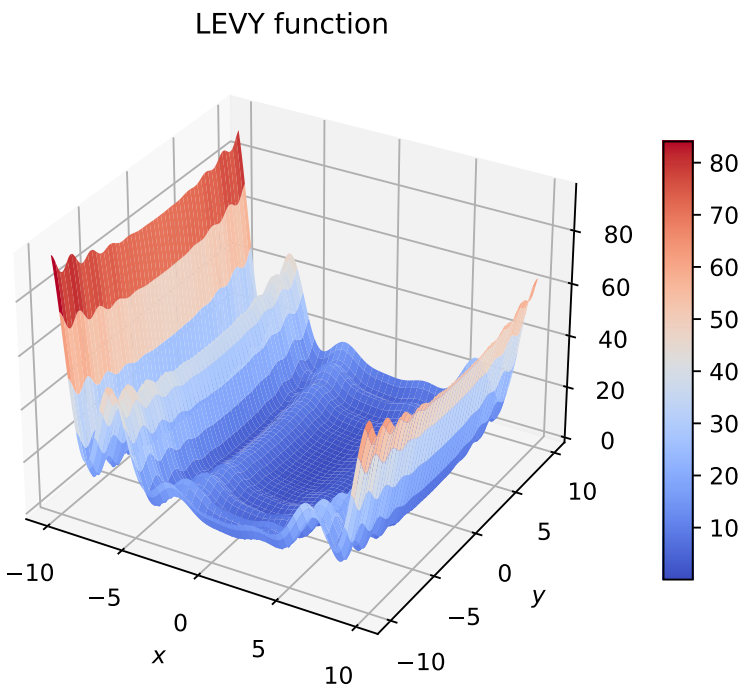
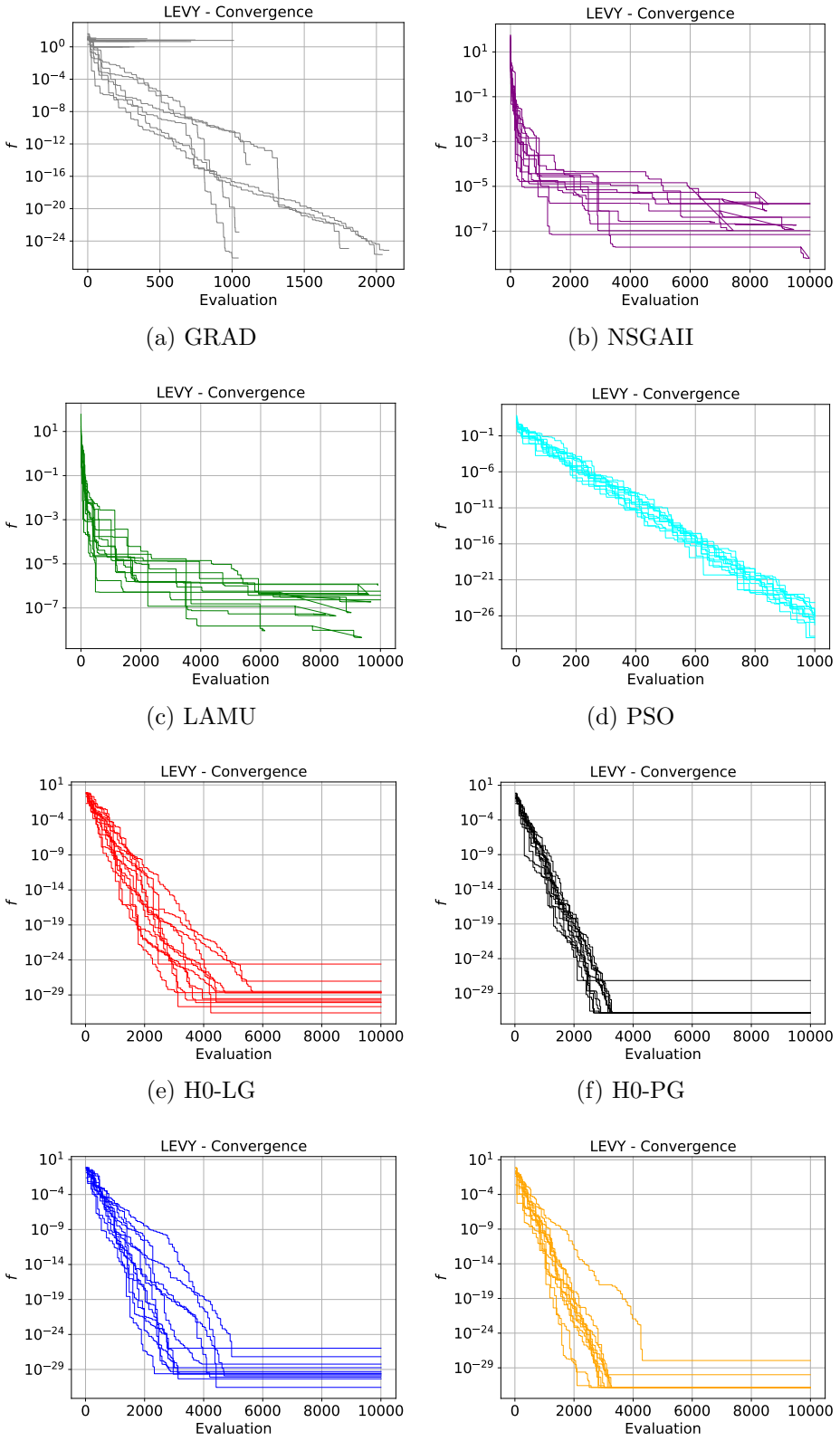


Figure 4.5: Levy function.

Figure 4.6 presents the convergence for the Levy function of all independent runs grouped by the optimization method. The Particle Swarm Optimization has clearly outperformed the other optimization methods in this test case. It reached the global optimum in all runs, consistently and with the higher rate of convergence. The Conjugate Gradient performance shown in this case is the best so far with 6 out of 20 runs achieving the global minimum with high accuracy, but it still shows a high dependence on the starting point, as expected. The two Genetic Algorithms performed very similar between them, achieving the global minimum in all cases, but without the impressive accuracy shown by the Particle Swarm Optimization. The LAMU algorithm shows a slight higher accuracy than the NSGAII.



(a) GRAD (b) NSGAII

(c) LAMU (d) PSO

(e) H0-LG (f) H0-PG

(g) H1-LG (h) H1-PG

Figure 4.6: Convergence of each solver for all runs of the Levy function.

The Hybrid Methods which contain the Particle Swarm Optimization player (H0-PG and H1-PG) have performed worst than the pure PSO Method. On the other hand the ones that contain the LAMU player have improved the pure algorithm in both accuracy and rate of convergence. Figure 4.7 presents the convergence of all runs and solvers in the same graphic. Figure 4.8 shows the mean value of the convergence for each method and the Table 4.2 shows the mean and standard deviation of the results.

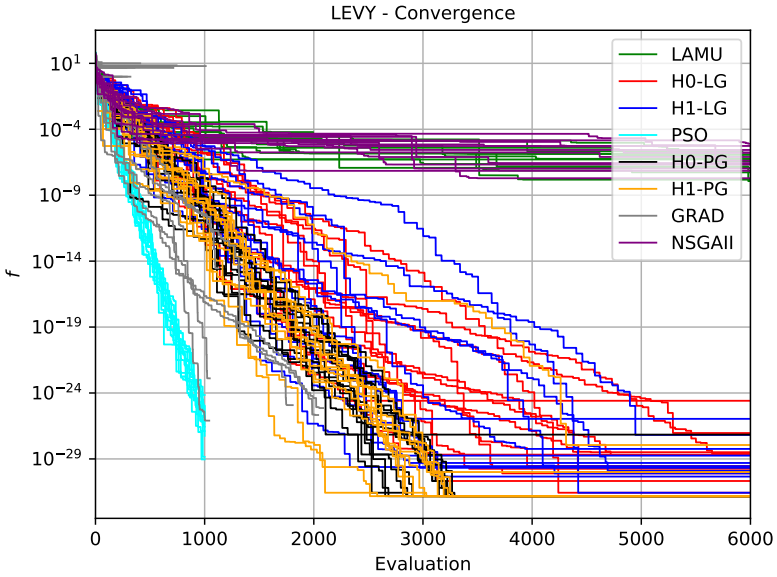


Figure 4.7: Levy function convergence.

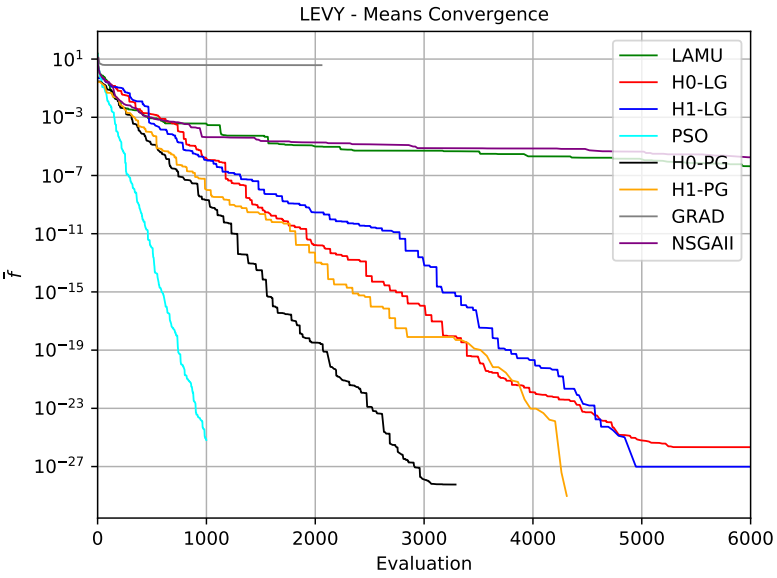


Figure 4.8: Levy function means convergence.

Method	$\overline{f}_{\text{best}}$	$\sigma_{f_{\text{best}}}$
LAMU	4.281×10^{-7}	1.639×10^{-13}
H0-LG	2.152×10^{-26}	5.509×10^{-51}
H1-LG	9.743×10^{-28}	9.880×10^{-54}
PSO	6.642×10^{-26}	4.074×10^{-50}
H0-PG	5.809×10^{-29}	4.047×10^{-56}
H1-PG	9.565×10^{-30}	1.074×10^{-57}
GRAD	3.817	1.370×10^{1}
NSGAII	1.785×10^{-6}	6.319×10^{-12}

Table 4.2: Mean and standard deviation of the best result comparison for the LEVY problem.

4.2.3 Eggholder function

The Eggholder function was proposed by Mishra [72]. It is a complex function due to its multiple valleys and a large number of local minimums. The function defines a single objective function with two design variables. Figure 4.9 shows the shape of the function. The objective function of the problem is defined as

$$f(\mathbf{x}) = -(x_2 + 47)\sin\left(\sqrt{\left|x_2 + \frac{x_1}{2} + 47\right|}\right) - x_1 \sin\left(\sqrt{|x_1 - (x_2 + 47)|}\right) \tag{4.4}$$

The Eggholder function is usually evaluated in the range

$$x_i \in [-512, 512], \text{for all } i = 1, 2.$$

and the global minimum is located at

$$f(\mathbf{x}^*) = -959.6407, \text{ at } \mathbf{x}^* = (512, 404.2319).$$

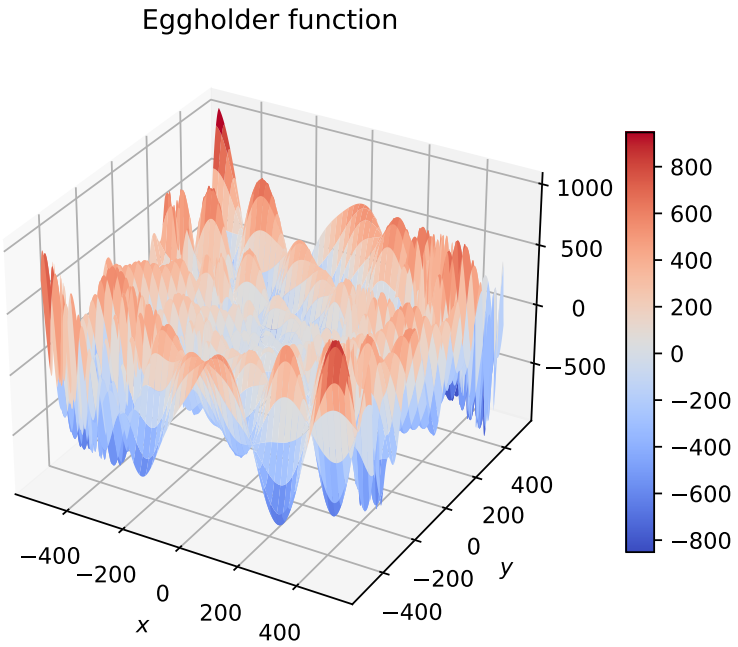
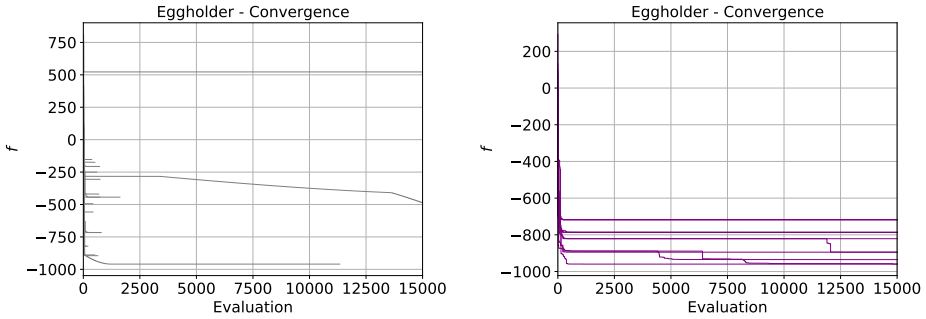


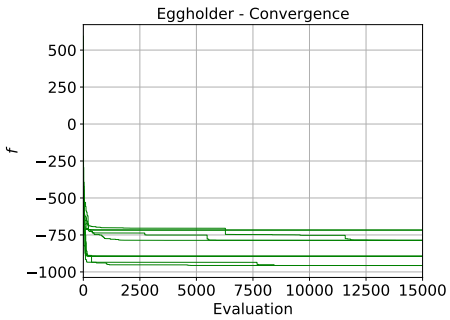
Figure 4.9: Eggholder function.

Figure 4.10 presents the convergence for the Eggholder function of all independent runs grouped by the optimization method. Only 1 of the 20 runs made with the Conjugate Gradient really converged to the global minimum. This result can be anticipated as the function has a lot of local minimums and the Conjugate Gradient dos not have the mechanism to escape from them. Considering that most of the runs stopped early and only 3 runs took a large number of iterations to finish, the method did not performed poorly, although it only achieved the global minimum once, the total amount of objective evaluations is not too big. The NSGAII algorithm achieved the global minimum in 2 of the 12 runs. In another 2 runs it got close in terms of the objective function value. The downside of the NSGAII algorithm is the high number of evaluations needed to achieve this result. The LAMU Genetic Algorithm achieved slightly better results than the NSGAII, probably because of the tracking of already computed individuals. In this test case the Particle Swarm Optimization was outperformed by the Genetic Algorithms, as it got stuck in local minimums.

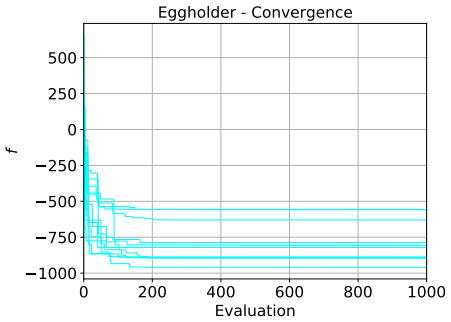


(a) GRAD

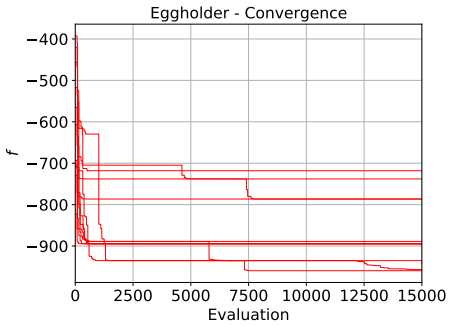
(b) NSGAII



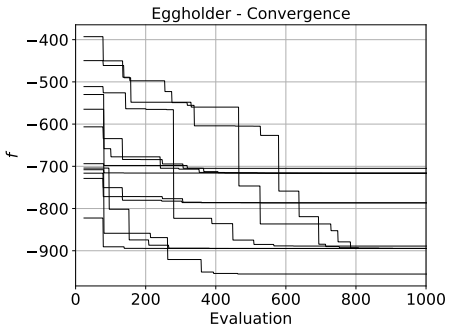
(c) LAMU



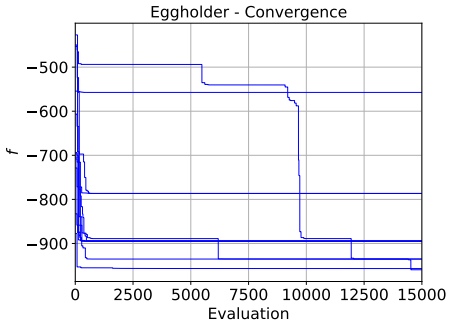
(d) PSO



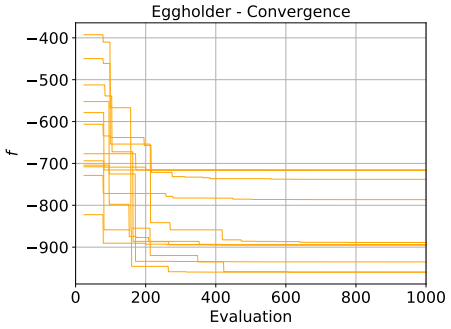
(e) H0-LG



(f) H0-PG



(g) H1-LG



(h) H1-PG

Figure 4.10: Convergence of each solver for all runs of the Eggholder function.

In this test case the Hybrid Methods did not stood out as in the previous one. Its performance is comparable to those of the Evolutionary Algorithms tested. Con-

sidering the means of the convergence, graphic in Figure 4.12, it shows a slight improvement in accuracy by the H0-LG and H1-LG Hybrid Methods over the rest of the algorithms. It also shows that the Hybrid Methods based which contain the Particle Swarm Optimization (H0-PG and H1-PG) also improves the performance over the pure Particle Swarm Optimization. Figure 4.11 presents the convergence of all runs and solvers in the same graphic. Table 4.3 shows the mean and standard deviation of the results.

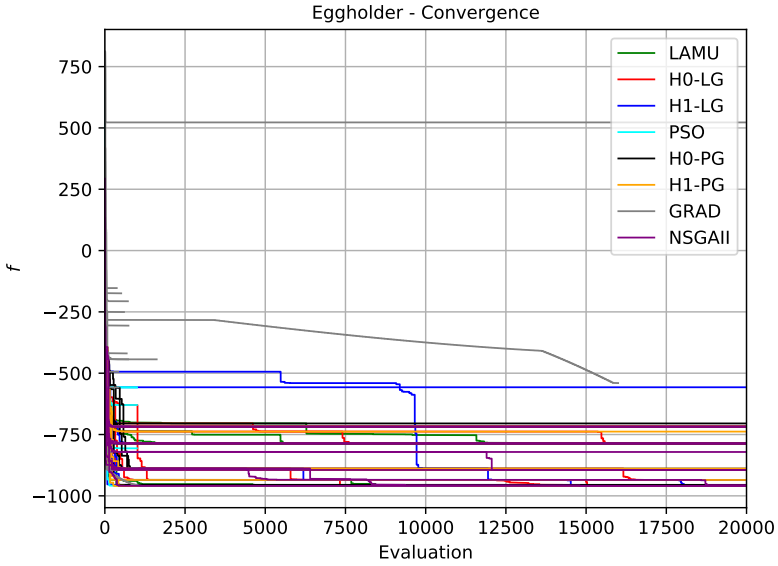


Figure 4.11: Eggholder function convergence.

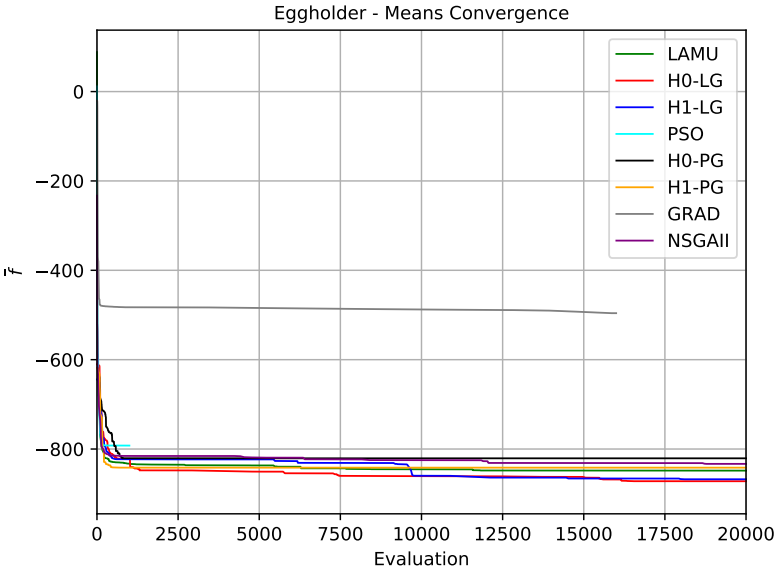


Figure 4.12: Eggholder function means convergence.

Method	$\overline{f}_{\text{best}}$	$\sigma_{f_{\text{best}}}$
LAMU	-8.483×10^2	7.283×10^3
H0-LG	-8.721×10^2	6.703×10^3
H1-LG	-8.677×10^2	1.291×10^4
PSO	-7.923×10^2	1.871×10^4
H0-PG	-8.208×10^2	8.458×10^3
H1-PG	-8.417×10^2	9.848×10^3
GRAD	-4.959×10^2	1.217×10^5
NSGAII	-8.332×10^2	9.219×10^3

Table 4.3: Mean and standard deviation of the best result comparison for the EGG problem.

4.2.4 Holder table function

The Holder table function [72] is a complex function with a large number of local minima in the center of the search space. The function has four global minima located at the four corners of the search space. The shape of the function is shown in Figure 4.13. The objective function of the problem is defined as

$$f(\mathbf{x}) = -\left|\sin(x_1)\cos(x_2)\exp\left(\left|1 - \frac{\sqrt{x_1^2 + x_2^2}}{\pi}\right|\right)\right| \tag{4.5}$$

The Holder table function is usually evaluated in the range

$$x_i \in [-10, 10], \text{for all } i = 1, 2.$$

and the four global minimums which are located at

$$\begin{aligned} f(\mathbf{x}^*) = -19.2085, \text{ at } \mathbf{x}^* = &(8.05502, 9.66459), \\ &(8.05502, -9.66459), \\ &(-8.05502, 9.66459), \\ &(-8.05502, -9.66459). \end{aligned}$$

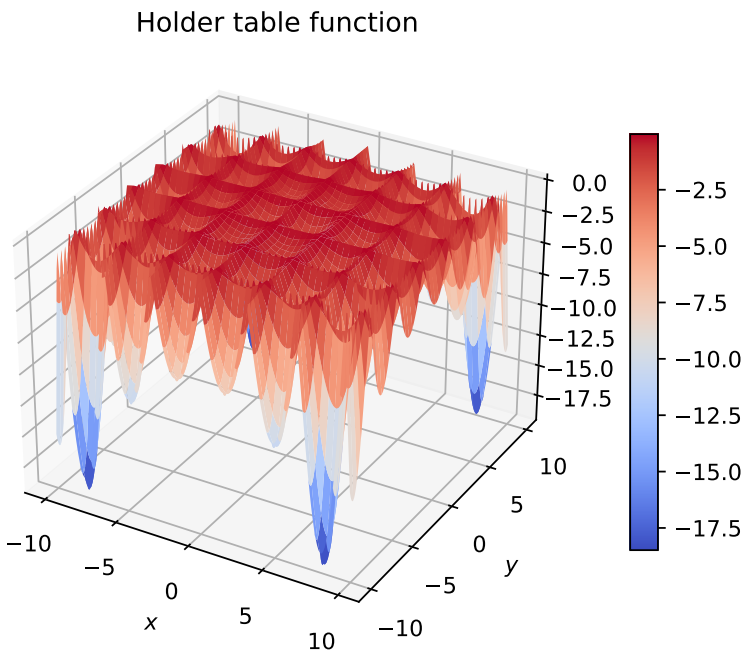
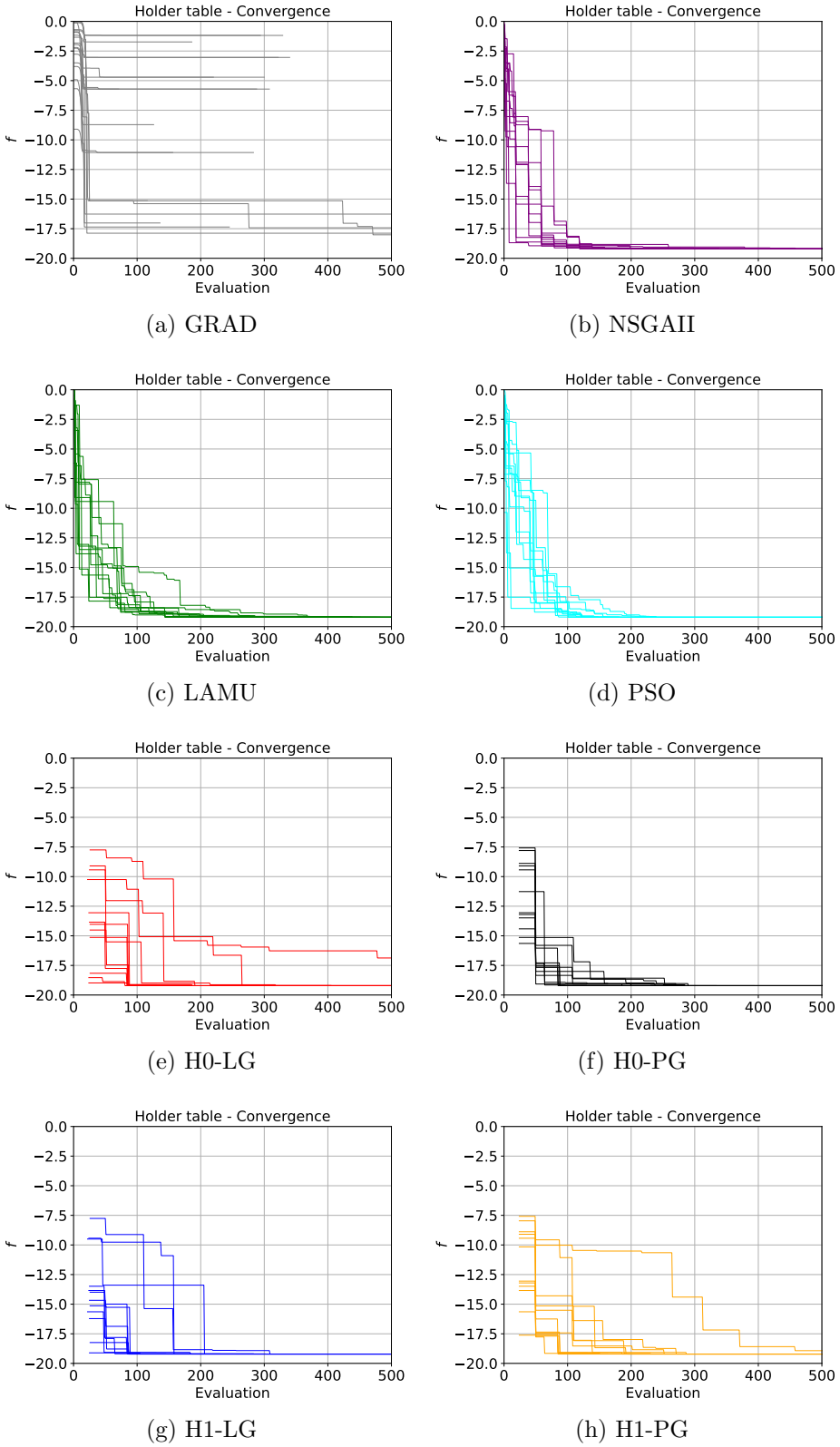


Figure 4.13: Holder table function.

Figure 4.14 presents the convergence for the Holder table function of all independent runs grouped by the optimization method. Only 2 of the 20 runs made with the Conjugate Gradient really converged to one of the 4 possible global minimum. This result could have been anticipated as the function has a lot of local minimums and the Conjugate Gradient dos not have the mechanism to escape from them. In this case, the runs of the Conjugate gradient took longer to converge to local minimums consuming more resources than in the previous test case. The NSGAII algorithm achieved the global minimum in all of the runs, as well as the LAMU algorithm, although the LAMU method shows less accuracy than the NSGAII in this test case. The Particle Swarm Optimization also achieved a global minimum in all runs and slightly outperformed the Genetic Algorithms in terms of accuracy.



(a) GRAD (b) NSGAII

(c) LAMU (d) PSO

(e) H0-LG (f) H0-PG

(g) H1-LG (h) H1-PG

Figure 4.14: Convergence of each solver for all runs of the Holder table function.

In this test case the Hybrid Methods performed better in terms of accuracy and

similar in the rate of convergence with a similar behavior than the Evolutionary Algorithms but increasing the accuracy, as expected because of the Conjugate Gradient contribution. Although, two Hybrid Methods have a run that did not reach the Global Minimums, the H0-LG and H1-PG. Figure 4.15 presents the convergence of all runs and solvers in the same graphic. Figure 4.16 shows the mean value of the convergence for each method and the Table 4.4 shows the mean and standard deviation of the results.

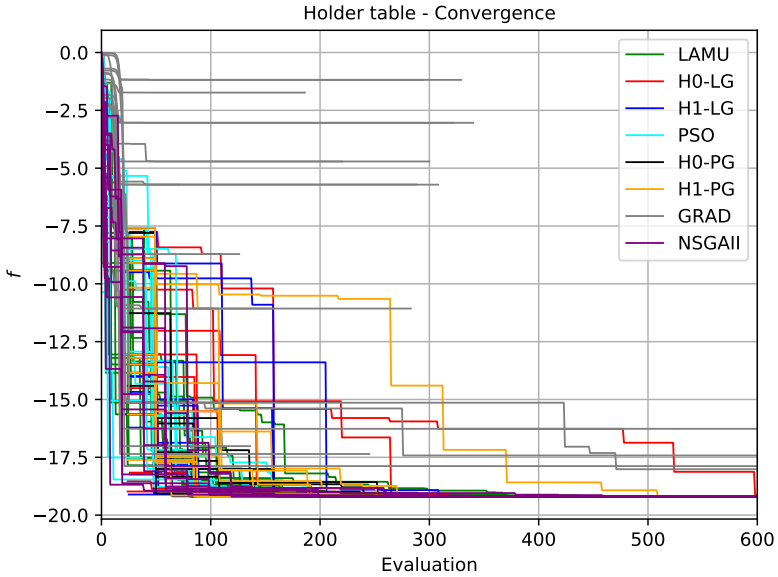


Figure 4.15: Holder table function convergence.

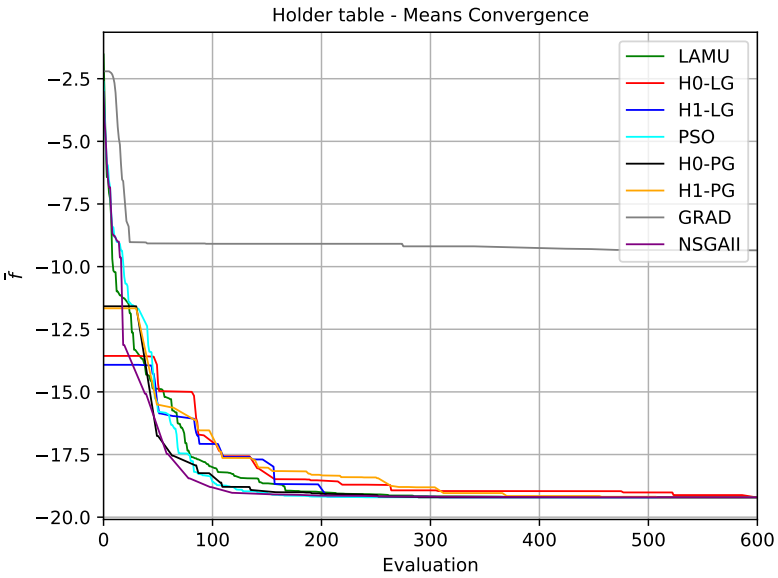


Figure 4.16: Holder table function means convergence.

Method	$\overline{f}_{\text{best}}$	$\sigma_{f_{\text{best}}}$
LAMU	-1.921×10^{1}	1.905×10^{-5}
H0-LG	-1.920×10^{1}	4.759×10^{-4}
H1-LG	-1.921×10^{1}	0.000
PSO	-1.921×10^{1}	7.014×10^{-9}
H0-PG	-1.921×10^{1}	0.000
H1-PG	-1.921×10^{1}	2.552×10^{-10}
GRAD	-9.334	4.092×10^{1}
NSGAII	-1.921×10^{1}	1.486×10^{-5}

Table 4.4: Mean and standard deviation of the best result comparison for the HOLDERTABLE problem.

4.2.5 Michalewicz function

The Michalewicz function [73] is a multimodal function that can be used with d variables and in this study two have been used. It has a single global minimum near the center of the search space but has $d!$ local minimums. Surrounding the valley that contains the global minimum the function is flat with the exemption of some lineal valleys in both directions which generates near local minimums with the value of the gradient close to zero, this increasing the difficulty to find the global minimum. Figure 4.17 shows the shape of the function for two variables. The objective function of the problem is defined as

$$f(\mathbf{x}) = -\sum_{i=1}^{d} \sin(x_i) \sin^{2m}\left(\frac{i x_i^2}{\pi}\right) \tag{4.6}$$

The recommended value of m is 10 and it is usually evaluated in the range

$$x_i \in [0, \pi], \text{for all } i = 1, ..., d.$$

The location of the global minimum depends on the number of variables

$$\begin{aligned}
&\text{at } d = 2: \ f(\mathbf{x}^*) = -1.8013, \text{ at } \mathbf{x}^* = (2.20, 1.57) \\
&\text{at } d = 5: \ f(\mathbf{x}^*) = -4.687658 \\
&\text{at } d = 10: \ f(\mathbf{x}^*) = -9.66015.
\end{aligned}$$

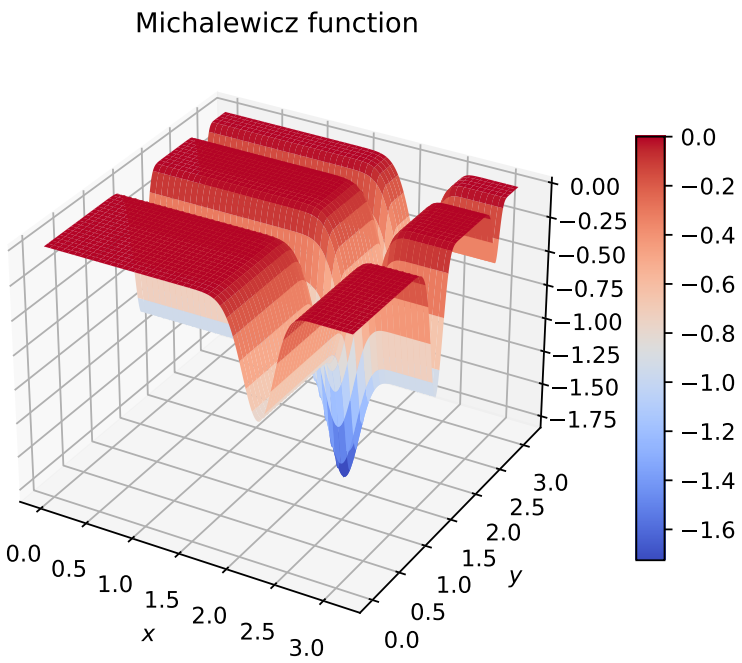
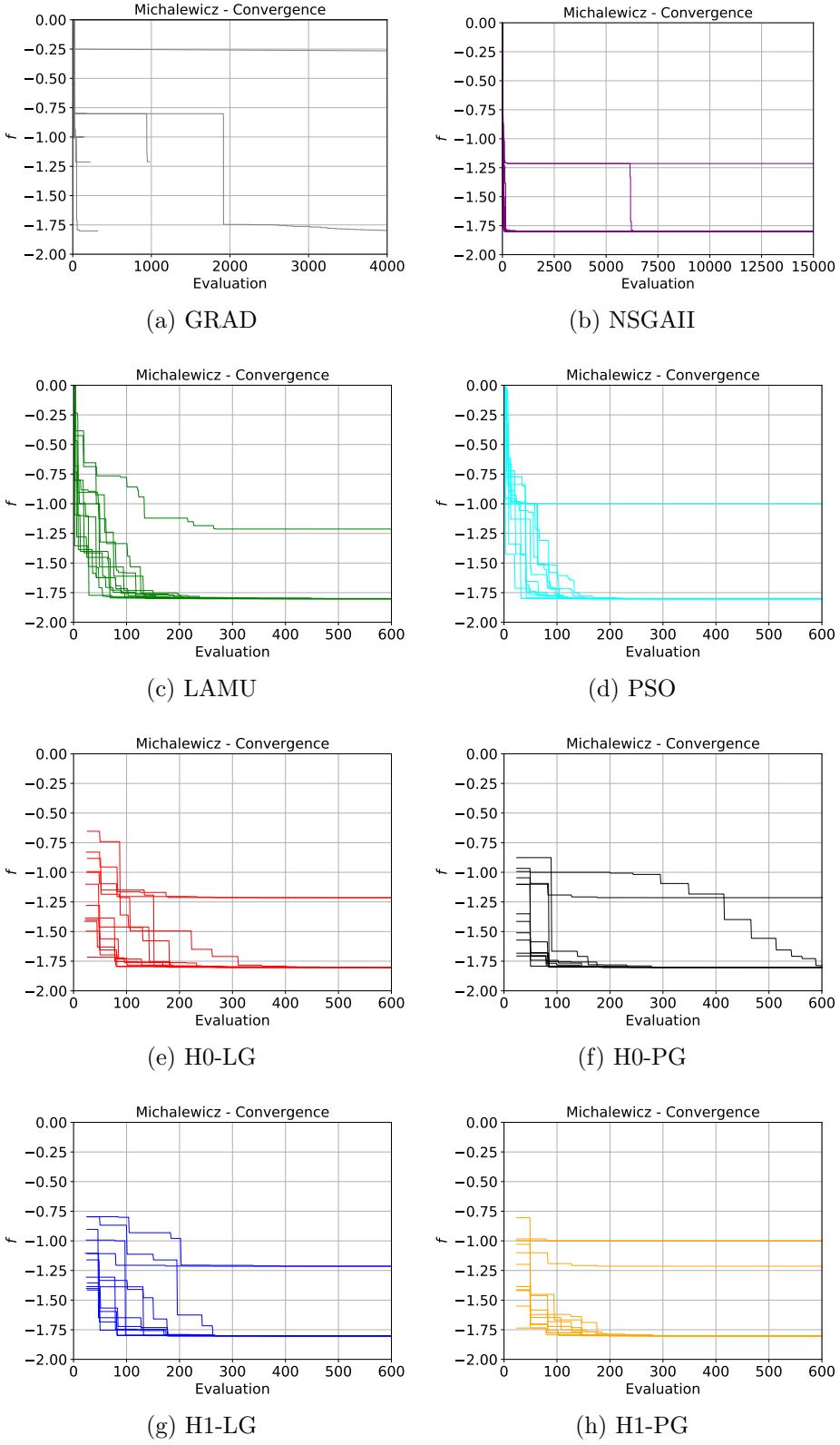


Figure 4.17: Michalewicz function.

Figure 4.18 presents the convergence for the Michalewicz function of all independent runs grouped by the optimization method. The Conjugate Gradient performance shown in this case is also as expected. It shows a strong dependence on the starting point. The run that fell into the global minimum valley shows a lot of accuracy as it successfully reached the global minimum with a high rate of convergence. One of the starting points fell into one of the canals and slowly reached the global minimum, the rest where trapped in the flat areas of the function. The Particle Swarm Optimization and the Genetic Algorithms (the three Evolutionary Algorithms) show a similar behavior in the three aspects, accuracy, robustness and rate of convergence. The three of them have a similar accuracy, although the LAMU algorithm is performing slightly worst in this case and aspect. One run of each of the three Evolutionary Algorithms failed to converge to the minimum in the firsts ≈ 600 objective function evaluations, although, the Genetic Algorithm converged at ≈ 6000 objective function evaluations, see Figure 4.19.



(a) GRAD (b) NSGAII

(c) LAMU (d) PSO

(e) H0-LG (f) H0-PG

(g) H1-LG (h) H1-PG

Figure 4.18: Convergence of each solver for all runs of the Michalewicz function.

In this test case the Hybrid Methods have performed worst than the Evolutionary

Algorithms, as 2 runs of each Hybrid Method test have failed to converge in the firsts ≈ 300 objective function evaluations. Although they achieved the convergence of those cases at between $\approx 10k$ to $15k$ objective function evaluations, except for one run of the H1-PG that did not converge. Figure 4.19 presents the convergence of all runs and solvers in the same graphic. Figure 4.20 shows the mean value of the convergence for each method, and can be noticed that the mean value of the Hybrid Methods, with the exception of the H1-PG, is highly penalized by the run that did not converge, along with the LAMU algorithm achieve a slightly higher mean accuracy at the expense of more evaluations, thus over performing the particle Swarm in this aspect. Table 4.5 shows the mean and standard deviation of the results.

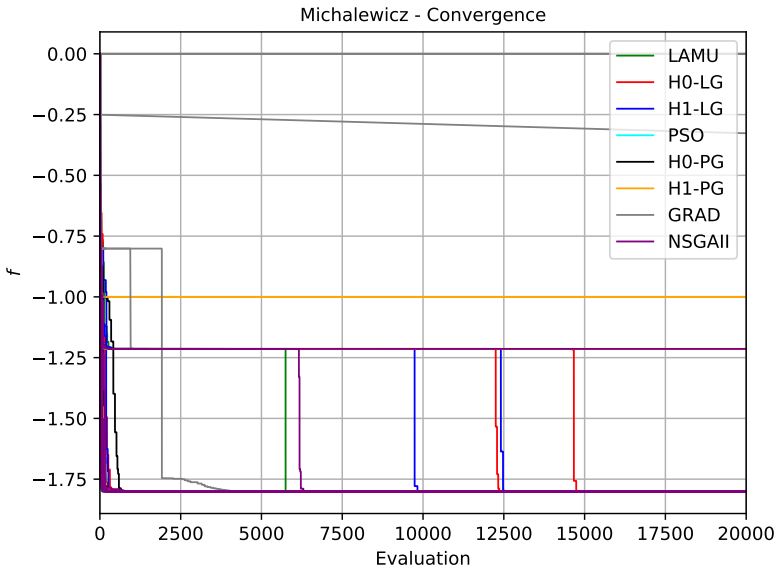


Figure 4.19: Michalewicz function convergence.

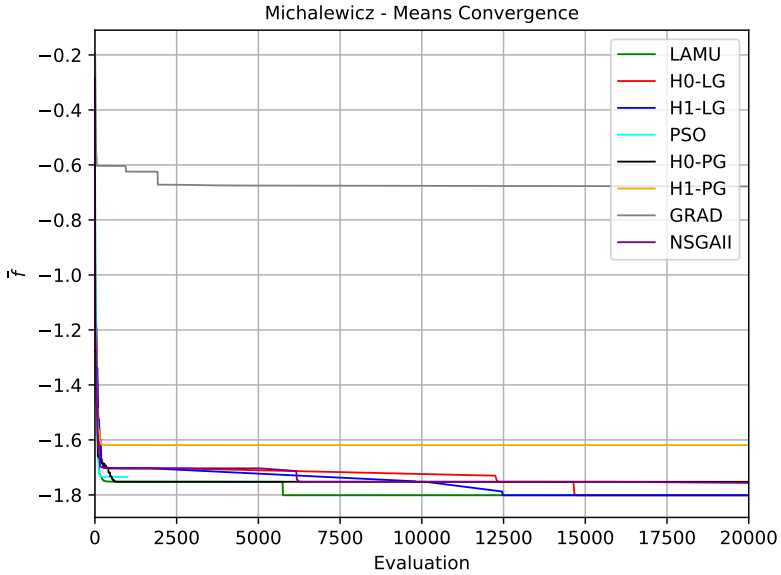


Figure 4.20: Michalewicz function means convergence.

Method	$\overline{f}_{\text{best}}$	$\sigma_{f_{\text{best}}}$
LAMU	-1.801	1.058×10^{-14}
H0-LG	-1.801	0.000
H1-LG	-1.801	0.000
PSO	-1.735	5.351×10^{-2}
H0-PG	-1.752	2.874×10^{-2}
H1-PG	-1.619	1.118×10^{-1}
GRAD	-6.780×10^{-1}	3.638×10^{-1}
NSGAII	-1.752	2.874×10^{-2}

Table 4.5: Mean and standard deviation of the best result comparison for the MICHALEWICZ problem.

4.2.6 Rosenbrock function

The Rosenbrock function was introduced by Rosenbrock [74]. It is a non-convex function that can be used with d variables and in this study two have been used. It has a single global minimum near the center of the search space. The function is very flat near the global minimum which makes the gradient evaluation producing values close to zero, this increasing the difficulty to find the global minimum. Figure 4.21 shows the shape of the function for two variables and a detail near the minimum. The objective function of the problem is defined as

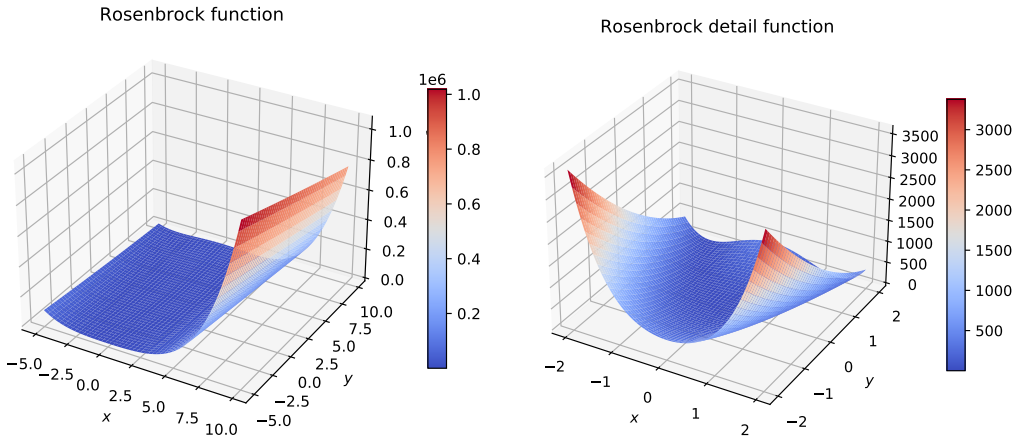
$$f(\mathbf{x}) = \sum_{i=1}^{d-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right] \tag{4.7}$$

The Rosenbrock function is usually evaluated in the range

$$x_i \in [-5, 10], \text{for all } i = 1, ..., d$$

and the global minimum is located at

$$f(\mathbf{x}^*) = 0, \text{ at } \mathbf{x}^* = (1, ..., d).$$



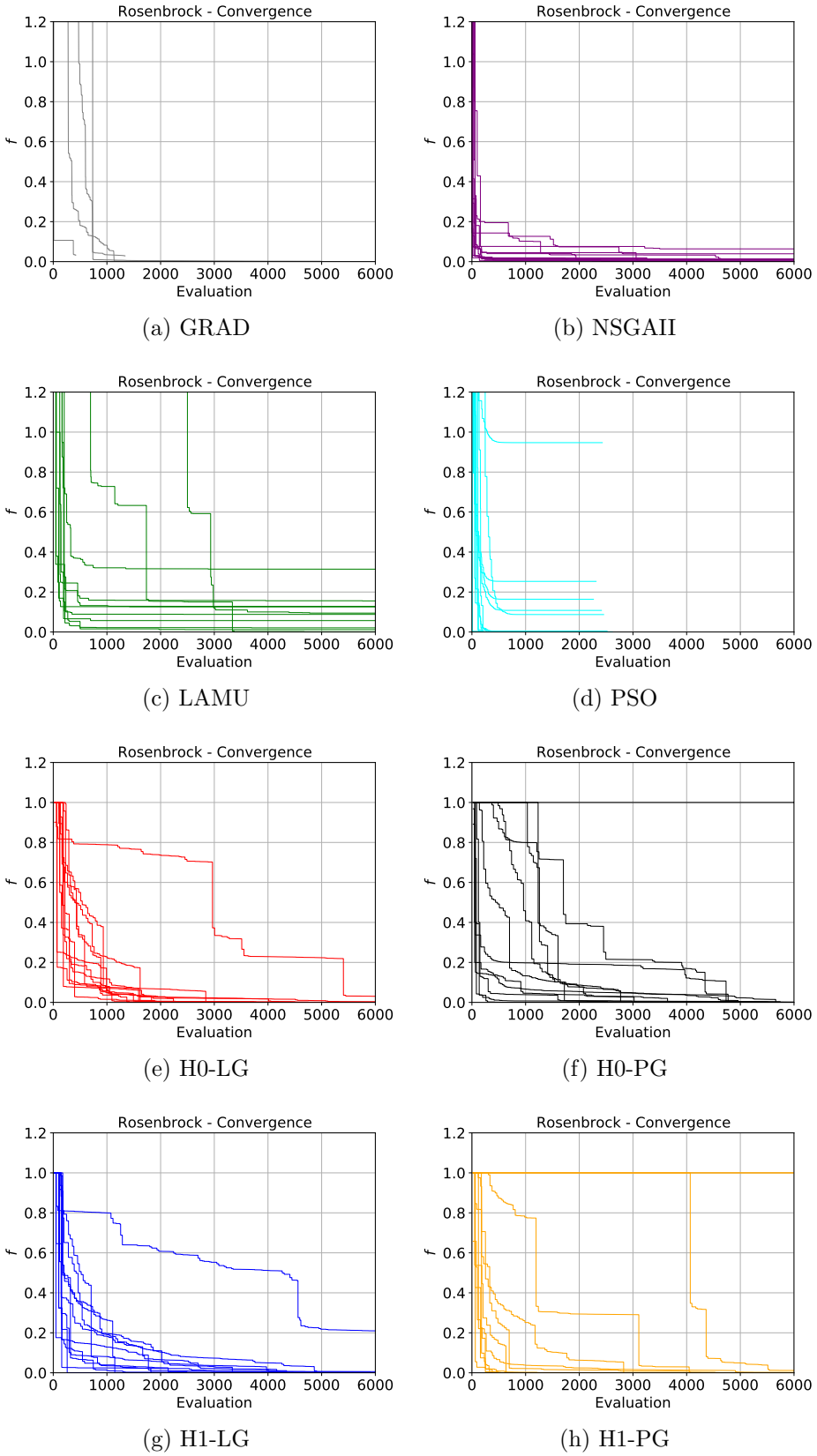
(a) (b)

Figure 4.21: (a) Rosenbrock function and (b) zoomed view of the minimum location.

Figure 4.22 presents the convergence for the Rosenbrock function of all independent runs grouped by the optimization method. The Conjugate Gradient performance shown in this test case is the best so far with about 10 out of 20 runs getting really close to the global minimum, but it still shows a high dependence on the starting point, as expected. . The higher performance on this function with respect to the other ones, is because the Rosenbrock function has a single valley. In this test case the difficulty for the Conjugate Gradient is that although there is a single minimum the function is very flat around it. This makes the gradient evaluation producing values close to zero, thus making it difficult to estimate an appropriate step size as it tends to zero along with the gradient. The Evolutionary Algorithms are behaving similarly. The NSGAII has performed better in this test case with higher accuracy and robustness. The Particle Swarm Optimization and LAMU method are performing very similar, with poor accuracy on most runs.

Some runs of the Hybrid Methods based on the Genetic Algorithm (i.e. H0-LG and H1-LG) are achieving a slower rate of convergence at the beginning of the optimizations but are being more robust and accurate as the number of evaluations

increase, and there is only one run of each method one run that is not as converged as the others. The Hybrid Methods are spending more evaluations in the Conjugate Gradient player as it slowly improves the objective function value until a better individual is found closer to the global minimum, which is slowing the initial convergence down as the Genetic Algorithm is capable to find a better starting with less objective functions evaluations. The two Hybrid Methods with the Particle Swarm Optimization players (i.e. H0-PG and H1-PG) are performing with less robustness than the based on the GA. One run of the H0-PG and two of the H1-PG got stuck at $f = 1$.



(a) GRAD

(b) NSGAII

(c) LAMU

(d) PSO

(e) H0-LG

(f) H0-PG

(g) H1-LG

(h) H1-PG

Figure 4.22: Convergence of each solver for all runs of the Rosenbrock function.

Figure 4.23 presents the convergence of all runs and solvers in the same graphic. Figure 4.24 shows the mean value of the convergence for each method, and can be noticed the outstanding accuracy of the NSGAII in this test case. The Hybrid Methods with the Genetic Algorithm as one of the players is achieving a similar mean accuracy than the NSGAII. In this test case the Hybrid Methods also improves the performance of both Evolutionary Algorithms running alone. The H0-LG achieved a the best overall accuracy passed the 5000 objective functions evaluations. Table 4.6 shows the mean and standard deviation of the results.

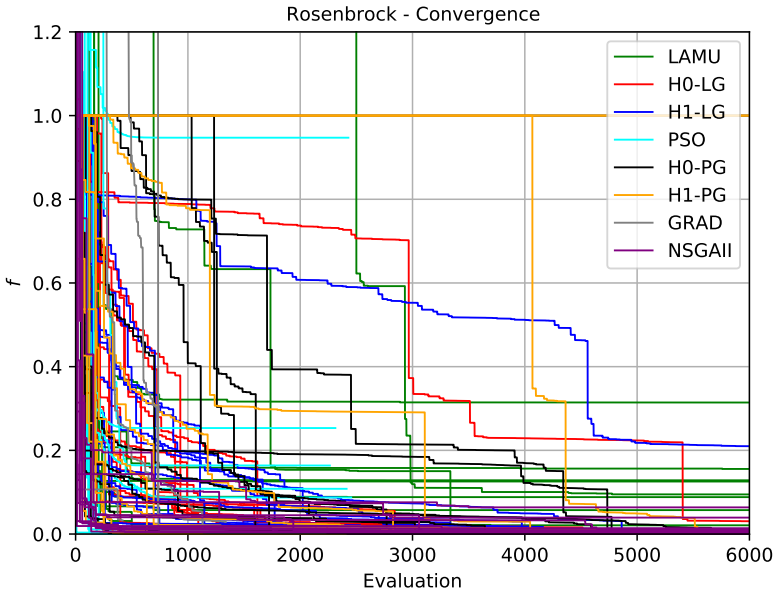


Figure 4.23: Rosenbrock function convergence.

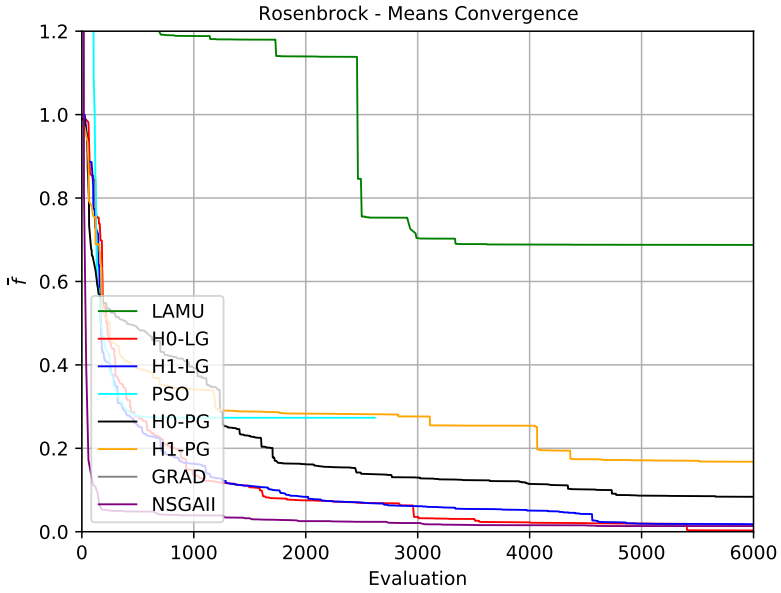


Figure 4.24: Rosenbrock function means convergence.

Method	$\overline{f}_{\text{best}}$	$\sigma_{f_{\text{best}}}$
LAMU	6.876×10^{-1}	1.894
H0-LG	3.013×10^{-3}	7.624×10^{-5}
H1-LG	1.804×10^{-2}	3.651×10^{-3}
PSO	2.732×10^{-1}	2.769×10^{-1}
H0-PG	8.374×10^{-2}	8.326×10^{-2}
H1-PG	1.678×10^{-1}	1.511×10^{-1}
GRAD	1.108×10^{4}	1.724×10^{8}
NSGAII	1.360×10^{-2}	3.648×10^{-4}

Table 4.6: Mean and standard deviation of the best result comparison for the Rosenbrock problem.

4.2.7 Summary

The optimization algorithms have been tested in six single objective mathematical benchmark tests. In the Ackley problem the Hybrid Methods clearly outperformed the rest of the methods in accuracy and rate of convergence. In this case the Hybrid Methods increased the performance of the optimization methods that internally uses. The Particle Swarm optimization performed better than both Genetic Algorithms in this case.

The method that performed better in the Levy benchmark is the Particle Swarm Optimization. The Hybrid Methods that contain the PSO (i.e. H0-PG and H1-PG) performed worst in the rate of convergence, although they reached the solution with the same accuracy as the PSO but needed more evaluations. The Hybrid Methods that contain the Genetic Algorithm (i.e. H0-LG and H1-LG) improved the pure Genetic Algorithm in rate of convergence and accuracy.

Regarding the Eggholder problem, the Particle Swarm Optimization method performed worst, as did not find the global minimums consistently. The Hybrid Methods based on the PSO (i.e. H0-PG and H1-PG) improved its performance. The Genetic Algorithm performed better than the PSO and the H0-PG and H1-PG hybrid methods. The Hybrid Methods based on the GA (i.e H0-LG and H1-LG) also improved its based Evolutionary Algorithm and are the methods that performed better in this benchmark.

All method performed similarly in the Holder table benchmark, although three Hybrid Methods, the H0-LG, H1-LG and H1-PG performed slightly worst in terms of rate of convergence specially on the first iterations.

In the Michalewicz benchmark the Particle Swarm Optimizaiton and H1-PG performed worst than than the rest of Hybrids and Evolutionary Algorithms as one run of each method did not converge to the global minimum. The LAMU method is the algorithm the performed best in this case. The NSGAII and the rest of the Hybrid

methods performed slightly worst than the NSGAII in terms rate of convergence.

The NSGAII is the method that achieved the highest initial rate of convergence in the Rosenbrock benchmark, although the H0-LG achieved a better accuracy passed the 5000 evaluations. The Hybrid Methods based on the GA (i.e. H0-LG and H1-LG) performed better than the based on the Particle Swarm Optimization.

4.3 Multi-objective mathematical test cases

As proposed for the single-objective test problems, a set of multi-objective benchmark problems have been selected from the literature in order to compare performance (mainly convergence) and accuracy of the proposed methodologies. Again, the reader will identify from simple functions with several design variables, to complex functions with few design variables, and some intermediate cases. The reader will also realize that all the selected problems present two objective functions. Considering the fact that these problems have been used to test the performance of the presented hybrid optimization method, it has been considered that an increase of the number of objective functions would have increased the complexity of the results making more difficult their interpretation.

Each optimization algorithm has been executed 12 times for each test problem.

4.3.1 ZDT1

The ZDT1 is a convex synthetic test problem proposed by Zitzler, Deb and Thiele [75]. It can be used with n design variables and it has 2 objective functions. In this tests it has been configured with $n = 30$ variables. The problem is defined as

$$\text{ZDT1}: \begin{cases} \text{Minimize} & f_1(x_1) = x_1 \\ \text{Minimize} & f_2(\mathbf{x}) = g \cdot h \\ \text{Domain} & 0 \leq x_i \leq 1 \quad , \quad i = 1, 2, \cdots, n \end{cases} \tag{4.8a}$$

$$g(x_2, \ldots, x_n) = 1 + 9 \sum_{i=2}^{n} \frac{x_i}{n-1} \tag{4.8b}$$

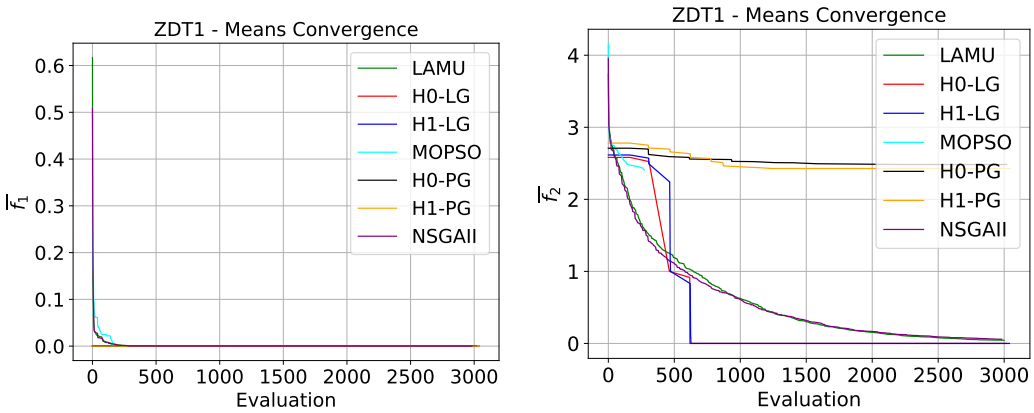
$$h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}} \tag{4.8c}$$

The Pareto optimal front complies with $g(\mathbf{x}) = 1$ which yields

$$f_2 = 1 - \sqrt{f_1} \tag{4.9}$$

Figure 4.25 presents the mean convergence of the two objective functions achieved by each optimization algorithm within its 12 runs. This information only gives information on the extremes of the Pareto front. All the optimization algorithms have converged to the best-f_1 extreme of the Pareto optimal front with high precision and

accuracy, see Figure 4.25a. For the best-f_2 extreme the Particle Swarm Optimization method and the Hybrid Methods that have a MOPSO player (i.e., H0-PG and H1-PG), are fairly far from the optimal extreme. The two Genetic Algorithms got close to the optimum in average at the end of the optimization (3000 objective functions evaluations). The Hybrid Methods based on the Genetic Algorithm presents a high rate of convergence and converged to the exact extreme of f_2 at around 600 objective functions evaluations.



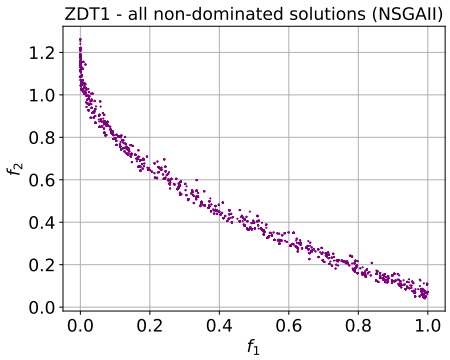
(a) $\overline{f_1}$ convergence.

(b) $\overline{f_2}$ convergence.

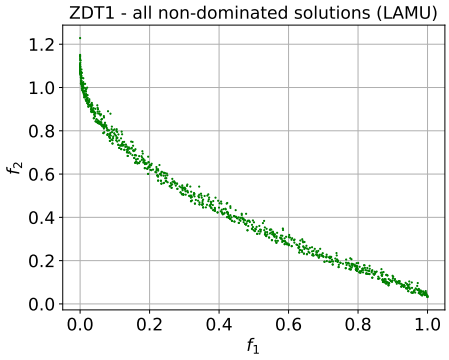
Figure 4.25: Mean convergence (ZDT1).

The Pareto fronts obtained by the 12 runs of each optimization algorithm are presented in Figure 4.26. Each figure contains all individuals from the Pareto fronts of each optimization test run, thus some individuals are dominated by individuals from another optimization run. The graphics show the robustness of each method and in some cases allow to visualize which region of the Pareto front are captured with better precision between different runs.

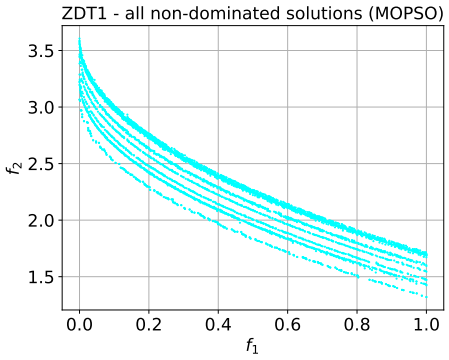
The Particle Swarm Optimization and the Hybrid Methods based on it (i.e., H0-PG and H1-PG) show a lack of robustness as the different runs created Pareto fronts far one from each other, with scattered bands. The NSGAII, the LAMU method and the Hybrid Methods based on the Genetic Algorithm (i.e., H0-LG and H1-LG) are performing much better than the others with more robustness, although the Hybrid Methods have less density of individuals in the central region of the Pareto front as shown in Figures 4.26d and 4.26f.



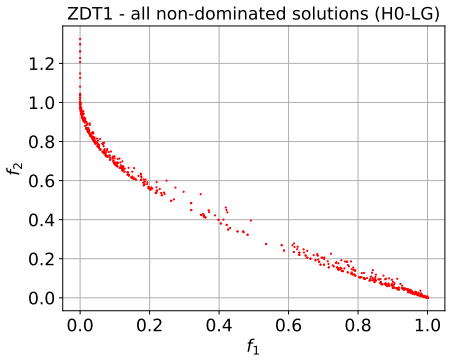
(a) NSGAII



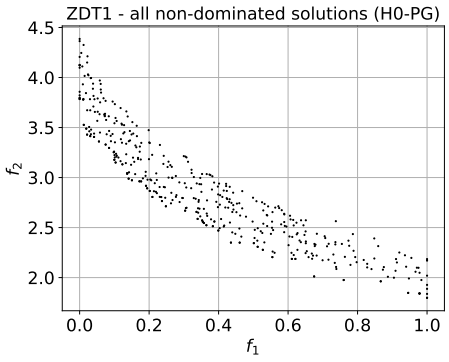
(b) LAMU



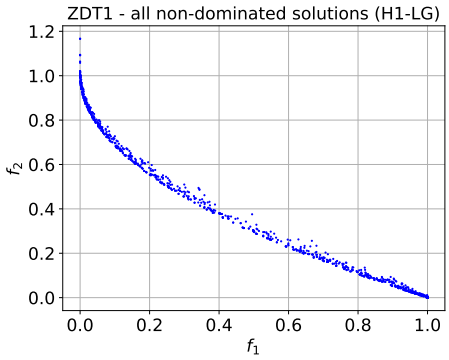
(c) MOPSO



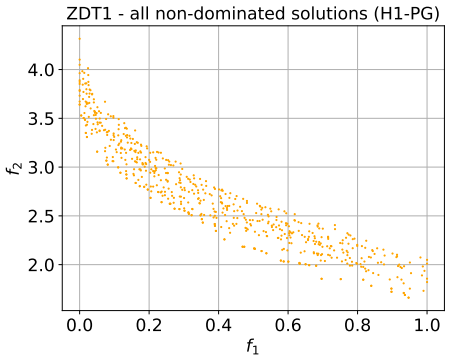
(d) H0-LG



(e) H0-PG



(f) H1-LG

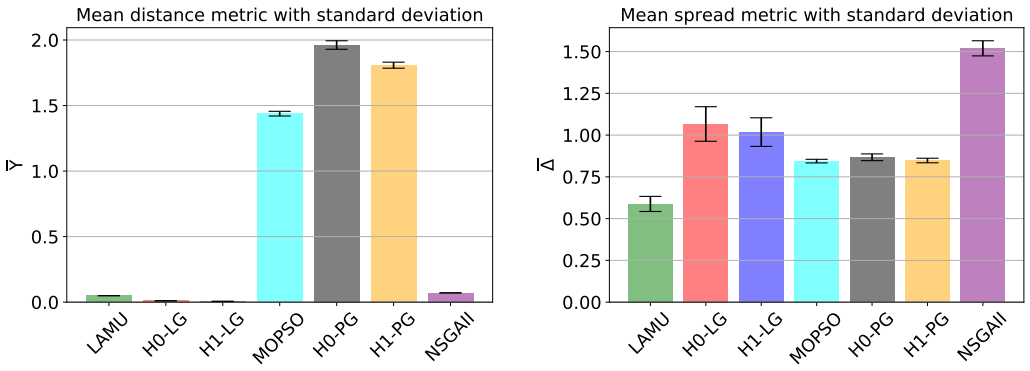


(g) H1-PG

Figure 4.26: Pareto front of each solver for all runs of the ZDT1 function.

Figure 4.27 and Table 4.7 presents the mean value of the performance metrics and the standard deviation for each optimization algorithm. The metrics have been computed for each one of the 12 runs of each optimization algorithm. With the values of the metrics the mean and standard deviation have been calculated for each optimization algorithm.

According to the distance metric (Table 4.7) the optimization algorithm that achieved a better convergence to the Pareto optimal front is the H1-LG with a value of $\overline{\Upsilon} = 0.006$, closely followed by the H0-LG $\overline{\Upsilon} = 0.01$. The two Genetic Algorithms also performed well: LAMU method $\overline{\Upsilon} = 0.049$ and NSGAII $\overline{\Upsilon} = 0.071$. These methods also demonstrated a high robustness with low values of the standard deviation compared to the other methods.



(a) Distance metric.

(b) Spread metric.

Figure 4.27: ZDT1 metrics.

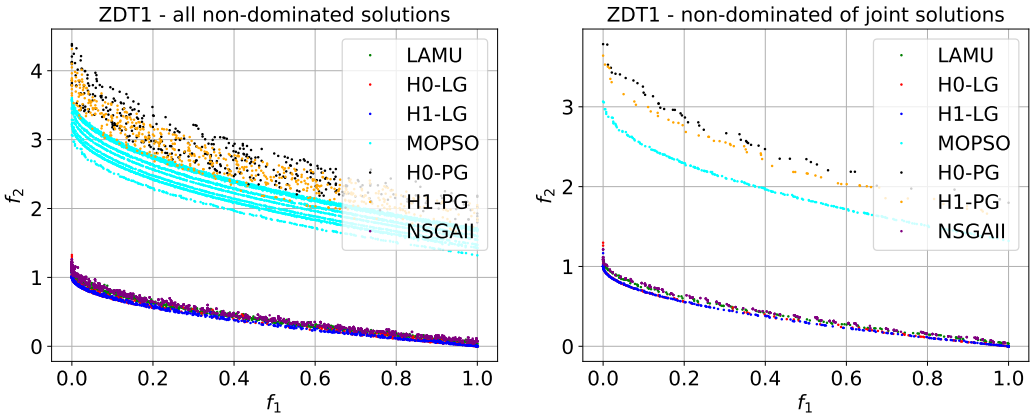
The best spread metric is achieved by the LAMU method which performed $\overline{\Delta} = 0.588$.

Method	$\overline{\Upsilon}$	σ_Υ	$\overline{\Delta}$	σ_Δ
LAMU	0.049	6.22×10^{-5}	0.588	4.52×10^{-2}
H0-LG	0.010	4.49×10^{-5}	1.066	1.03×10^{-1}
H1-LG	0.006	4.11×10^{-6}	1.018	8.56×10^{-2}
MOPSO	1.438	1.79×10^{-2}	0.844	1.09×10^{-2}
H0-PG	1.962	3.25×10^{-2}	0.867	2.02×10^{-2}
H1-PG	1.808	2.30×10^{-2}	0.848	1.36×10^{-2}
NSGA2	0.071	2.22×10^{-4}	1.519	4.53×10^{-2}

Table 4.7: Solver metrics comparison for ZDT1 problem.

Figure 4.28a presents the Pareto fronts of all runs, as stated before the Hybrid Methods based on the Genetic Algorithm are not performing well as a some individuals of

these methods are dominated by the other methods. Figure 4.28b presents the non-dominated solutions among the different runs for each solver. For each optimization method the solutions of their runs are joined in a single population. With this population the dominance is computed to obtain the global set of non-dominated individuals achieved by each optimization method. Figure 4.28b clearly shows the difference in performance between the different optimization methods.



(a) (b)

Figure 4.28: (a) Overlapped Pareto fronts from all runs. (b) Non-dominated solutions of the combined Pareto fronts of all runs for each optimization method. (ZDT1).

4.3.2 ZDT2

The ZDT2 is a non-convex synthetic test problem proposed by Zitzler, Deb and Thiele [75]. It can be used with n design variables and it has 2 objective functions. In this tests it has been configured with $n = 30$ variables. The problem is defined as

$$\text{ZDT2}: \begin{cases} \text{Minimize} & f_1(x_1) = x_1 \\ \text{Minimize} & f_2(\mathbf{x}) = g \cdot h \\ \text{Domain} & 0 \leq x_i \leq 1 \quad , \quad i = 1, 2, \cdots, n \end{cases} \tag{4.10a}$$

$$g(x_2, \ldots, x_n) = 1 + 9 \sum_{i=2}^{n} \frac{x_i}{n-1} \tag{4.10b}$$

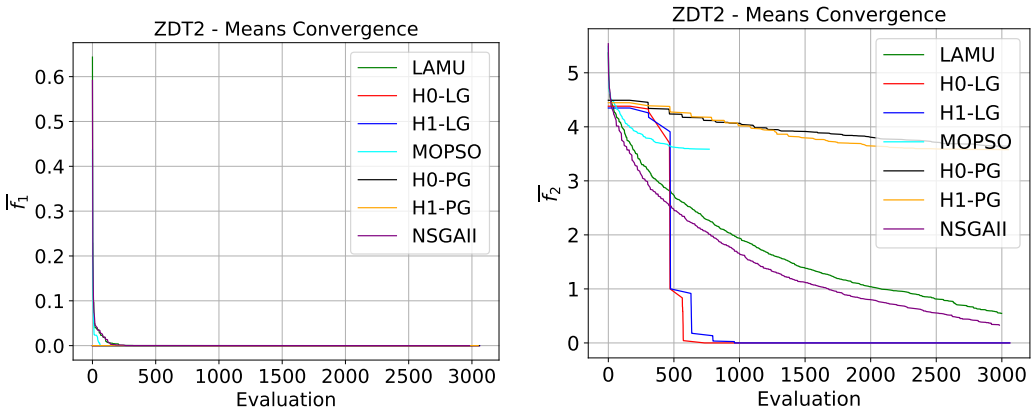
$$h(f_1, g) = 1 - \left(\frac{f_1}{g}\right)^2 \tag{4.10c}$$

The Pareto optimal front complies with $g(\mathbf{x}) = 1$ which yields

$$f_2 = 1 - (f_1)^2 \tag{4.11}$$

The behavior and performance of the optimization algorithms in the ZDT2 test problem is very similar to the observed in the ZDT1 case.

Figure 4.29 presents the mean convergence of the two objective functions achieved by each optimization algorithm within its 12 runs. This information only gives information on the extremes of the Pareto front. All the optimization algorithms have converged to the best-f_1 extreme of the Pareto optimal front with high precision and accuracy, see Figure 4.29a. For the best-f_2 extreme the Particle Swarm Optimization method and the Hybrid Methods that have a MOPSO player (i.e., H0-PG and H1-PG), are fairly far from the optimal extreme. The two Genetic Algorithms got close to the optimum in average at the end of the optimization (3000 objective functions evaluations). The Hybrid Methods based on the Genetic Algorithm presents a high rate of convergence and converged to the exact extreme of f_2 at around 1000 objective functions evaluations.



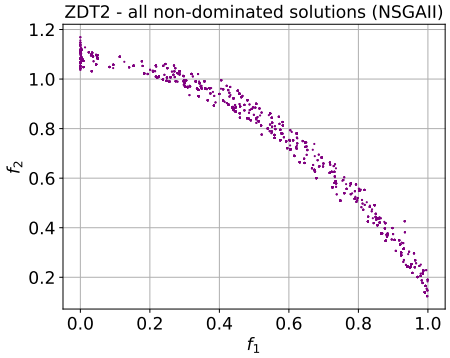
(a) $\overline{f_1}$ convergence.

(b) $\overline{f_2}$ convergence.

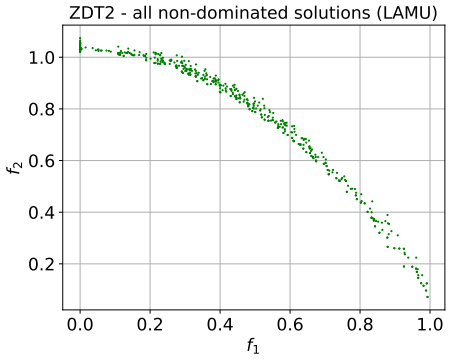
Figure 4.29: Mean convergence (ZDT2).

The Pareto fronts obtained by the 12 runs of each optimization algorithm are presented in Figure 4.30. Each figure contains all individuals from the Pareto fronts of each optimization test run, thus some individuals are dominated by individuals from another optimization run. The graphics show the robustness of each method and in some cases allow to visualize which region of the Pareto front are captured with better precision between different runs.

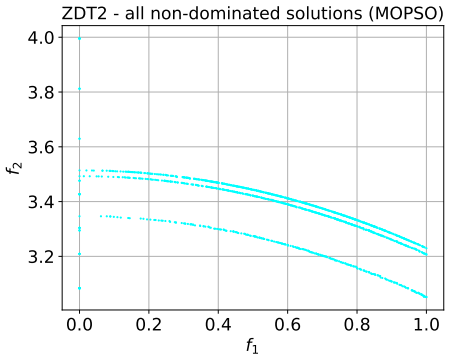
The Particle Swarm Optimization and the Hybrid Methods based on it (i.e., H0-PG and H1-PG) show a lack of robustness as the different runs created Pareto fronts far one from each other, with scattered bands in the MOPSO and scattered points in the MOPSO based Hybrid Methods. The NSGAII, the LAMU method and the Hybrid Methods based on the Genetic Algorithm (i.e., H0-LG and H1-LG) are performing much better than the others with more robustness, although the Hybrid Methods have less density of individuals in the central region of the Pareto front as shown in Figures 4.30d and 4.30f.



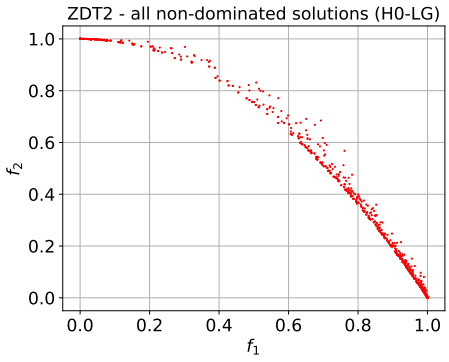
(a) NSGAII



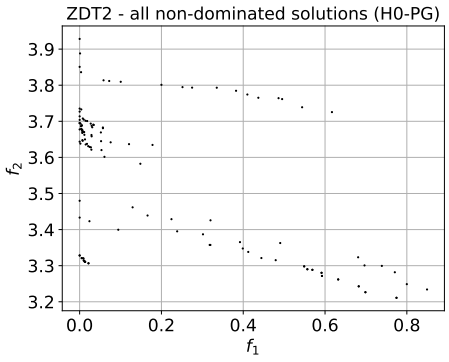
(b) LAMU



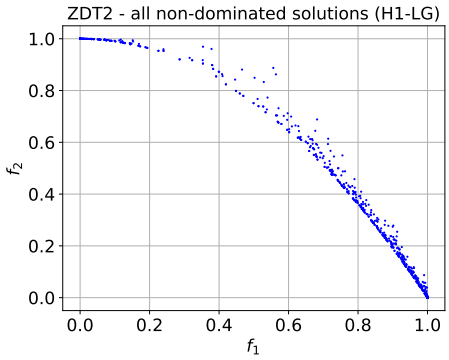
(c) MOPSO



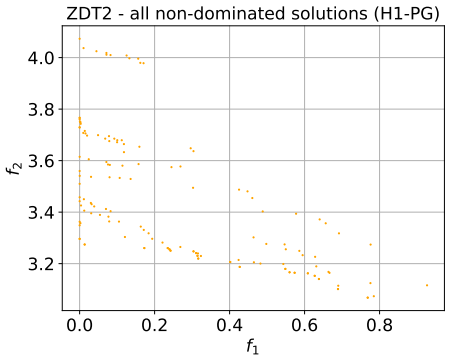
(d) H0-LG



(e) H0-PG



(f) H1-LG

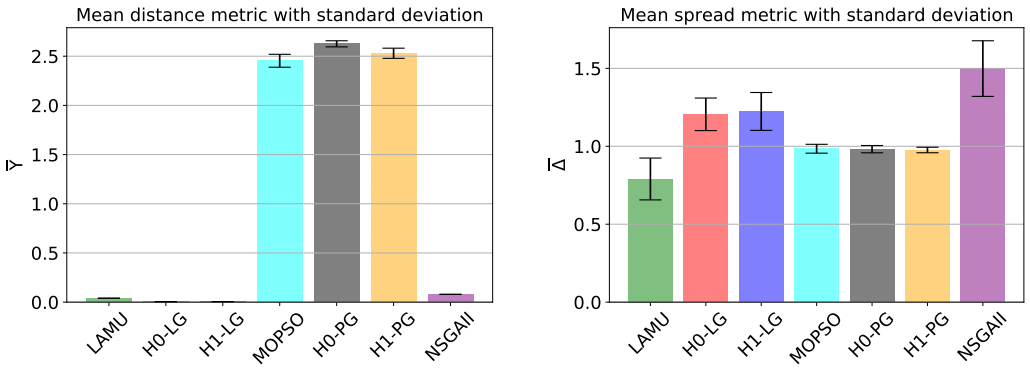


(g) H1-PG

Figure 4.30: Pareto front of each solver for all runs of the ZDT2 function.

Figure 4.31 and Table 4.8 presents the mean value of the performance metrics and the standard deviation for each optimization algorithm. The metrics have been computed for each one of the 12 runs of each optimization algorithm. With the values of the metrics the mean and standard deviation have been calculated for each optimization algorithm.

According to the distance metric (Table 4.8) the optimization algorithm that achieved a better convergence to the Pareto optimal front is the H0-LG with a value of $\overline{\Upsilon} = 0.005$, closely followed by the H1-LG $\overline{\Upsilon} = 0.006$. The two Genetic Algorithms also performed well: LAMU method $\overline{\Upsilon} = 0.040$ and NSGAII $\overline{\Upsilon} = 0.079$. These methods also demonstrated a high robustness with low values of the standard deviation compared to the other methods.



(a) Distance metric.

(b) Spread metric.

Figure 4.31: ZDT2 metrics.

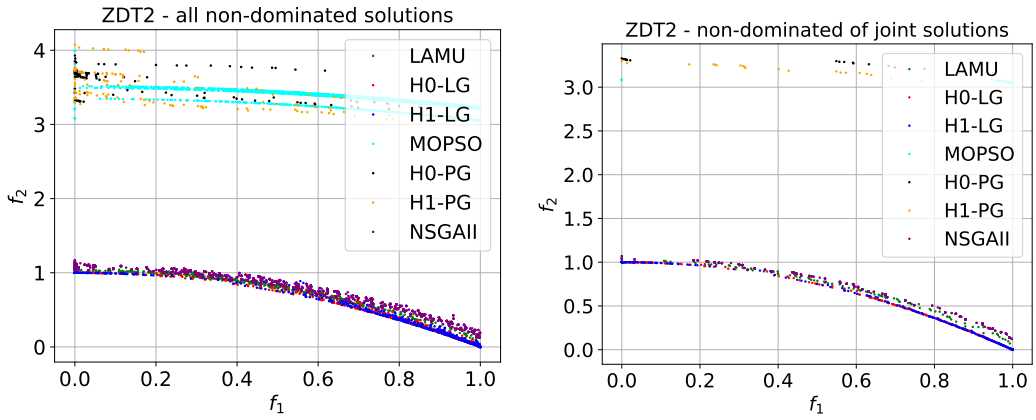
The best spread metric is achieved by the LAMU method which performed $\overline{\Delta} = 0.79$.

Method	$\overline{\Upsilon}$	σ_Υ	$\overline{\Delta}$	σ_Δ
LAMU	0.040	1.15×10^{-4}	0.790	1.34×10^{-1}
H0-LG	0.005	2.48×10^{-6}	1.205	1.04×10^{-1}
H1-LG	0.006	6.77×10^{-6}	1.224	1.22×10^{-1}
MOPSO	2.454	6.57×10^{-2}	0.984	2.85×10^{-2}
H0-PG	2.626	3.10×10^{-2}	0.982	2.34×10^{-2}
H1-PG	2.530	5.17×10^{-2}	0.976	1.77×10^{-2}
NSGA2	0.079	3.12×10^{-4}	1.499	1.78×10^{-1}

Table 4.8: Solver metrics comparison for ZDT2 problem.

Figure 4.32a presents the Pareto fronts of all runs, as stated before the Hybrid Methods based on the Genetic Algorithm are not performing well as a some individuals of

these methods are dominated by the other methods. Figure 4.32b presents the non-dominated solutions among the different runs for each solver. For each optimization method the solutions of their runs are joined in a single population. With this population the dominance is computed to obtain the global set of non-dominated individuals achieved by each optimization method. Figure 4.32b clearly shows the difference in performance between the different optimization methods.



(a) (b)

Figure 4.32: (a) Overlapped Pareto fronts from all runs. (b) Non-dominated solutions of the combined Pareto fronts of all runs for each optimization method. (ZDT2).

4.3.3 ZDT3

The ZDT3 is a convex and disconnected synthetic test problem proposed by Zitzler, Deb and Thiele [75]. It can be used with n design variables and it has 2 objective functions. In this tests it has been configured with $n = 30$ variables. The problem is defined as

$$\text{ZDT3}: \begin{cases} \text{Minimize} & f_1(x_1) = x_1 \\ \text{Minimize} & f_2(\mathbf{x}) = g \cdot h \\ \text{Domain} & 0 \leq x_i \leq 1 \quad , \quad i = 1, 2, \cdots, n \end{cases} \tag{4.12a}$$

$$g(x_2, \ldots, x_n) = 1 + 9 \sum_{i=2}^{n} \frac{x_i}{n-1} \tag{4.12b}$$

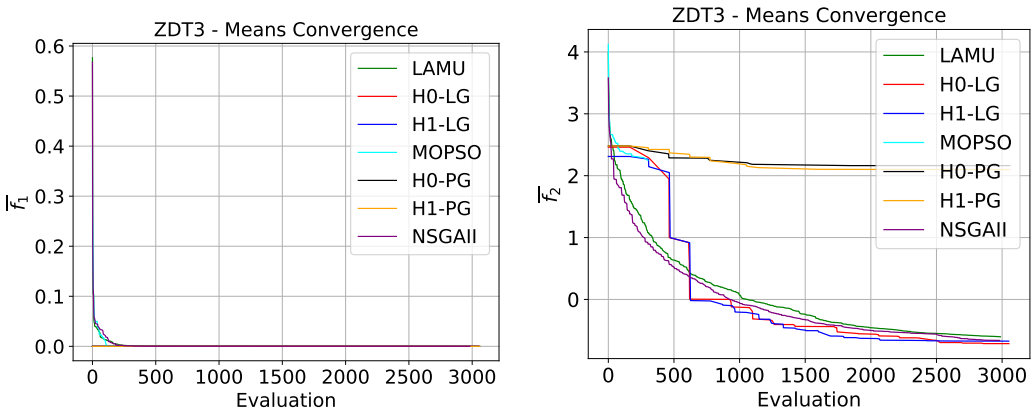
$$h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}} - \frac{f_1}{g} \sin(10\pi f_1) \tag{4.12c}$$

The Pareto optimal front complies with $g(\mathbf{x}) = 1$ which yields

$$f_2 = 1 - \sqrt{f_1} - f_1 \sin(10\pi f_1) \tag{4.13}$$

The behavior and performance of the optimization algorithms in the ZDT3 test problem is very similar to the observed in the ZDT1 and ZDT2 test problems.

Figure 4.33 presents the mean convergence of the two objective functions achieved by each optimization algorithm within its 12 runs. This information only gives information on the extremes of the Pareto front. All the optimization algorithms have converged to the best-f_1 extreme of the Pareto optimal front with high precision and accuracy, see Figure 4.33a. For the best-f_2 extreme the Particle Swarm Optimization method and the Hybrid Methods that have a MOPSO player (i.e., H0-PG and H1-PG), are fairly far from the optimal extreme. The two Genetic Algorithms and the Hybrid Methods based on the Genetic Algorithm got close to the optimum in average at the end of the optimization (3000 objective functions evaluations).



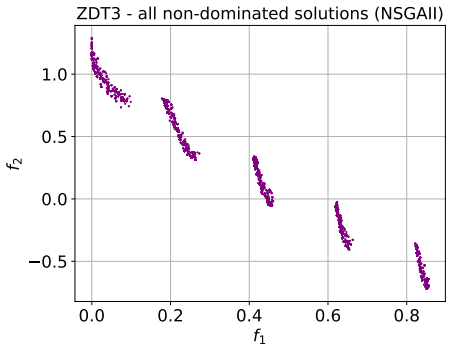
(a) $\overline{f_1}$ convergence.

(b) $\overline{f_2}$ convergence.

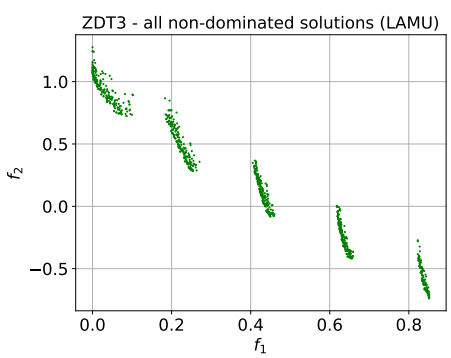
Figure 4.33: Mean convergence (ZDT3).

The Pareto fronts obtained by the 12 runs of each optimization algorithm are presented in Figure 4.34. Each figure contains all individuals from the Pareto fronts of each optimization test run, thus some individuals are dominated by individuals from another optimization run. The graphics show the robustness of each method and in some cases allow to visualize which region of the Pareto front are captured with better precision between different runs.

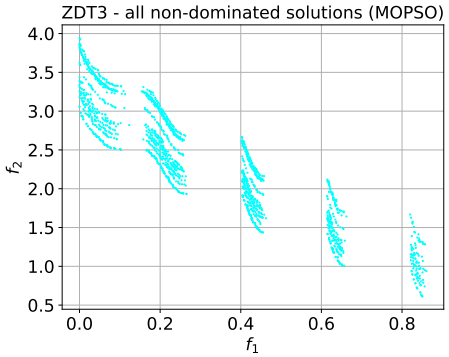
The Particle Swarm Optimization and the Hybrid Methods based on it (i.e., H0-PG and H1-PG) show a lack of robustness as the different runs created Pareto fronts far one from each other, with scattered bands in the MOPSO and scattered points in the MOPSO based Hybrid Methods. The NSGAII, the LAMU method and the Hybrid Methods based on the Genetic Algorithm (i.e., H0-LG and H1-LG) are performing much better than the others with more robustness, although as in the ZDT2 test problem the Hybrid Methods have less density of individuals in the central region of the Pareto front as shown in Figures 4.34d and 4.34f.



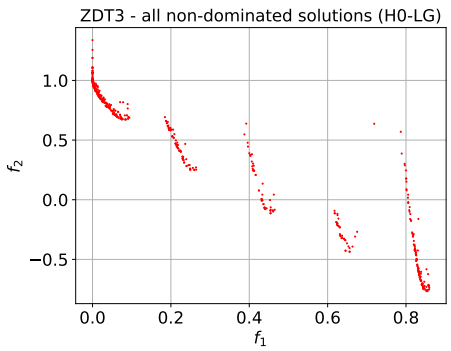
(a) NSGAII



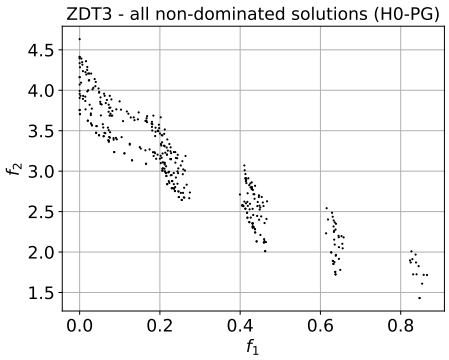
(b) LAMU



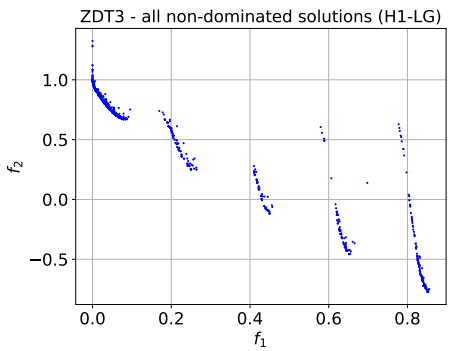
(c) MOPSO



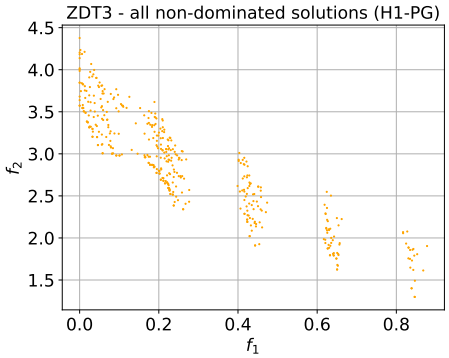
(d) H0-LG



(e) H0-PG



(f) H1-LG

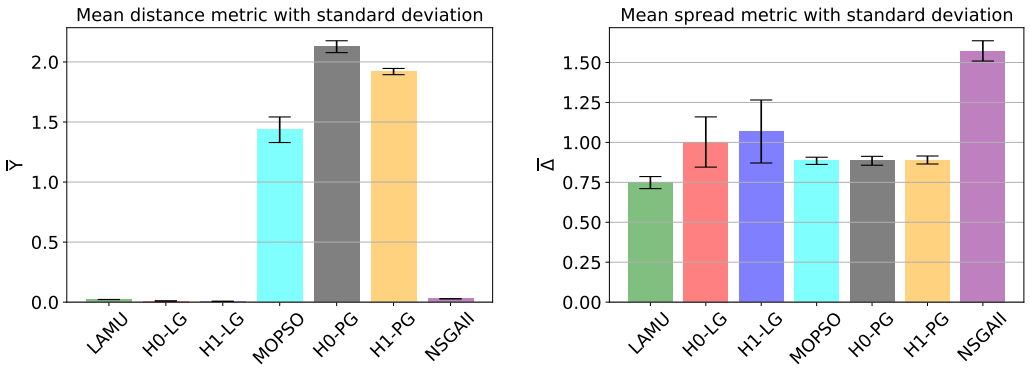


(g) H1-PG

Figure 4.34: Pareto front of each solver for all runs of the ZDT3 function.

Figure 4.35 and Table 4.9 presents the mean value of the performance metrics and the standard deviation for each optimization algorithm. The metrics have been computed for each one of the 12 runs of each optimization algorithm. With the values of the metrics the mean and standard deviation have been calculated for each optimization algorithm.

According to the distance metric (Table 4.9) the optimization algorithm that achieved a better convergence to the Pareto optimal front is the H0-LG with a value of $\overline{\Upsilon} = 0.007$, closely followed by the H1-LG $\overline{\Upsilon} = 0.01$. The two Genetic Algorithms also performed well: LAMU method $\overline{\Upsilon} = 0.022$ and NSGAII $\overline{\Upsilon} = 0.028$. These methods also demonstrated a high robustness with low values of the standard deviation compared to the other methods.



(a) Distance metric.

(b) Spread metric.

Figure 4.35: ZDT3 metrics.

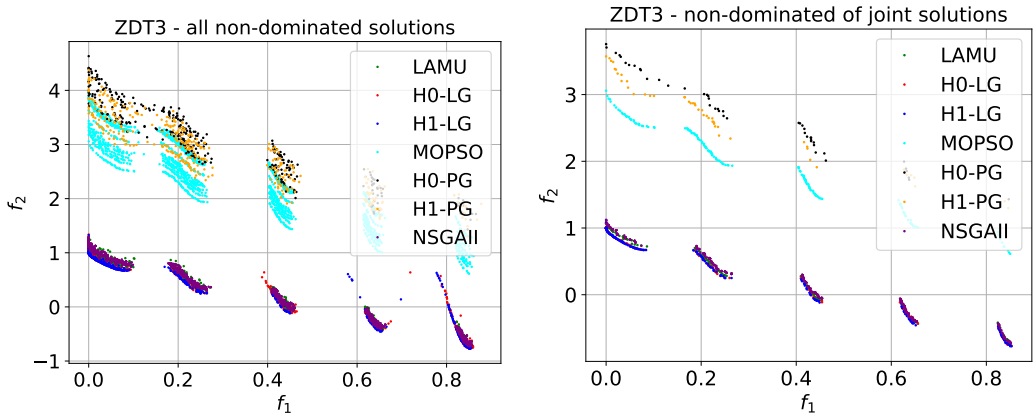
The best spread metric is achieved by the LAMU method which performed $\overline{\Delta} = 0.79$.

Method	$\overline{\Upsilon}$	σ_Υ	$\overline{\Delta}$	σ_Δ
LAMU	0.022	1.02×10^{-4}	0.748	3.78×10^{-2}
H0-LG	0.010	3.67×10^{-5}	1.002	1.57×10^{-1}
H1-LG	0.007	4.06×10^{-5}	1.068	1.97×10^{-1}
MOPSO	1.436	1.06×10^{-1}	0.884	2.26×10^{-2}
H0-PG	2.127	4.94×10^{-2}	0.885	2.78×10^{-2}
H1-PG	1.920	2.59×10^{-2}	0.890	2.50×10^{-2}
NSGA2	0.028	8.37×10^{-5}	1.573	6.35×10^{-2}

Table 4.9: Solver metrics comparison for ZDT3 problem.

Figure 4.36a presents the Pareto fronts of all runs, as stated before the Hybrid Methods based on the Genetic Algorithm are not performing well as a some individuals of

these methods are dominated by the other methods. Figure 4.36b presents the non-dominated solutions among the different runs for each solver. For each optimization method the solutions of their runs are joined in a single population. With this population the dominance is computed to obtain the global set of non-dominated individuals achieved by each optimization method. Figure 4.36b clearly shows the difference in performance between the different optimization methods.



(a) (b)

Figure 4.36: (a) Overlapped Pareto fronts from all runs. (b) Non-dominated solutions of the combined Pareto fronts of all runs for each optimization method. (ZDT3).

4.3.4 ZDT4

The ZDT4 is a non-convex synthetic test problem proposed by Zitzler, Deb and Thiele [75]. It can be used with n design variables and it has 2 objective functions. In this tests it has been configured with $n = 10$ variables. The problem is defined as

$$\text{ZDT4}: \begin{cases} \text{Minimize} & f_1(x_1) = x_1 \\ \text{Minimize} & f_2(\mathbf{x}) = g \cdot h \\ \text{Domain} & 0 \leq x_i \leq 1 \quad , \quad i = 1, 2, \cdots, n \end{cases} \tag{4.14a}$$

$$g(x_2, \ldots, x_n) = 1 + 10(n-1) + \sum_{i=2}^{n} \left(x_i^2 - 10\cos(4\pi x_i)\right) \tag{4.14b}$$

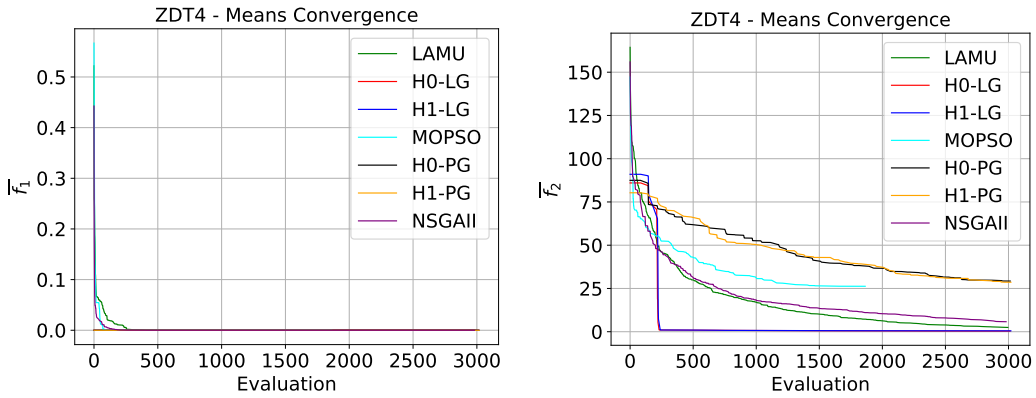
$$h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}} \tag{4.14c}$$

The Pareto optimal front complies with $g(\mathbf{x}) = 1$ which yields

$$f_2 = 1 - \sqrt{f_1} \tag{4.15}$$

The behavior and performance of the optimization algorithms in the ZDT4 test problem is very similar to the observed in the ZDT1, ZDT2 and ZDT3 test problems.

Figure 4.37 presents the mean convergence of the two objective functions achieved by each optimization algorithm within its 12 runs. This information only gives information on the extremes of the Pareto front. All the optimization algorithms have converged to the best-f_1 extreme of the Pareto optimal front with high precision and accuracy, see Figure 4.37a. For the best-f_2 extreme the Particle Swarm Optimization method and the Hybrid Methods that have a MOPSO player (i.e., H0-PG and H1-PG), are fairly far from the optimal extreme. The two Genetic Algorithms got close to the optimum in average at the end of the optimization (3000 objective functions evaluations). The Hybrid Methods based on the Genetic Algorithm presents a high rate of convergence and converged to the exact extreme of f_2 at around 600 objective functions evaluations.



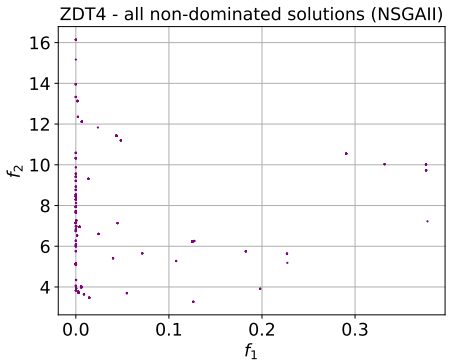
(a) $\overline{f_1}$ convergence.

(b) $\overline{f_2}$ convergence.

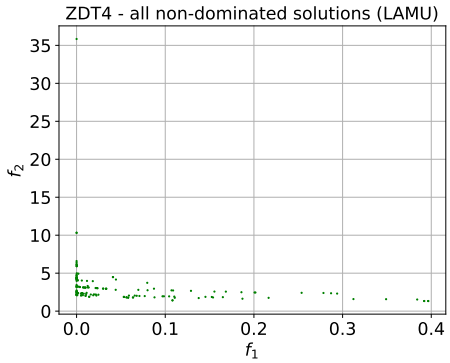
Figure 4.37: Mean convergence (ZDT4).

The Pareto fronts obtained by the 12 runs of each optimization algorithm are presented in Figure 4.38. Each figure contains all individuals from the Pareto fronts of each optimization test run, thus some individuals are dominated by individuals from another optimization run. The graphics show the robustness of each method and in some cases allow to visualize which region of the Pareto front are captured with better precision between different runs.

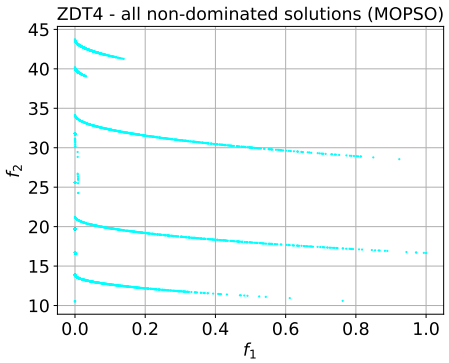
The Particle Swarm Optimization and the Hybrid Methods based on it (i.e., H0-PG and H1-PG) show a lack of robustness as the different runs created Pareto fronts far one from each other, with scattered bands in the MOPSO and scattered points in the MOPSO based Hybrid Methods. The NSGAII, the LAMU method and the Hybrid Methods based on the Genetic Algorithm (i.e., H0-LG and H1-LG) are performing much better than the others with more robustness.



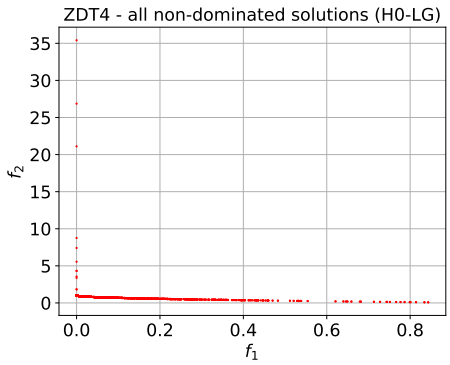
(a) NSGAII



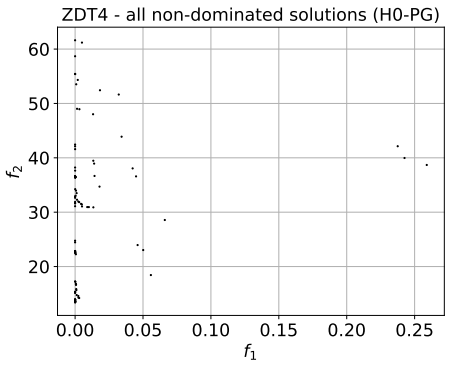
(b) LAMU



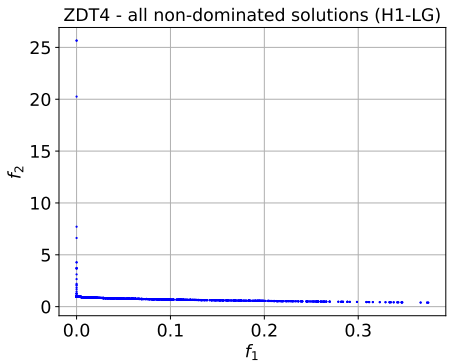
(c) MOPSO



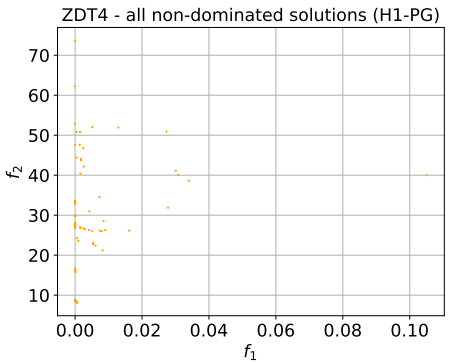
(d) H0-LG



(e) H0-PG



(f) H1-LG

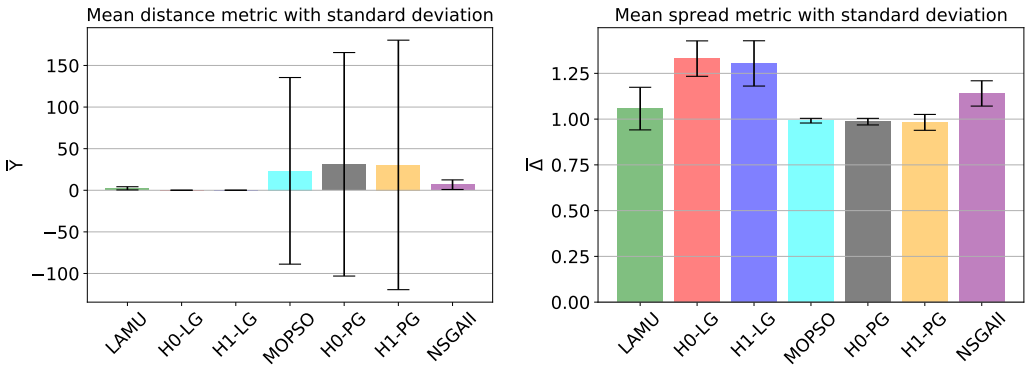


(g) H1-PG

Figure 4.38: Pareto front of each solver for all runs of the ZDT4 function.

Figure 4.39 and Table 4.10 presents the mean value of the performance metrics and the standard deviation for each optimization algorithm. The metrics have been computed for each one of the 12 runs of each optimization algorithm. With the values of the metrics the mean and standard deviation have been calculated for each optimization algorithm.

According to the distance metric (Table 4.10) the optimization algorithm that achieved a better convergence to the Pareto optimal front is the H0-LG with a value of $\overline{\Upsilon} = 0.045$, closely followed by the H1-LG $\overline{\Upsilon} = 0.058$. The two Genetic Algorithms also performed well: LAMU method $\overline{\Upsilon} = 2.337$ and NSGAII $\overline{\Upsilon} = 5.77$. These methods also demonstrated a high robustness with low values of the standard deviation compared to the other methods.



(a) Distance metric.

(b) Spread metric.

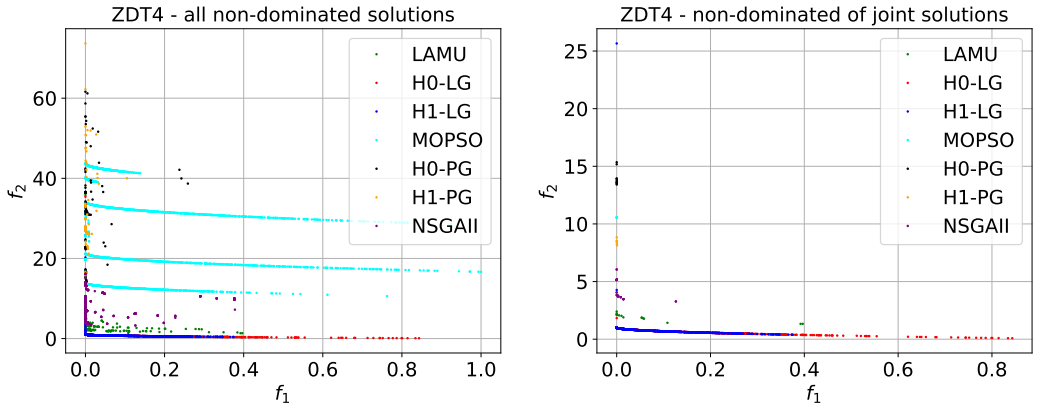
Figure 4.39: ZDT4 metrics.

The best spread metric is achieved by the H1-PG method which performed $\overline{\Delta} = 0.982$, although in this case is irrelevant, as the method is not performing well and all the individuals in the solution od this method is dominated by the Genetic Algorithm ones and the Hybrid Methods based on the Genetic Algorithm (i.e., H0-LG and H1-LG).

Method	$\overline{\Upsilon}$	σ_Υ	$\overline{\Delta}$	σ_Δ
LAMU	2.337	1.92	1.057	1.16×10^{-1}
H0-LG	0.045	8.87×10^{-3}	1.331	9.68×10^{-2}
H1-LG	0.058	7.15×10^{-3}	1.304	1.24×10^{-1}
MOPSO	23.352	1.12×10^{2}	0.991	1.28×10^{-2}
H0-PG	31.209	1.34×10^{2}	0.986	1.80×10^{-2}
H1-PG	30.470	1.50×10^{2}	0.982	4.34×10^{-2}
NSGA2	6.709	5.77	1.140	6.91×10^{-2}

Table 4.10: Solver metrics comparison for ZDT4 problem.

Figure 4.40a presents the Pareto fronts of all runs, as stated before the Hybrid Methods based on the Genetic Algorithm are not performing well as a some individuals of these methods are dominated by the other methods. Figure 4.40b presents the non-dominated solutions among the different runs for each solver. For each optimization method the solutions of their runs are joined in a single population. With this population the dominance is computed to obtain the global set of non-dominated individuals achieved by each optimization method. Figure 4.40b clearly shows the difference in performance between the different optimization methods.



(a) (b)

Figure 4.40: (a) Overlapped Pareto fronts from all runs. (b) Non-dominated solutions of the combined Pareto fronts of all runs for each optimization method. (ZDT4).

4.3.5 ZDT6

The ZDT6 is a non-convex and non-uniformly spaced synthetic test problem proposed by Zitzler, Deb and Thiele [75]. It can be used with n design variables and it has 2 objective functions. In this tests it has been configured with $n = 10$ variables. The problem is defined as

$$\text{ZDT4}: \begin{cases} \text{Minimize} & f_1(x_1) = 1 - \exp(-4x_1)\sin^6(6\pi x_1) \\ \text{Minimize} & f_2(\mathbf{x}) = g \cdot h \\ \text{Domain} & 0 \leq x_i \leq 1 \quad , \quad i = 1, 2, \cdots, n \end{cases} \tag{4.16a}$$

$$g(x_2, \ldots, x_n) = 1 + 9\left(\frac{\sum_{i=2}^{n} x_i}{n-1}\right)^{0.25} \tag{4.16b}$$

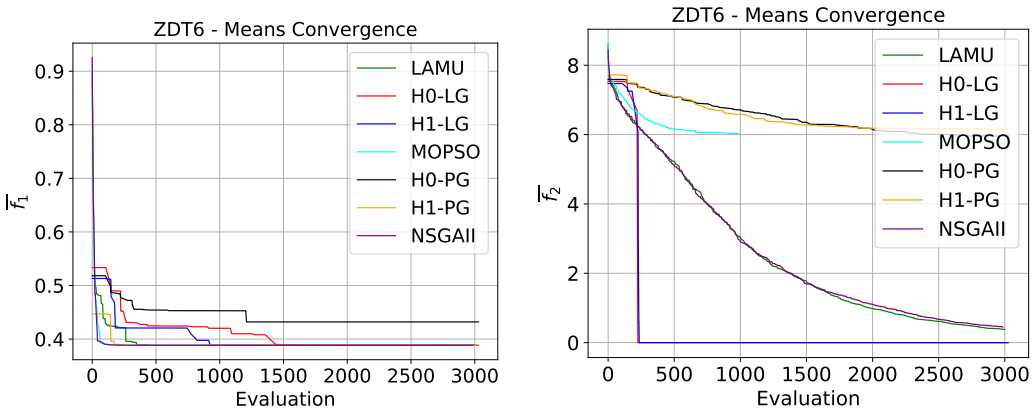
$$h(f_1, g) = 1 - \left(\frac{f_1}{g}\right)^2 \tag{4.16c}$$

The Pareto optimal front complies with $g(\mathbf{x}) = 1$ which yields

$$f_2 = 1 - (f_1)^2 \tag{4.17}$$

The behavior and performance of the optimization algorithms in the ZDT6 test problem is very similar to the observed in the ZDT1, ZDT2, ZDT3 and ZDT4 test problems.

Figure 4.41 presents the mean convergence of the two objective functions achieved by each optimization algorithm within its 12 runs. This information only gives information on the extremes of the Pareto front. All the optimization algorithms have converged to the best-f_1 extreme of the Pareto optimal front with high precision and accuracy, see Figure 4.41a. For the best-f_2 extreme the Particle Swarm Optimization method and the Hybrid Methods that have a MOPSO player (i.e., H0-PG and H1-PG), are fairly far from the optimal extreme. The two Genetic Algorithms got close to the optimum in average at the end of the optimization (3000 objective functions evaluations). The Hybrid Methods based on the Genetic Algorithm presents a high rate of convergence and converged to the exact extreme of f_2 at around 250 objective functions evaluations.

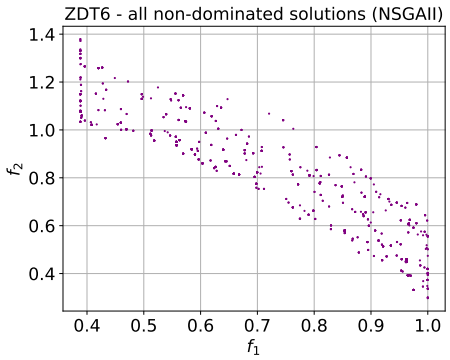


(a) $\overline{f_1}$ convergence. (b) $\overline{f_2}$ convergence.

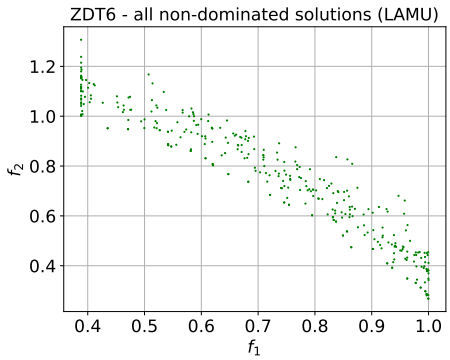
Figure 4.41: Mean convergence (ZDT6).

The Pareto fronts obtained by the 12 runs of each optimization algorithm are presented in Figure 4.42. Each figure contains all individuals from the Pareto fronts of each optimization test run, thus some individuals are dominated by individuals from another optimization run. The graphics show the robustness of each method and in some cases allow to visualize which region of the Pareto front are captured with better precision between different runs.

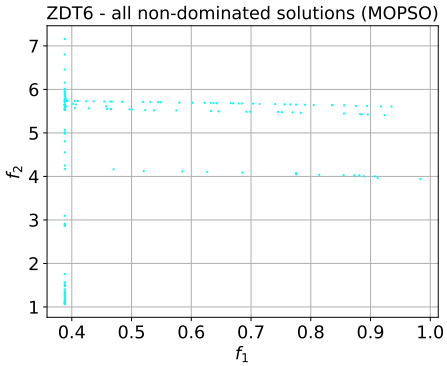
The Particle Swarm Optimization and the Hybrid Methods based on it (i.e., H0-PG and H1-PG) show a lack of robustness as the different runs created Pareto fronts far one from each other, with scattered bands in the MOPSO and scattered points in the MOPSO based Hybrid Methods. The NSGAII, the LAMU method and the Hybrid Methods based on the Genetic Algorithm (i.e., H0-LG and H1-LG) are performing much better than the others with more robustness.



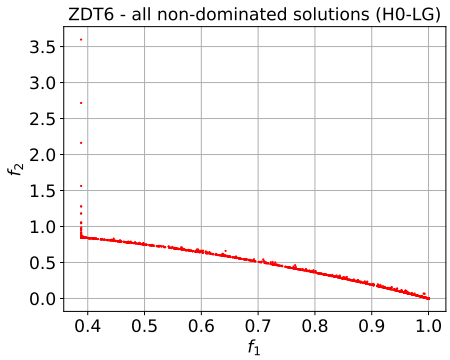
(a) NSGAII



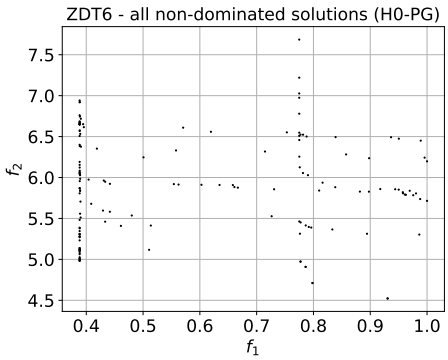
(b) LAMU



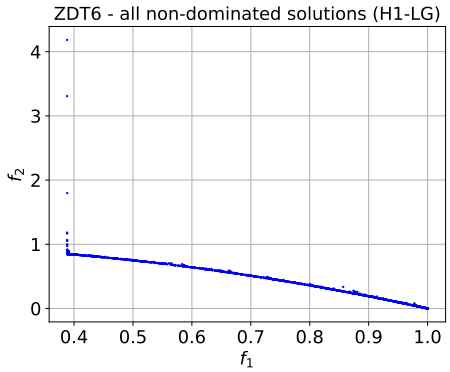
(c) MOPSO



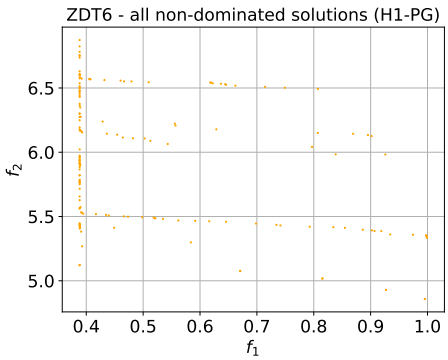
(d) H0-LG



(e) H0-PG



(f) H1-LG

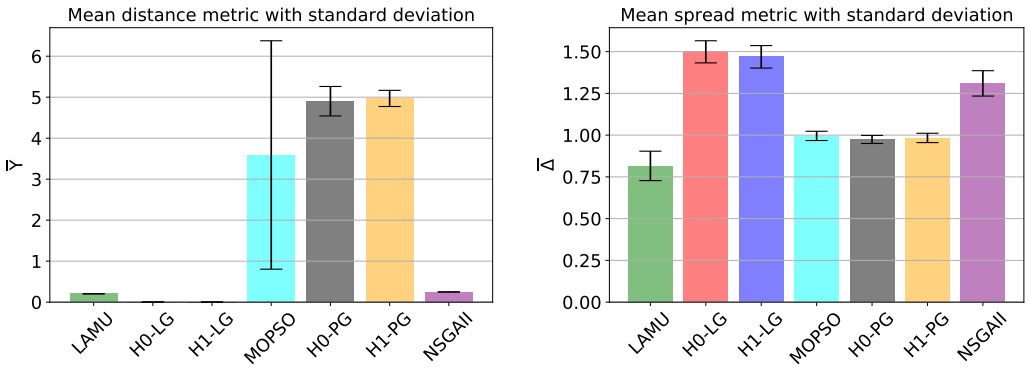


(g) H1-PG

Figure 4.42: Pareto front of each solver for all runs of the ZDT6 function.

Figure 4.43 and Table 4.11 presents the mean value of the performance metrics and the standard deviation for each optimization algorithm. The metrics have been computed for each one of the 12 runs of each optimization algorithm. With the values of the metrics the mean and standard deviation have been calculated for each optimization algorithm.

According to the distance metric (Table 4.11) the optimization algorithm that achieved a better convergence to the Pareto optimal front is the H0-LG with a value of $\overline{\Upsilon} = 0.045$, closely followed by the H1-LG $\overline{\Upsilon} = 0.058$. The two Genetic Algorithms also performed well: LAMU method $\overline{\Upsilon} = 2.337$ and NSGAII $\overline{\Upsilon} = 5.77$. These methods also demonstrated a high robustness with low values of the standard deviation compared to the other methods.



(a) Distance metric.

(b) Spread metric.

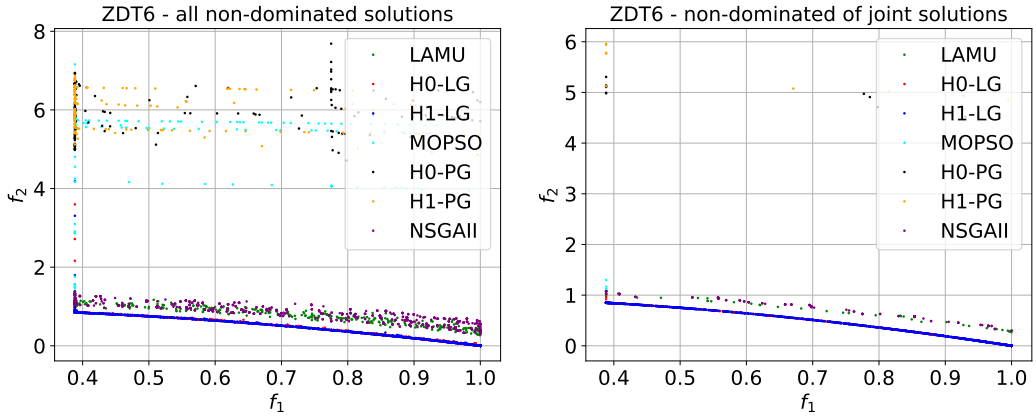
Figure 4.43: ZDT6 metrics.

The best spread metric is achieved by the H1-PG method which performed $\overline{\Delta} = 0.982$, although in this case is irrelevant, as the method is not performing well and all the individuals in the solution od this method is dominated by the Genetic Algorithm ones and the Hybrid Methods based on the Genetic Algorithm (i.e., H0-LG and H1-LG).

Method	$\overline{\Upsilon}$	σ_Υ	$\overline{\Delta}$	σ_Δ
LAMU	0.208	2.20×10^{-3}	0.816	8.81×10^{-2}
H0-LG	0.006	6.69×10^{-5}	1.499	6.63×10^{-2}
H1-LG	0.005	4.81×10^{-5}	1.469	6.74×10^{-2}
MOPSO	3.708	2.86	0.995	2.77×10^{-2}
H0-PG	5.028	3.57×10^{-1}	0.974	2.42×10^{-2}
H1-PG	5.101	1.99×10^{-1}	0.983	2.80×10^{-2}
NSGA2	0.256	4.99×10^{-3}	1.310	7.60×10^{-2}

Table 4.11: Solver metrics comparison for ZDT6 problem.

Figure 4.44a presents the Pareto fronts of all runs, as stated before the Hybrid Methods based on the Genetic Algorithm are not performing well as a some individuals of these methods are dominated by the other methods. Figure 4.44b presents the non-dominated solutions among the different runs for each solver. For each optimization method the solutions of their runs are joined in a single population. With this population the dominance is computed to obtain the global set of non-dominated individuals achieved by each optimization method. Figure 4.44b clearly shows the difference in performance between the different optimization methods.



(a) (b)

Figure 4.44: (a) Overlapped Pareto fronts from all runs. (b) Non-dominated solutions of the combined Pareto fronts of all runs for each optimization method. (ZDT6).

4.3.6 FON

The Fonseca-Fleming problem (FON) [76, 67] is a non-convex two objective optimization problem. The Pareto optimal front is continuous. In the tests of this study three design variables have been used for the FON test problem. The corresponding optimization problem is defined as

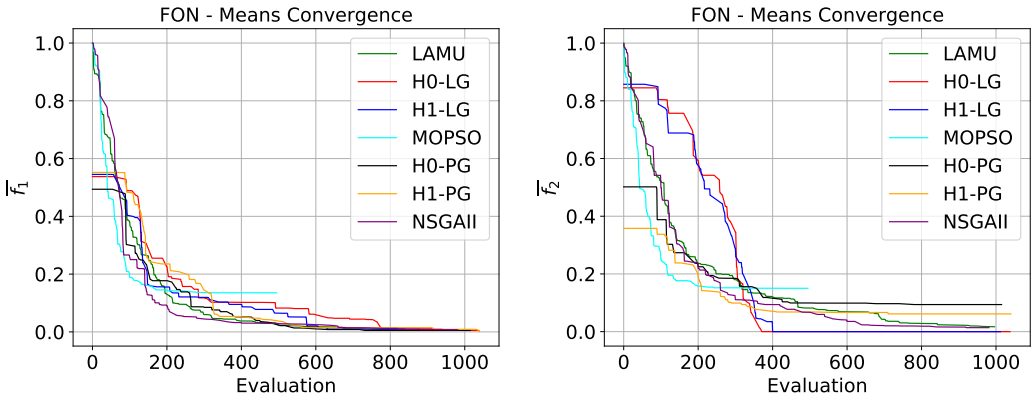
$$\text{FON}: \begin{cases} \text{Minimize} & f_1(\mathbf{x}) = 1 - exp\left[-\sum_{i=1}^{n}\left(x_i - \frac{1}{\sqrt{n}}\right)^2\right] \\ \text{Minimize} & f_2(\mathbf{x}) = 1 - exp\left[-\sum_{i=1}^{n}\left(x_i + \frac{1}{\sqrt{n}}\right)^2\right] \\ \text{Domain} & -4 \leq x_i \leq 4 \quad , \quad i = 1, 2, \cdots, n \end{cases} \tag{4.18}$$

and the Pareto optimal region is

$$x_0^* = x_1^* = x_i^* ... = x_n^* \quad \in \left[\frac{-1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right] \quad , \quad i = 1, 2, \cdots, n.$$

Figure 4.45 presents the mean convergence through the minimum of the two objective functions achieved by each optimization algorithm within its 12 runs. This

information only gives information about the extremes of the Pareto front. All optimization algorithms have achieved a similar rate of convergence for the first objective function, see Figure 4.45a. The Particle Swarm Optimization has not achieved the same accuracy as the rest of the optimization algorithms. Regarding the second objective function the Evolutionary Algorithms have achieved a higher initial rate of convergence, but the two Hybrid Methods with the Genetic Algorithm as a player (i.e., H0-LG and H1-LG) have converged to the extreme faster and more accurately as they finally achieved a higher rate of convergence.

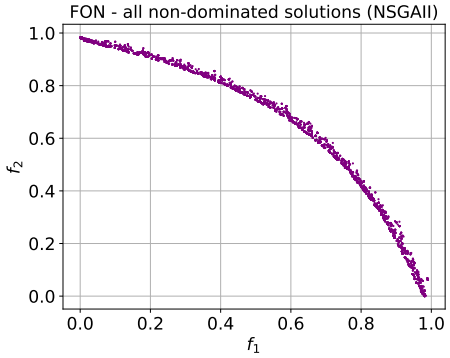


(a) $\overline{f_1}$ convergence. (b) $\overline{f_2}$ convergence.

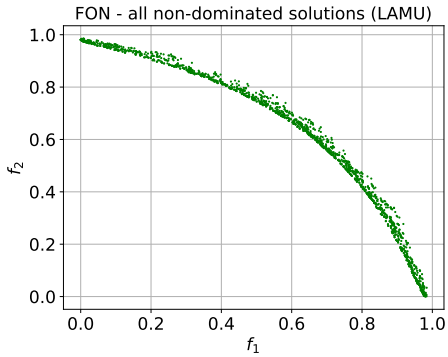
Figure 4.45: Mean convergence (FON).

The Pareto fronts obtained by the 12 runs of each optimization algorithm are presented in Figure 4.46. Each figure contains all individuals from all the the Pareto fronts of each optimization test run, thus some individuals are dominated by individuals from another optimization run. The graphics show the robustness of each method and in some cases allow to visualize which region of the Pareto front are captured with better precision between different runs.

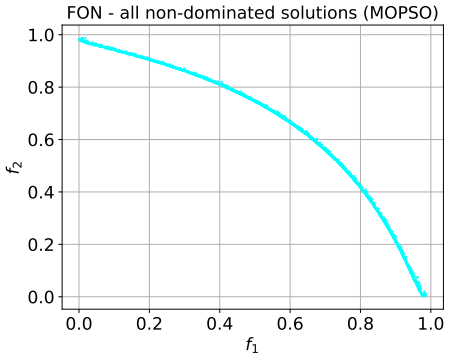
The two Hybrid Methods with the Genetic Algorithm as a player (i.e., H0-LG and H1-LG), although they captured the extremes of the Pareto front, are the less accurate and precise finding the rest of the Pareto optimal front. The Particle Swarm Optimization seems the method with better precision as the metric confirms, which is discussed below.



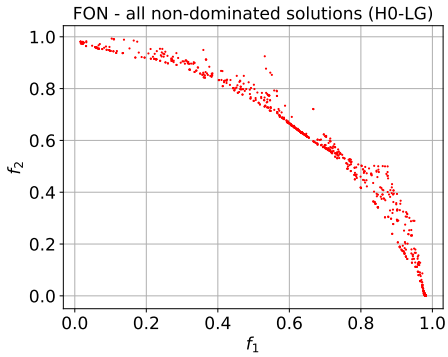
(a) NSGAII



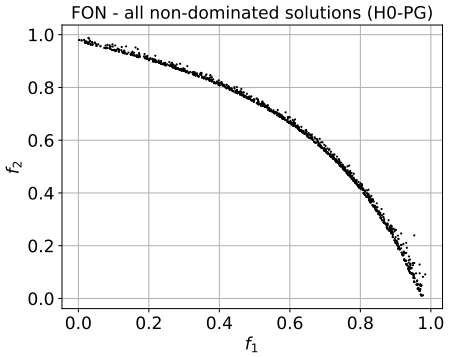
(b) LAMU



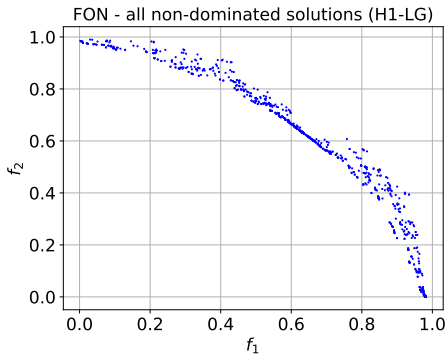
(c) MOPSO



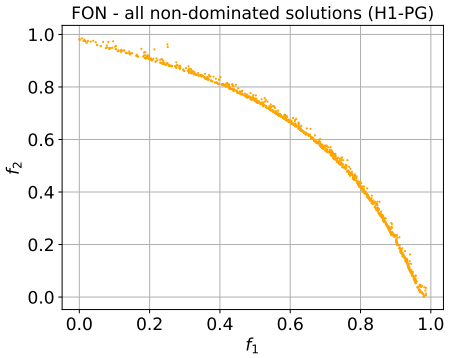
(d) H0-LG



(e) H0-PG



(f) H1-LG

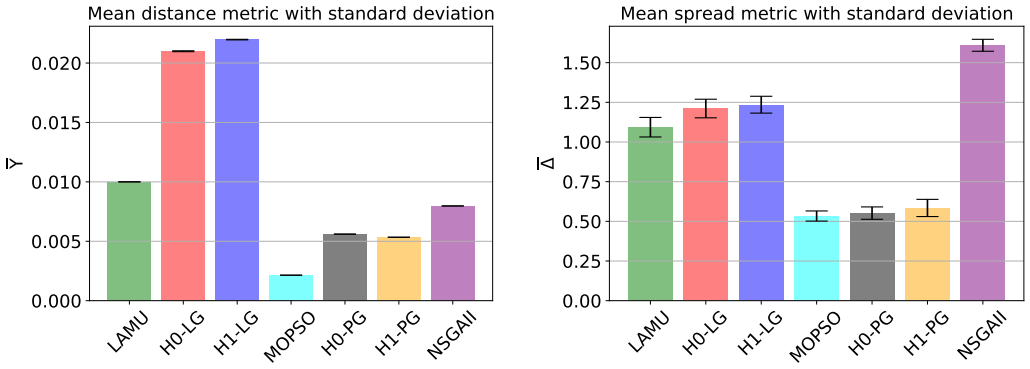


(g) H1-PG

Figure 4.46: Pareto front of each solver for all runs of the FON function.

Figure 4.47 and Table 4.12 presents the mean value of the performance metrics and the standard deviation for each optimization algorithm. The metrics have been computed for each one of the 12 runs of each optimization algorithm. With the values of the metrics the mean and standard deviation have been calculated for each optimization algorithm.

According to the distance metric (Table 4.12) the optimization algorithm that achieved a better convergence to the Pareto optimal front is the Particle Swarm Optimization with a value of $\overline{\Upsilon} = 0.005$. It is also the most robust optimization method with with the best standard deviation. The NSGAII method ($\overline{\Upsilon} = 0.008$) performed slightly better than the LAMU method $\overline{\Upsilon} = 0.01$. The Hybrid Methods have performed worst that its Evolutionary Algorithms running alone, specially in the distance metric.



(a) Distance metric.

(b) Spread metric.

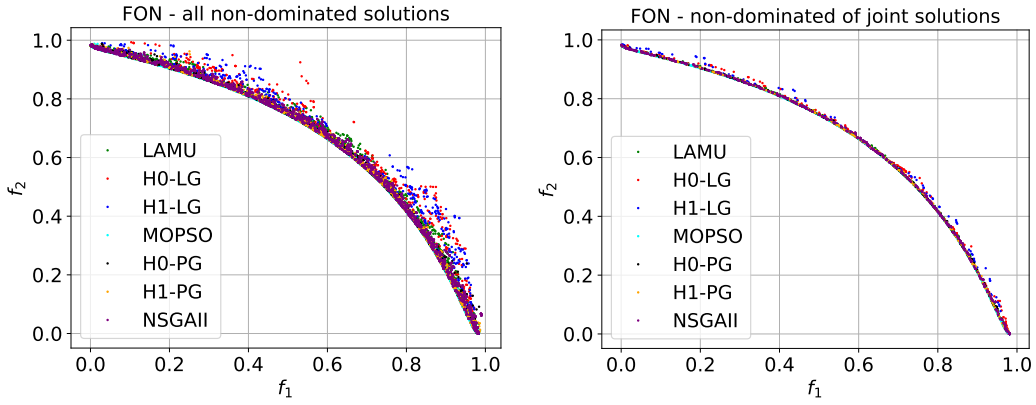
Figure 4.47: FON metrics.

The best spread metric is also achieved by the Particle Swarm Optimization method, see Figure 4.47b. Followed by the LAMU Genetic Algorithm method and NSGAII. The spread metric of the Hybrid Methods are very similar to the one of the Evolutionary Algorithm that is composed by as a player, although they have a slightly worst performance in the spread metric.

Method	$\overline{\Upsilon}$	σ_Υ	$\overline{\Delta}$	σ_Δ
LAMU	0.010	4.82×10^{-6}	1.093	6.16×10^{-2}
H0-LG	0.021	3.78×10^{-5}	1.211	5.86×10^{-2}
H1-LG	0.022	2.49×10^{-5}	1.235	5.31×10^{-2}
MOPSO	0.002	4.47×10^{-8}	0.533	3.20×10^{-2}
H0-PG	0.006	1.80×10^{-6}	0.552	3.90×10^{-2}
H1-PG	0.005	1.98×10^{-6}	0.584	5.42×10^{-2}
NSGA2	0.008	1.39×10^{-6}	1.609	3.77×10^{-2}

Table 4.12: Solver metrics comparison for FON problem.

Figure 4.48a presents the Pareto fronts of all runs. As stated before, the Hybrid Methods based on the Genetic Algorithm are not performing well as some individuals of these methods are dominated by the other methods. Figure 4.48b presents the non-dominated solutions among the different runs for each solver. For each optimization method the solutions of their runs are joined in a single population. With this population the dominance is computed to obtain the global set of non-dominated individuals achieved by each optimization method. As it can be seen in Figure 4.48b several individuals of the Pareto fronts of the H0-LG and H1-LG methods are dominated by individuals of other methods.



(a) (b)

Figure 4.48: (a) Overlapped Pareto fronts from all runs. (b) Non-dominated solutions of the combined Pareto fronts of all runs for each optimization method. (FON).

4.3.7 KUR

The KUR problem was proposed by Kursawe [77]. Deb [19] also provides a detailed study of this test problem. It is a non-convex problem with a disconnected Pareto optimal front. The problem has three variables and two objective functions. The problem is defined as

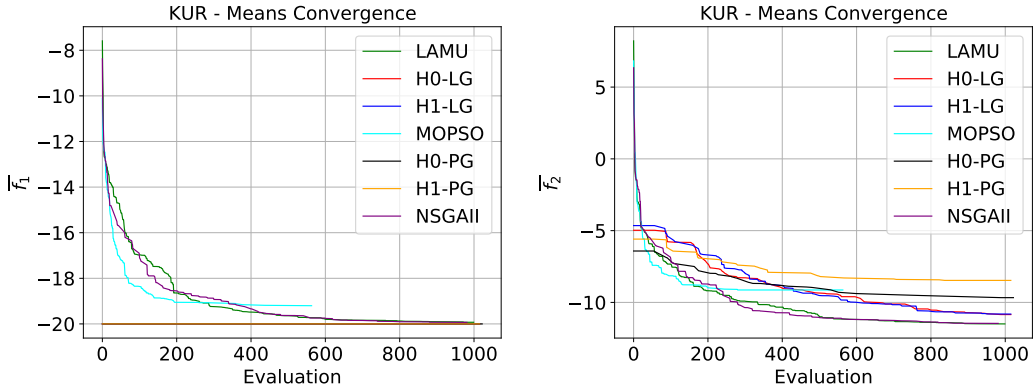
$$\text{KUR}: \begin{cases} \text{Minimize} & f_1(\mathbf{x}) = \sum_{i=1}^{2} \left[-10 \exp\left(-0.2\sqrt{x_i^2 + x_{i+1}^2}\right)\right] \\ \text{Minimize} & f_2(\mathbf{x}) = \sum_{i=1}^{3} \left[|x_i|^{0.8} + 5\sin\left(x_i^3\right)\right] \\ \text{Domain} & -5 \leq x_i \leq 5 \quad , \quad i = 1, 2, 3 \end{cases} \tag{4.19}$$

The Pareto optimal front complies with

$$\begin{aligned} \exp\left(-0.2x_3^*\right)\left[15\left(x_1^*\right)^2\cos\left(x_1^*\right)^3 - 0.8\left(-x_3^*\right)^{-0.2}\right] = \\ \exp\left(-0.2x_1^*\right)\left[15\left(x_3^*\right)^2\cos\left(x_3^*\right)^3 - 0.8\left(-x_3^*\right)^{-0.2}\right]. \end{aligned} \tag{4.20}$$

Figure 4.49 presents the mean convergence of the two objective functions achieved by each optimization algorithm within its 12 runs. This information only gives

information on the extremes of the Pareto front. The Evolutionary Algorithms have achieved a similar rate of convergence for the first objective function, see Figure 4.49a. The Particle Swarm Optimization has not achieved the same precision as the rest of the optimization algorithms. The four Hybrid Methods have converged extremely fast to the minimum of the first objective function. Regarding the second objective function all optimization algorithms have achieved a similar rate of convergence. The two Hybrid Methods based on the Particle Swarm Optimization (i.e., H0-PG and H1-PG) and the Particle Swarm Optimization have not reached the extreme of the Pareto optimal front. The Hybrid Methods based on the Genetic Algorithm (i.e., H0-LG and H1-LG) have not converged to the Pareto optimal extreme of f_2, but obtained better results than the other two Hybrid Methods.



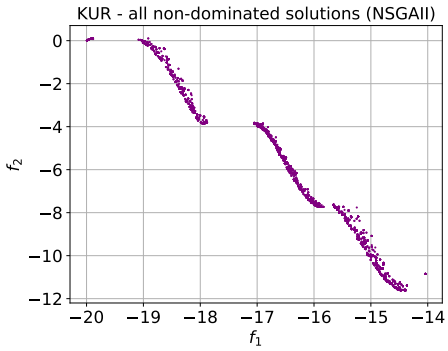
(a) $\overline{f_1}$ convergence.

(b) $\overline{f_2}$ convergence.

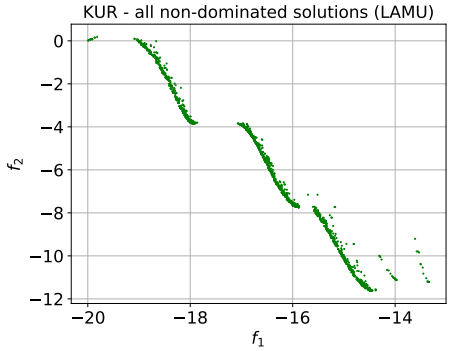
Figure 4.49: Mean convergence (KUR).

The Pareto fronts obtained by the 12 runs of each optimization algorithm are presented in Figure 4.50. Each figure contains all individuals from the Pareto fronts of each optimization test run, thus some individuals are dominated by individuals from another optimization run. The graphics show the robustness of each method and in some cases allow to visualize which region of the Pareto front are captured with better precision between different runs.

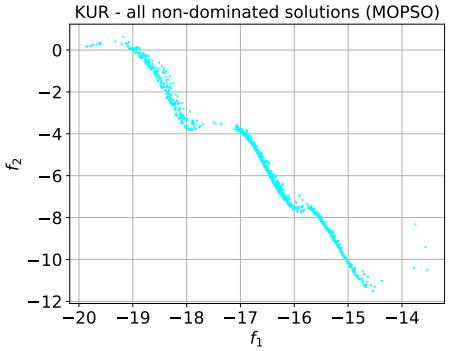
The Hybrid Methods are the least precise (least robust) as the dispersion of the individuals from the Pareto front is higher than the Evolutionary Algorithms results. The three Evolutionary Algorithm performed fairly similar although the LAMU method has more dispersion near the best values of the f_2 (see bottom-right of the graphic Figure 4.50b).



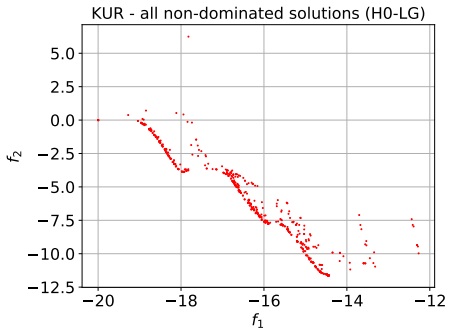
(a) NSGAII



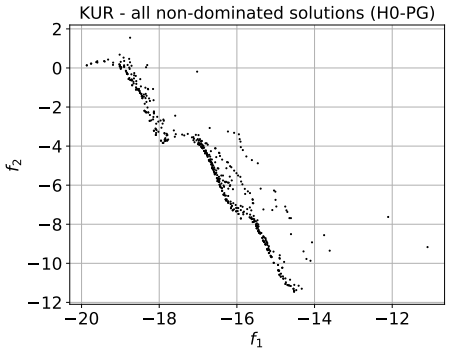
(b) LAMU



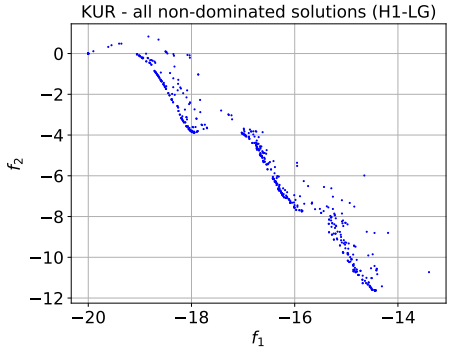
(c) MOPSO



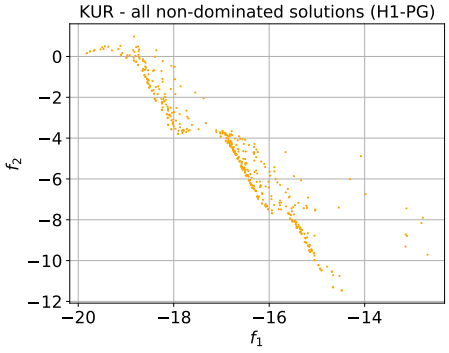
(d) H0-LG



(e) H0-PG



(f) H1-LG

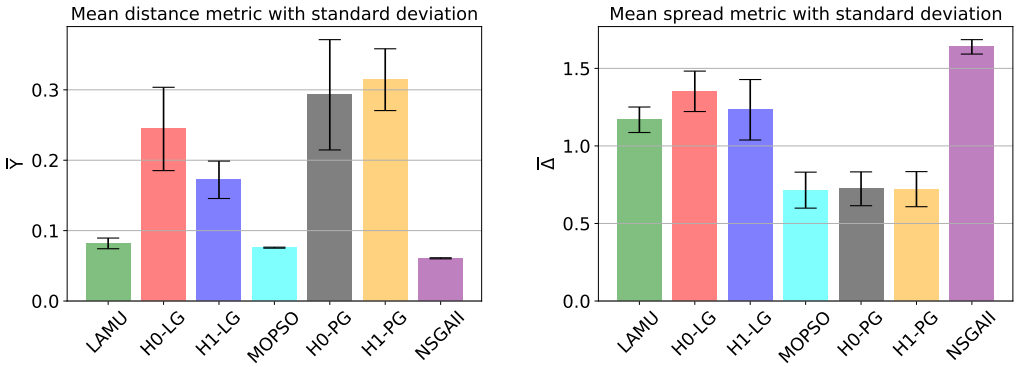


(g) H1-PG

Figure 4.50: Pareto front of each solver for all runs of the KUR function.

Figure 4.51 and Table 4.13 presents the mean value of the performance metrics and the standard deviation for each optimization algorithm. The metrics have been computed for each one of the 12 runs of each optimization algorithm. With the values of the metrics the mean and standard deviation have been calculated for each optimization algorithm.

According to the distance metric (Table 4.13) the optimization algorithm that achieved a better convergence to the Pareto optimal front is the NSGAII with a value of $\overline{\Upsilon} = 0.061$, closely followed by the Particle Swarm Optimization $\overline{\Upsilon} = 0.076$ and the LAMU method $\overline{\Upsilon} = 0.082$. The NSGAII and Particle Swarm Optimization methods over performed the other methods with the best accuracy and also precision with the lower values of the standard deviation as well. The Hybrid Methods performed worst than their based Evolutionary Algorithm in terms of the distance metric mean and standard deviation.



(a) Distance metric.

(b) Spread metric.

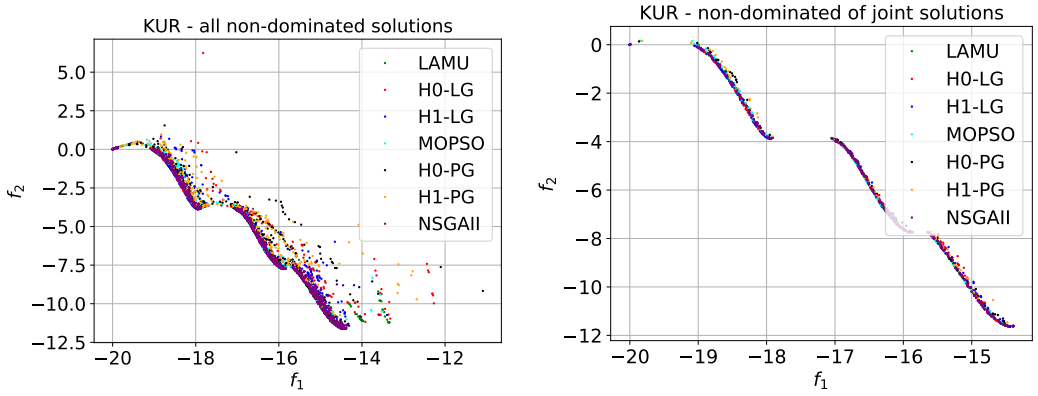
Figure 4.51: KUR metrics.

The best spread metric is also achieved by the Particle Swarm Optimization method, see Figure 4.51b. Followed by the LAMU Genetic Algorithm method and NSGAII. The spread metric of the Hybrid Methods are very similar to the one of the Evolutionary Algorithm that is composed by as a player, although in the Genetic Algorithm ones have a slightly worst performance in the spread metric.

Method	$\overline{\Upsilon}$	σ_Υ	$\overline{\Delta}$	σ_Δ
LAMU	0.082	7.59×10^{-3}	1.169	8.22×10^{-2}
H0-LG	0.244	5.92×10^{-2}	1.352	1.30×10^{-1}
H1-LG	0.172	2.66×10^{-2}	1.233	1.95×10^{-1}
MOPSO	0.076	7.21×10^{-4}	0.715	1.16×10^{-1}
H0-PG	0.293	7.83×10^{-2}	0.724	1.09×10^{-1}
H1-PG	0.314	4.39×10^{-2}	0.721	1.13×10^{-1}
NSGA2	0.061	7.87×10^{-4}	1.639	4.65×10^{-2}

Table 4.13: Solver metrics comparison for KUR problem.

Figure 4.52a presents the Pareto fronts of all runs. As stated before the Hybrid Methods based on the Genetic Algorithm are not performing well as a some individuals of these methods are dominated by the other methods. Figure 4.52b presents the non-dominated solutions among the different runs for each solver. For each optimization method the solutions of their runs are joined in a single population. With this population the dominance is computed to obtain the global set of non-dominated individuals achieved by each optimization method. As it can be seen in Figure 4.52b several individuals of the Pareto fronts of the four Hybrid Methods are dominated by individuals of other methods.



(a) (b)

Figure 4.52: (a) Overlapped Pareto fronts from all runs. (b) Non-dominated solutions of the combined Pareto fronts of all runs for each optimization method. (KUR).

4.3.8 POL

Poloni et al. [78] proposed another non-convex test problem with a disconnected Pareto optimal front. The problem has two objective functions and two design

variables. The problem is defined as

$$\text{POL}: \begin{cases} \text{Minimize} & f_1(\mathbf{x}) = 1 + (A_1 - B_1)^2 + (A_2 - B_2)^2 \\ \text{Minimize} & f_2(\mathbf{x}) = (x_1 + 3)^2 + (x_2 + 1)^2 \\ \text{Domain} & -\pi \leq x_i \leq \pi \quad , \quad i = 1, 2 \end{cases} \tag{4.21a}$$

$$A_1 = 0.5 * \sin(1) - 2.0\cos(1) + \sin(2) - 1.5\cos(2) \tag{4.21b}$$

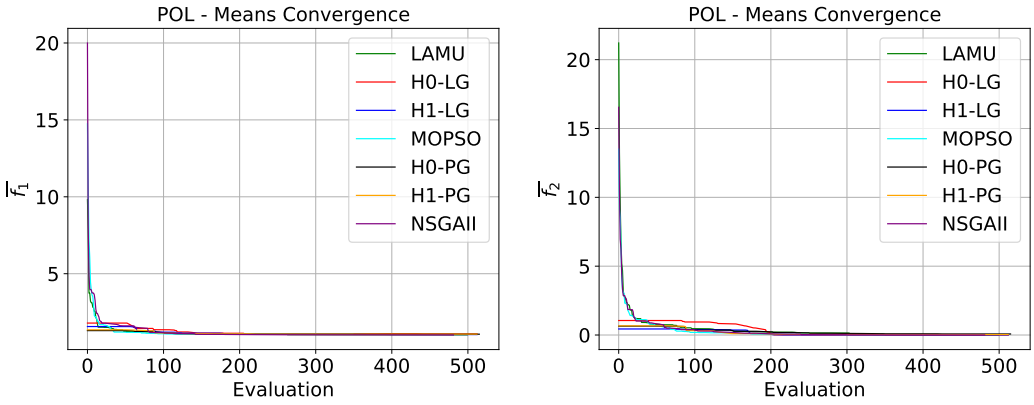
$$A_2 = 1.5\sin(1) - \cos(1) + 2.0\sin(2) - 0.5\cos(2) \tag{4.21c}$$

$$B_1 = 0.5\sin(x_1) - 2.0\cos(x_1) + \sin(x_2) - 1.5\cos(x_1) \tag{4.21d}$$

$$B_2 = 1.5\sin(x_1) - \cos(x_1) + 2.0\sin(x_2) - 0.5\cos(x_2) \tag{4.21e}$$

For the Pareto optimal front solution refer to [19].

Figure 4.53 presents the mean convergence of the two objective functions achieved by each optimization algorithm within its 12 runs. This information only gives information on the extremes of the Pareto front. The Evolutionary Algorithms have achieved a similar rate of convergence for the first objective function, see Figure 4.53a. In this test case all optimization algorithms have achieved the Pareto optimal front extremes in both functions.



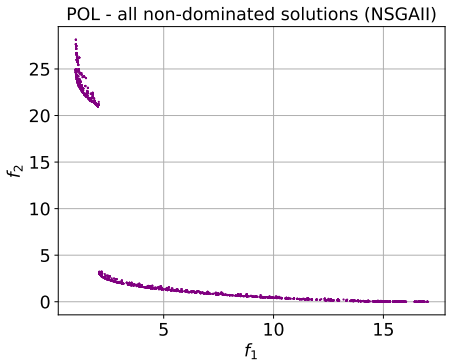
(a) $\overline{f_1}$ convergence. (b) $\overline{f_2}$ convergence.

Figure 4.53: Mean convergence (POL).

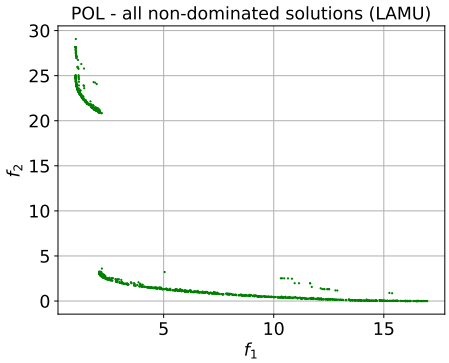
The Pareto fronts obtained by the 12 runs of each optimization algorithm are presented in Figure 4.54. Each figure contains all individuals from the Pareto fronts of each optimization test run, thus some individuals are dominated by individuals from another optimization run. The graphics show the robustness of each method and in some cases allow to visualize which region of the Pareto front are captured with better precision between different runs.

The Hybrid Methods based on the Genetic Algorithm (i.e., H0-LG and H1-LG) and the LAMU method are the least precise (least robust) as there is more dispersion of individuals away from the Pareto front. The other Hybrid Methods based on the

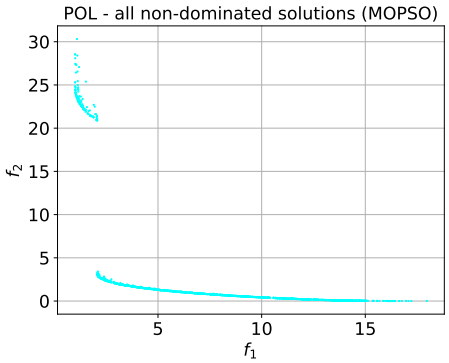
Particle Swarm Optimization performs better in terms of robustness than the other Hybrid Methods.



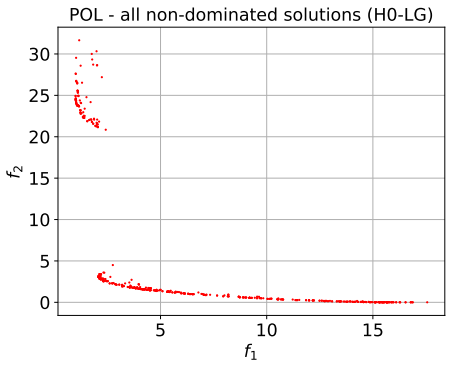
(a) NSGAII



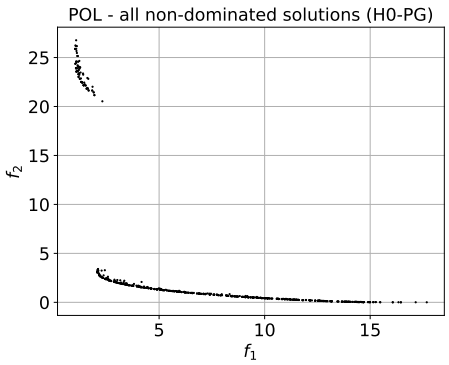
(b) LAMU



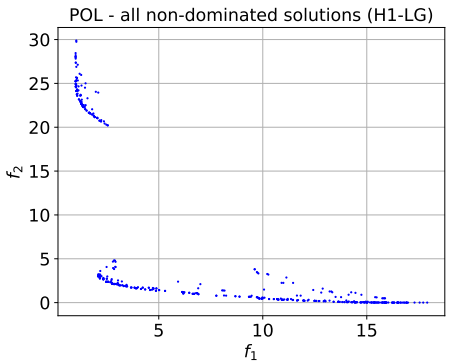
(c) MOPSO



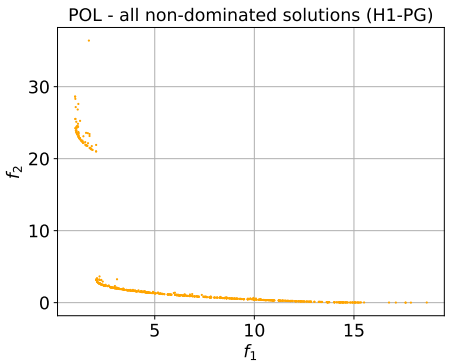
(d) H0-LG



(e) H0-PG



(f) H1-LG

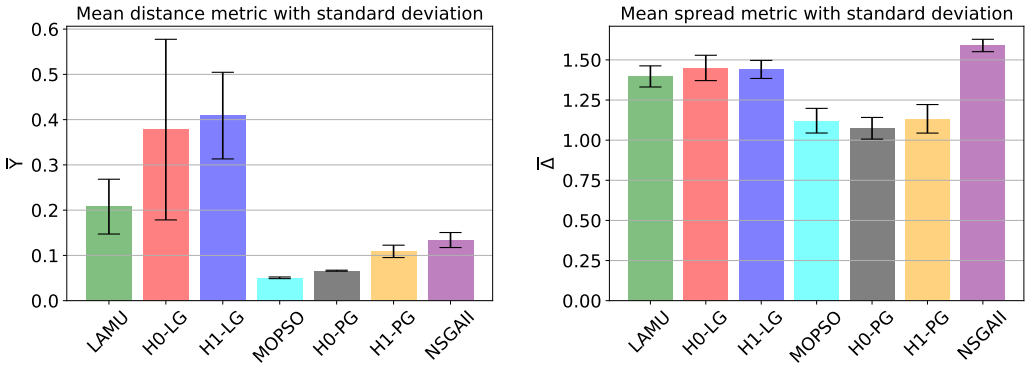


(g) H1-PG

Figure 4.54: Pareto front of each solver for all runs of the POL function.

Figure 4.55 and Table 4.14 presents the mean value of the performance metrics and the standard deviation for each optimization algorithm. The metrics have been computed for each one of the 12 runs of each optimization algorithm. With the values of the metrics the mean and standard deviation have been calculated for each optimization algorithm.

According to the distance metric (Table 4.14) the optimization algorithm that achieved a better convergence to the Pareto optimal front is the Particle Swarm Optimization with a value of $\overline{\Upsilon} = 0.051$, closely followed by the Hybrid Methods based on the Particle Swarm Optimization (i.e., H0-PG $\overline{\Upsilon} = 0.066$ and H1-PG $\overline{\Upsilon} = 0.109$) and the LAMU method $\overline{\Upsilon} = 0.134$. The LAMU method and the Hybrid Methods based on the Genetic Algorithm performed significantly worst in accuracy and precision.



(a) Distance metric.

(b) Spread metric.

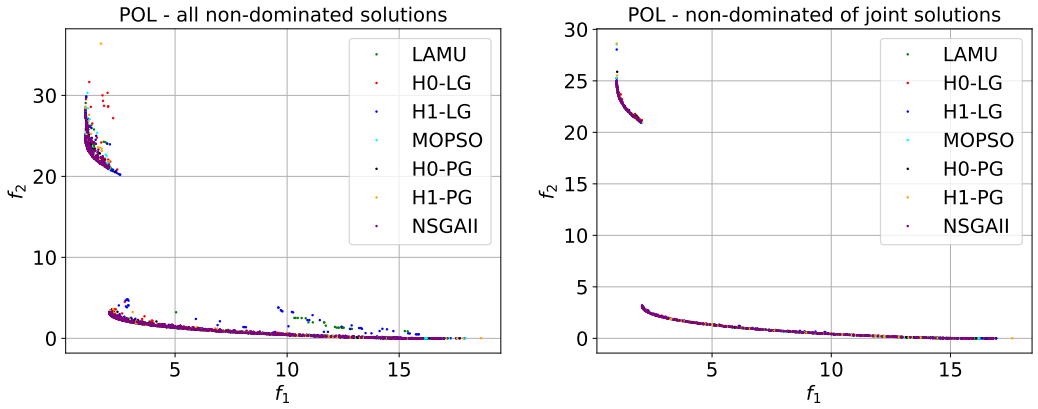
Figure 4.55: POL metrics.

The best spread metric is achieved by the H0-PG method which performed $\overline{\Delta} = 1.074$ slightly better than the Particle Swarm Optimization and the H1-PG method, see Figure 4.55b. The rest of the methods performed similarly between them with a spread metric $\overline{\Delta} \approx 1.4$.

Method	$\overline{\Upsilon}$	σ_Υ	$\overline{\Delta}$	σ_Δ
LAMU	0.208	6.05×10^{-2}	1.397	6.58×10^{-2}
H0-LG	0.378	1.99×10^{-1}	1.450	7.89×10^{-2}
H1-LG	0.409	9.57×10^{-2}	1.441	5.63×10^{-2}
MOPSO	0.051	1.84×10^{-3}	1.121	7.68×10^{-2}
H0-PG	0.066	1.10×10^{-3}	1.074	6.75×10^{-2}
H1-PG	0.109	1.37×10^{-2}	1.133	8.90×10^{-2}
NSGA2	0.134	1.66×10^{-2}	1.590	3.85×10^{-2}

Table 4.14: Solver metrics comparison for POL problem.

Figure 4.56a presents the Pareto fronts of all runs, as stated before the Hybrid Methods based on the Genetic Algorithm are not performing well as a some individuals of these methods are dominated by the other methods. Figure 4.56b presents the non-dominated solutions among the different runs for each solver. For each optimization method the solutions of their runs are joined in a single population. With this population the dominance is computed to obtain the global set of non-dominated individuals achieved by each optimization method.



(a) (b)

Figure 4.56: (a) Overlapped Pareto fronts from all runs. (b) Non-dominated solutions of the combined Pareto fronts of all runs for each optimization method. (POL).

4.3.9 Summary

The Hybrid Methods performed extremely well in the family of functions ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6, specially the hybrids based on the Genetic Algorithm which increased the rate of convergence of the Evolutionary Algorithm. The spread metric of the Hybrid Methods based on the GA (i.e. H0-LG and H1-LG) is higher (worst) than the achieved by the LAMU method in this functions, but if the results are closely analyzed the spread metric is penalizing the high density of non-dominated solutions on some regions of the Pareto front and not the lack of solutions. The Hybrid Methods based on the GA dominate the rest of the methods with more density of solutions, so in this case the spread metric is not indicative that a higher metric is better.

In the FON, KUR and POL benchmarks the Hybrid Methods performed worst than the pure Evolutionary Algorithms as the hybrid non-dominated solutions where highly dominated by the ones from the pure methods, although the convergence at the extremes was captured faster by the hybrids. This is probably because once the exact extremes where found by the hybrids they kept dedicating resources to that regions and not allowing to explore the rest of the search space.

4.4 Optimization of Synthetic Jet device for active flow control

In order to evaluate the performance of the proposed Hybrid Methods when applied to an industrial interest test case, it has been compared against two classical optimization methods, a plain conjugate gradient and a plain genetic algorithm. The conjugate gradient method is the same that forms part of the Hybrid Method, but running on its own. The genetic algorithm used to compare the hybrid algorithm is also the same that forms part of the Hybrid Method, but running alone too. To take into account the random component of the genetic algorithm and the strong dependence on the starting point of the conjugate gradient, multiple optimizations with each algorithm have been conducted.

The optimization methods have been tested against the same test case. The test case consists on an optimization of an active flow control optimization problem based on the work of [79]. The objective of that work is to determine the optimum parameters of the Synthetic Jet Actuator design at different angles of attack in a multiple objective optimization problem. The test case details for the comparison of optimization algorithms are presented in Section 4.4.1.

4.4.1 Test case description

The test case focuses on the optimization of an active flow control device, more precisely a synthetic jet actuator. The device is tested on a SD7003 airfoil at an angle of attack of 14 degrees. For the comparison between the optimization algorithms, which is the main objective of this work, a single optimization problem and a two objectives problem have been defined. Both problems use the same 5 design variables.

At high angles of attack the synthetic jet can greatly affect the flow structure improving the lift coefficient. The synthetic jet actuator, if set properly, can help to reattach the flow to the airfoil or almost avoid the flow to detach.

The design variables are the same as the previous work by [79], for a full explanation and detail of the synthetic jet actuator design variables meaning refer to [79]. The five design variables are:

F^+ Non-dimensional frequency.

C_μ Momentum coefficient.

θ Jet inclination angle.

x/C Non-dimensional jet position.

h/C Non-dimensional jet width.

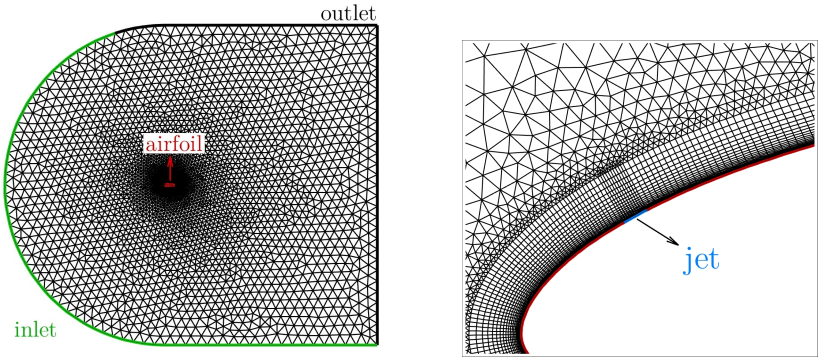
The evaluation range of each design variable is shown in Table 4.15. The same ranges are used with all optimization algorithms.

Table 4.15: Active flow control design variables and their evaluation ranges.

Design variable	Minimum value	Maximum value
F^+	0.1	10
C_μ	0.0001	0.02
θ^o	5	175
x/C	0.001	0.3
h/C	0.005	0.015

The flow has been solved with an unsteady Reynolds averaged Navier-Stokes model (URANS), using the OpenFOAM software. Other models like Direct Numerical Simulation (DNS) or Large Eddy Simulation (LES) could be used to solve the synthetic jet simulation, but its computational cost is too high to perform so many optimizations with the available resources. In addition, there is no need to use such precise solvers to evaluate the performance of the Hybrid Method. More details on the solver used can be consulted at [79], as this study uses the same model.

The mesh used, see Figure 4.57a, is one of the meshes previously evaluated in [79] although having a smaller number of cells 34,448 than the final one employed in that paper, the maximum y^+ after the simulation was $y^+ = 1$. The mesh nearby the synthetic jet actuator is presented in Figure 4.57b. The run time of the simulation has been adjusted to 30 time units, which as shown in [79] is sufficient to reach convergence. It is important to notice, that this study is not about the synthetic jet actuator optimization, but to compare the optimization algorithms in a real world application with a significant computational cost and complexity. The study of the physical problem is not the main purpose of this study, which justifies reducing the precision of each CFD simulation in order to reduce the overall computational cost.



(a) Full mesh of the domain.

(b) Mesh nearby the synthetic jet actuator.

Figure 4.57: Entire mesh domain (a) and zoomed view of the jet location (b).

In Figure 4.58, the temporal averaged streamlines and pressure field for the non-actuated case is presented. This configuration is called the baseline. From the

streamlines it can be seen that the flow is fully separated, the airfoil is under stall conditions.

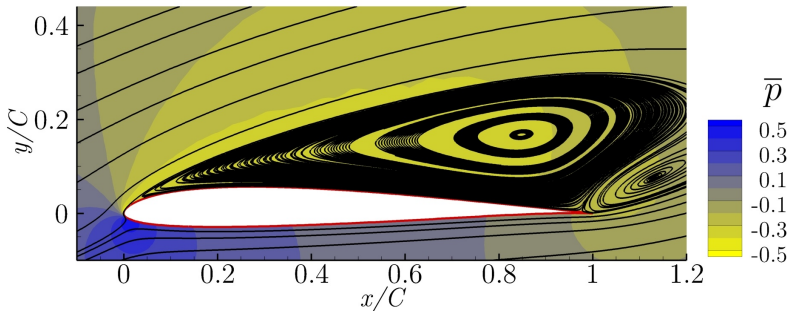


Figure 4.58: Averaged streamlines and pressure field of baseline case ($C_l = 0.80$).

The time to solve the flow problem is around 15 hours for each individual, although not all cases takes the same amount of time as it depends on the exact conditions and the convergence of the iterative solver. With the solution of the flow the objective functions can be computed. The hardware used to compute this optimizations are four Intel(R) Xeon(R) CPU E7- 4860 @ 2.27GHz, which gives a total of 40 cores (80 threads) with 224 GB in a shred memory configuration. The solver of the flow has been configured to use a single thread. The optimizer used multiple cores to evaluate individuals in parallel.

Two set of optimizations have been conducted to test the algorithms, a single objective optimization and a two objectives optimization which are described in the following sections.

4.4.2 One objective function results

This section introduces the results obtained for the single objective test case. The objective of the design problem is to maximize the lift coefficient C_l, to do so the objective function of the minimization problem is set to

$$f = -C_l \tag{4.22}$$

This case has been run two times with the LAMU method, two times each of the two Hybrid Methods based on it (i.e. H0-LG and H1-LG), six times the Conjugate Gradient and two the Particle Swarm Optimization. The rest of the algorithms have not been tested with this problem due to the high computational resources that it requires. The rest of the algorithms have not been tested in this case due to the high computational cost, the available resources and time.

The convergence of the different optimization algorithms are shown in Figure 4.59. The results shown in the graph of Figure 4.59 reflect the problems encountered by the gradient based method. Most runs of the conjugate gradient, initially, improve faster than the genetic algorithm but then get stuck between $C_l \approx 1.35$ and $C_l \approx$

1.45 (except for two runs that get stuck at $C_l \approx 0.8$ and $C_l \approx 1.25$, respectively). Those lift coefficient values are achieved with almost 100 evaluations of the objective function for each optimization of the conjugate gradient. The strong dependence of the conjugate gradient on the starting point is also reflected on these results, as it presents much different solutions between runs than the other methods. In all cases, the conjugate gradient method was stopped because the algorithm found a local minimum and could not compute the gradient to further improve the results.

On the other hand, the genetic algorithm provides optimal solutions similar to the conjugate gradient, but with a higher computational cost, approximately 4 times higher. Both runs of the genetic algorithm achieve values of the objective function in the range of the conjugate gradient results. One of the runs achieves a better objective function than all of the conjugate gradient runs, with a value of $C_l = 1.49$. It is important to note that the genetic algorithm optimizations could run additional iterations and achieve better results, but with a high computational cost.

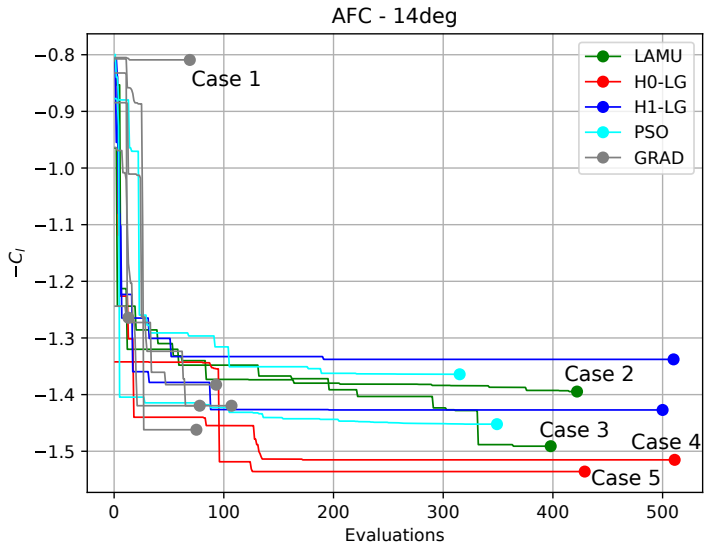


Figure 4.59: Comparative of the convergence of the different optimization methods.

The Hybrid Method is the method that achieved better results, outperforming all runs of the other algorithms in both of its runs. During the first iterations, it achieved a convergence rate similar to the conjugate gradient. But, the improvement of the solution has continued, avoiding being trapped in any local minimum. It is also the most robust, as both runs are very similar in its performance. Both Hybrid Method runs outperformed all the other optimization methods with a $C_l = 1.52$ and $C_l = 1.54$. One run of the genetic algorithm achieved a comparable solution ($C_l = 1.49$) but it took more than twice the computational cost of the Hybrid Method. Case 3 with a lift coefficient of $C_l = 1.49$, obtained by the genetic algorithm with around 375 objective function evaluations, improves the baseline lift coefficient by 86%. Cases 4 and 5, obtained by the Hybrid Method runs achieved a lift coefficient of $C_l = 1.52$ and $C_l = 1.54$ respectively. Both runs needed around 125 objective

functions evaluations, which is 67% the number of evaluations of case 3 with an increase of lift coefficient of 90% and 93% respectively, from the baseline.

Looking at these results, one can conclude that the proposed Hybrid Method H0-LG performs much more robustly than the conjugate gradient method and much faster than the genetic algorithms in this test problem.

Flow results and AFC performance

This subsection provides a comparison between the characteristics of the flow field corresponding to each of the optimal solutions labeled in Figure 4.59. The lift coefficient and design variables of each of the optimal solutions are presented in Table 4.16. For a full explanation of the flow structure and details on the synthetic jet actuator performance, the reader should have a look at [79].

Table 4.16: Values of the lift coefficient and design variables of the 5 labeled cases.

Case	C_l	F^+	C_μ	θ^o	x/C	h/C
1	0.79	5.9	1.74×10^{-02}	120.0	1.00×10^{-03}	1.0×10^{-02}
2	1.39	0.3	1.47×10^{-02}	24.0	1.00×10^{-03}	1.4×10^{-02}
3	1.49	3.2	1.92×10^{-02}	8.0	2.00×10^{-02}	5.0×10^{-03}
4	1.52	6.7	2.00×10^{-02}	7.0	2.60×10^{-02}	5.0×10^{-03}
5	1.54	9.9	2.00×10^{-02}	5.0	1.97×10^{-02}	5.0×10^{-03}

As explained in the Section 4.4.1, the flow without the synthetic jet actuator is fully detached, see Figure 4.58. The objective of the synthetic jet actuator is to prevent or minimize flow separation. The averaged streamlines and pressure field of the optimized cases are presented and discussed in this section. The flow field corresponding to Case 1 is presented in Figure 4.60. Despite the fact the flow separation is slightly delayed versus the baseline case, a large vortical structure is still observed over the airfoil. In fact, the lift coefficient is worst for this particular case than the one obtained in the baseline case.

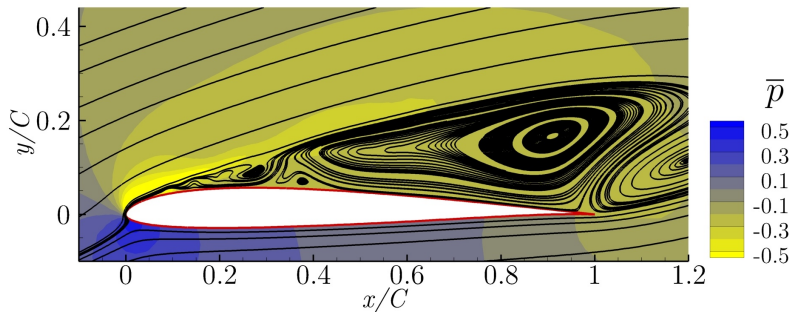


Figure 4.60: Averaged streamlines and pressure field of the Case 1 ($C_l = 0.79$).

In Figure 4.61 the averaged streamlines and pressure field obtained from case 2

is presented. It shows a late reattachment of the flow, which improves the lift coefficient of the baseline by 74%. This solution has been obtained by one of the genetic algorithm's runs with around 400 evaluations of the objective function. The resulting lift coefficient is $C_l = 1.39$.

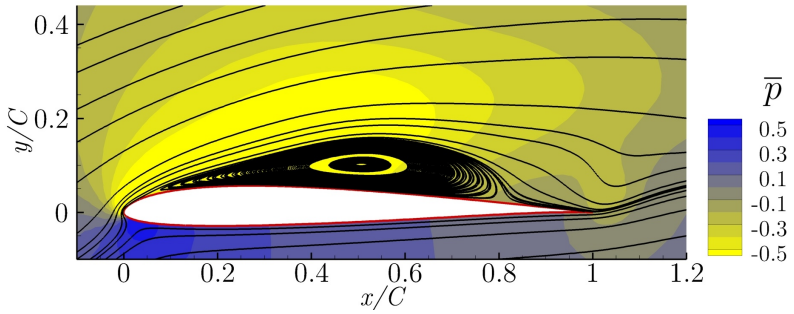


Figure 4.61: Averaged streamlines and pressure field of the Case 2 ($C_l = 1.39$).

The averaged streamlines and pressure fields of cases 3, 4 and 5 are presented in Figures 4.62, 4.63 and 4.64, respectively. All of them show a complete flow reattachment, with very minor differences in the size of the laminar bubble appearing close to the airfoil leading edge. The bubbles are in fact located near the synthetic jet position, just downstream of it, as can be seen in the above mentioned figures. The optimization of the flow control actuation parameters has managed to successfully reattached the flow along the entire airfoil chord.

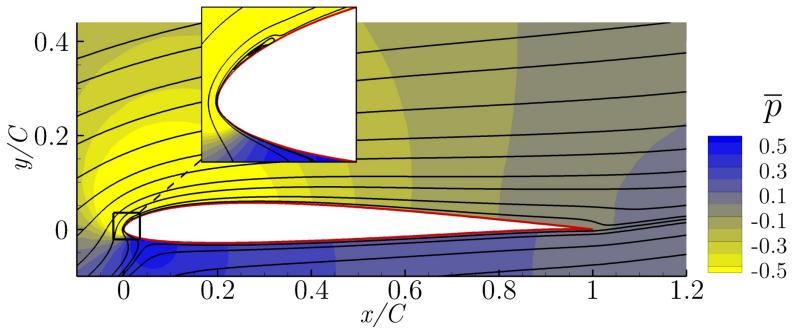


Figure 4.62: Averaged streamlines and pressure field of the Case 3 ($C_l = 1.49$).

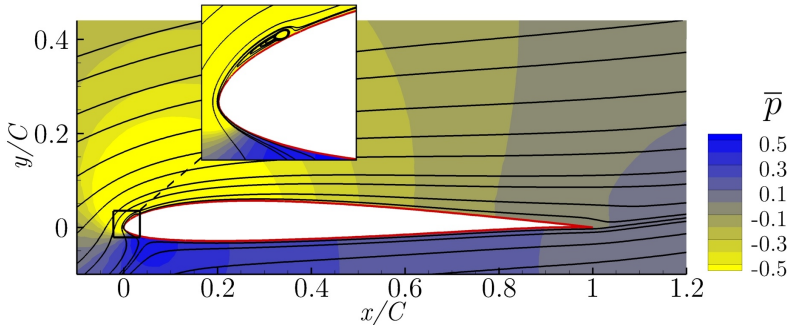


Figure 4.63: Averaged streamlines and pressure field of the Case 4 ($C_l = 1.52$).

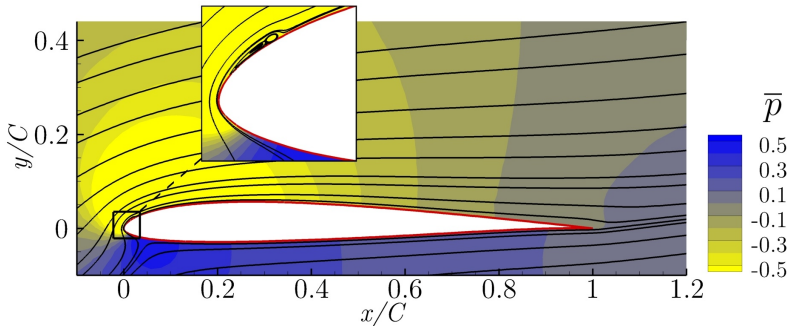


Figure 4.64: Averaged streamlines and pressure field of the Case 5 ($C_l = 1.54$).

4.4.3 Two objective functions results

This section introduces the results obtained for the two objectives AFC test case. The objective of the design problem is to maximize the lift coefficient C_l and the airfoil efficiency η which is defined as the lift to drag ratio

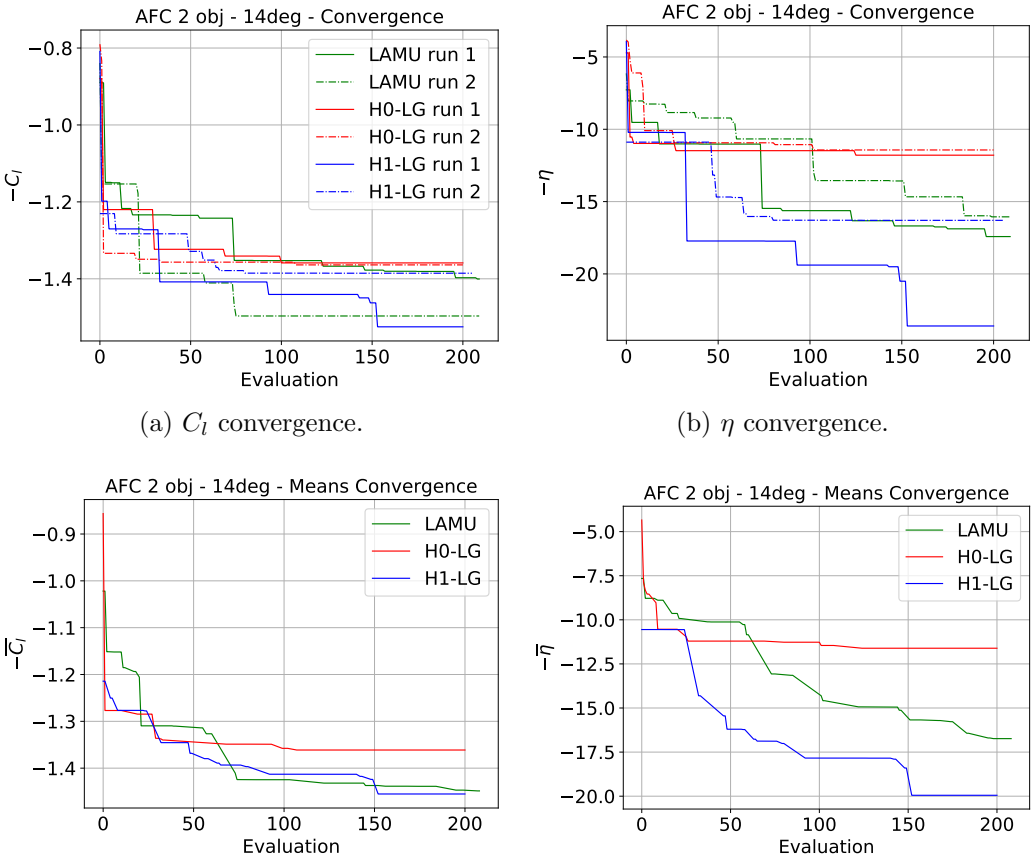
$$\eta = \frac{C_l}{C_d} \tag{4.23}$$

To do so the objective functions of the minimization problem are set to

$$\text{Minimize} : \begin{cases} f_1 = -C_l \\ f_2 = -\eta \end{cases} \tag{4.24}$$

This case has been run with the LAMU method and the two Hybrid Methods based on it (i.e. H0-LG and H1-LG). The rest of the algorithms have not been tested with this problem due to the high computational resources that it requires. Each of the three methods have been run two times with different seeds. All runs performed around 200 evaluations.

Figure 4.65 shows the convergence of each objective function and the mean values. In this test problem the mean values should not be taken with too thoroughness as they are calculated with only 2 samples. The method that performed worst in the extremes is the H0-LG specially in the airfoil efficiency η objective function. In the C_l objective function the LAMU and H1-LG methods performed similarly as both methods have achieved similar results. It appears to be a local minimum at around $C_l = 1.4$ which all methods found in one run. In the other run, the LAMU method achieved a $C_l = 1.496$ and the H1-LG method achieved a $C_l = 1.524$.



(a) C_l convergence.

(b) η convergence.

(c) $\overline{C_l}$ convergence.

(d) $\overline{\eta}$ convergence.

Figure 4.65: Convergence of the Active Flow Control with 2 objective functions. (b) has the same legend as (a).

Figure 4.66 show all the individuals from all methods and runs in the objective functions space. The H0-LG method is the method with the worst performance in this case. It did not got close to the extremes of the Pareto front as showed above and in the objective functions space did not perform well either as all individuals are far from the optimal ones found by the other methods. The Genetic algorithm is the method that more space has explored, see the top left region of Figure 4.66 which only has been explored by the LAMU method. The bottom left region of the objective functions space has been explored by the LAMU and the H1-LG methods, although the Hybrid Method achieved better solutions in that area.

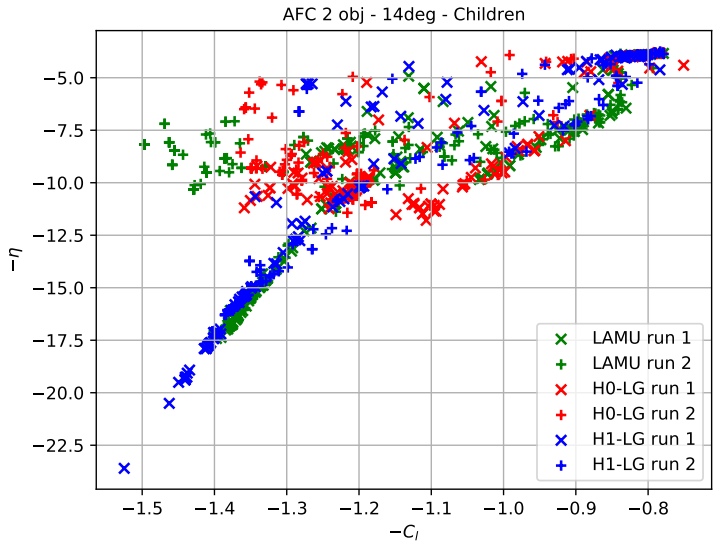
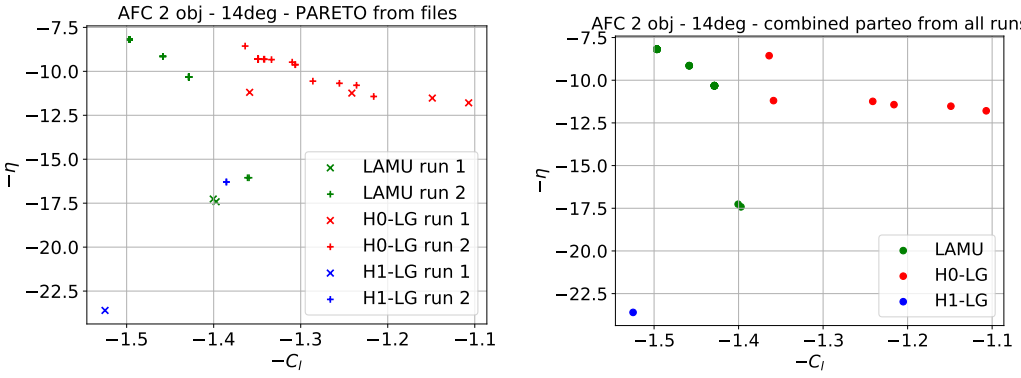


Figure 4.66: Individuals of all tests in the objective function space.

The non-dominated solutions of each run are plotted in Figure 4.67a. It is clear that the H0-LG is the algorithm that performed worst as it did not found the regions where the best individuals are and most of its individuals are dominated by the ones of the other methods. The H1-LG found a solution that dominates all other solutions, with $C_l = 1.254$ and $\eta = 23.6$, its design variables are presented at Table 4.17. Figure 4.67b shows the non-dominated solutions of the joint populations for the 2 runs of each method.



(a) (b)

Figure 4.67: Non-dominated solutions. (a) Overlapped Pareto fronts from all runs. (b) Non-dominated solutions of the combined Pareto fronts of all runs for each optimization method.

The Pareto front obtained by the H1-LG method in this optimization does not have the shape that is expected, as a single point dominates the rest of solutions and the non-dominated solution is composed by a single individual. The Genetic

Algorithm did find a more conventional Pareto front containing multiple points with a trade off between the maximum lift coefficient and the maximum airfoil efficiency. In the work of Tousi et al. [79] a Pareto front with multiple non-dominated solutions is obtained. The main differences between the configuration of the test case and the one in the article are that the mesh in this work is coarser to decrease the computational cost as it has been run with three different algorithms, and the optimization of the article performed 400 objective functions evaluations in front of the 200 evaluations that performed the algorithms in this study. Another difference is the range of the x/c, although the non-dominated solutions have similar values in both studies ($x/c \approx 2 \times 10^{-2}$). This will be studied further to determine which of this differences cause the differences in the solutions. Even if there are differences with the study from Tousi et al. [79] the comparison in this study is still valid as all optimization algorithms have used the exact same mesh and configuration which makes the comparison between algorithms fair.

Table 4.17: Values of the lift coefficient, airfoil efficiency and design variables of the overall best individual found by the H1-LG Hybrid Method.

C_l	η	F^+	C_μ	θ^o	x/C	h/C
1.524	23.6	9.3	2×10^{-02}	8.34	2.23×10^{-01}	7.19×10^{-02}

The method that has performed better is the H1-LG as it has achieved the most optimal solution in one of the two runs, and has also performed the best in the average of the two runs.

4.4.4 Summary

The single objective AFC problem has been solved by the LAMU, H0-LG, H1-LG, PSO and Conjugate gradient. The rest of methods have not been tested in this case due to the high computational resources that it requires. The H0-LG has solved the the single objective AFC problem very efficiently in comparison with the rest of the methods. It has achieves the highest rate of convergence, the highest robustness and the combines successfully the best characteristics of both optimization algorithms. It outperformed the rest of the algorithms in convergence rate, robustness and accuracy. The H1-LG has not performed so well in this case, actually worst it performed worst than the GA. The increase of robustness is provided by the genetic algorithm player which avoid local minimums.

The two objective AFC problem has been solved by the LAMU, H0-LG and H1-LG. In the two objective AFC problem, the method that has performed better is the H1-LG which has obtained a single solution that dominates all the solutions obtained by the rest of the methods. The H0-LG has performed worst than the pure Genetic Algorithm.

The strategy to exchange information between the two optimization algorithms proposed in this Hybrid Method has proven efficient to overcome the main drawbacks

of both classical optimization methods and the result is an optimization method capable of exploring and exploiting the full search space.

Chapter 5

Final remarks

In this chapter, the conclusions of the work are discussed, the future work is presented and the publications related to this work are listed.

5.1 Conclusions

Industrial optimization problems are usually high computational demanding and can be challenging to obtain satisfactory results in a reasonable time span. In high demanding problems the efficiency of the optimization algorithms becomes one of the main requirements to select one optimization method. The behavior of the optimization algorithms are highly dependent on the test problem and in some industrial problems there is little to no information about the characteristics of the objective functions of the problem. Evolutionary algorithms usually works well in the scenarios where there is not much information about the objective functions, but they usually have slowness problems to converge to the exact minimum.

The expected behavior of Evolutionary Algorithms and the Conjugate Gradient has been confirmed throughout the single objective mathematical benchmarks. The Conjugate Gradient performed with a lack of robustness in all cases, and only some runs that started in the global valley converged to the global minimum. The Evolutionary Algorithms have also have shown the expected behavior, finding good solutions in most cases and runs, although not always with high accuracy.

Two metrics have been used to measure the performance of the optimization algorithms in the two objective mathematical benchmarks. The distance metric, that calculates a mean Euclidean distance of the non-dominated solutions to a sample of the exact solutions, and the spread metric, which measures how uniformly distributed are the non-dominated solutions across the Pareto front. The distance metric has proven to give a very valuable information, as the comparison between methods using this method is intuitive and there is a strong correlation with the graphical interpretation of the non-dominated solutions plot in the objective space. The spread metric has proven to not be very reliable for performance comparisons, as may penalize methods with higher density of points on some regions, although

the overall density is higher. In this case it is possible to increase its spread metric dropping some solutions from the higher density regions, and if there are enough non-dominated solutions the rest of the performance evaluation may not be affected. For this reason, the conclusion is that the spread metric gives a good measure of the uniformity of the non-dominated solutions, but a higher value does not mean that one method is worst than another.

Four different version of the Hybrid Method have been implemented and tested successfully in both mathematical benchmarks and engineering applications. The proposed Hybrid Methods have proven very efficient and robust in most cases, although there is not a single version of the Hybrid Method, nor a pure Evolutionary Algorithm, that performed the best across all tests. The results of each optimization method are highly dependent on the problem.

In the two objective functions problems the proposed Hybrid Methods have consistently helped to improve the convergence of the extremes of the Pareto front. The convergence of the rest of the Pareto front regions has not been improved consistently through all tests. The Hybrid Methods have solved the ZDT family problems with much more efficiency than the pure Evolutionary Algorithms tested. In the KUR, FON and POL problems the results in the whole Pareto front where worst than the pure Evolutionary Algorithms.

The AFC problem which is a CFD problem with 5 design variables has been solved successfully with the proposed Hybrid Methods. The best results of the single objective problem have been achieved with the H0-LG algorithm over performing the EA algorithms in both runs showing a high robustness and rate of convergence. It has showed a great convergence rate, similar to a gradient based method, but without the lack of robustness that usually comes with gradient based methods applied to complex applications without good initial solutions. The increase in robustness is provided by the Genetic Algorithm player which avoids local minimums. The strategy to exchange information between the two optimization algorithms proposed in this Hybrid Method has proven efficient to overcome the main drawbacks of both classical optimization methods, and the result is an optimization method capable of exploring and exploiting the full search space. The best results of the two objective problem have been achieved with the H1-LG algorithm which has found a single solution that dominates all the solutions proposed by the other methods. The Hybrid Methods successfully combines the best characteristics of both optimization algorithms, although the performance is dependent on the problem.

The new implementation of the CIMNE's in-house optimization tool RMOPv2 is a result of this thesis. The tool will be used as a research platform in hybridization to further test the developed algorithms and to develop new ones. It also will be used and further tested as a production tool in the industrial and research projects to come.

5.2 Future work

The future work aims at deepen the comprehension of all the generated results and understand the difference in performance of the Hybrid Methods between the different test cases to modify the algorithms and search for more universal methods that behaves better in a wider sample of benchmarks.

One specific point of research is already identified in the KUR, FON and POL benchmarks, where the Hybrid Methods wasted many evaluations in the extremes of the Pareto front without achieving further improvements. A strategy to detect the convergence of the extremes solutions may help to improve the performance in those cases. A possible improvement is to stop the Conjugate Gradient when this is detected and allocate all resources to the Evolutionary Algorithm player. Another possible approach is to change the the objective function of the Conjugate Gradient to exploit other regions of the Pareto front, for example using a weighted sum of the two objectives as the Conjugate Gradient objective. The weights could be changed during the optimization process, for example when the convergence for a particular weights is detected.

Another aspect to improve is the restart capability of the methods. It has not been discussed before, because it does not directly affect the optimization algorithm, but when dealing with optimizations that takes months to compute, as the Active Flow Control problem, power outages become a real problem. A restart routine had to be implemented during the evaluation of the AFC tests. A more thought out strategy should be developed to allow a restart of the Hybrid Methods with less loss of information as it has become a problem during the development of the work.

The evaluation of the performance comparison could be improved including the Hypervolume indicator [80, 81] and to include statistical significance tests, for example see . In single objective problems, for example see [82] for reference. In the single objective problems, use the best value of each independent run and for the multi-objective problems use the Hypervolume indicator as the metrics for the significance tests.

The Hybrid Methods based on the Genetic Algorithm could be modified to use a single objective Genetic Algorithm, instead of the LAMU method which is based on the NSGAII, to solve the single objective problems.

The mutation probability used in this study has been fixed to $P_m = 0.1$. The influence of this value should be tested against the usually recommended value of $P_m = 1/(\text{number of design variables})$.

Finally, the algorithm could be tested in optimization problems with restrictions. In that case, the Conjugate Gradient player will have to be studied carefully as the current algorithm is not capable to handle restrictions, and it may be necessary to make some modification or to test other local search strategies.

5.3 Related publications

During the development of the thesis the following the author has contributed to the following publication and conferences.

5.3.1 Journal papers

1. Martí Coma, Navid Monshi Tousi, Jordi Pons-Prats, Gabriel Bugeda and Josep M. Bergada. "A New Hybrid Optimization Method, Application to a Single Objective Active Flow Control Test Case". In: *Applied Sciences* 12.8 (2022). ISSN: 2076-3417. DOI: 10.3390/app12083894. URL: https://www.mdpi.com/2076-3417/12/8/3894

2. N.M. Tousi, M. Coma, J.M. Bergadà, J. Pons-Prats, F. Mellibovsky and G. Bugeda. "Active flow control optimisation on SD7003 airfoil at pre and post-stall angles of attack using synthetic jets". In: *Applied Mathematical Modelling* 98 (2021), pp. 435–464. ISSN: 0307-904X. DOI: https://doi.org/10.1016/j.apm.2021.05.016. URL: https://www.sciencedirect.com/science/article/pii/S0307904X21002614

3. Nerea Mangado, Jordi Pons-Prats, Martí Coma, Pavel Mistrík, Gemma Piella, Mario Ceresa and Miguel A Gonzalez Ballester. "Computational evaluation of cochlear implant surgery outcomes accounting for uncertainty and parameter variability". In: *Frontiers in physiology* 9 (2018), p. 498

5.3.2 Book Chapters

1. Jordi Pons-Prats, Martí Coma, Jaume Betran, Xavier Roca and Gabriel Bugeda. "Industrial application of genetic algorithms to cost reduction of a wind turbine equipped with a tuned mass damper". In: *Evolutionary and Deterministic Methods for Design Optimization and Control With Applications to Industrial and Societal Problems*. Springer, 2019, pp. 419–436

5.3.3 Contribution to conferences

A list of publications made by the author in the optimization field is presented below.

During the PhD candidature:

1. Martí Coma, Navid Monshi Tousi, Jordi Pons-Prats, Josep M. Bergada and Gabriel Bugeda. "Performance of hybrid optimization methods applied to active flow control devices". In: *AeroBest* (July 2021). Lisbon, Online.

2. J. Pons-Prats, M. Coma and G. Bugeda. "Optimization hybridization with multiple populations and optimization methods". In: *Eurogen 2019* (Sept. 2019). Guimares, Portugal.

3. M. Coma, J. Pons-Prats and G. Bugeda. "Hybrid optimization methods". In: *Congreso de Métodos Numéricos en Ingeniería (CMN) 2019* (July 2019). Guimares, Portugal.

4. M. Coma, J. Pons-Prats and G. Bugeda. "Programming strategies for high performance Genetic Algorithms for demanding applications, Nash and Hybrid games revisited". In: *Eurogen 2017* (Sept. 2017). Madrid.

5. J. Pons-Prats, M. Coma, J. Betran, X. Roca and G. Bugeda. "Industrial Application of Genetic Algorithms to cost reduction of a Wind Turbine equipped with a Tuned Mass Damper". In: *Eurogen 2017* (Sept. 2017). Madrid.

Previously to start the PhD candidature:

6. Dong-Seop Lee, M. Coma, H. Espinoza, O. Fruitós and J. Pons-Prats. "Multi-objective design optimisation of a 3D-rail stamping process using a robust multi-objective optimisation platform (RMOP)". in: *International Conference on Computational Plasticity XII. Fundamentals and Applications* (2013), pp. 1–12

Bibliography

[1] David G Luenberger, Yinyu Ye et al. *Linear and nonlinear programming*. Vol. 2. Springer, 1984.

[2] Mokhtar S Bazaraa, Hanif D Sherali and Chitharanjan M Shetty. *Nonlinear programming: theory and algorithms*. John Wiley & Sons, 2013.

[3] David M Himmelblau. "Applied Nonlinear Programming McGraw-Hill Book Company". In: *New York* (1972).

[4] S. L. S. Jacoby, Janusz S. Kowalik and J. T. Pizzo. "Iterative methods for nonlinear optimization problems". In: 1972.

[5] Zlobec S and Petric J. *Nonlinear programming*. Naučna Knjiga, Beograd, 1989 (in serbian).

[6] Joaquim RRA Martins and Andrew B Lambe. "Multidisciplinary design optimization: a survey of architectures". In: *AIAA journal* 51.9 (2013), pp. 2049–2075.

[7] Pradeep Kumar Gudla and Ranjan Ganguli. "An automated hybrid genetic-conjugate gradient algorithm for multimodal optimization problems". In: *Applied Mathematics and Computation* 167.2 (2005), pp. 1457–1474.

[8] Fred Glover. "Future paths for integer programming and links to artificial intelligence". In: *Computers & operations research* 13.5 (1986), pp. 533–549.

[9] Fred Glover. "Tabu search—part I". In: *ORSA Journal on computing* 1.3 (1989), pp. 190–206.

[10] Fred Glover. "Tabu search—part II". In: *ORSA Journal on computing* 2.1 (1990), pp. 4–32.

[11] TH Cormen et al. *Greedy Algorithms, Chapter 16*. 2001.

[12] Martin Pincus. "A Monte Carlo method for the approximate solution of certain types of constrained optimization problems". In: *Operations research* 18.6 (1970), pp. 1225–1228.

[13] A Khachaturyan, S Semenovsovskaya and B Vainshtein. "The thermodynamic approach to the structure analysis of crystals". In: *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography* 37.5 (1981), pp. 742–754.

[14] Rui Chibante. *Simulated annealing: theory with applications*. BoD–Books on Demand, 2010.

[15] Scott Kirkpatrick, C Daniel Gelatt Jr and Mario P Vecchi. "Optimization by simulated annealing". In: *science* 220.4598 (1983), pp. 671–680.

[16] Kalyanmoy Deb. "Multi-objective optimisation using evolutionary algorithms: an introduction". In: *Multi-objective evolutionary optimisation for product design and manufacturing*. Springer, 2011, pp. 3–34.

[17] Kalyanmoy Deb and Ram Bhushan Agrawal. "Simulated binary crossover for continuous search space". In: *Complex systems* 9.2 (1995), pp. 115–148.

[18] Kalyanmoy Deb et al. "A fast and elitist multiobjective genetic algorithm: NSGA-II". In: *IEEE transactions on evolutionary computation* 6.2 (Apr. 2002), pp. 182–197. ISSN: 1089-778X. DOI: 10.1109/4235.996017.

[19] Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms*. Vol. 16. Chichester, UK: John Wiley & Sons, 2001.

[20] Paolo Rocca, Giacomo Oliveri and Andrea Massa. "Differential evolution as applied to electromagnetics". In: *IEEE Antennas and Propagation Magazine* 53.1 (2011), pp. 38–49.

[21] Giacomo Oliveri, Paolo Rocca and Andrea Massa. "Differential evolution as applied to electromagnetics: Advances, comparisons, and applications". In: *2012 6th European Conference on Antennas and Propagation (EUCAP)*. IEEE. 2012, pp. 3058–3059.

[22] Jason Brownlee et al. "Clonal selection algorithms". In: *Complex Intelligent Systems Laboratory, Swinburne University of Technology, Australia* (2007).

[23] Leandro N De Castro and Fernando J Von Zuben. "Learning and optimization using the clonal selection principle". In: *IEEE transactions on evolutionary computation* 6.3 (2002), pp. 239–251.

[24] Ignacio Segovia Domínguez, Arturo Hernández Aguirre and S Ivvan Valdez. "A new EDA by a gradient-driven density". In: (2014), pp. 352–361.

[25] James Kennedy and Russell Eberhart. "Particle swarm optimization". In: *Proceedings of ICNN'95-international conference on neural networks*. Vol. 4. IEEE. 1995, pp. 1942–1948.

[26] A Colorni, M Dorigo and V Maniezzo. "Distributed Optimization by Ant Colonies, actes de la première conférence européenne sur la vie artificielle". In: *Elsevier Publishing*. 1991, pp. 134–142.

[27] Marco Dorigo. "Optimization, learning and natural algorithms". In: *Ph. D. Thesis, Politecnico di Milano* (1992).

[28] Duc Truong Pham et al. "The bees algorithm—a novel tool for complex optimisation problems". In: *Intelligent production machines and systems*. Elsevier, 2006, pp. 454–459.

[29] Jonathan Richard Shewchuk et al. *An introduction to the conjugate gradient method without the agonizing pain*. 1994.

[30] Reeves Fletcher and Colin M Reeves. "Function minimization by conjugate gradients". In: *The computer journal* 7.2 (1964), pp. 149–154.

[31] Roger Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2013.

[32] Elijah Polak and Gerard Ribiere. "Note sur la convergence de méthodes de directions conjuguées". In: *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique* 3.R1 (1969), pp. 35–43.

[33] Jack Kiefer. "Sequential minimax search for a maximum". In: *Proceedings of the American mathematical society* 4.3 (1953), pp. 502–506.

[34] Richard P Brent. *Algorithms for minimization without derivatives*. Courier Corporation, 2013.

[35] Russell Eberhart and James 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.

[36] Russell Eberhart and James Kennedy. "Particle swarm optimization". In: *Proceedings of the IEEE international conference on neural networks*. Vol. 4. Citeseer. 1995, pp. 1942–1948.

[37] Russ Eberhart, Pat Simpson and Roy Dobbins. *Computational intelligence PC tools*. Academic Press Professional, Inc., 1996.

[38] Yuhui Shi and Russell C Eberhart. "Empirical study of particle swarm optimization". In: *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*. Vol. 3. IEEE. 1999, pp. 1945–1950.

[39] Konstantinos E Parsopoulos and Michael N Vrahatis. "Particle swarm optimization method in multiobjective problems". In: *Proceedings of the 2002 ACM symposium on Applied computing*. 2002, pp. 603–607.

[40] CA Coello Coello and Maximino Salazar Lechuga. "MOPSO: A proposal for multiple objective particle swarm optimization". In: *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*. Vol. 2. IEEE. 2002, pp. 1051–1056.

[41] Runwei Cheng, Mitsuo Gen and Yasuhiro Tsujimura. "A tutorial survey of job-shop scheduling problems using genetic algorithms, part II: hybrid genetic search strategies". In: *Computers & Industrial Engineering* 36.2 (1999), pp. 343–364.

[42] Yuping Wang et al. "A new hybrid genetic algorithm for job shop scheduling problem". In: *Computers & Operations Research* 39.10 (2012), pp. 2291–2299.

[43] Rupert Weare, Edmund Burke and Dave Elliman. "A hybrid genetic algorithm for highly constrained timetabling problems". In: *Department of Computer Science* (1995).

[44] Vicente Valls, Francisco Ballestin and Sacramento Quintanilla. "A hybrid genetic algorithm for the resource-constrained project scheduling problem". In: *European Journal of Operational Research* 185.2 (2008), pp. 495–508.

[45] William Ho et al. "A hybrid genetic algorithm for the multi-depot vehicle routing problem". In: *Engineering Applications of Artificial Intelligence* 21.4 (2008), pp. 548–557.

[46] N. Shahidi et al. "Self-adaptive memetic algorithm: an adaptive conjugate gradient approach". In: *IEEE Conference on Cybernetics and Intelligent Systems, 2004.* Vol. 1. 2004, 6–11 vol.1. DOI: 10.1109/ICCIS.2004.1460378.

[47] Diego Fernando Páez Bautista. "Study of hybrid strategies for multi-objective optimization using gradient based methods and evolutionary algorithms". PhD thesis. The University of North Carolina at Charlotte, 2013.

[48] James D Kelly Jr and Lawrence Davis. "A Hybrid Genetic Algorithm for Classification." In: *IJCAI.* Vol. 91. 1991, pp. 645–650.

[49] Wan-Rong Jih and J Yung-Jen Hsu. "Dynamic vehicle routing using hybrid genetic algorithms". In: *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C).* Vol. 1. IEEE. 1999, pp. 453–458.

[50] Tarek A El-Mihoub et al. "Hybrid Genetic Algorithms: A Review." In: *Engineering Letters* 13.2 (2006), pp. 124–137.

[51] Matthias Kulcke and Wolfgang E. Lorenz. "Utilizing Gradient Analysis within Interactive Genetic Algorithms". In: *eCAAD.* Vol. 2. 2016, pp. 359–364.

[52] Ahmad B. A. Hassanat et al. "Enhancing Genetic Algorithms using Multi Mutations: Experimental Results on the Travelling Salesman Problem". In: *International Journal of Computer Science and Information Security (IJCSIS)* 7.1 (2016), p. 785.

[53] Wang Pan et al. "Study on a novel hybrid adaptive genetic algorithm embedding conjugate gradient algorithm". In: *Proceedings of the 3rd World Congress on Intelligent Control and Automation (Cat. No.00EX393).* Vol. 1. 2000, 630–633 vol.1. DOI: 10.1109/WCICA.2000.860048.

[54] Jean Berger and Mohamed Barkaoui. "A parallel hybrid genetic algorithm for the vehicle routing problem with time windows". In: *Computers & Operations Research* 31.10 (2004), pp. 2037–2053.

[55] DongSeop Lee et al. "Double Shock Control Bump design optimization using hybridised evolutionary algorithms". In: *Evolutionary Computation (CEC), 2010 IEEE Congress on.* IEEE. 2010, pp. 1–8.

[56] Danilo Vasconcellos Vargas et al. "General subpopulation framework and taming the conflict inside populations". In: *Evolutionary computation* 23.1 (2015), pp. 1–36.

[57] Ferrante Neri, Carlos Cotta and Pablo Moscato. *Handbook of memetic algorithms.* Vol. 379. Springer, 2011.

[58] Erick Cantu-Paz. *Efficient and accurate parallel genetic algorithms.* Vol. 1. Springer Science & Business Media, 2000.

[59] Jacques Periaux, Felipe Gonzalez and Dong Seop Chris Lee. *Evolutionary optimization and game strategies for advanced multi-disciplinary design: applications to aeronautics and UAV design.* Vol. 75. Springer, 2015.

[60] El-Ghazali Talbi. "A unified view of parallel multi-objective evolutionary algorithms". In: *Journal of Parallel and Distributed Computing* 133 (2019), pp. 349–358.

[61] DongSeop Lee et al. "Hybrid-Game Strategies for multi-objective design optimization in engineering". In: *Computers & Fluids* 47.1 (2011), pp. 189–204.

[62] Jacques Periaux et al. "Fast reconstruction of aerodynamic shapes using evolutionary algorithms and virtual Nash strategies in a CFD design environment". In: *Journal of computational and applied mathematics* 232.1 (2009), pp. 61–71.

[63] Dong-Seop Lee et al. "Hybrid-Game Strategies for multi-objective design optimization in engineering". In: *Computers and Fluids* 47.1 (2011), pp. 189–204. ISSN: 0045-7930. DOI: http://dx.doi.org/10.1016/j.compfluid.2011.03.007. URL: http://www.sciencedirect.com/science/article/pii/S0045793011000934.

[64] Dong-Seop Lee et al. "Double-shock control bump design optimization using hybridized evolutionary algorithms". In: *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* 225.10 (2011), pp. 1175–1192. DOI: 10.1177/0954410011406210. eprint: http://dx.doi.org/10.1177/0954410011406210. URL: http://dx.doi.org/10.1177/0954410011406210.

[65] Maurice Clerc. *Particle swarm optimization*. Vol. 93. John Wiley & Sons, 2010.

[66] Douglas Thain, Todd Tannenbaum and Miron Livny. "Distributed computing in practice: the Condor experience". In: *Concurrency and computation: practice and experience* 17.2-4 (2005), pp. 323–356.

[67] K. Deb et al. "A fast and elitist multiobjective genetic algorithm: NSGA-II". In: *Evolutionary Computation, IEEE Transactions on* 6.2 (Apr. 2002), pp. 182–197. ISSN: 1089-778X. DOI: 10.1109/4235.996017.

[68] Eckart Zitzler. *Evolutionary algorithms for multiobjective optimization: Methods and applications*. Vol. 63. Citeseer, 1999.

[69] Carlos M Fonseca and Peter J Fleming. "On the performance assessment and comparison of stochastic multiobjective optimizers". In: *International Conference on Parallel Problem Solving from Nature*. Springer. 1996, pp. 584–593.

[70] David Ackley. *A connectionist machine for genetic hillclimbing*. Vol. 28. Springer Science & Business Media, 2012.

[71] Manuel Laguna and Rafael Martí. "Experimental testing of advanced scatter search designs for global optimization of multimodal functions". In: *Journal of Global Optimization* 33.2 (2005), pp. 235–255.

[72] Sudhanshu K Mishra. "Some new test functions for global optimization and performance of repulsive particle swarm method". In: *Available at SSRN 926132* (2006).

[73] Marcin Molga and Czesław Smutnicki. "Test functions for optimization needs". In: *Test functions for optimization needs* 101 (2005).

[74] HoHo Rosenbrock. "An automatic method for finding the greatest or least value of a function". In: *The Computer Journal* 3.3 (1960), pp. 175–184.

[75] Eckart Zitzler, Kalyanmoy Deb and Lothar Thiele. "Comparison of multiobjective evolutionary algorithms: Empirical results". In: *Evolutionary computation* 8.2 (2000), pp. 173–195.

[76] Carlos M Fonseca and Peter J Fleming. "An overview of evolutionary algorithms in multiobjective optimization". In: *Evolutionary computation* 3.1 (1995), pp. 1–16.

[77] Frank Kursawe. "A variant of evolution strategies for vector optimization". In: *International Conference on Parallel Problem Solving from Nature*. Springer. 1990, pp. 193–197.

[78] Carlo Poloni et al. "Hybridization of a multi-objective genetic algorithm, a neural network and a classical optimizer for a complex design problem in fluid dynamics". In: *Computer Methods in Applied Mechanics and Engineering* 186.2-4 (2000), pp. 403–420.

[79] N.M. Tousi et al. "Active flow control optimisation on SD7003 airfoil at pre and post-stall angles of attack using synthetic jets". In: *Applied Mathematical Modelling* 98 (2021), pp. 435–464. ISSN: 0307-904X. DOI: https://doi.org/10.1016/j.apm.2021.05.016. URL: https://www.sciencedirect.com/science/article/pii/S0307904X21002614.

[80] Zitzler Eckart and Thiele Lothar. "Multiobjective evolutionary algorithms a comparative case study and the strength pareto approach". In: *IEEE transactions on evolutionary computation* 3.4 (1999), pp. 257–271.

[81] Michael Emmerich and André H Deutz. "A tutorial on multiobjective optimization: fundamentals and evolutionary methods". In: *Natural computing* 17.3 (2018), pp. 585–609.

[82] Salvador Garcia and Francisco Herrera. "An Extension on" Statistical Comparisons of Classifiers over Multiple Data Sets" for all Pairwise Comparisons." In: *Journal of machine learning research* 9.12 (2008).

[83] Martí Coma et al. "A New Hybrid Optimization Method, Application to a Single Objective Active Flow Control Test Case". In: *Applied Sciences* 12.8 (2022). ISSN: 2076-3417. DOI: 10.3390/app12083894. URL: https://www.mdpi.com/2076-3417/12/8/3894.

[84] Nerea Mangado et al. "Computational evaluation of cochlear implant surgery outcomes accounting for uncertainty and parameter variability". In: *Frontiers in physiology* 9 (2018), p. 498.

[85] Jordi Pons-Prats et al. "Industrial application of genetic algorithms to cost reduction of a wind turbine equipped with a tuned mass damper". In: *Evolutionary and Deterministic Methods for Design Optimization and Control With Applications to Industrial and Societal Problems*. Springer, 2019, pp. 419–436.

[86] Martí Coma et al. "Performance of hybrid optimization methods applied to active flow control devices". In: *AeroBest* (July 2021). Lisbon, Online.

[87] J. Pons-Prats, M. Coma and G. Bugeda. "Optimization hybridization with multiple populations and optimization methods". In: *Eurogen 2019* (Sept. 2019). Guimares, Portugal.

[88] M. Coma, J. Pons-Prats and G. Bugeda. "Hybrid optimization methods". In: *Congreso de Métodos Numéricos en Ingeniería (CMN) 2019* (July 2019). Guimares, Portugal.

[89] M. Coma, J. Pons-Prats and G. Bugeda. "Programming strategies for high performance Genetic Algorithms for demanding applications, Nash and Hybrid games revisited". In: *Eurogen 2017* (Sept. 2017). Madrid.

[90] J. Pons-Prats et al. "Industrial Application of Genetic Algorithms to cost reduction of a Wind Turbine equipped with a Tuned Mass Damper". In: *Eurogen 2017* (Sept. 2017). Madrid.

[91] Dong-Seop Lee et al. "Multi-objective design optimisation of a 3D-rail stamping process using a robust multi-objective optimisation platform (RMOP)". In: *International Conference on Computational Plasticity XII. Fundamentals and Applications* (2013), pp. 1–12.