
Politecnico di Torino

**Politecnico di Torino**

Master's Degree in Aerospace Engineering
Academic Year 2021/2022
Graduation Session March-April 2022

# Development of Hybrid Evolutionary Algorithms Tool Applied to Lunar Landing Mission -ESA EL3- Study Phase

**Supervisor**
Prof. Paolo Maggiore

**Candidate**
Mattia Topini

**Co-Supervisors**
Eng. Francesco Simeoni
Eng. Massimiliano Saponara

# Acknowledgements

Throughout this master's thesis work I have received a great deal of support and assistance. It has been an incredibly enriching experience.

I would first like to thank my supervisor, Professor Paolo Maggiore, that gave me the opportunity to realize this thesis in a major space industry company.

I would like to acknowledge my supervisors at Thales Alenia Space. Eng. Francesco Simeoni, you followed me throughout all this work. It was a wonderful collaboration and and I hope it will continue in the future. Your insightful feedback pushed me to sharpen my thinking and brought my work to a higher level. A big thank also to Eng. Ferdinando Cometto, who followed me with regard to the EL3 application case of this thesis, giving me all the data and feedback needed. And thanks to Eng. Massimiliano Saponara, head of the GNC/AOCS group, who welcomed me and allowed me to work daily within the Euclid office.

In addition, I would like to thank my parents, you are always there for me, grazie mamma. A thought for my brother Luca who has proven himself able to support me and bear me for all these years. Finally, I could not have completed this thesis without the support of relatives and friends. I'm so lucky I can't quote them all, otherwise I would fill in another page. C'è il gruppo di sempre, le Zollette®. Stani, Simo, Gabri and other friends met at the university who rejoiced these years. And all the friendships never ended since high school. The friendships born more recently and those that were born during my experience in Erasmus. En particular un saludo y un abrazo a Nacho, Viki, Marcos, Belu y Flor, que me acompañaron durante mi Erasmus en Argentina.

I strongly believe, and I always will, in the ability to seize each and every opportunity, even the unexpected ones. And each brings others. I don't know if I have taken all the one that arose or if I have been able to fully live them. Anyway I wish for myself to continue grasp them and live a future rich of opportunities.

# Abstract

The primary purpose of this thesis is to further advance the development of a modular hybrid meta-heuristic optimization tool and then apply it for the ESA European Large Logistic Lander (EL3) RCS thrusters' orientation optimization problem. This study was conducted in Thales Alenia Space (TAS), in response to the desire of the GNC group to have a tool for fastest optimization of complex functions during different project phases.

Most real-world optimizations are highly nonlinear and multimodal, under various complex constraints. A study has been carried on hybrid optimization, focusing on the island model, coupled with a survey on metaheuristic algorithms used in this field. A first version of the tool implementing Differential Evolution and Evolution Strategy was already developed in TAS. During the thesis a Particle Swarm Optimization algorithm, a mechanism of mass mutation and the possibility to vary the cloning topology has been implemented. The tool has been tested on a series of benchmark functions and EL3 thrusters' orientation to show its effectiveness and make a comparison between the use of individual algorithms and a combined use of them. ESA EL3 mission has the purpose to support human and robotic exploration by providing a multipurpose lander. The study conducted on the orientation of EL3 RCS thrusters required the development of an approximate solution method. It was then possible to apply the developed tool into the studying of 2 configurations, 4 and 8 thrusters, and different combinations of degrees of freedom for the orientation, with the aim of obtaining an acceptable torque margin in the lander control.

The results on the benchmark functions demonstrate an increase in the efficiency and effectiveness of hybrid optimization in some benchmarks problems, in conjunction with the use of a suitable cloning topology, and an improvement in the exploration of the domain thanks to the presence of mass mutation. For the EL3 application problem, the tool was able to show the correlation between the degrees of freedom in the 4 thrusters' configuration, while in the 8 thrusters' configuration was not able to improve the solution in all the cases analyzed. To improve the tool's results on high non-linear problems, future work can focus on algorithms' parameters tuning and the analysis of different algorithms and topologies.

[This Page Intentionally Left Blank]

# Contents

# List of Figures

# List of Tables

[This Page Intentionally Left Blank]

# Chapter 1

# Introduction

The work I have done for this master's thesis has been carried out within the company Thales Alenia Space (TAS), in the GNC/AOCS working group. It addresses the need of the Guidance Navigation Control (GNC) group to have an optimization tool available to be used for the study in different project phases for the missions they work on. Within the Polytechnic of Turin heuristic methods are regularly used, to exploit their qualities, in combination with methods of indirect optimization. The goal of this work was to:

- Further develop a hybrid global optimization tool, implemented in MATLAB® script language, that uses evolutionary algorithms for a single parameter optimization. The tool has to be modular, easy to use and without dependencies from the MATLAB optimization toolbox.

- Review the state of the art of hybrid optimization systems that implement evolutionary algorithms and the evolutionary algorithms used with particular regard to the island model.

- Implement some of the algorithms and mechanisms studied. A global mutation mechanism within the tool and the possibility to choose the cloning topology to be used for optimization. As well as the addition of Particle Swarm Optimization to the available algorithms. Moreover to prepare the tool for the addition of a function the implement an approximate function. It will describes the problem analyzed using the data obtained from the algorithms used. The choice fell on the use of Artificial Neural Networks (ANN). The idea behind this development of the tool will be briefly introduced.

- Apply the developed tool for the ESA European Large Logistic Lander (EL3) RCS thrusters' orientation optimization problem. Studying various configurations with different degrees of freedom.

Moreover, the participation within the group of the Polytechnic of Turin, to the Global Trajectory Optimization Competition 11 held between October and November 2021, allowed me to apply ongoing the work done for this thesis. The tool has helped the validation of an algorithm used for the calculation of an optimum orbital plane to locate the Dyson ring.

Numerous application problems encountered everywhere can be modeled as global optimization problems. In addition most real-world optimizations are highly nonlinear and underlie various complex constraints. It is therefore important to develop and study numerical methods and tools that are able to identify the overall optimal of the function analyzed.

Heuristic methods are problem-dependent and with the objective to adopt problem independent strategies researchers studied meta-heuristic algorithms. This algorithms are used as a high level search strategy and can be applied to a wide range of problems [1].

Many heuristic algorithms have been developed since the 1970s. Among these algorithms we encounter: Genetic algorithms, Evolution strategy, Differential evolution, Neuroevolution, Ant Colony optimization, Artificial bee Colony algorithm, Particle Swarm Optimization, simulated annealing, Tabu search, etc. But recently these algorithms have been combined within optimization tools with each other and with other traditional approaches to solve even more complex problems. The idea behind this thesis is to obtain the combination of algorithms through a island model. This types of work have been carried out over the years with various contributions. These include:

- PaGMO[2][3]: a parallel global multiobjective framework for optimization;
- Hemoglop[4]: acronym of Hybrid Evolutionary Multi-Optimiser Global Optimisation tool;
- ParadisEO[5]: a Framework for the Reusable Design of Parallel and Distributed Metaheuristics;
- Various works about parallel optimization and island model[1][6][7][8].

These will be the works to which most reference will be made during the work presented. The island model paradigm allows to efficiently distribute evolutionary algorithms over multiple algorithms while introducing a new evolutionary operator, the migration operator. The migration operator, also known as clonation, is able to improve the overall algorithmic performance. The Generalized island model can be applied to a broad class of optimization algorithms. In this thesis in

particular will be applied to evolutionary algorithms (EA), including Differential Evolution, Evolution Strategy and Particle Swarm Optimization.

A first version of the tool implementing Differential Evolution and Evolution Strategy was already developed in Thales Alenia Space. During the thesis a Particle Swarm Optimization algorithm, a mechanism of mass mutation and the possibility to vary the cloning topology has been implemented. The tool will be tested on a series of benchmark functions and on the optimization of ESA European Large Logistic Lander (EL3) RCS thrusters' orientation to show is effectiveness and make a comparison between the use of individual algorithms and a combined use of them.

ESA EL3 mission has the purpose to support human and robotic exploration by providing a multipurpose lander. The study conducted on the orientation of EL3 RCS thrusters required the development of an approximate solution method. That's because in most real applications of EAs, computational complexity is a prohibiting factor. This computational complexity is due to fitness function evaluation. It was then possible to apply the developed tool into the studying of 2 configurations, 4 and 8 thrusters, and different combinations of degrees of freedom for the orientation, with the aim of obtaining an acceptable torque margin in the lander control.

Within the chapter "Hybrid heuristic optimization" a study will be carried on hybrid optimization, focusing on the island model, coupled with a survey on some metaheuristic algorithms used in this field.

Then in the chapter "Hybrid Evolutionary algorithms tool" will be described the reasons that led to the development of the tool object of the thesis. The work was carried out within Thales Alenia Space. It describes the architecture of the tool, the algorithms that compose it and the parameters on which they depend. Also within this chapter, through the results of the tests carried out, the effects of the mass mutation mechanism implemented and the variation of the cloning topology are shown.

The chapter "Results on numerical Benchmark Problems" presents the results of the tests carried out on the tool using the benchmark functions in the literature. These results were obtained following a tuning of the parameters, work that is certainly not conclusive and that therefore leaves room for further improvement.

The next chapter, European Space Agency EL3 Application Problem, presents the mission of the European Space Agency whose orientation of the Thrusters of the RCS system represents the application problem of the thesis. ESA EL3 mission has the purpose to support human and robotic exploration by providing a multipurpose lander. The study conducted on the orientation of EL3 RCS thrusters required the development of an approximate solution method. It was then pos-

sible to apply the developed tool into the studying of 2 configurations, 4 and 8 thrusters, and different combinations of degrees of freedom for the orientation, with the aim of obtaining an acceptable margin in the lander control.

The conclusions of the work carried out are then presented.

# Chapter 2

# Hybrid Metaheuristic Optimization

The work of this thesis is focused on a single parameter global optimization tool based on the island model, using meta-heuristic algorithms. This chapter introduce the optimization problem, some of the algorithms presented in literature and the hybrid mechanism. It is organized as follows:

- Section 2.1 presents the global optimization problem;
- Section 2.2 gives an introduction about some meta-heuristic algorithms present in literature and about the artificial neural networks.
- Section 2.3 presents the hybrid meta-heuristic optimization and focuses on the parallel island model.

## 2.1 Global Optimization

Many application problems encountered in the scientific field can be modeled as global optimization problems. Namely problems in which you have to determine the overall minimum or the maximum of a function with real values. Global optimization has as its objectives:

- analysis of non-linear models with multiple optimal solutions;
- the design and study of efficient algorithms capable of identifying the best solution globally.

The difficulties in solving this type of problem depend on the properties of the function and on the domain boundaries. From a mathematical point of view,

the problem of global optimization in $\mathbb{R}_n$ can be formulated as in 2.1. Where $f : S \to \mathbb{R}$ is a function defined on $S \subseteq \mathbb{R}_n$ set. The function $f$ is called the objective function and the set $S$ is usually called the eligible solution space.

$$\begin{aligned} &\text{determine } x^* \in S | f(x^*) \leq f(x), \quad \forall x \in S \\ &\text{or minimize } f(x), x \in S \end{aligned} \tag{2.1}$$

A maximization problem is easily attributable to a minimization problem because:

$$\max f(x) = -\min(-f(x)), \quad x \in S \tag{2.2}$$

Frequently the number of local minimum is unknown and also be very large. Also local solutions can be very far from the global one. In this case solving the global optimization problem means identifying the best of all solutions, avoiding being trapped in a non-global local minimum.

The classes of problems to be solved in the field of optimization, both local and global, are varied and very different from each other: from problems of combinatorial optimization to concave minimization, convex differential programming, lipschitz optimization, etc. As a result, the proposed approaches to solving these problems are very diverse: while a very general strategy could be suitable in all cases, obviously being inefficient in specific cases, on the other hand strictly specialized methods are applicable only to the class of problems for which they were designed for.

The main difficulties of solving the global optimization problem depend on:

- the function to be minimised, in particular the computational cost required for each function evaluation;
- by the number of local minimum and their distribution: the greater the number of local minimum points of a function, the smaller the measures of the respective regions of attraction, thus the probability of identifying the overall minimum;
- from the size of the research space: it is possible to demonstrate that the computational effort required to generate and store the test points grows exponentially as the size increases. Moreover, as the size increases, the number of local minimum increases and the probability of finding an optimal solution decreases;
- the lack of a characterisation of the overall minimum that is easy to verify: this does not allow an easy definition of the stop criteria for iterative methods of minimization.

A further element of difficulty to consider in addressing the problem 2.1 is the fact that, in general, the problem of global optimization cannot be solved with absolute certainty in a finite number of steps. This result, formulated by Dixon in the 1978[9].

The first algorithms for solving the global optimization problem date back to the first half of the twentieth century. In the seventies, the first heuristic strategies were proposed, trying with little success, to extend convergent algorithms in the one-dimensional case to higher dimensions. Depending on whether or not they use probabilistic elements, the methods developed for the resolution of a global optimization problem can be divided in principle into two main classes:

1. Deterministic methods;
2. Stochastic methods.

Typically, all methods of the Branch and Bound type belong to the class of deterministic methods, such as the Grid Search method, interval partition methods, methods based on Peano curves and partition strategies for lipschitz functions. Some methods that use perturbations of the target function, such as the tunneling method or the filled function method, also belong to the same class. These methods provide an absolute guarantee of success but require restrictive assumptions about the function. An algorithm is determistic if repeating the run from the same initial conditions are always found the same iterates. In practice the direction is deterministically found, the step as well.

Stochastic methods typically require weaker assumptions than the deterministic ones and ensure probabilistic convergence to global optimal. This class includes, for example, Two-Phase methods, Random Search conceptual methods, Simulated Annealing methods and Random Directions methods. An algorithm is stochastic when the iterate is updated using something probabilistic. For example the descent direction can be stochastic, or the step. And it can be stochastic for example because the exact gradient calculation would be expensive and so its used a smaller batchsize. If an algorithm is stochastic, repeating the run from the same initial conditions are obtained different iterations. Methods based on stochastic procedures exist in the literature which, under particular conditions, are empirically competitive but do not provide guarantees of convergence. In some cases, a method may be required to be empirically competitive without providing guarantees of convergence, as is the case for heuristic/meta-heuristic methods. Heuristic methods are problem-dependent while meta-heuristic algorithms are used as a high level search strategy and can be applied to a wide range of problems. These methods include for example all those approaches that simulate the processes of the biological evolution of a system, such as natural selection,

and apply to populations of points that are recombined sequentially to generate the optimal solution. These include evolutionary methods and genetic algorithms, an introduction about these algorithm is given in section 2.2.

## 2.2 Algorithms

### 2.2.1 Swarm Intelligence

Swarm intelligence arises from biological intuitions, about the incredible ability of social insects to solve problems of everyday life. There are several examples of complex behaviors among social insects, from the construction of efficient structures, to the dynamic distribution of tasks among workers.

Individual insects do not need any representation, map or explicit knowledge of the overall structure they produce. A single insect is not able to assess a global situation, to centralize information on the state of the entire colony to which it belongs, nor to control the tasks that must be carried out by other workers, as, within these there is no supervisor. A colony of social insects is like a decentralized system, consisting of autonomous units distributed in the environment, and can be defined as a set of simple behaviors of response to stimuli. The rules governing interactions between individuals are performed on the basis of local information, without knowledge of the global model. Each insect follows a small set of behavioral rules. The organization of the colony comes from the sum of the simple behaviors of each individual. All these interactions ensure the propagation of information through the colony and, in turn, organize the activity of each individual. Thanks to this network of interactions social insects are able to offer a very flexible and robust system.

#### 2.2.1.1 Ant Colony Optimization

The study of intelligent swarms, led to the formation of the Ant Colony Optimization (ACO) algorithm, first introduced in 1992 by Marco Dorigo[10]. It is used for solving complex computation problems. A simplified pseudocode for ACO is preseted with Algorithm 1.

#### 2.2.1.2 Artificial Bee Colony

Another swarm intelligence algorithm is the Artificial Bee Colony Algorithm (ABC) that mimics the behavior of honey bees to perform research, prioritize, and other activities. It was developed in 2005 and has been applied to a number

**Algorithm 1** ACO Algorithm

```
Define End Condition
while End condition not satisfied do
  schedule activities
  ants activity
  pheromone evaporation
  daemon actions
end while
```

of optimization problems. The goal is to determine the best solution to a problem. The decision-making processes used by bees in nature to solve problems related to the management of hives can be equally effective in other environments.

A single hive uses a combination of two search methods to return data, in this case, information about food sources. The first is the use of scouts, which randomly scan a region to locate specific areas, or neighborhoods, which will likely give good results. Scouts return to the hive and the other bees decide which neighborhoods to look for more intensively to identify useful resources. This combination of random and local search patterns can be optimal for some search environments. In the algorithm of the bees, the programmer can decide how many scouts to send, launching them to carry out random searches in all directions. They identify the most likely sources of useful data or the most optimal solutions in a range of choices and report with this data. More intense localized searches in these regions can yield the best results, classified in terms of relevance, effectiveness and other features that the programmer can set.

#### 2.2.1.3 Particle Swarm Optimization

Particle Swarm Optimization is a population-based stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995[11][12][13][14]. It is a heuristic method of research and optimization, inspired by the social behavior of flocks of birds and schools of fish. Individuals follow a very simple behavior: they emulate the success of neighboring individuals. Flocks of birds stay together, coordinate when they need to change direction, avoiding collisions with obstacles and each other. Shoals of fish behave in a similar way. Each bird or fish follows three simple rules:

- Keeps a specific minimum distance from the object or other nearest birds;
- Equals speed with the neighboring birds;
- It's close to the center of the flock.

The flock is a self-binding structure to which all individual actions respond simultaneously and change the general structure of the swarm. Although each bird or fish only feels the movements of its nearest peers, its reactions to these movements propagate to the other entities of the structure, so the system as a whole exhibits a global coordination. Such optimization shares many similarities with evolutionary computing techniques, such as genetic algorithms.

The system is initialized with a population of random solutions and searches for the optimal one by updating the generations. However, unlike genetic algorithms, there are no evolutionary operators such as crossovers and mutations in particle swarm optimization. In this optimization, the possible solutions called particles, fly through the space of the problem following the optimal current particles. At each interaction the algorithm identifies a new candidate in the space of the problem considered, and it does so in reference to a specific measure of quality called fitness. Optimization with swarms of particles allows exploration of large spaces of solutions, given the structure of algorithm there is no guarantee that the optimal solution will ever be found. This type of optimization uses a population of candidate solutions, called particles, that move in the research space on the basis of simple formulas, which take into account their current travel speed, their knowledge of the fitness space (the best solution found so far) and shared knowledge (the best overall solution identified).

A particle i in a time t, assumes a position and a velocity. There is a function called fitness function that evaluates the quality of the position of the particles. Each particle has a memory in which it keeps track of the best position reached, and is able to know what is the best position taken by its neighbors. The social component reflects the information exchanged between neighbors, highlighting the local nature of the system. The best solution found so far by the particle is called $P_{best}$ and the best overall solution identified is called $G_{best}$.

After finding the two best values, the particle updates its velocity and positions with following equation 2.3 and 2.4.

$$v = v + c_1 \cdot RAND1 \cdot (P_{\text{best}} - p) + c_2 \cdot RAND2 \cdot (G_{best} - p) \tag{2.3}$$

$$p = p + v \tag{2.4}$$

$v$ is the particle velocity, $p$ is the current particle or solution. $P_{\text{best}}$ and $G_{\text{best}}$ are defined as stated before. RAND1 RAND2 and random numbers between $(0, 1)$.$c_1$, $c_2$ are learning factors. Usually $c_1 = c_2 = 2$[15]. A pseudo-code of the algorithm is presented in Algorithm 2. Particles' velocities on each dimension are clamped to a maximum velocity $V_{max}$. If the sum of accelerations would cause the velocity on that dimension to exceed $V_{max}$, which is a parameter specified by the user, then the velocity on that dimension is limited to $V_{max}$.

**Algorithm 2** PSO Algorithm

```
Define End Condition
for each particle do
  Initialize particle
end for
while End condition not satisfied do
  for each particle do
    Calculate fitness value
    If the fitness value is better than the best fitness value (pBest) in historyset
    current value as the new pBest
  end for
  Choose the particle with the best fitness value of all the particles as the gBest

  for each particle do
    Calculate particle velocity according equation 2.3
    Update particle position according equation 2.4
  end for
end while
```

### 2.2.2 Tabu Search

The Tabu Search represents an evolution of the classic descent method that is, used to find the minimum of a real function f on a set S, solutions space. This classical methodology consists in starting from an initial solution and performing a series of moves that lead to a new solution within the surrounding of the current solution, in which the objective function f assumes a value less than the present value. The defect of the descent method lies in the fact that, if in the set of adjacency there are no better solutions than the current one, the search stops. The optimal solution identified by the descent method is therefore associated with a minimum local space of the solutions, which is often far, especially in terms of quality, from the overall optimal solution.

The Tabu Search technique was proposed by Fred Glover as a way to continue the search beyond local lows. First, in order to escape the trap of local lows, the Tabu Search allows worsening moves. However, by doing so, there is the risk of falling back immediately afterwards into the local minimum, unless you somehow prevent the moves recently made. The fundamental concept of Tabu Search consists then in making forbidden or, in fact tabu, the last moves made in the search path, so that the algorithm can not go back on its feet and fall back into the local minimum. The basic feature of Tabu Search is therefore the systematic use of memory: to increase the effectiveness of the search process, not only local

information is kept track (such as the current value of the target function)but also some information about the itinerary. This information is used to guide the move from the current solution to the next solution, to be chosen within the adjacency set. A simplified pseudocode version of the algorithm is presented in algorithm 3.

**Algorithm 3** Tabu Search

```
Define End Condition
while End Condition not satisfied do
    Update End Condition
    Search the neighborhood
    Evaluate candidate solutions
    Update Tabu list
end while
```

### 2.2.3 Simulated Annealing

Simulated Annealing originates from the analogy between the physical process of cooling solids, called annealing, and the problem of finding minimal solutions for optimization problems in the discrete. Annealing denotes a physical process in which a solid immersed in a hot bath reaches a state of minimum energy by a slow lowering of the system temperature.

The first solid annealing simulation algorithm is due to Metropolis in 1953. In 1983, Kirkpatrick used this algorithm for combinatorial optimization problems by replacing energy with a cost function and physical system states with solutions to an optimization problem. For its success in this type of problem, the study of Simulated Annealing has also been extended to problems of continuous optimization. Apart from the purely heuristic reason for the origin of the method, the peculiarity of Simulated Annealing is that the algorithm avoids stabilization in a local minimum that is not global by accepting, as well as transitions corresponding to a decrease in the value of the function, also transitions that correspond to an increase in the value of f; these occur however in a limited way. The general idea of the algorithm is to generate at each test a point based on a given probability distribution D and its acceptance or rejection depends on an assigned function. This function constitutes the so-called criterion of acceptance and depends on a parameter, called temperature, that controls the acceptance of those points to which corresponds an increase in the value of the objective function. As the temperature decreases, the probability of transitions decreases. The general scheme of the algorithm is presented in the algorithm 4.

**Algorithm 4** Simulated Annealing

determine the starting point $x_0$ arbitrarily in $S$
$z_0 \leftarrow \{x_0\}$
determine $T_0 > 0$, initial temperature value
$k \leftarrow 0$
**while** End condition not satisfied **do**
  generate a point $y_{k+1}$ according to a distribution $D(z_k; \cdot)$
  choose $p$ evenly in $[0, 1]$
  **if** $p \leq A(x_k, y_{k+1}, T_k)$ **then**
    $x_{k+1} \leftarrow y_{k+1}$
    $z_{k+1} \leftarrow z_k \cup \{x_{k+1}\}$
  **else**
    $x_{k+1} \leftarrow x_k$
    $z_{k+1} \leftarrow z_k$
  **end if**
  $T_{k+1} \leftarrow U(z_{k+1})$
  $k \leftarrow k + 1$
**end while**

### 2.2.4 Evolutionary Algorithms

Within artificial intelligence, an evolutionary algorithm (EA) is a subset of evolutionary computation, a generic population-based metaheuristic optimization algorithm. An EA borrows their mechanisms from biological evolution: reproduction, mutation, recombination, natural selection and survival of the fittest. Candidate solutions to the optimization problem play the role of individuals in a population, and the cost function determines the way in which the person is adapted to the environment. Evolution of the population then takes place after the repeated application of the above operators. Evolutionary algorithms consistently perform well in approximating solutions to all types of problems because they do not make any assumption about the underlying fitness landscape.

Particularly, major elements of the Genetic Algorithm are [16][17][18][19]:

- Population Representation and Initialisation. EAs and GAs operate on a number of potential solutions, called a population, consisting of a set of solutions to the problem that evolve during the algorithms run. The most commonly used representation of chromosomes in the GA is that of the single-level binary string. But it is also common to use real value vector representation. And that will be the case for the tool developed. This increases efficiency because you save a conversion before you can call the function evaluation, that is based on a real number problem. The first step

in the simple genetic algorithm is to create an initial population. This is usually achieved by generating the required number of individuals using a uniform or gaussian random number generator that distributes numbers in the desired range.

- The Objective or Fitness Functions. The objective function is used to provide a measure of how individuals have performed in the problem domain. The result value from the evaluation will be considered the fitness value associated with the individual.

- Selection process. Selection is the process of determining the number of times, or trials, a particular individual is chosen for reproduction and, thus, the number of offspring that an individual will produce.

- Crossover or Recombination. The basic operator for producing new chromosomes in the GA is crossover. Like its counterpart in nature, crossover produces new individuals that have some parts of both parent genetic material.

- Mutation. It is a random process where one gene is replaced by another to produce a new genetic structure. In GAs, mutation is randomly applied with low probability, typically in the range 0.001 and 0.01, and modifies elements in the individual.

- Reinsertion. Once a new population has been produced by selection and recombination of individuals from the old population, the fitness of the individuals in the new population may be determined. Then a reinsertion scheme must be used to determine which individuals are to exist in the new population.

- Termination conditions. Being stochastic methods It is difficult to formally specify convergence criteria. Different criteria are then proposed, based on fitness function evaluation, number of generations, time, or relative evolution of the solution.

A general Evolutionary Algorithm procedure is presented in figure 2.1.

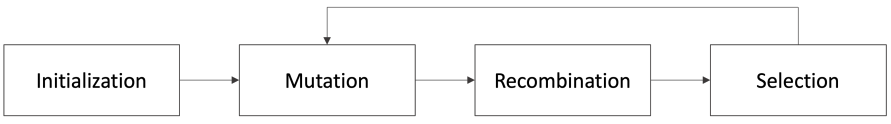



Figure 2.1: General Evolutionary Algorithm Procedure

#### 2.2.4.1 Differential Evolution

Differential Evolution (DE) generates new vectors of parameters by adding the weighted difference between two population vectors to a third one[20][21][22]. If the resulting individual provides a lower objective function than a predetermined population member, in the next generation the new individual replaces the one with which it was compared; otherwise, the old individual is retained. G. The population size does not change during the optimisation process. For each target vector a mutant vector for the next generation is generated according to 2.5.

$$v_{i,G+1} = x_{r1,G} + F \cdot (x_{r2,G} - x_{r3,G}) \tag{2.5}$$

The chosen vector are random, and have to be different. F is a real factor varying between 0 and 2 that controls the amplification of the difference vector. This is the basic strategy for Differential Evolution. By modifying the F value is possible to obtain DE algorithm that are more focused on the domain exploration or more focused on exploiting of the solution.

The target vector is mixed with the mutated vector thanks to the crossover. This operation is governed by the factor CR. CR is a probability that usually vary from 0 to 1. CR = 0 means no crossover.

#### 2.2.4.2 Evolution Strategy

Evolution strategies are evolutionary algorithms that date back to the 1960s and that are most commonly applied to black-box optimization problems in continuous search spaces. Inspired by biological evolution, their original formulation is based on applying mutations, recombinations and population selections of candidate solutions.

Individuals are also denoted as parents or offspring, depending on the context. In a generational procedure:

1. one or several parents are picked from the population and new offspring are generated by duplication and recombination of these parents;

2. the new offspring undergo mutation and become new members of the population;

3. environmental selection reduces the population to its original size.

Within this procedure, evolution strategies employ the following main principles that are specified and applied some fundamentals operators.

The parameters that characterize the algorithm are[23]:

- $\lambda \in \mathbb{N}$ number of offspring, offspring population size.

- $\mu \in \mathbb{N}$ number of parents, parental population size. Survive after the environmental selection.
- For recombination, $\rho$ (out of $\mu$) parent individuals are used. We have therefore $\rho \leq \mu$.
- $\sigma$ and $\tau$ govern the mutation. The mutation operator introduces small variations by adding a point symmetric perturbation to the result of recombination. $\tau$ is the learning rate.

### 2.2.5 Artificial Neural Networks

Neural networks, also known as artificial neural networks (ANN) or simulated neural networks (SNN) are a subset of machine learning and are the central element of deep learning algorithms. Their name and structure are inspired by the human brain, mimicking the way biological neurons send signals.

Artificial neural networks (ANN) are composed of layer nodes that contain an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any single node is above the specified threshold value, that node is activated, sending the data to the next network level. Otherwise, no data is passed to the next level of the network. Neural networks rely on training data to learn and improve their accuracy over time. However, once optimized for accuracy, these learning algorithms are powerful tools in computer science and AI, allowing us to classify and cluster high-speed data.

Each individual node is a linear regression model, consisting of input data, weights, a distortion (or threshold) and an output. The formula that defines the node is near what presented in 2.6.

$$\sum_{i=1}^{m} w_i x_i + \text{ bias } = w_1 x_1 + w_2 x_2 + w_3 x_3 + \text{ bias } \tag{2.6}$$

Once an input level is determined, the weights are assigned. These weights help determine the importance of any given variable, with larger ones contributing more significantly to the output than other inputs. All inputs are then multiplied by their respective weights and then summed. Next, the output is passed through an activation function, which determines the output. If it exceeds a certain threshold, that output activates the node, passing the data to the next level on the network. The output of a node then becomes the input of the next node. This process of passing data from one level to the next defines this neural network as a feedforward

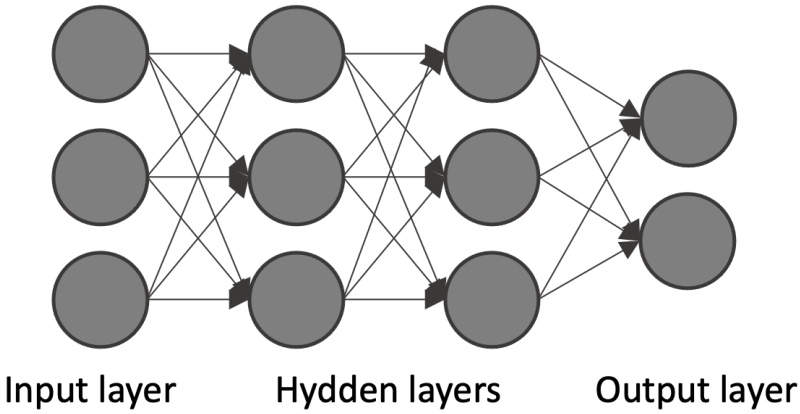



Figure 2.2: Feedforward Artificial Neural Network

network. Specifically feedforward neural networks are formed by an input level, one or more hidden levels and an output level as represented in figure 2.2.

Within this thesis were used the ML tools provided by MATLAB for the development of a neural network, based on the feedforward mechanism. The goal of this work, only started and which is not the focus of the thesis, is to give the possibility to the tool to create a surrogate model, an approximate function, of the analyzed problem. This model will be created by exploiting the information coming from the individuals analyzed during the evolution of the tool within the present islands. This allows not to lose the information found but to use them to train the neural network.

## 2.3 Hybrid Algorithms Metaheuristic Optimization

In recent years there has been a proliferation of investigation that shifted interest from exact approaches to near-optimal heuristics and metaheuristics. The aim of these studies was to achieve an optimal or almost optimal solution for this NP-hard problem within a minimum time budget[24][25][26][27].

Although metaheuristics has met some expectations, the quest for a high-quality, near-optimal solution has led researchers to design hybrid methods. Consequently, research is evolving towards the hybridisation of metaheuristics. Hybrid metaheuristics have been promising efforts to transcend the boundaries of metaheuristics by leveraging the strength of complementary methods to overcome base algorithm shortcomings. Inadequacy in traditional metaheuristics such as slow or

premature convergence and stochastic behaviour.

The hybrid strategies for metaheuristics were born from the need to reduce the inadequacy of metaheuristics algorithms in some situations. The objectives of the hybridized model can be summarize in:

- To improve solution qualities.
- To accelerate convergence.
- To avoid local entrapment.
- To reduce search space.
- To improve exploration or exploitation.

Following the survey carried out by a group of researchers in December 2021[1], which covers 202 resolution methods published between 2003 and 2020, it is possible to analyse the current state of metaheuristic hybrid optimisation.

The result is that while 25% of solutions use exact or heuristic methods, 40% use metaheuristic methods and 35% use hybrid metaheuristic methods. Recent application of hybrid metaheuristic has shown promising results.

One of the most sought-after approaches to improving meta-heuristics. efficiency has been hybridization with other techniques to attain a better search mechanism with minimal time complexity. The results of the analysis indicate that the choice of the algorithm falls on 42% of the cases on Evolutionary Algorithms (EA), 21% of the cases on Particle Swarm Optimization (PSO), 10% of the cases on the Ant Bee Colony (ABC) and the remaining 27% of the cases on other types of algorithms. Also the tool object of this thesis will be developed mainly with Evolutionary Algorithms and in addition PSO and other characteristics will be added.

The hybrid strategies can be divided in:

- Algorithm operators, 54%;
- Metaheuristics, 20%;
- QoS Modeling, 13%;
- Data Clustering, 8%;
- Machine Learning, 5%.

The 54% of the studies proposed a hybrid method by replacing or modifying one or a few operators that characterize. Therefore, much research is dedicated to proposing a hybrid alternative with dynamic operators. As reported above Evolutionary Algorithms, particularly genetic algorithms and swarm intelligence techniques, are the basis of the vast majority of algorithms. Empirical evidence suggests that genetic algorithm suffers from slow convergence while PSO-based algorithms diverge at local optima in some circumstances. Operator modification is shown to be a potent remedy to overcome algorithm weakness by integrating another algorithm operator. Even randomicity that governs the stochastic optimization process of traditional metaheuristics has been the root of inefficiency. Therefore, one major effort has been to modify the search operator to generate a quality initial population with adaptive search mechanisms. These practices accelerate the convergence, improve solution quality and provide immunity against local entrapment.

Around 20% of research efforts have improved a metaheuristic algorithm by combining it with another metaheuristic to minimize the weakness and maximize their strength. The survey shows that a population-based metaheuristic frequently employed as a base algorithm due to its hybridization capacity. The major disadvantage of combining multiple metaheuristics is sophistication, which leads to higher time complexity. Therefore, this strategy has been less popular in comparison to the practice of operator modifications.

Only 5% of articles incorporate a hybridized fitness function with a combination of a few techniques to improve the algorithm performance. That is the QoS modeling.

There is a close association between search space dimensions and search method efficacy. Therefore, some researchers used service clustering techniques to reduce the search space in order to improve algorithm performance. Service clustering has been a proven hybrid strategy to accelerate the search process.

Furthermore in recent years, Machine Learning (ML) has gradually become a viable hybrid strategy to incorporate into metaheuristics. The machine learning techniques including Q-learning, deep learning, reinforcement learning has been employed to improve the search mechanism on different grounds.

The tool, argument of this thesis, uses more strategies. It presents changes in the operators applied to the available algorithms and these will act in a combined way to try to maximize their strengths. Machine learning has also been used, although this is not the focus of the work done. In particular, the hybrid strategy

used is that of a parallel composition with the Island Model.

### 2.3.1 Parallel Metaheuristic and Island Model

All methods analyzed in the study have a proposed composition approach that is semi-automated or fully automated. Regarding the choice of parameters and algorithms. While as for the way in which to compose the different algorithms the possible choices are 4 and are reported in the figure 2.3.

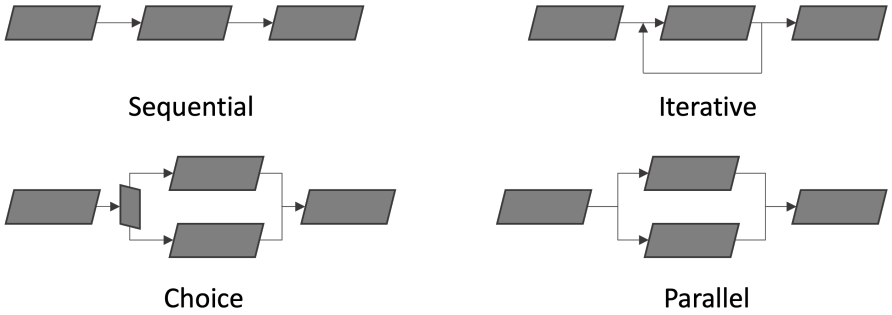



Figure 2.3: Composition Modes

The reference models for this work are all based on the method of parallel composition. This method in fact lends itself optimally to a parallel computation implemented within the various algorithms. These works are:

- PaGMO[2][3] that is a parallel global multiobjective framework for optimization. It is built to tackle high-dimensional global optimisation problems, and it has been successfully used to find solutions to real-life engineering problems among which the preliminary design of inter- planetary spacecraft trajectories, both chemical and low-thrust, the inverse design of nanostructured radiators and the design of non-reactive controllers for planetary rovers.

- Hemoglop[4]: acronym of Hybrid Evolutionary Multi-Optimiser Global Optimisation tool. A Phd work, that is a multipopulation algorithm that exploits three evolutionary algorithms running in parallel and exchanging information between them during the evolution.

- ParadisEO[5]: a Framework for the Reusable Design of Parallel and Distributed Metaheuristics. A white-box object-oriented framework dedicated to the reusable design of parallel and distributed metaheuristics (PDM). ParadisEO provides a broad range of features including evolutionary algorithms (EA), local searches (LS), the most common parallel and distributed models and hybridization mechanisms.

The island model[28] paradigm allows to efficiently distribute genetic algorithms over multiple processors while introducing a new genetic operator, the migration operator, able to improve the overall algorithmic performance. The analogy with the islands of an archipelago, where each island hosts a population that evolves over time governed by an algorithm, gives the name to the Island Model. The islands will communicate with each other during the evolution of the analysis, exchanging information thanks to mechanism called migration. This exchange mechanism involves copying an individual from one island to another, according to various migration topologies. The information exchange process is done with an operator called Clonation, that is something similar to a migration operator for the basic multi-population algorithm. This thesis work will refer to this operation as migration or clonation considering them synonymous. The exchange of information influences the analysis that allows therefore to obtain better results going to reduce the incidence of the difficulties of the single algorithms on the analysis. This mechanism that regulates the exchange of information will depend on various factors that will influence the performance of the analysis, these are:

- The number of islands;
- The migration topology that determines feasible migration paths
- migration rate that tells how many individuals migrate from the source population at a time;
- The migration frequency that tells how often the migration occurs;
- The migration algorithm that specify all remaining details.

It has been shown that employing the Island Model[29] leads to increased algorithm performance what can be explained in terms of improved balance between exploitation and exploration of the solution space. In the Island Model, each island can exchange information with its neighbor island as defined in the graph of possible links between islands, commonly referred to as migration topology. If the calculation, during the analysis, is divided between several processors, with the increase in CPU number and power, the number of islands contributing to one optimization can grow significantly and the resulting optimization is thus affected more clearly by the way the information is exchanged between the islands and in particular by the migration topology.

The migration topology has two effects on the underlying optimization process. The first one, beneficial, is the super-linear speed-up caused by the information

exchange, and the second one, being sometimes an issue, is the BUS and CPU overhead caused by the required information flow. Furthermore, if the information is synchronously exchanged, it will be necessary to wait until all the islands have evolved so that information can be exchanged before continuing. This does not happen in the case of asynchronous migration.

A very interesting and exhaustive migration topology study was conducted by Rucinski, Izzo and Biscani [7]. In the 3.3 paragraph the migration topologies presented in the tool will be tested, after having introduced the tool in the chapter 2, following a similar approach.

# Chapter 3

# Hybrid Evolutionary Algorithms Tool

Numerous application problems encountered everywhere can be modeled as global optimization problems. It is therefore important to develop and study numerical methods and tools that are able to identify the overall optimal of the function analyzed. The difficulties in solving this type of problem depend on the nature of the function and the research domain.

The idea of developing this global optimization tool based on meta-heuristic methods was born within the GNC/AOCS group of the company Thales Alenia Space (TAS). The team works according to the guidelines imposed by the ESA documentation and follows the mission life cycle (MLC) process. Following the different phases of a mission from pre-phase A. The problems that are encountered are therefore varied and in different fields.

And it is precisely the need to study the real problems that has raised the interest in the design and implementation of an optimization tool. The tool have to be modular and easy to use. The tool can be useful in different aspects of the design process and review. Facing a new problem to be optimized, not having an analytical model that shows the correlation between the DOF, a meta-heuristic tool could be able to find an optimum solution and possibly show a correlation between the DOF. This correlation could allows to reduce the DOF necessary for the study of the problem in successive phases.

The problem addressed in this thesis is the study of an RCS configuration for the thrusters orientation study. Given 8 angles to define the orientation of 4 thrusters has been the tool itself to find an ideal configuration in which the thrusters are arranged with symmetrically direction with respect to the center of gravity. That while not making preliminary assumptions about the type of function to optimize. This result will be better explained in chapter 5. Other

examples analyzed with a previous version of the tool are the research of the worst case for the release of a satellite, or the worst condition for the angular velocity of the same satellite.

Moreover, as the project progresses, the refinement in the knowledge of the problem data and the solution model will allow the tool to refine the solution looking for an optimal one.

Some of the objectives of the tool are then improving knowledge of the problem, showing a correlation between the DOF, and to refine the solution optimizing it with increasing knowledge of the problem data. Furthermore a possible development of the tool can give to it the possibility to provide a robust solution. It is important within the project of realization of the tool the possibility that a model of the problem is created that allows to obtain a robust solution, able to remain valid even with a small change of the boundary conditions, situation that usually occurs during the project phases for which the tool will be used. The achievement of this robustness of the solution is conditioned to the creation of an approximate function. To realize this approximate function it was decided to use a neural network model provided by MATLAB. The approximate model will be obtained using the data resulting from the optimization analyses carried out, implementing a surrogate model.

In order to avoid having making assumptions about the problem the optimization tool will be based on stochastic methods. It consists of a variable number of algorithms combined with each other. Each algorithm represent an island and the tool is therefore called ISOLE (the Italian word for islands). Each algorithm implemented evaluates the target function on a set of attempt solutions, which will be referred to as individuals, forming together a population and representing an island. The algorithms/islands can interact with each other in a system that is called archipelago, a name derived from the analogy between the system of algorithms and the set of a group of islands. The number and type of algorithms used by the tool can be chosen by the user, depending on the type of analysis to be carried out. In order for the tool to be easy to use, the necessary parameters are inserted through an input spreadsheet, in addition to this it is necessary to prepare a cost function and an objective function (which may coincide) in a format that will be described. It is possible to select the output data in addition to the individual that corresponds to the optimum found.

The chosen programming language to develop the tool is MATLAB. This is not only a programming language but also a development environment for numerical calculation and statistical analysis. Within this environment you can manipulate matrices, view functions and data, implement algorithms, create user interfaces and interface with other programs. It is used by millions of engineers and scientists

for modelling, data analysis and algorithm development and is regularly used within the GNC/AOCS group in Thales Alenia Space where I have conduct my thesis experience.

## 3.1 Tool Structure

For ISOLE to be able to work, the user is required to define an input file and a file that contains the cost function and the objective function. Once run, the program will give as output the optimized parameter and in addition the data required. They must be representative of the optimization problem. The objective function will be the function that the ISOLE tool will optimize. The two functions can coincide if the cost function is able to give as result a single value, otherwise if the cost function result consists of more values it will be the task of the objective function to put together these values to obtain a single parameter to optimize. This is because the optimization tool uses a single parameter optimization approach. It is therefore possible to define through the input file all the parameters involved in the optimization, starting from the number of islands and the type of algorithms that will work on each island.

The ideal algorithm presents at the same time the characteristics of speed, convergence and accuracy. Defined input parameters several algorithms will run in parallel and exchanging information between them with constancy is possible to create a synergistic union that improves the result. When the end conditions chosen among those available is reached by the tool it will be provided the results in output, with the amount of information desired in addition to the optimum found.

Upon my arrival in Thales Alenia Space the tool consisted of two algorithms, Differential Evolution and Evolution Strategy, with sub-versions that vary some of the parameters that define the algorithms. The input was managed by modifying the parameters directly from the code. The output was fixed. Upon my arrival at the company I was asked to work initially on the organization of the tool, so as to make it modular, easy to use and easily modifiable. The work was started in MATLAB® and I decided to continue on this path with the intention of achieving another of the company's objectives, namely to have an open optimization tool available. I was given the opportunity to choose which additional algorithms to implement and I was required to implement the mass mutation mechanism and the possibility to vary cloning topology. So the work that I did during my thesis was focused on the organization of the work already carried out to date, the implementation of new algorithms and operators with the aim to improve the tool.

The tool was then evaluated through various benchmark functions found in the literature. Input has become settable through a spreadsheet file, as well as tool parameter management. The output has been standardized and it is possible to select the data of interest to be obtained at the end of the analysis. A version of the PSO algorithm has been implemented, a global approximation function based on the neural network tool offered by the MATLAB® software, the possibility to choose between various cloning mechanisms and an operator of mass mutation has been added to reduce the stagnation of solutions in local minimums.

At the end of this work the tool is working and have been implemented the functions required in the development phase. A tuning of the parameters of the individual algorithms was performed. Anyway the purpose of this work was not to propose an optimal tuning of the parameters, a more in-depth work can be done in this regard. The advantage of the hybrid optimization using the island model is to improve efficiency and effectiveness thanks to the interaction between the algorithms. The results obtained by the tool through the benchmark functions are therefore considered acceptable. During the development of the tool many ideas were born about how to reuse the data that arises from the algorithms evolution through a surrogate model applied to machine learning. In particular, a work that will lead to the union of a neural network and an evolutionary algorithm has been sketched. The candidates are currently Evolution Strategy and Particle Swarm Optimization. This will allow to exploit the information obtained from all the islands for the evolution of a single algorithm thanks to the approximate function created with the neural network.

This chapter describes the architecture of the tool, the available settings, and the parameters governing the optimization.

### 3.1.1 Input

The input requires the user to specify, in a defined script format, the variables of the problem and to define a cost function and an objective function, which may coincide, depending on the case, to allow the single parameter optimization. In figure 3.1 are shown the blocks of which it is composed the input and the parameters that it is possible to set in phase of setting of the program. The yellow colored blocks represent the areas where the work carried out has led to an improvement of the initial project, while the green blocks represent the contribution given to the tool in these months of work.

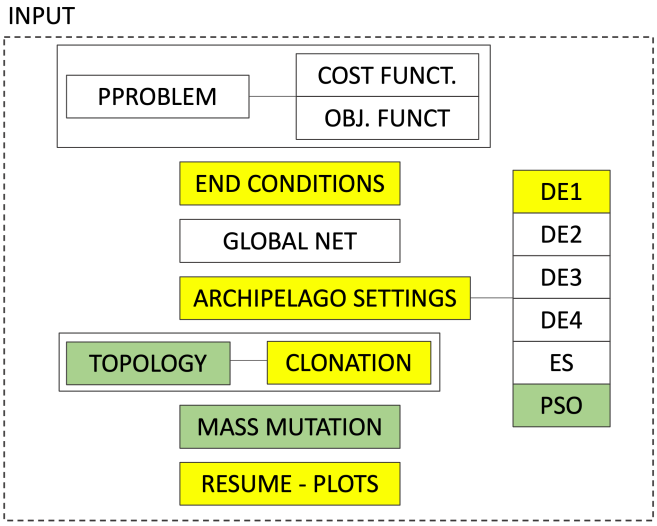



Figure 3.1: Input

#### 3.1.1.1 Problem

The problem considered is in the form:

$$\begin{aligned} &\text{find } \text{lb} \leq x \leq \text{ub} \\ &\text{to minimize } f(x) \in R_n \end{aligned} \tag{3.1}$$

where x belonging to $R_n$ is the vector of unknowns that will also be referred to as individual. It's composed of n real elements that represent the degrees of freedom of the problem and then define the size of the problem that will be analyzed. lb and ub belong to $R_n$ and define lower and upper boundaries of the domain. For example for the search of the thrusters best orientation in an RCS system they represent the physical limits due to the mounting of the thrusters and the cone of thrust direction. $f(x)$ represents the function that represents the problem object of the optimization.

To clarify which data should be defined, the mock-up of a test function is presented (3.1.1.1), in which the parameters that need to be defined are shown.

```
1      case PS.OPT_FUN1
2          % Dimension of the problem
3          D=30;
4
5          % Define Lower Boundaries
6          lb = -5.12*ones(D,1);
7          % Define Upper Boundaries
8          ub = 5.12*ones(D,1);
```

```
9           % Define Weight used for managing boundaries (if any)
10          Weight=ones(D,2);
11          % Define Weight = 0 for debug purpose
12          Weight_pure=zeros(D,2);
13          % Define anonymous function
14          Cost_function=@(x) (f1_opt(x,lb,ub,Weight));
15          Cost_function_pure=@(x) (f1_opt(x,lb,ub,Weight_pure));
```

While the cost function and objective function must present the form reported in the example of the same problem of mock-up (3.1.1.1). In this case the two functions coincide and the value resulting from the evaluation is modified by an augmented cost that aims to keep the exploration of the domain within the constraints imposed. The user can choose to run the program with the evaluation of the test functions, which are defined and integrated within it. Each will be identified by a number and can be selected during the launch of the analysis. Or can define a new function.

```
1 function J=f1_opt(x,lb,ub,Weight)
2
3 J_pure=sum(x.^2);
4
5 lb_cons=lb-x;
6 ub_cons=x-ub;
7 Augmented_cost = sum(sum(Weight.*[lb_cons.*(lb_cons>0), ...
      ub_cons.*(ub_cons>0)]));
8
9 J=J_pure + Augmented_cost;
```

#### 3.1.1.2 End conditions

The meta-heuristic optimization is conditioned to the satisfaction of an end condition that stops the search. Different criteria are used in the literature, the most used being the number of generations and the number of evaluations of the target function. When I started working on the tool it was possible to set a maximum number of generations for the evolution of the populations analyzed. During the reorganization of the tool other methods have been implemented and therefore those currently available are:

- Maximum number of generations. It is a classic condition that foresees the evolution of algorithms for each island for a fixed number of generations. Upon reaching the limit, the program asks if the user wants the analysis to continue and for how many generations, or if the user wants the analysis to stop.

- Maximum number of objective function evaluations. This is a very useful method when the tool is evaluated. Whenever new individuals are produced and these evolve, their evaluation through the objective function is required. In this way they can be classified according to the score obtained and they will continue the evolution. A counter is updated every time the evaluation function is called, in this way it is possible to compare the runs of algorithms with different parameters with a fixed number of objective function evaluations.
- Time of execution of the program. Other possibilities to terminate program executions can be determined by a tolerance between one generation and the next or by a maximum execution time. Given the large number of executions that required the tool to be tested and developed, a limit for the execution time was implemented.

#### 3.1.1.3 Global Net

The tool aims to provide a function approximating the problem analyzed, using the results of the evaluation of the objective function carried out for each generation by the algorithms. This is the idea behind a surrogate model. For this reason, the libraries for the realization of neural networks made available by MATLAB are used. The FeedForwardNet mechanism has been used. During the parameter definition it will be possible to decide the following parameters:

- Whether or not to activate the use of neural networks within the tool. If they are activated, two neural networks will be produced. The first network that will receive the input of the problem data and will output the value sought by the optimization (then from $R_n$ to $R_1$). The second one that will receive in input the expected value and will give in output the values of the variables that allow to obtain that value (the working from $R_1$ to $R_n$).
- The number and type of hidden layers that define the neural network from $R_n$ to $R_1$.
- The number and type of hidden layers that define the neural network from $R_1$ to $R_n$.

#### 3.1.1.4 Archipelago

A fundamental part for the execution of the tool is the definition of what are called archipelago variables. It is left to the user the choice of how many and which algorithms to use and the number of islands whose populations will evolve with the generations. The parameters to choose are then:

- The number of islands and therefore of populations used by the tool. For the analyses carried out within this thesis the number of islands has been made to vary from 1, for the evaluation of the single algorithm, to even more than 10, for the evaluation of the clonation topologies.
- The type of algorithm used for each island. The following describes the types of algorithms among which it is possible to decide and which are the characteristics that distinguish them. The choice will be between:
  - Differential Evolution: DE1, DE2, DE3, DE4
  - Evolution Strategy: ES, ES2
  - Particle Swarm Optimization: PSO
  - Particle Swarm Optimization + Artificial Neural Newtwork: PSO_ANN. This algorithm was a surrogate model trial. The results at the moment are not encouraging and will not be discussed further during the thesis.

The characteristics of the implemented algorithms were discussed in Chapter 2.

#### 3.1.1.5 Clonation

Regarding the clonation, the user can choose whether or not to apply it during the analysis. If is chosen to disable it the algorithms will run in parallel without actually exploiting the island model possibilities. As for the parameters that are changeable, these are:

- The clonation rate;
- The clonotation topology;
- The clonation selection mechanism;
- How many individuals clone at every clonation.

The cloning mechanism and the tests carried out are treated in the paragraph 3.3.

#### 3.1.1.6 Mass Mutation

The Mass Mutation operator is an implementation resulting from the work of Sentinella[4]. Its purpose is to limit the possibilities that algorithms may stuck in a local minimum during analysis. In case this happens, if the analysis is not able to improve after a certain number of iterations then the operator proceed to the

re-initialization of part of the population. This mechanism favors the exploration of the domain.

The Mass Mutation is explained in more depth, defining the parameters and showing the tests carried out, in the paragraph 3.2.

#### 3.1.1.7 Resume – Plots

To the user is given the choice of whether or not to enable the presentation of a resume and some plots at the end of the analysis. The resume describes the selected problem, the settings of the analysis and the characteristics of each island. The evolution of the feval with the increase of generations. At the end of the run it also deals with saving information about the best individual for each island and its corresponding feval. It also indicates how many times has been activated the mass mutation mechanism and the duration of the analysis. The plots that present the tool at the end of the analysis are the average function value depending on the number of function evaluations for each island of the archipelago.

### 3.1.2 Initialization and Run

After obtaining the data necessary to set the parameters, the tool can perform the phase of initialization of the populations. Then they will evolve following the steps indicated by the algorithm 5.

### 3.1.3 Output

The tool performs a single-parameter optimization. The expected output is therefore the lowest value of the objective function found during the analysis. This value corresponds to an individual, who will have dimension D of the problem and represents the best solution found to the implemented problem. The tool is able to show the optimum achieved for each island and the optimum of the archipelago.

**Algorithm 5** ISOLE Optimization Tool

**input** Read input parameters
  Configuration of plots and resume
  Problem data definition: D, lb, ub, weight and cost function
  Set End Conditions
  Global Neural Net parameters configuration
  Define island number
  **for** each island **do**
    –Archipelago settings definition–
    Define island population
    Define island algorithm
  **end for**
  Clonation configuration
  Mass Mutation configuration
  –**Initialization**–
  **for** each island **do**
    Initialization of random individuals
    Evaluation of the populations
  **end for**
  Global neural net initialization
  –**Run**–
  **while** End condition not satisfied **do**
    **for** each island **do**
      **if** ES Algorithm **then**
        –**Mutation**–
      **end if**
      –**Evolution**–
      New generation
      Domain boundary check
      Function evaluation
      Save generation and best island individual
      –**Selection**–
      Snapshot
      **if** Aggregation Degree exceeds set value **then**
        –**Mass Mutation**–
      **end if**
    **end for**
    Save best archipelago individual
    Train global neural net
    –**Clonation**–
  **end while**
  Plot and resume

## 3.2 Mass Mutation

For evolutionary algorithms it can be difficult to exit a local minimum when searching on the domain. If an algorithm were to settle in a local minimum, it would lead to the advance of the number of generations a concentration of individuals in the minimum zone. This does not help the exploration of the domain and in general, not being guaranteed that the minimum found is absolute, could prevent the tool to find a better solution. This condition, in which all the individuals approach the same point, can be translated from the mathematical point of view by analyzing the average value of the objective function of individuals with respect to the value of the best individual. Having indicated the value of the objective function with f_i, for each of the individuals then it is possible to define the Aggregation Degree R:

$$R = \frac{f_{\text{best}}}{\sum f_i} \tag{3.2}$$

When R exceeds a set value called RMAXDEG and the minimum value is stuck by more than IMAXSTUCK number of iterations then the Mass Mutation is activated. Only a couple of individuals are saved for the next generation, all others are deleted and replaced with new, randomly chosen over the entire domain. It is also possible to choose when R exceeds the set value within an island whether to reset the island or the whole archipelago. The idea that has given rise to the inclusion of this parameter is to re-initialize a large part of the population, so as to introduce new individuals who can continue the research and possibly escape the local minimum. One effect is the possibility of working with a smaller population, decreasing the computational cost[4].

To evaluate the utility of mass mutation, a particularly complex 2D function is used, defined for the occasion with 16 sub-domains function. It appears as shown in figure 3.2 and 3.3.

The analyzed 2D problem has an absolute minimum at [0, -5.26, -81,87], but there is also an annoying local minimum at [-5.14, 18.82, -74,91]. To test the effectiveness of mass mutation, several runs of the tool were performed using the DE1, ES and PSO algorithms with 30 individuals. The general settings are the same defined for the statistics analysis presented in Chapter 4. The analysis end condition is a fixed number of generations.

The table 3.4 and 3.5 show the results obtained with the statistical analysis for the simple algorithms. The activation of the Mass Mutation (MM) operator drastically increase the efficiency for the DE1 and PSO. It also increases efficiency for the ES, but less so than for the other two algorithms. In addition, convergence towards the global optimum is faster. With active MM the algorithms will have a greater chance of not getting stuck in a local minimum. While without the

Figure 3.2: 2D Function

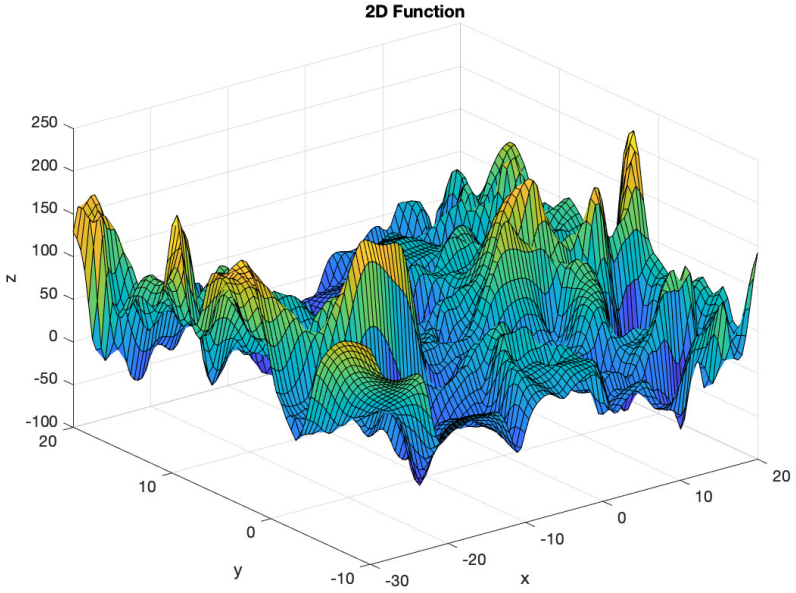



Figure 3.3: 2D Function xy plane

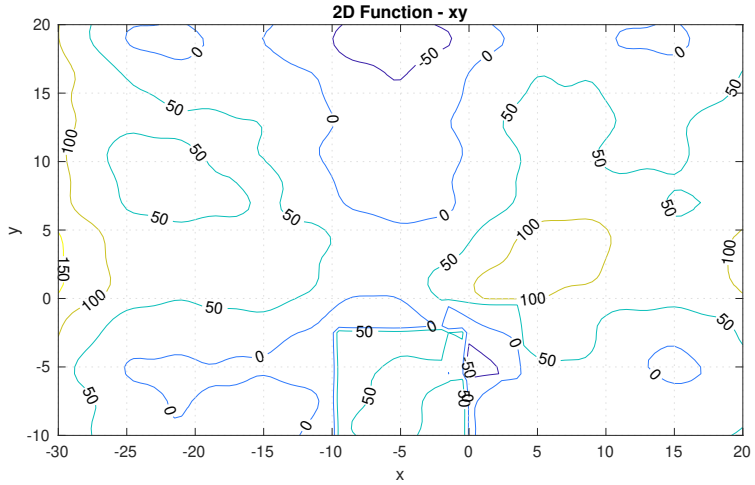


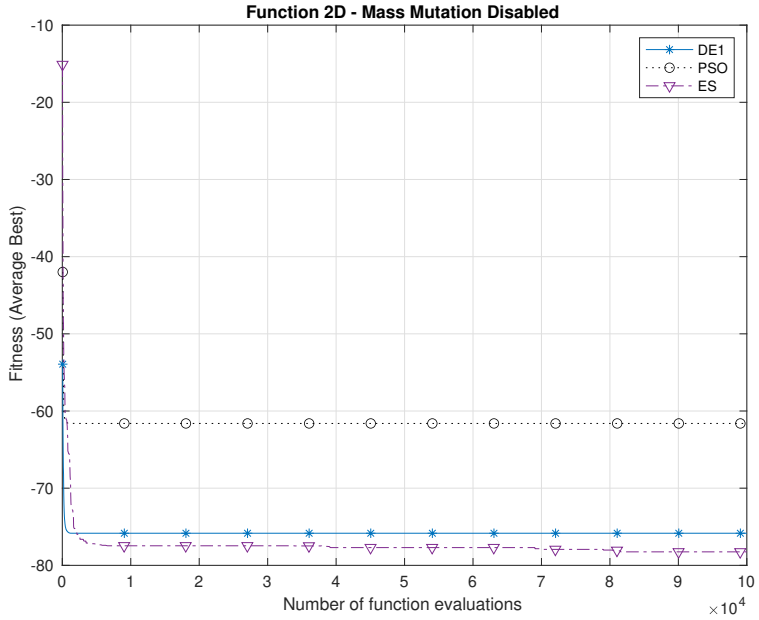
Table 3.1: Results for $F_{2D}$ without MM

| Method | Min value | Mean value | Std Dev V | Efficiency | NFEmin | NFE mean | STD Dev N |
|---|---|---|---|---|---|---|---|
| DE1 | -8.187854e+01 | -7.583791e+01 | 2.409835e+00 | 13.33 | 5070 | 2907 | 1037 |
| PSO | -8.187854e+01 | -6.161204e+01 | 1.842935e+01 | 26.67 | 21750 | 10808 | 4472 |
| ES | -8.187854e+01 | -7.825100e+01 | 3.452057e+00 | 43.33 | 25950 | 44303 | 25875 |

Table 3.2: Results for $F_{2D}$ with MM

| Method | Min value | Mean value | Std Dev V | Efficiency | NFEmin | NFE mean | STD Dev N |
|---|---|---|---|---|---|---|---|
| DE1 | -8.187854e+01 | -8.187724e+01 | 1.308262e-03 | 100 | 3450 | 51918 | 31920 |
| PSO | -8.187854e+01 | -7.978754e+01 | 3.248628e+00 | 70 | 11460 | 44056 | 31243 |
| ES | -8.187854e+01 | -7.860851e+01 | 3.420470e+00 | 50 | 30240 | 33557 | 25507 |

Figure 3.4: Function F2D - Average function values, Mass Mutation Disabled

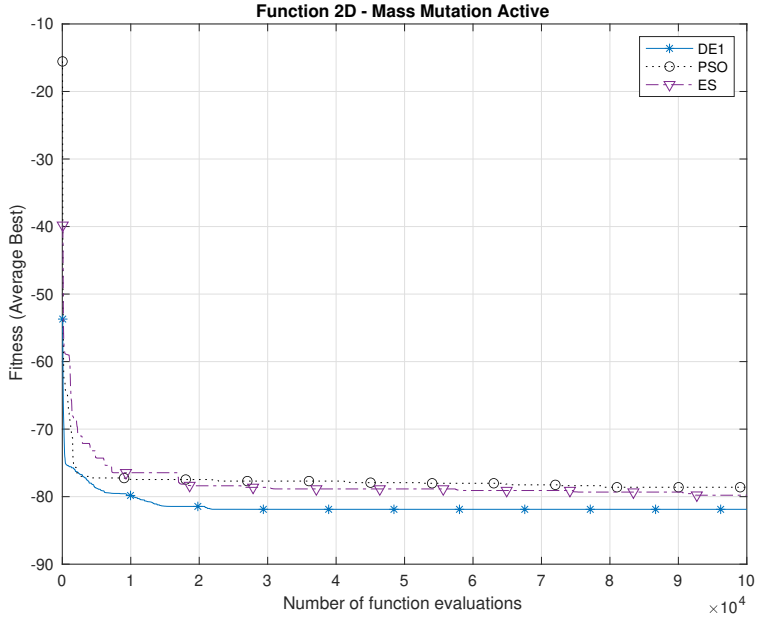



MM once stranded in a relative minimum it will be difficult to get out of it, with the risk that once the individuals are all concentrated in that point it becomes impossible to get out.

It is important to consider that the analyses were conducted with a maximum number of generations of $10^4$ and not with a maximum number of evaluations of $10^5$ as in the Chapter 4. This in order to demonstrate the increased efficiency and speed of convergence thanks to this introduced mechanism.

It is also interesting to look at the exploration of the domain. The use of the mechanism of Mass Mutation allows a greater exploration of the domain, as the

Figure 3.5: Function F2D - Average function values, Mass Mutation Active




two figures 3.6 and 3.7 visually demonstrate.

Figure 3.6: Function F2D - Domain Exploration, Mass Mutation Disabled

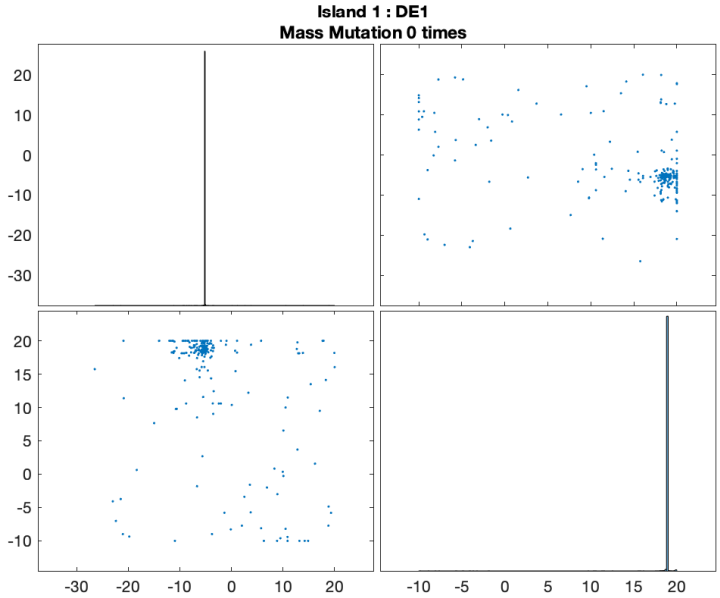
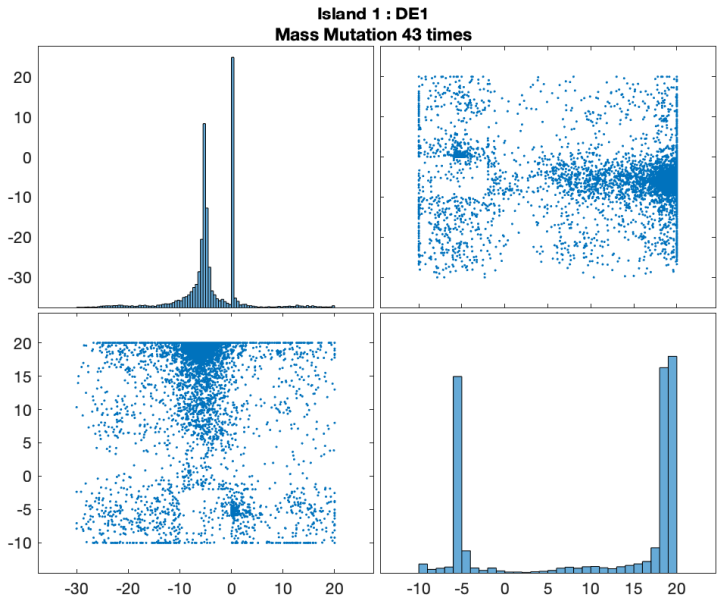


Figure 3.7: Function F2D - Domain Exploration, Mass Mutation Active



## 3.3 Clonation Topology

The information exchange process is done with an operator called Clonation, that is something similar to a migration operator for the basic multi-population algorithm. This thesis work will refer to this operation as migration or clonation considering them synonymous.

To test some of the different topologies studied have been implemented within the tool 3 cloning topologies with two selection mechanisms. The topologies are:

- Chain, where islands are organized in a sequence and the communication is allowed only between neighbor islands, figure 3.8(a);
- One-way Ring, where islands are organized in a ring, figure 3.8(b);
- Fully connected, where every island is connected to the others, figure 3.8(c).

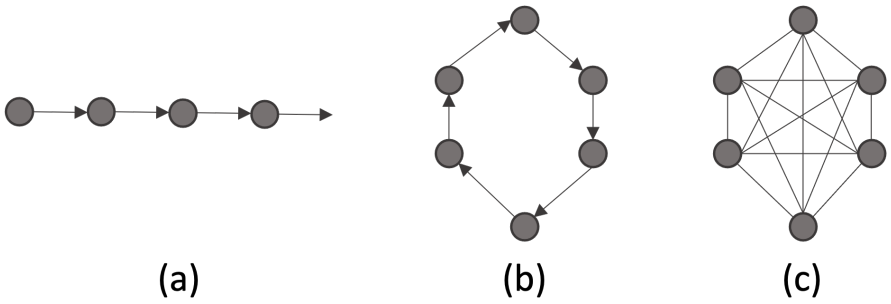



Figure 3.8: Clonation Topologies

The selection and insertion mechanisms are:

- Random: the selected individual and the replaced one are selected randomly;
- Elite: in the selection phase you choose the individual with the best feval. In the insertion phase you go to replace the individual with the worst feval.

To evaluate the potential offered by the selected cloning system, a statistical analysis was carried out on the $f_3$ function of the 4.1 table. With the general parameters presented in the paragraph 4.1. The test functions and the settings are better explained in the next chapter 4.

$$f_2(\vec{x}) = \sum_{i=0}^{n-1} \left( \sum_{j=0}^{i} x_i \right)^2 \tag{3.3}$$

Global minimum: 0

The analyses were carried out with 10 islands of 20 individuals for each one using the Differential Evolution DE1 algorithm. The cloning mechanism implemented is synchronous and the cloning frequency has been set to 20 generations.

The results obtained are shown in figure 3.9 and 3.10. The mechanism that proves to be better is the ring mechanism. While as expected ([7]) the fully connected performs less well than the other two solutions. The result is influenced by the characteristics of the algorithms and the frequency of cloning and it is not has and it's not a complete study like the one conducted by Rucinski, Izzo and Biscani. Probably 20 individuals represent a population too small to use a fully connected cloning type, the turnover of individuals is excessive and does not allow the algorithm to evolve as expected. The DE1 algorithm is the most exploratory among the algorithms analyzed, using only islands that implement exploratory algorithms is not the optimal method. Better results are expected by also modifying the topology of algorithms, going to analyze an archipelago composed of islands more dedicated to exploration and islands more dedicated to exploitation. The passage of the best individuals from the first islands to the second allows the former to continue exploring the domain and the latter to go deeper into the solutions found. The results obtained by the random selection mechanism were, as expected, worse than the results obtained by an elitist selection and substitution of individuals.

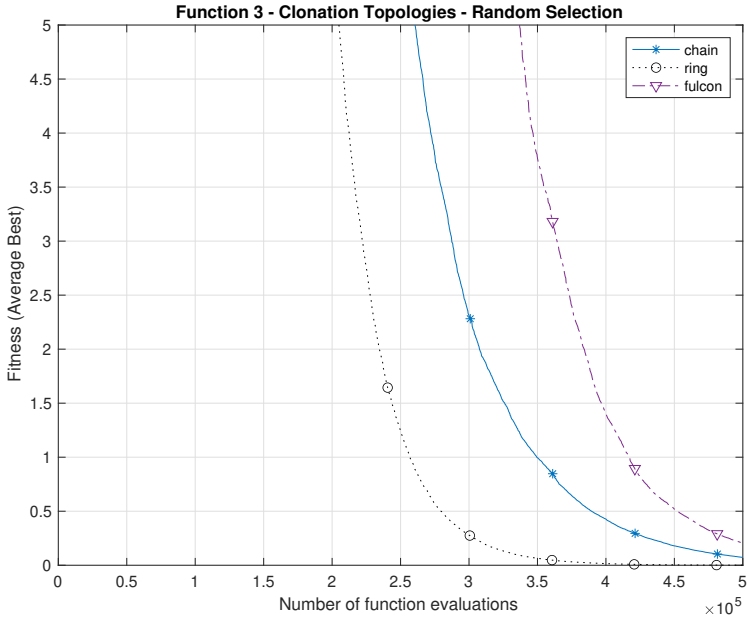



Figure 3.9: Function F3 - Average function values - Random Selection

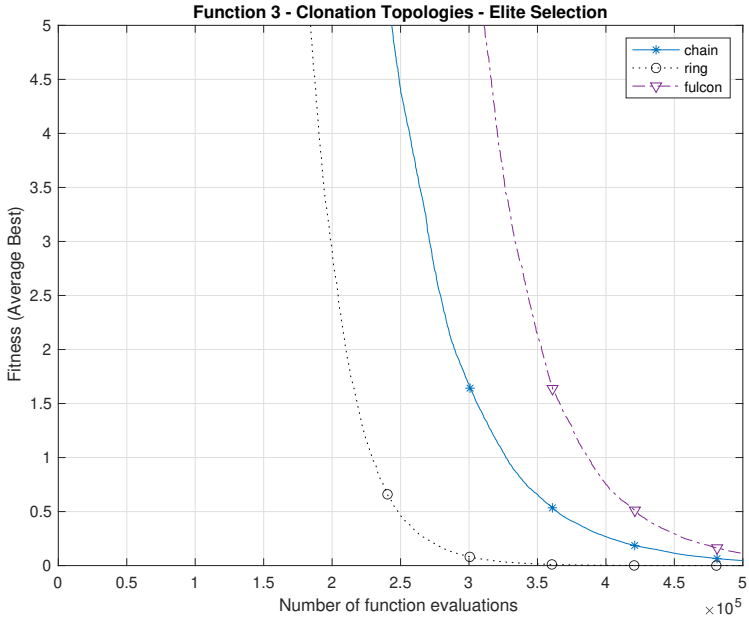



Figure 3.10: Function F3 - Average function values - Elite Selection

# Chapter 4

# Results on Numerical Benchmark Problems

By working with various optimization algorithms it is necessary to be able to evaluate their operation and their effectiveness. This is possible by comparing the results obtained on known problems treated in literature, in order to verify their performance. The inspiration for this work comes from two main sources: the work of J. Vesterstrom and R. Thomsen [30] and the doctoral work of Matteo Rosa Sentinella [4]. In Vesterstrom work the comparison was performed on a suite of 34 widely used benchmark problems. For this thesis work I chose 11 of these problems to perform the tests on the ISOLE tool. The objectives are to verify the performances of the algorithms with different types of functions, check performances and if the use of hybrid algorithms improves the effectiveness of the tool.

## 4.1 Functions and Experimental Settings

The comparison took place between the various algorithms implemented within the tool. The algorithms are DE, ES, PSO and their comnbinations: Hybrid DE-PSO, Hybrid DE-ES, Hybrid ES-PSO, Hybrid DE-ES-PSO. DE algorithm is implemented in four distinct versions: DE1, DE2, DE3, DE4. The versions differ for exploration and exploitation characteristics: DE1 is the more exploratory algorithm, DE4 the less exploratory one. They were initially designed for a hybrid mechanism in which the first islands face better the exploration, while the latter are designed to take advantage of the minimums encountered during research. The DE algorithms were already present in the ISOLE tool, so they are not extensive tested in the frame of this thesis. For the hybrid versions, algorithm DE1 is used. The settings used for the various algorithms are, depending on the case, those

obtained from previous development phases, those recommended in the literature or manually tuned parameters based on a few experiments.

Each algorithm has been tested on the functions presented in Table 4.1. An objective function evaluation limit of 500000 has been imposed for each test. Each experiment has been repeated for 40 times. The results presents the average suitability of the best solutions through optimization, the average number of function evaluations, their standard deviations and the efficiency of the optimizer quantified in the number of times the global minimum is reached in percentage. A solution with an absolute lower deviation than $10^{-6}$ with respect to the exact solution has been considered as exact.

Table 4.1: Numerical benchmark functions

| Function | Dim | Domain | Minimum Value |
|---|---|---|---|
| $f_1(\vec{x}) = \sum_{i=0}^{n-1} x_i^2$ | 30 | $-5.12 \leq x_i \leq 5.12$ | $f_1(\overrightarrow{0}) = 0$ |
| $f_2(\vec{x}) = \sum_{i=0}^{n-1} \left(\sum_{j=0}^{i} x_i\right)^2$ | 30 | $-100 \leq x_i \leq 100$ | $f_2(\overrightarrow{0}) = 0$ |
| $f_3(\vec{x}) = \max \lvert x_i \rvert, 0 \leq i \leq n$ | 30 | $-100 \leq x_i \leq 100$ | $f_3(\overrightarrow{0}) = 0$ |
| $f_4(\vec{x}) = \sum_{i=0}^{n-1} \left(100 \cdot \left(x_{i+1} - (x_i)^2\right)^2 + + (x_i - 1)^2\right)$ | 30 | $-30 \leq x_i \leq 30$ | $f_4(\overrightarrow{1}) = 0$ |
| $f_5(\vec{x}) = \sum_{i=0}^{n-1} \left(\lvert x_i + \frac{1}{2} \rvert\right)^2$ | 30 | $-100 \leq x_i \leq 100$ | $f_5(\vec{p}) = 0,$ $-\frac{1}{2} \leq p < \frac{1}{2}$ |
| $f_6(\vec{x}) = \sum_{i=0}^{n-1} -x_i \cdot \sin\left(\sqrt{\lvert x_i \rvert}\right)$ | 30 | $-500 \leq x_i \leq 500$ | $f_6(\overrightarrow{420.97})$ $= -12569.5$ |
| $f_7(\vec{x}) = \sum_{i=0}^{n-1} \left(x_i^2 - 10\cos\left(2\pi x_i\right) + 10\right)$ | 30 | $-5.12 \leq x_i \leq 5.12$ | $f_7(\overrightarrow{0}) = 0$ |
| $f_8(\vec{x}) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 +$ $+10\left(1 - \frac{1}{8\pi}\right)\cos\left(x_1\right) + 10$ | 2 | $-5 \leq x_i \leq 15$ | $f_8(9.42, 2.47)$ $= 0.398$ |
| $f_9(\vec{x}) = \left(1 - x_1\right)^2 + 100\left(x_2 - x_1^2\right)^2$ | 2 | $-2 \leq x_i \leq 2$ | $f_9(\overrightarrow{1}) = 0$ |
| $f_{10}(\vec{x}) = \frac{1}{4000}\sum_{i=0}^{n-1} x_i^2 + \left(\prod_{i=0}^{n-1} \cos\left(\frac{x_i}{\sqrt{i+1}}\right)\right) + 1$ | 30 | $-600 \leq x_i \leq 600$ | $f_{10}(\overrightarrow{0}) = 0$ |
| $f_{11}(\vec{x}) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$ | 2 | $-5 \leq x_i \leq 5$ | $f_{11}(-0.09, 0.71)$ $= -1.0316$ |

### 4.1.1 DE Settings

DE has three parameters: the crossover constant CR, the scaling factor F, and the size of the population NI. The values was set to: NI = 30, CR = 0.5:1, F = 0.1:1. CR and F were selected randomly within the interval for each individual. The generation of the individuals is random.

### 4.1.2 ES Settings

The ES parameters define the population size NI, the $\lambda$ factor, the $\frac{\lambda}{\mu}$ ratio, $\sigma$ and $\tau$. The choosen values are: NI = 30, $\frac{\lambda}{\mu} = 10$; $\sigma_{init}$ one-sixth of the interval in the search space, $\tau_{pi} = \frac{1}{(2*D)^{0.5}}$ and $\tau_{init} = \frac{1}{(2*(D^{0.5}))^{0.5}}$ using the problem dimension as D.

### 4.1.3 PSO Settings

PSO has different parameters: the maximum velocity Vmax, the number of particles in the swarm NI and the parameters for attraction towards personal best C1 and the neighborhoods best solutions C2. The settings used are: Vmax = 10% of the longest axis-parallel interval in the search space, NI = 30, C1 = 1.8, C2 = 1.8. The generation of the individuals is random.

### 4.1.4 Generic Settings

All individuals of the populations representing each island are generated by random generation. The cloning mechanism used is the fully connected mechanism with the exchange of a randomly selected individual every 10 iterations. For the mass-mutation operator, the 99% of the individuals are re-initialised when the aggregation degree R = 0.98 and the minimum value is stuck for more than 10 iterations.

## 4.2 Results

**Function $F_1$**

$$f_1(\vec{x}) = \sum_{i=0}^{n-1} x_i^2 \tag{4.1}$$

Global minimum: 0

Table 4.2: Results for $f_1$

| Method | Min value | Mean value | Std Dev V | Efficiency | NFEmin | NFE mean | STD Dev N |
|---|---|---|---|---|---|---|---|
| DE1 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 100 | 28340 | 42703 | 9867 |
| PSO | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 100 | 72060 | 82745 | 7565 |
| ES | 0.000000e+00 | 4.2695916e-05 | 1.1955398e-04 | 37.5 | 144030 | 290048 | 120124 |
| DE1 + PSO | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 100 | 16520 | 22011 | 3163 |
| DE1 + ES | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 100 | 35420 | 52397 | 10925 |
| PSO + ES | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 100 | 132370 | 163446 | 20082 |
| DE1 + PSO + ES | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 100 | 30910 | 40206 | 4956 |

Figure 4.1: Function F1 - Average function values, simple algorithms

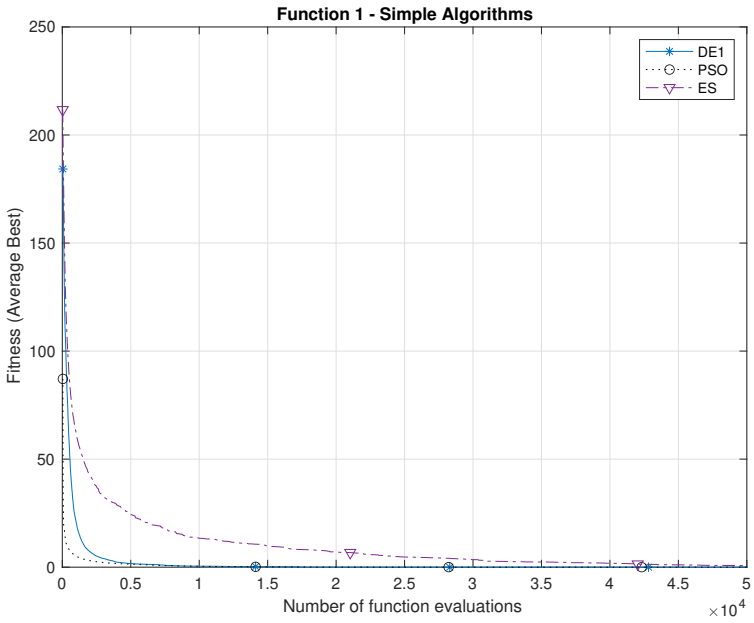
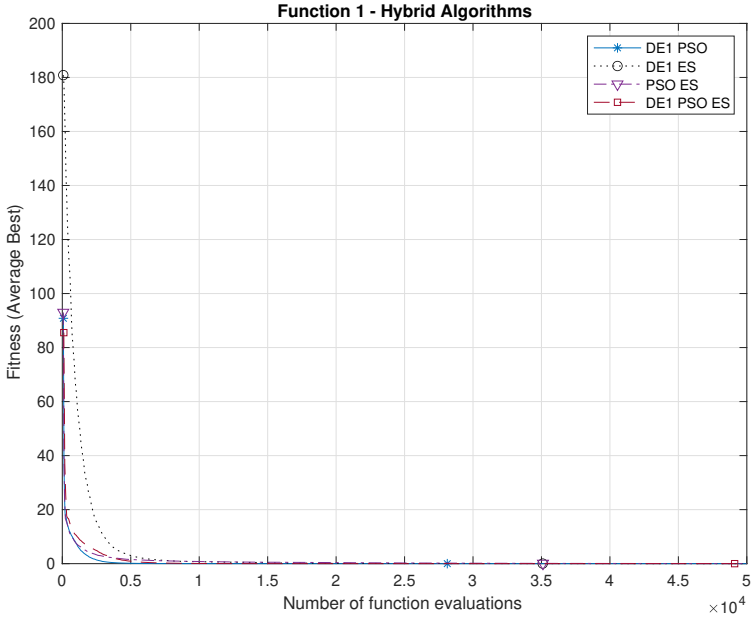



Figure 4.2: Function F1 - Average function values, hybrid algorithms



**Function $F_2$**

$$f_2(\vec{x}) = \sum_{i=0}^{n-1} \left( \sum_{j=0}^{i} x_i \right)^2 \tag{4.2}$$

Global minimum: 0

Table 4.3: Results for $f_2$

| Method | Min value | Mean value | Std Dev V | Efficiency | NFEmin | NFE mean | STD Dev N |
|---|---|---|---|---|---|---|---|
| DE1 | 2.561405e-02 | 2.956646e-01 | 1.950524e-01 | 0 | 499020 | 494034 | 16866 |
| PSO | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 100 | 361460 | 433411 | 27264 |
| ES | 4.268087e+02 | 2.028881e+03 | 1.487244e+03 | 0 | 411750 | 406929 | 111286 |
| DE1 + PSO | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 100 | 152340 | 172553 | 13711 |
| DE1 + ES | 1.303379e-06 | 9.518382e-03 | 3.104854e-02 | 0 | 499850 | 498898 | 4273 |
| ES + PSO | 4.292850e-06 | 1.488188e-04 | 2.073223e-04 | 0 | 499930 | 499979 | 50 |
| DE1 + PSO + ES | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 100 | 250280 | 308014 | 26220 |

Figure 4.3: Function F2 - Average function values, simple algorithms

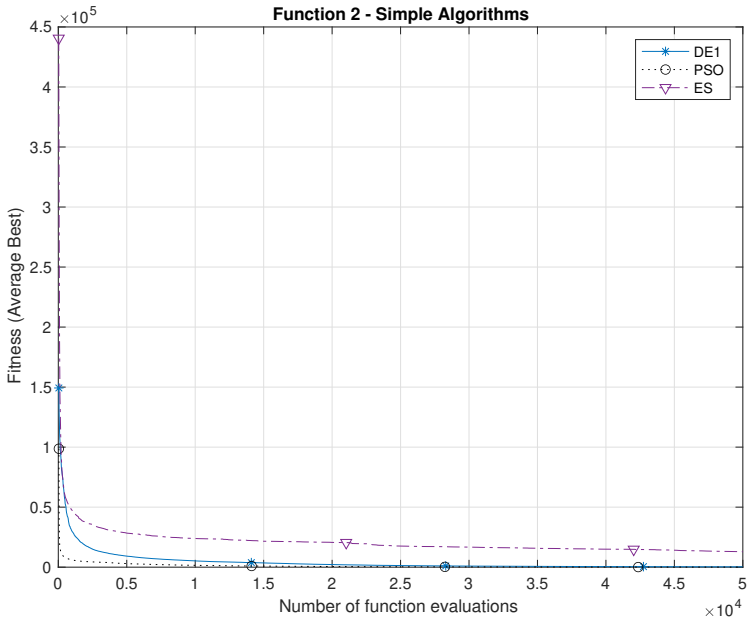
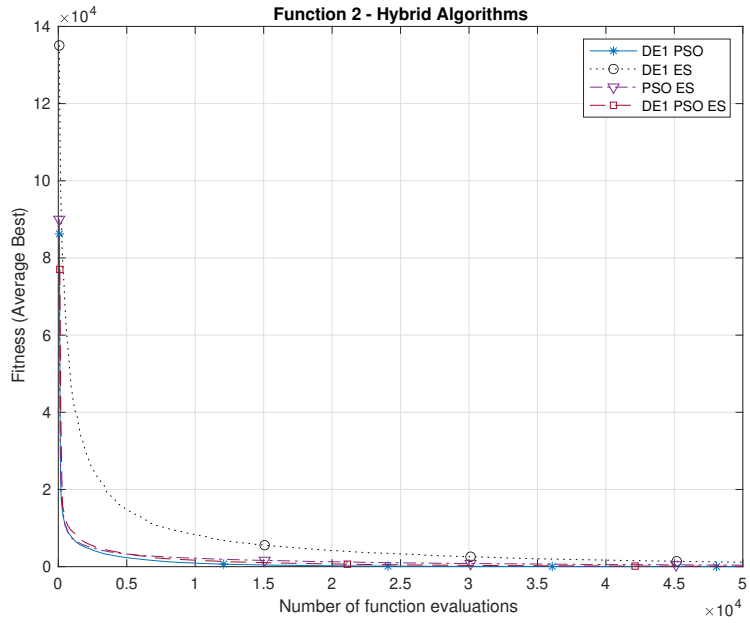



Figure 4.4: Function F2 - Average function values, hybrid algorithms



**Function $F_3$**

$$f_3(\vec{x}) = \max |x_i|, 0 \leq i \leq n \\ \text{Global minimum: } 0 \tag{4.3}$$

Table 4.4: Results for $f_3$

| Method | Min value | Mean value | Std Dev V | Efficiency | NFEmin | NFE mean | STD Dev N |
|---|---|---|---|---|---|---|---|
| DE1 | 3.087193e-02 | 6.820819e-02 | 2.019127e-02 | 0 | 497580 | 495283 | 4499 |
| PSO | 2.538844e-01 | 2.525510e+00 | 1.673269e+00 | 0 | 500000 | 499743 | 444 |
| ES | 1.500666e+01 | 2.631721e+01 | 7.545006e+00 | 0 | 468360 | 270474 | 145228 |
| DE1 + PSO | 2.597852e-03 | 1.160091e-02 | 7.067836e-03 | 0 | 499720 | 499827 | 302 |
| DE1 + ES | 1.106747e-02 | 7.144101e-02 | 6.267689e-02 | 0 | 499940 | 482638 | 55534 |
| ES + PSO | 1.067849e+00 | 2.441955e+00 | 5.648996e-01 | 0 | 500020 | 499946 | 107 |
| DE1 + PSO + ES | 1.705502e-02 | 7.945830e-02 | 2.920703e-02 | 0 | 499650 | 499971 | 102 |

Figure 4.5: Function F3 - Average function values, simple algorithms

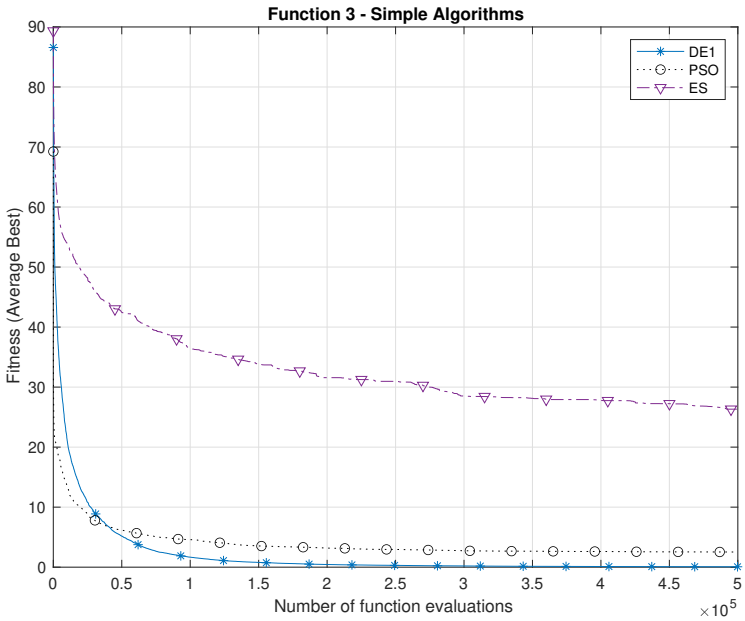
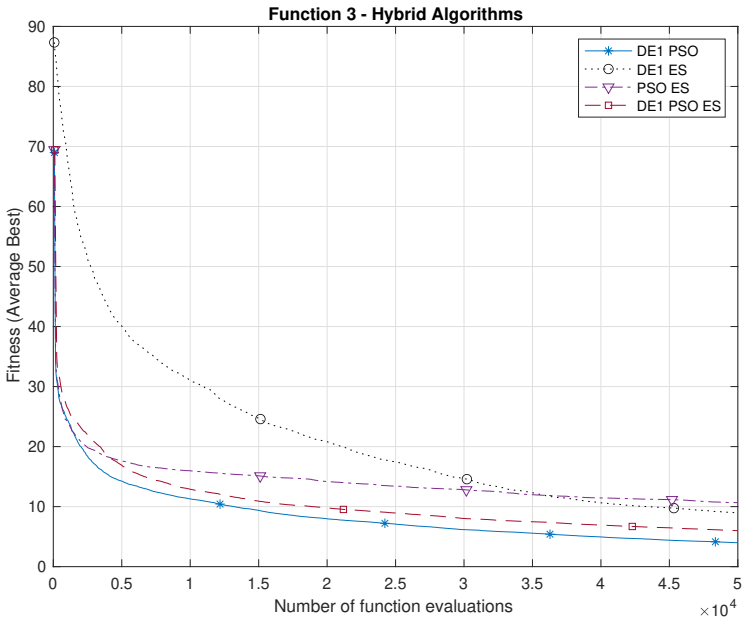



Figure 4.6: Function F3 - Average function values, hybrid algorithms



**Function** $F_4$

$$f_4(\vec{x}) = \sum_{i=0}^{n-1} \left(100 \cdot \left(x_{i+1} - (x_i)^2\right)^2 + (x_i - 1)^2\right) \tag{4.4}$$

Global minimum: 0

Table 4.5: Results for $f_4$

| Method | Min value | Mean value | Std Dev V | Efficiency | NFEmin | NFE mean | STD Dev N |
|---|---|---|---|---|---|---|---|
| DE1 | 2.021420e-03 | 4.422962e+01 | 3.464887e+01 | 0 | 499050 | 498814 | 1546 |
| PSO | 1.459481e+01 | 3.821823e+01 | 3.256155e+01 | 0 | 500010 | 499944 | 117 |
| ES | 1.447965e+01 | 2.128592e+02 | 1.886267e+02 | 0 | 404490 | 347887 | 116220 |
| DE1 + PSO | 7.310455e-03 | 3.448402e+01 | 2.736140e+01 | 0 | 500040 | 499932 | 131 |
| DE1 + ES | 1.487188e-01 | 5.663732e+01 | 5.622800e+01 | 0 | 500010 | 499417 | 2077 |
| ES + PSO | 1.089583e+01 | 3.847982e+01 | 2.586338e+01 | 0 | 500040 | 499933 | 179 |
| DE1 + PSO + ES | 1.005843e+01 | 4.046677e+01 | 2.745558e+01 | 0 | 500040 | 499879 | 206 |

Figure 4.7: Function F4 - Average function values, simple algorithms

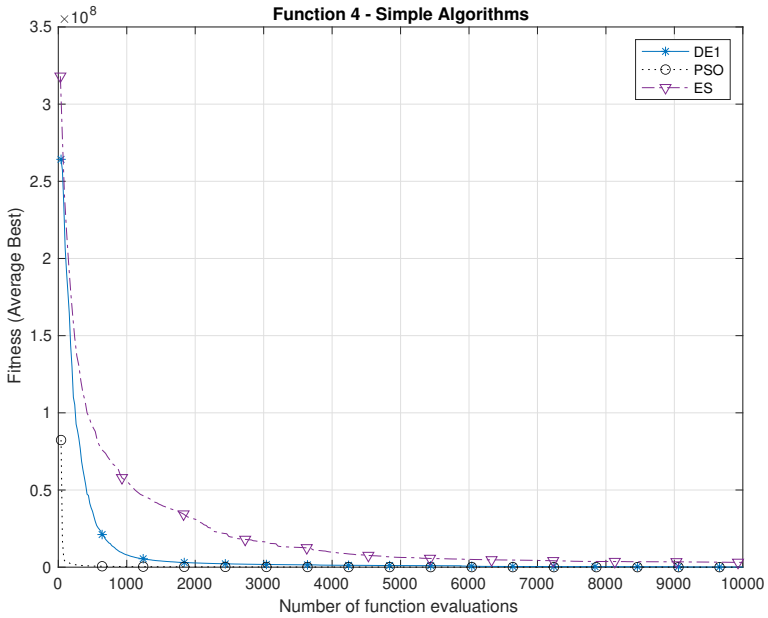
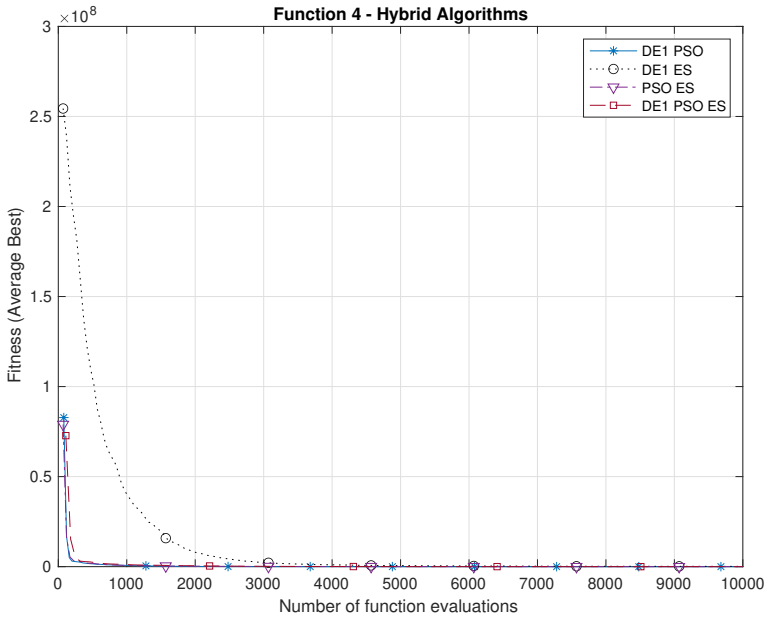



Figure 4.8: Function F4 - Average function values, hybrid algorithms



**Function $F_5$**

$$f_5(\vec{x}) = \sum_{i=0}^{n-1} \left( \left| x_i + \frac{1}{2} \right| \right)^2 \tag{4.5}$$

Global minimum: 0

Table 4.6: Results for $f_5$

| Method | Min value | Mean value | Std Dev V | Efficiency | NFEmin | NFE mean | STD Dev N |
|---|---|---|---|---|---|---|---|
| DE1 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 100 | 56880 | 119392 | 40297 |
| PSO | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 100 | 109960 | 124171 | 7629 |
| ES | 6.269644e-06 | 6.109726e-02 | 1.998486e-01 | 0 | 336180 | 343124 | 112708 |
| DE1 + PSO | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 100 | 27240 | 31490 | 2482 |
| DE1 + ES | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 100 | 32490 | 51966 | 17593 |
| ES + PSO | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 100 | 202260 | 255462 | 24775 |
| DE1 + PSO + ES | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 100 | 41730 | 50494 | 5837 |

Figure 4.9: Function F5 - Average function values, simple algorithms

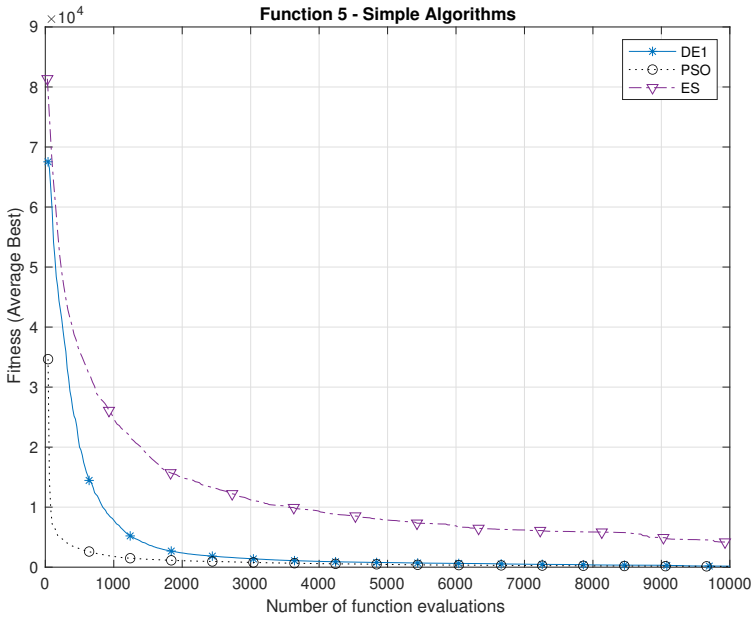
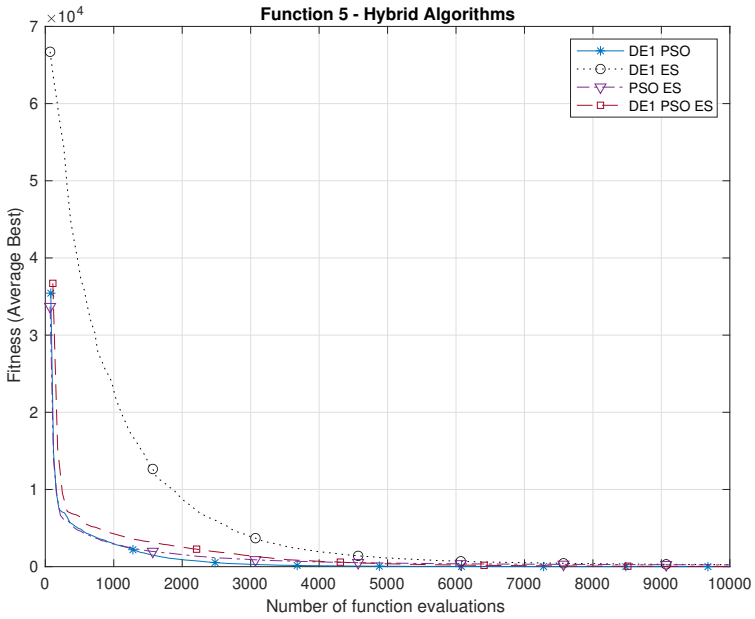



Figure 4.10: Function F5 - Average function values, hybrid algorithms



**Function $F_6$**

$$f_6(\vec{x}) \sum_{i=0}^{n-1} -x_i \cdot \sin\left(\sqrt{|x_i|}\right) \tag{4.6}$$
$$\text{Global minimum: } -1.25695e+04$$

Table 4.7: Results for $f_6$

| Method | Min value | Mean value | Std Dev V | Efficiency | NFEmin | NFE mean | STD Dev N |
|---|---|---|---|---|---|---|---|
| DE1 | -1.256945e+04 | -1.256929e+04 | 1.144972e-01 | 0 | 499360 | 490490 | 9423 |
| PSO | -7.317997e+03 | -5.212119e+03 | 9.298708e+02 | 0 | 498160 | 495151 | 6487 |
| ES | -1.030867e+04 | -9.632713e+03 | 4.237569e+02 | 0 | 316650 | 382224 | 129989 |
| DE1 + PSO | -1.256841e+04 | -1.218623e+04 | 2.077064e+02 | 0 | 499470 | 499690 | 541 |
| DE1 + ES | -1.256113e+04 | -1.235832e+04 | 1.382769e+02 | 0 | 449400 | 496966 | 9816 |
| ES + PSO | -1.104923e+04 | -1.025051e+04 | 4.509264e+02 | 0 | 500010 | 499818 | 377 |
| DE1 + PSO + ES | -1.245105e+04 | -1.199686e+04 | 2.548768e+02 | 0 | 499640 | 499699 | 656 |

Figure 4.11: Function F6 - Average function values, simple algorithms

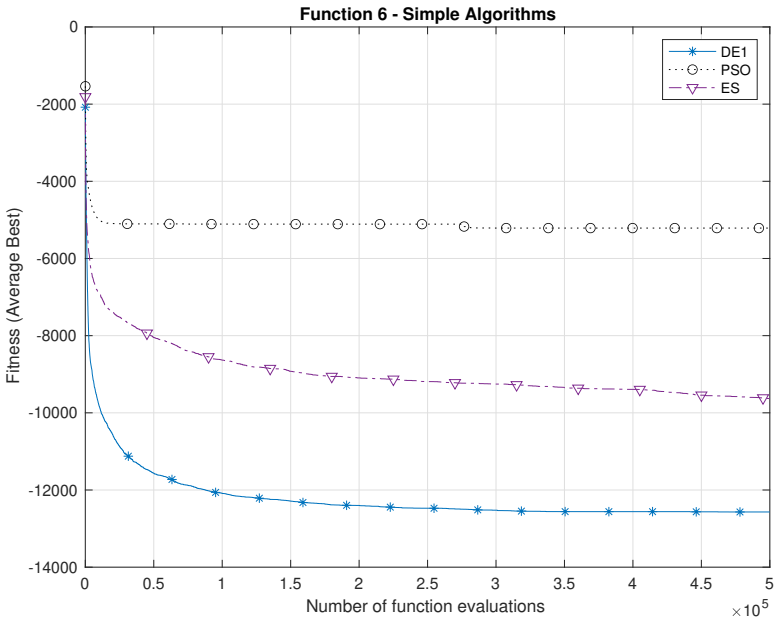
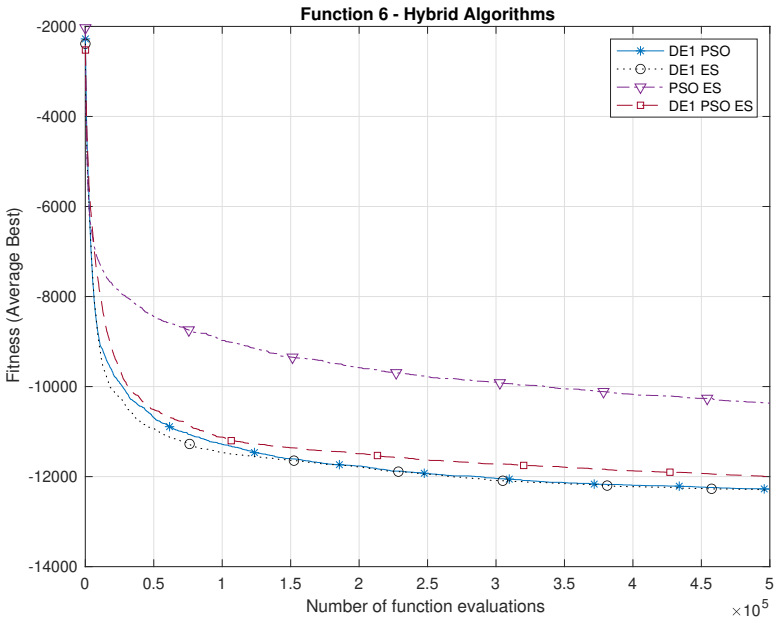



Figure 4.12: Function F6 - Average function values, hybrid algorithms



**Function $F_7$**

$$f_7(\vec{x}) = \sum_{i=0}^{n-1} \left(x_i^2 - 10\cos\left(2\pi x_i\right) + 10\right) \tag{4.7}$$

Global minimum: 0

Table 4.8: Results for $f_7$

| Method | Min value | Mean value | Std Dev V | Efficiency | NFEmin | NFE mean | STD Dev N |
|---|---|---|---|---|---|---|---|
| DE1 | 0.000000e+00 | 3.316621e-02 | 1.816577e-01 | 83.3 | 233320 | 371467 | 79988 |
| PSO | 2.487397e+01 | 3.946666e+01 | 1.236733e+01 | 0 | 499000 | 497732 | 2644 |
| ES | 1.990009e+00 | 7.851659e+00 | 2.122590e+00 | 0 | 425220 | 357416 | 103965 |
| DE1 + PSO | 5.430263e-04 | 2.677581e+00 | 1.502977e+00 | 0 | 500000 | 499777 | 365 |
| DE1 + ES | 0.000000e+00 | 1.107802e+00 | 1.016416e+00 | 23.3 | 362090 | 457330 | 51410 |
| ES + PSO | 9.953409e-01 | 4.941792e+00 | 1.837903e+00 | 0 | 499920 | 498955 | 3077 |
| DE1 + PSO + ES | 0.000000e+00 | 2.355046e+00 | 1.466199e+00 | 33.3 | 398750 | 495936 | 18378 |

Figure 4.13: Function F7 - Average function values, simple algorithms

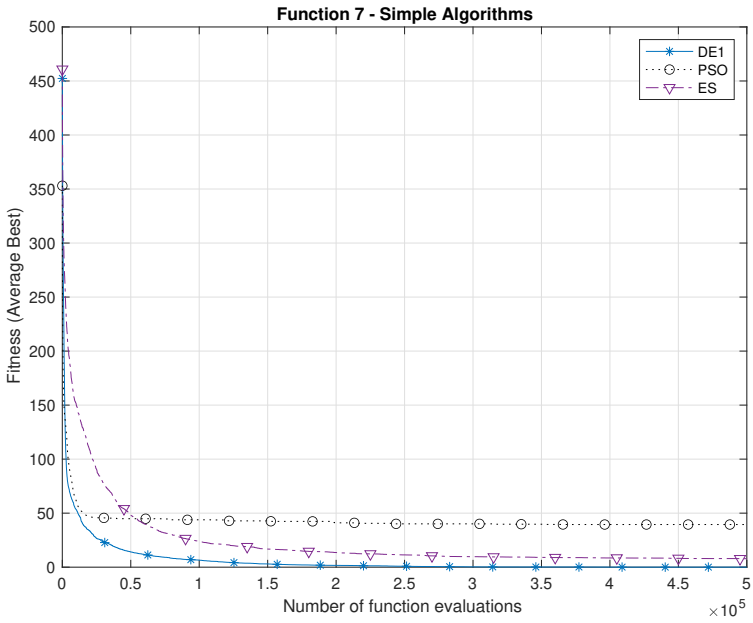
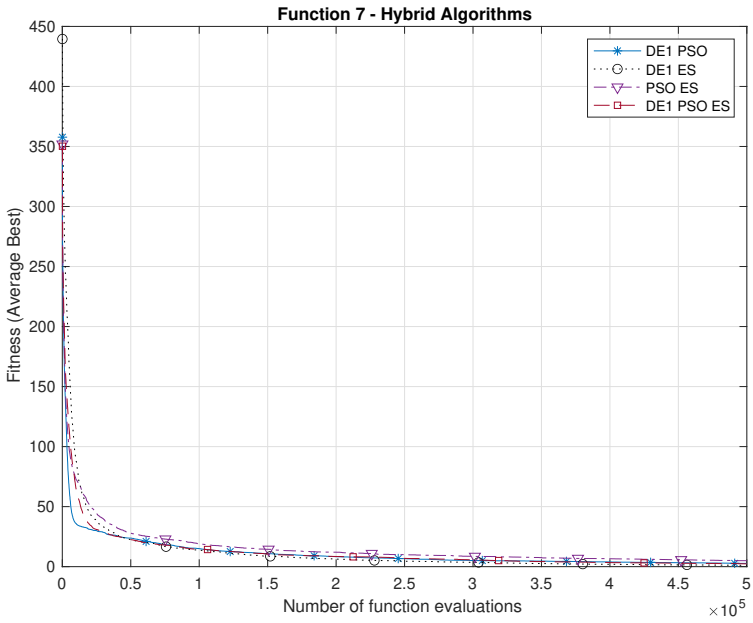



Figure 4.14: Function F7 - Average function values, hybrid algorithms



**Function $F_8$**

$$f_8(\vec{x}) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10 \tag{4.8}$$

Global minimum: 0.3978

Table 4.9: Results for $f_8$

| Method | Min value | Mean value | Std Dev V | Efficiency | NFEmin | NFE mean | STD Dev N |
|---|---|---|---|---|---|---|---|
| DE1 | 3.978874e-01 | 3.978874e-01 | 3.295327e-10 | 100 | 174330 | 314092 | 114354 |
| PSO | 3.978874e-01 | 3.978874e-01 | 0 | 100 | 5580 | 24054 | 27861 |
| ES | 3.978874e-01 | 3.978874e-01 | 0 | 100 | 2640 | 38335 | 32938 |
| DE + PSO | 3.978874e-01 | 3.978874e-01 | 1.201596e-09 | 100 | 10920 | 42761 | 55129 |
| DE + ES | 3.978874e-01 | 3.978874e-01 | 0 | 100 | 5310 | 30320 | 34515 |
| ES + PSO | 3.978874e-01 | 3.978874e-01 | 0 | 100 | 7380 | 14435 | 3713 |
| DE + PSO + ES | 3.978874e-01 | 3.978874e-01 | 0 | 100 | 12330 | 23505 | 9490 |

Figure 4.15: Function F8 - Average function values, simple algorithms

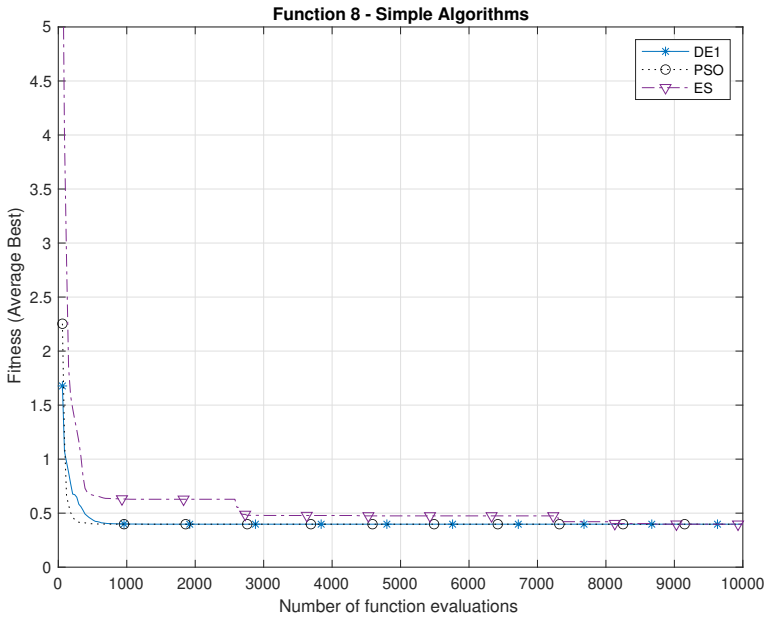
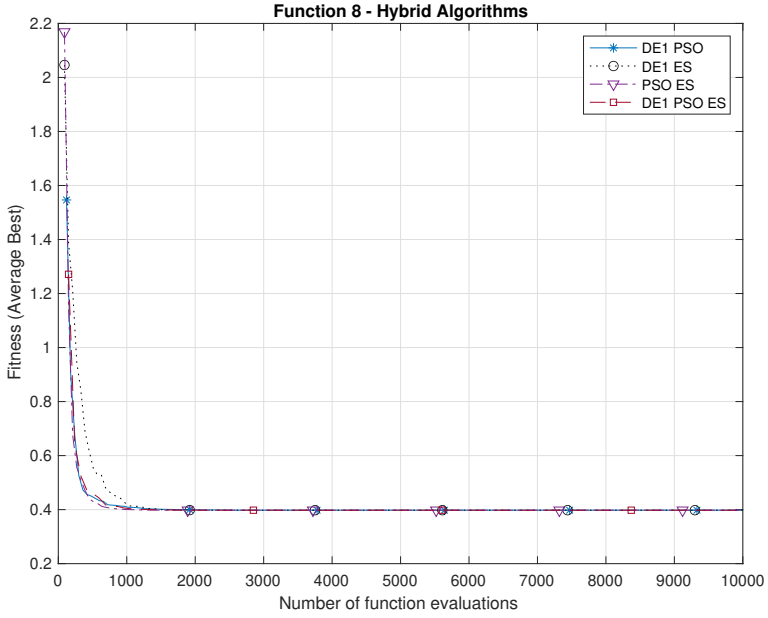



Figure 4.16: Function F8 - Average function values, hybrid algorithms



**Function $F_9$**

$$f_9(\vec{x}) = (1 - x_1)^2 + 100\left(x_2 - x_1^2\right)^2 \qquad (4.9)$$
$$\text{Global minimum: } 0$$

Table 4.10: Results for $f_9$

| Method | Min value | Mean value | Std Dev V | Efficiency | NFEmin | NFE mean | STD Dev N |
|---|---|---|---|---|---|---|---|
| DE1 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 100 | 930 | 1315 | 255 |
| PSO | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 100 | 990 | 1788 | 808 |
| ES | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 100 | 8670 | 109341 | 69188 |
| DE + PSO | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 100 | 1740 | 2524 | 386 |
| DE + ES | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 100 | 2010 | 2582 | 547 |
| ES + PSO | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 100 | 1830 | 2916 | 571 |
| DE + PSO + ES | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 100 | 2400 | 3405 | 434 |

Figure 4.17: Function F9 - Average function values, simple algorithms

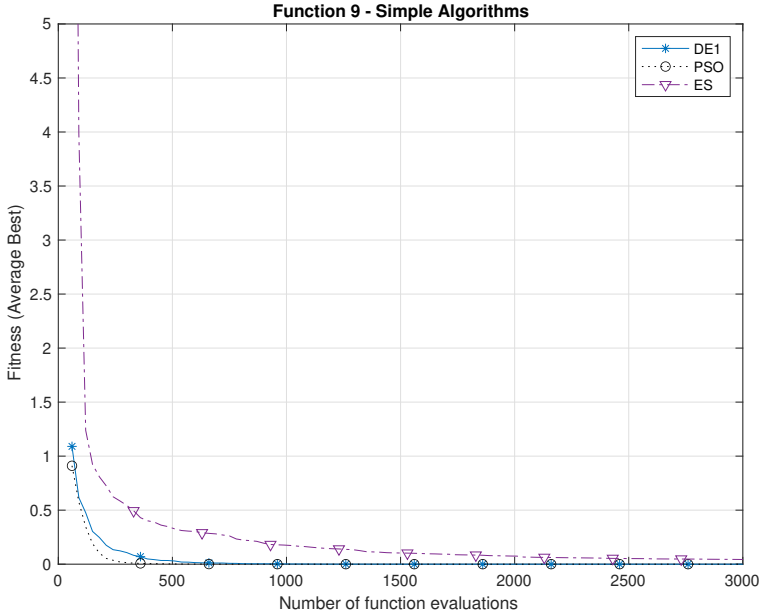
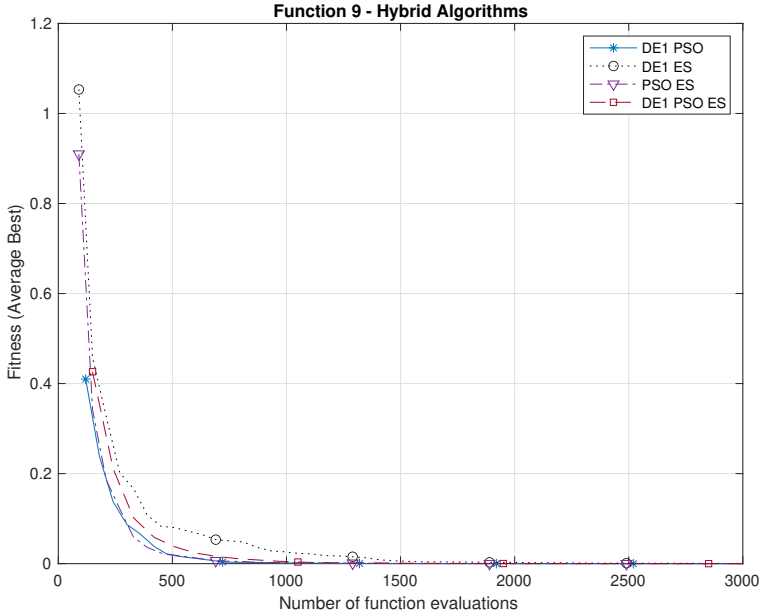



Figure 4.18: Function F9 - Average function values, hybrid algorithms



**Function** $F_{10}$

$$f_{10}(\vec{x}) = \frac{1}{4000}\sum_{i=0}^{n-1} x_i^2 + \left(\prod_{i=0}^{n-1}\cos\left(\frac{x_i}{\sqrt{i+1}}\right)\right) + 1 \tag{4.10}$$

Global minimum: 0

Table 4.11: Results for $f_{10}$

| Method | Min value | Mean value | Std Dev V | Efficiency | NFEmin | NFE mean | STD Dev N |
|---|---|---|---|---|---|---|---|
| DE1 | 2.466174e-03 | 3.552292e-02 | 3.274118e-02 | 0 | 489270 | 481826 | 19274 |
| PSO | 2.466168e-03 | 2.058058e-02 | 1.802784e-02 | 0 | 499440 | 498906 | 1376 |
| ES | 2.336941e-01 | 7.167426e-01 | 3.535200e-01 | 0 | 192570 | 281547 | 121662 |
| DE + PSO | 2.466168e-03 | 8.523999e-03 | 1.020188e-02 | 0 | 406650 | 461953 | 35864 |
| DE + ES | 2.466168e-03 | 1.944423e-02 | 2.582350e-02 | 0 | 488490 | 382009 | 80019 |
| ES + PSO | 2.466168e-03 | 1.564104e-02 | 1.455466e-02 | 0 | 495390 | 498609 | 1625 |
| DE + PSO + ES | 2.466168e-03 | 1.253275e-02 | 1.510965e-02 | 0 | 494880 | 498661 | 1679 |

Figure 4.19: Function F10 - Average function values, simple algorithms

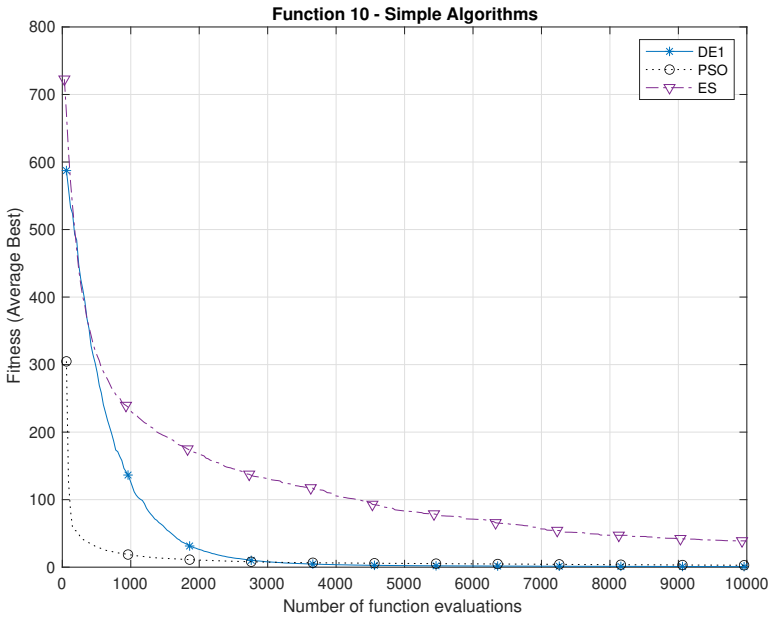
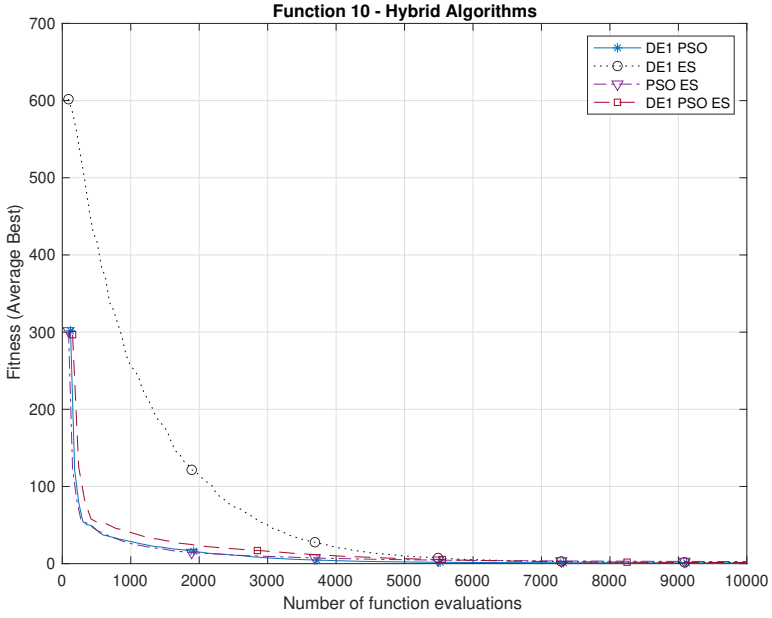



Figure 4.20: Function F10 - Average function values, hybrid algorithms



**Function** $F_{11}$

$$f_{11}(\vec{x}) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4 \qquad (4.11)$$
$$\text{Global minimum: } -1,03163$$

Table 4.12: Results for $f_{11}$

| Method | Min value | Mean value | Std Dev V | Efficiency | NFEmin | NFE mean | STD Dev N |
|---|---|---|---|---|---|---|---|
| DE1 | -1.031628e+00 | -1.031605e+00 | 2.668678e-05 | 67 | 337920 | 296486 | 154300 |
| PSO | -1.031628e+00 | -1.031628e+00 | 3.232900e-08 | 100 | 13380 | 110215 | 150003 |
| ES | -1.031628e+00 | -1.031628e+00 | 3.933344e-16 | 100 | 131850 | 203205 | 116062 |
| DE + PSO | -1.031628e+00 | -1.031628e+00 | 1.716618e-07 | 100 | 38880 | 99038 | 131187 |
| DE + ES | -1.031628e+00 | -1.031628e+00 | 4.896106e-16 | 100 | 21480 | 194260 | 145640 |
| ES + PSO | -1.031628e+00 | -1.031628e+00 | 5.531940e-16 | 100 | 60060 | 122468 | 152443 |
| DE + PSO + ES | -1.031628e+00 | -1.031628e+00 | 5.376080e-16 | 100 | 57630 | 132154 | 115696 |

Figure 4.21: Function F11 - Average function values, simple algorithms

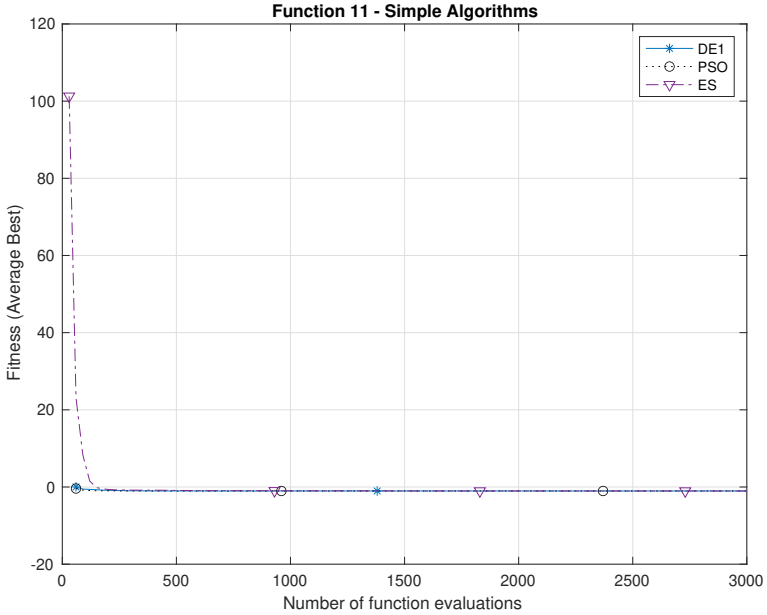
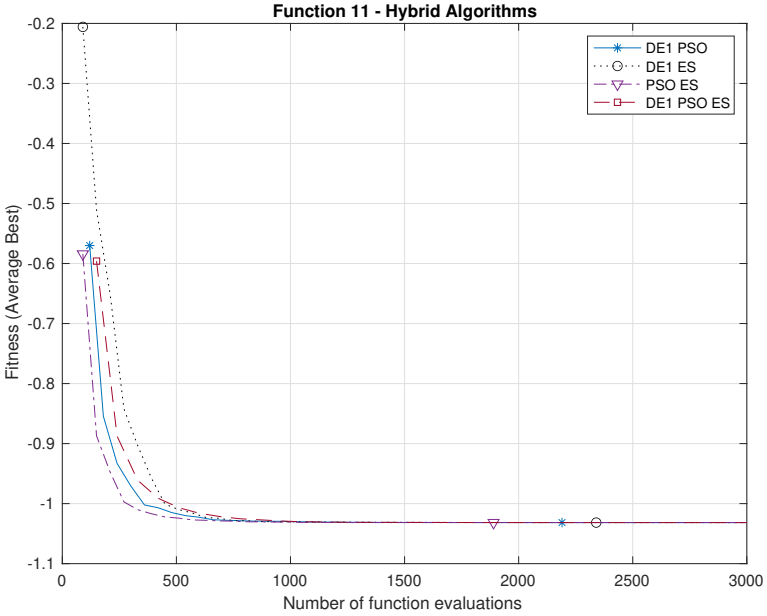



Figure 4.22: Function F11 - Average function values, hybrid algorithms



The results of the analysis performed on the F1-F11 benchmark problems are presented in the previous pages. For each function is shown the average function values for simple algorithms (DE,ES,PSO) and hybrid algorithms. For the first problem there is also a graph that illustrates the behavior of the different DE algorithms, as can be deduced the only effective DE for this type of analysis is the DE1, referred to below by the abbreviation DE. For function $F_1$ the ES algorithm behaves worse than the others and DE is faster than PSO. While all the hybrids converge faster than the single algorithms. For function $F_2$ instead the PSO is the only one that always finds the best solution, the DE get closer than the ES. The bad behavior of the ES affects the convergence of the hybrids, while where all 3 algorithms are present the help of DE and PSO allows the convergence. In functions $F_3$ and $F_4$ the best results are obtained from DE and PSO, their combination gives the best result, while the ES is the worst and slows convergence in hybrids. In function $F_5$ the ES behaves worse than the others, DE is faster than PSO. All the hybrids converge to global optimum and with fewer iterations. Also in function $F_6$, the ES slows down the convergence of hybrids and is the one that arrives farthest from the global optimal. DE finds better solutions than PSO on average. For function $F_7$ the worst is PSO. The best is DE, and in hybrids the PSO slows the convergence towards the global optimum more than the ES. In function $F_8$ the best are ES and PSO, the ES is the fastest and hybrids also converge to the global optimum. The function $F_9$ is the Rosenbrock's function and all algorithms have excellent convergence, but the simple ones are faster than the hybrids, except for the ES. In function $F_{10}$ DE and PSO are the best but no single algorithm or combination is able to find the optimum. In function $F_{11}$ the ES is the best, while DE the worst. DE does not always converge individually, while with hybrids the efficiency is increased and make it faster requiring less iterations. Functions $F_6$, $F_7$ and $F_10$ are highly multimodal functions.

The purpose of this work was not to propose an optimal tuning of the parameters, a more in-depth work can be done in this regard. Overall, the experimental results show that DE was far more efficient and robust compared to PSO and the ES that appears to be the worst tuned algorithm. However in some cases PSO and even ES perform better than DE. While in almost every cases the hybrid algorithms obtain better results than the simple ones. This results show that on average with hybrid algorithms is possible to obtain a better efficiency without knowing which is the best optimiser for the problem. Eventually the optimum will be achieved more slowly, with a small increase in computational cost.

# Chapter 5

# European Space Agency EL3 Application Problem

The primary purpose of this thesis is to further advance the development of a modular hybrid meta-heuristic optimization tool and then apply it for the ESA European Large Logistic Lander (EL3) RCS thrusters' orientation optimization problem. This study was conducted in Thales Alenia Space (TAS).

The tool has been tested on a series of benchmark functions and it will be applied on the EL3 thrusters' orientation to show if it suitable for these types of applications. The study conducted on the orientation of EL3 RCS thrusters required the development of an approximate solution method. It was then possible to apply the developed tool into the studying of 2 configurations, 4 and 8 thrusters, and different combinations of degrees of freedom for the orientation, with the aim of obtaining an acceptable torque margin in the lander control.

The project is passing through a study phase, to conduct the analysis necessary for this thesis were used the data available in October 2021. The work done is part of the EL3 descent and landing GNC study and is organized as follows:

- Section 5.1 presents the mission and the geometry of the problem;
- Section 5.2 describes the associated algorithm and the results.

## 5.1 EL3 Mission

The purpose of the European Large Logistic Lander (EL3) mission is to support human and robotic exploration by providing a multipurpose lander, the Lunar Descent Element, capable of carrying different types of cargo through the use of

specific interfaces for the mission, nominally the Cargo Platform Element. The mission will be launched from the Kourou site in French Guiana with an Ariane 64. With different types of missions available including cargo, science rover, sample return stage, technology demonstration packages, in-situ resources, power generation equipment and more [31].

The EL3 Space System consists of the following elements:

- Lunar Descent Element (LDE):
  - It is recurring lander, common to all EL3 missions;
  - It is designed to be compatible with different delivery missions without undergoing modifications;
  - The LDE provides the functions of Payload transport to the Moon's surface as well as the main mission functions common to all the different missions.
- Cargo Platform Element (CPE):
  - The mission-specific element that provides a standard interface between the modular LDE and different Payloads;
  - This element is intended to accomplish the functions proper to specific missions allowing for a better design of both the LDE and CPE.
- Payload:
  - It is intended to be either a Lunar Rover for south pole exploration or a Cargo Delivery mission.

The three elements are the constituents of the Robotic Landing Stack (RLS), which is the generic name for an EL3 composite, a render is shown if figure 5.1.

In this stage of development of the project the physical architecture is presented in figure 5.2 that let identify the elements that make up the LDE layout. The propulsion system is analyzed with characteristics that can be summarized in this way:

- A cluster of three 7 kN Main Engines, to meet thrust level requested for EL3 lunar descent maneuver and to guarantee a 1FT (failure tolerance) configuration;
- In an independent propulsion system, 4 (main) + 4 (redundant) RCS (Reaction Control System) bipropellant thrusters (400N thrust level) and 8 (main) + 8 (redundant) FCS bipropellant thrusters (10N thrust level for fine maneuvers and rolling control) shall be implemented;

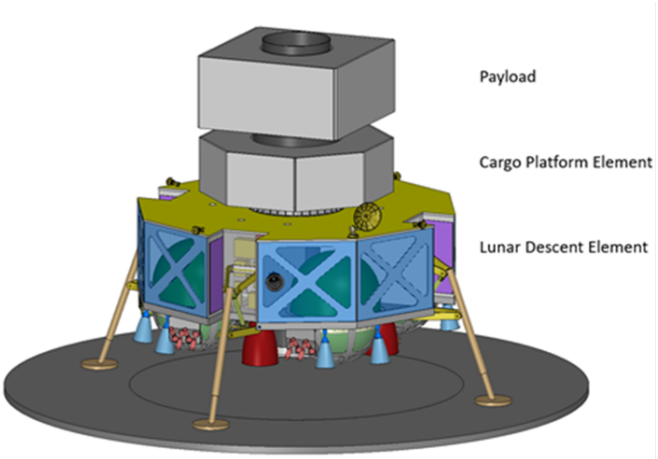



Figure 5.1: Robotic Landing Stack overview

- RCS and GNC thrusters shall operate in pressure regulated mode;
- Propulsion system shall be 1 Failure Tolerance (1FT), including Main Engine cluster.

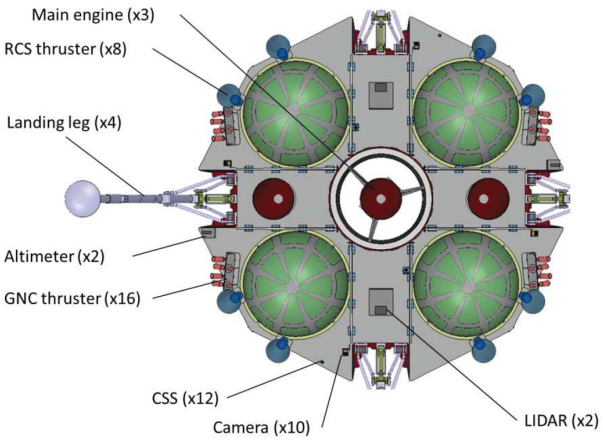



Figure 5.2: EL3 LDE bottom view

The Lunar Descent Element reference frame is described in figure 5.3.

### 5.1.1 Reaction Control System

The Reaction Control System (RCS), is able to provide small amounts of thrust in any direction or combination of directions desired, to allow control of rotation

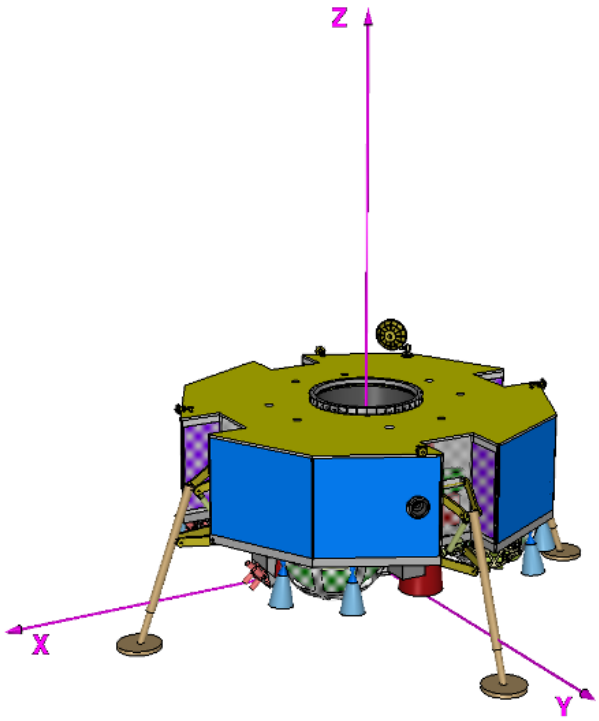



Figure 5.3: LDE S/C reference frame

(rolling, pitch and yaw). The Reaction Control System is first intended to compensate the disturbances and control the S/C attitude during the maneuvers with the Main Engines (ME). These disturbances are mostly associated with the ME, the S/C CoG dispersions and the sloshing.

At this early phase of the project, the control torque available should exceed the worst case disturbances by a margin of 40%. Following the adoption of a simplified solution method, this requirement will subsequently be expressed as a torque margin for the system. The sections below present the ME subsystem, the RCS thrusters data, the FCS data and the data used for the Monte Carlo campaigns all based on pre-phase A data.

#### 5.1.1.1 Main Engines Disturbance Torques

The Main propulsion system is composed of three 7 kN engines with position and orientation as defined in table 5.1.

The disturbance torque resulting from the activation of the engines is analysed with a Monte Carlo campaign of 5000 runs by considering the uncertainties reported in talbe 5.2. The data concerning the centre of gravity consider the con-

Table 5.1: Main Engines Position and Orientation

| **Main Engine** | **Position [mm]** | | | **Orientation** | | |
|---|---|---|---|---|---|---|
| | $X_{SC}$ | $Y_{SC}$ | $Z_{SC}$ | $X_{SC}$ | $Y_{SC}$ | $Z_{SC}$ |
| **1** | -1250 | 0 | 830 | 0 | 0 | -1 |
| **2** | 0 | 0 | 830 | 0 | 0 | -1 |
| **3** | 1250 | 0 | 830 | 0 | 0 | -1 |

figuration (LDE+CPE+P/L). The uncertain range for ME thrust is set at 5% of the nominal thrust value.

Table 5.2: Monte Carlo Campaign Data

| | | **Nominal value** | **Uncertain range (uniform distribution)** |
|---|---|---|---|
| **CoG position BOL [mm]** | $X_{SC}$ | 0 | [-50, 50] |
| | $Y_{SC}$ | 0 | [-50, 50] |
| | $Z_{SC}$ | 965 | [-500, 500] |
| **CoG position EOL [mm]** | $X_{SC}$ | -1250 | [-50, 50] |
| | $Y_{SC}$ | 0 | [-50, 50] |
| | $Z_{SC}$ | 1972 | [-500, 500] |
| **Main engines position [mm]** | $X_{SC}$ | As reported in Table5.1 | [-5, 5] |
| | $Y_{SC}$ | | [-5, 5] |
| | $Z_{SC}$ | | [-5, 5] |
| **Thrust [N]** | | 7000 | [-350, 350] |

The components of the disturbance torques produced by the EM and resulting from the Monte Carlo campaign are shown in the graphs 5.4 for the case Begin of Life (BOL) and 5.5 for the case End of Life (EOL). Both are analysed with a thrust of 7 kN for each engine. This value will then be reduced during the analysis

to reach an acceptable torque margin.

Figure 5.4: Distribution of the components of the disturbance torque at BOL

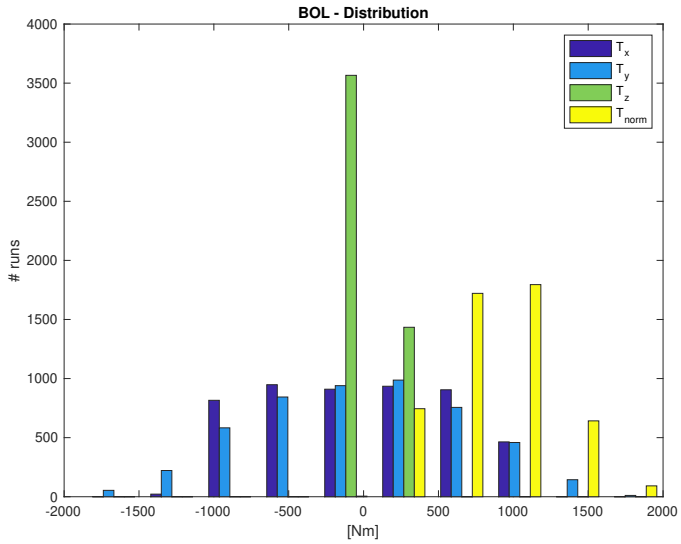
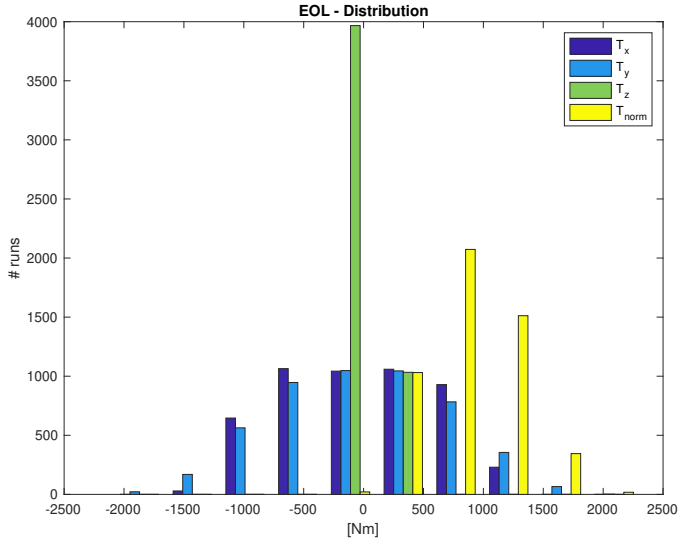



Figure 5.5: Distribution of the components of the disturbance torque at EOL



#### 5.1.1.2 Reaction Control System Thrusters

The baseline Reaction Control system is composed of four 400 N thrusters, and a redundant branch of four 400 N thrusters with positions presented in table 5.3.

In order for the RCS to be able to compensate the disturbance torques, the thruster of the RCS must be able to generate a torque higher than the disturbance torques resulting from the main engine, with a given margin. The analyses are carried out using either the 4 main thrusters or using 8 total thrusters considering the 4 redundants as if they were in the nominal branch. In this case it will be necessary to consider the presence of a 8 thrusters redundant branch for failure tolerance. In addition the analysis consider different levels of thrust produced by the Main Engines until an acceptable torque margin is reached.

Table 5.3: Reaction Control System Thrusters

| | **Position [mm]** | | |
|---|---|---|---|
| | **XSC** | **YSC** | **ZSC** |
| **Nominal Branch** | | | |
| **1a** | -1712 | -1168 | 504 |
| **2a** | -1712 | 1168 | 504 |
| **3a** | 1712 | 1168 | 504 |
| **4a** | 1712 | -1168 | 504 |
| **Redundant Branch** | | | |
| **1b** | -1168 | -1712 | 504 |
| **2b** | -1168 | 1712 | 504 |
| **3b** | 1168 | 1712 | 504 |
| **4b** | 1168 | -1712 | 504 |

#### 5.1.1.3 Fine Control System Thrusters

The Fine Reaction Control system is composed by a baseline of eight 10N thrusters, and a redundant branch of eight 10N thrusters. The FCS system is not the subject of this study.

## 5.2 RCS Orientation Analysis

The tool object of this thesis has been evaluated through the analysis of different test functions. Each with some peculiarities and with variable problem sizes between 2 and 30. The results served to show the strengths and weaknesses of the algorithms used and their combination through the use of an island model with a clonation mechanism between the various islands for each generation.

A further aim of the thesis was to evaluate the ISOLE tool on a real problem applying it for the ESA European Large Logistic Lander (EL3) RCS thrusters' orientation optimization problem. The work done is part of the EL3 descent and landing GNC study during this study phase.

### 5.2.1 Objective function

In most real applications of evolutionary algorithms, the computational complexity due to fitness function evaluation is a prohibiting factor, since an evaluation must be made for each individual for each generation. For this reason, a solution to the problem must often be proposed through an approximate solution. Hereafter, the exact problem and the associated approximate solution for the case of RCS orientation analysis are presented.

On board the spacecraft, the RCS system is activated as a result of a calculation performed using the resolution of the linear system 5.1. The unknown value is represented by the ignition times of the thrusters ($t_{on}$), which can act within the time window represented by the $T_{ctrl}$. The torque produced by the thrusters of the RCS system ($RCS_T$) must be able to compensate the torque produced by the Main Engines ($ME\_DIST\_T$).

$$RCS_T \cdot t_{on} = ME\_DIST\_T \cdot T_{ctrl} \tag{5.1}$$

Instead of solving the linear system to find an exact solution, that is computationally very expensive, it is possible to verify whether a positive margin with respect to $ME\_DIST\_T$ exists or not.

This approximate solution becomes the cost function to be implemented in the tool. The problem to solve is to be able to consider the presence of a torque margin in the direction of the disturbance. If one of the components of the disturbance exceed the torque potential in that direction than it is not possible to ensure a margin for control.

The Objective function is then structured as the Algorithm 6, within which is implemented the function that directly deals with calculating the torque margin in the considered direction (Algorithm 7).

The idea behind the solution model presented with Algorithm 6 and Algorithm 7 is as follow:

- A Monte Carlo campaign (MC) with data from table 5.2 is run, 5000 runs are included. The MC campaign will result in the identification of many directions around which the ME disturbance torque can act.

- For each thruster of the RCS system the orientation is defined given the 2 angles considered in the analysis. The analysis is conducted with a total number of RCS thrusters ranging from 2 to 8. Is the computed the RCS matrix.

- For each identified direction, the maximum RCS torque capability is evaluated. The search for the maximum torque is performed in two steps with Algorithm 7:

  1 The maximum torque along the ME disturbance is searched in the xy plane;

  2 In order to find the third component, a similar search is performed in the plane perpendicular to the xy direction.

  The computation of the maximum torque achievable in a certain direction, given the disturbance torques generated by the MEs, is obtained by considering all possible combinations of positive and negative RCS torques with coefficients variable between 0 and 1.

- Then, the difference between the maximum RCS torque and the ME disturbance torque is computed. The figure of merit used to assess the validity of the RCS configuration (i.e. define mounting angles of each thruster) is the minimum among all these torque differences. If the minimum value of this difference is above zero, then the considered orientation provides a positive margin between RCS torque and ME torque even in the worst case, and therefore is acceptable. If the minimum value is negative, it means that a direction exist, around which the disturbance torque produced by the ME is bigger than the one of the RCS. Therefore, that orientation give a negative margin and it is not valid.

**Algorithm 6** Objective function

**input** RCS parameters
**input** ME parameters
**input** CoG parameters
**input** Set Monte Carlo runs: $MCn$
  **for** $i = 1$ to $MCn$ **do**
    Set random CoG position
    **for** each $ME$ **do**
      Set random ME position and direction error
      Compute ME disturbance torque: $ME\_DIST\_T$
    **end for**
  **end for**
**input** RCS orientation Compute RCS matrix
  **for** $i = 1$ to $MCn$ **do**
    **for** each RCS Thrusters **do**
      Compute torque matrix: $RCS_T$
    **end for**
    Compute max RCS torque opposite to ME disturbance torque(Algorithm 7)
    Compute torque margin: max RCS - ME_DIST_T
    Store minimum margin
  **end for**

**Algorithm 7** Compute Max RCS torque opposite to ME

**input** $RCS_T$, ME_DIST_T
  **for** each axis: x,y,z **do**
    Compute torques available from mounting matrix
  **end for**
  Compute disturbance direction
  Compute max torque in xy plane
  **for** each quadrant **do**
    check which is the limiting component
    compute max torque with limited components
  **end for**
  Add z component
  Compute max torque for provided Me direction

### 5.2.2 RCS Configurations and Results

The objective of the analysis is to find the mounting angles that maximize the torque margin with respect to disturbances. The design statement originally required that the margin should exceed the worst case disturbances by a margin of 40%, this requirement is translated into the request for a margin of 400 Nm. Given that, various analyses are carried out, for each configuration, by reducing the thrust provided by the main engines up to the thrust that allows to obtain the required positive margin. The analyses are carried out for the two configurations of Begin Of Life (BOL) and End Of Life (EOL) of the centre of gravity.

For each configuration it will be indicated how the angles are measured. The reference system for EL3 has been presented in the figure 5.3. The analyzed configurations are:

- 4 Thrusters with a total of 2 DOF: two angles are used to define the thrust direction of all the four thrusters;
- 4 Thrusters with a total of 8 DOF: two angles are used to define the thrust direction for each of the four thrusters;
- 8 Thrusters with a total of 2 DOF: two angles are used to define the thrust direction of all the eight thrusters;
- 8 Thrusters with a total of 3 DOF: three angles are used to define the thrust direction the thrusters.

#### 5.2.2.1 4 Thrusters 2 DOF

The first configuration analyzed involves the use of the 4 main Thrusters with 2 degrees of freedom, the $\psi$ angle and the $\theta$ angle. The analysis includes four cases. The first considers the current situation with three 7 kN MEs, and then the other configurations have been studied to obtain an adequate torque positive margin reducing the thrust of the main engines.

The angles are measured as shown in Figure 5.6. The angle $\psi$ is measured as shown in the figure for the third thruster. Optimization is carried out with the use of a single value of $\psi$, with which the thrust directions of the four thrusters are obtained through the following transformation:

$$\begin{aligned} \psi_1 &= 180° + \psi \\ \psi_2 &= 180° - \psi \\ \psi_3 &= \psi \\ \psi_4 &= -\psi \end{aligned} \tag{5.2}$$

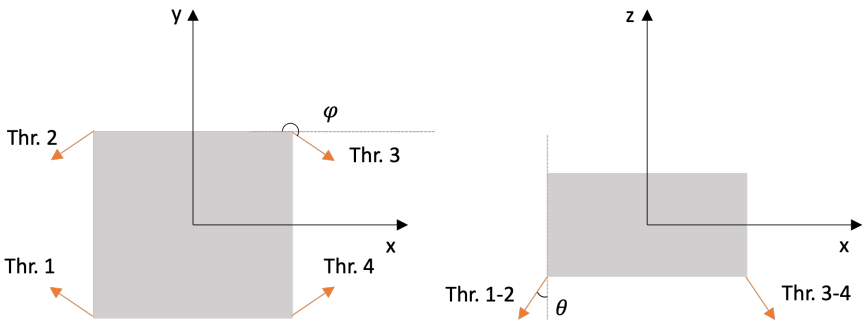



Figure 5.6: 4 Thr. 2 DOF Configuration Angles Reference Frame

The optimization procedure obtained the results presented in table 5.2.2.1, the corresponding angles are listed from 5.3 to 5.10. The margin increment is graphed in figure 5.7.

Table 5.4: 4 Thr. 2 DOF Torque Margin Results

| | ME Thrust | BOL | EOL |
|---|---|---|---|
| Option A | 7 kN | - 505.03 Nm | - 493.8 Nm |
| Option B | 5 kN | 13.129 Nm | 14.709 Nm |
| Option C | 4 kN | 217.99 Nm | 227.55 Nm |
| Option D | 3 kN | 410.86 Nm | 427.39 Nm |

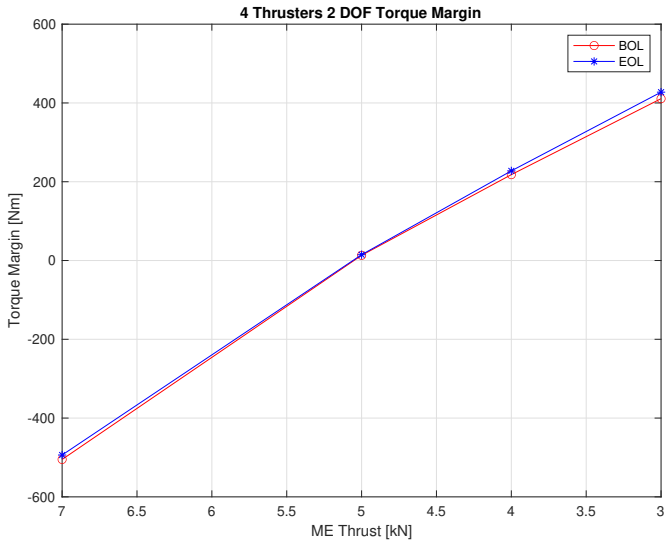



Figure 5.7: 4 Thr. 2 DOF Torque Margin

Option A BOL:

$$\begin{aligned} \theta &= 12^\circ \\ \psi &= 175^\circ \end{aligned} \tag{5.3}$$

Option A EOL:

$$\begin{aligned} \theta &= 44^\circ \\ \psi &= -157^\circ \end{aligned} \tag{5.4}$$

Option B BOL:

$$\begin{aligned} \theta &= 14^\circ \\ \psi &= 180^\circ \end{aligned} \tag{5.5}$$

Option B EOL:

$$\begin{aligned} \theta &= 35^\circ \\ \psi &= -132^\circ \end{aligned} \tag{5.6}$$

Option C BOL:

$$\begin{aligned} \theta &= 16^\circ \\ \psi &= -90^\circ \end{aligned} \tag{5.7}$$

Option C EOL:

$$\begin{aligned} \theta &= 35^\circ \\ \psi &= -122^\circ \end{aligned} \tag{5.8}$$

Option D BOL:

$$\begin{aligned} \theta &= 19^\circ \\ \psi &= -54^\circ \end{aligned} \tag{5.9}$$

Option D EOL:

$$\begin{aligned} \theta &= 38^\circ \\ \psi &= -114^\circ \end{aligned} \tag{5.10}$$

#### 5.2.2.2 4 Thrusters 8 DOF

The second configuration analyzed considers for the analysis 8 degrees freedom, 2 for each of the thrusters used.

The optimization procedure obtained the results presented in table 5.5, the corresponding angles are listed from 5.11 to 5.18. The margin increment is graphed in figure 5.8. There is a slight improvement in the torque margin obtained. It's interesting to see how the results present the same symmetry that was shown in the previous analysis. The results are presented with decimals, as obtained from the analysis, it is important to underline that this precision can not be obtained in the real mounting of the thrusters due to mechanical limitations. The reference frame for the angles is the same presented in figure 5.6, but for this analysis each thruster has its own 2 angles to define the orientation.

Table 5.5: 4 Thr. 8 DOF Torque Margin Results

| | ME Thrust | BOL | EOL |
|---|---|---|---|
| Option A | 7 kN | -504.97 Nm | -488.77 Nm |
| Option B | 5 kN | 15.14 Nm | 15.49 Nm |
| Option C | 4 kN | 218.66 Nm | 229.08 Nm |
| Option D | 3 kN | 412.07 Nm | 427.38 Nm |

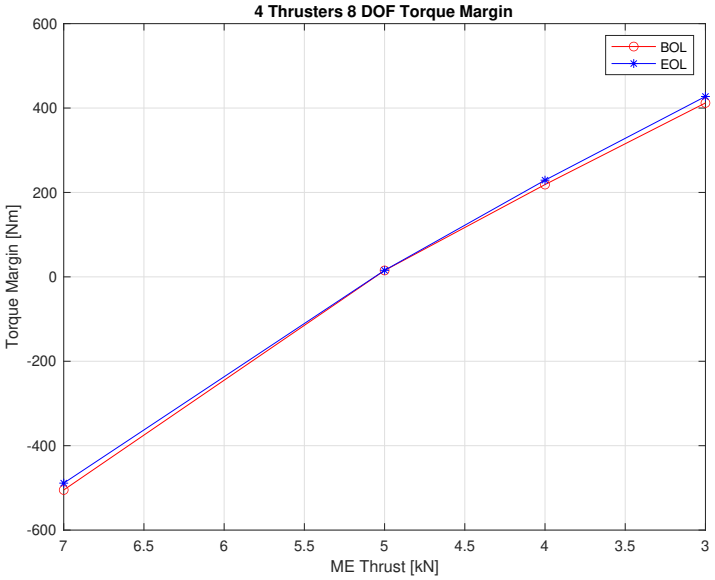



Figure 5.8: 4 Thr. 8 DOF Torque Margin

Option A BOL:

$$\begin{aligned} &Thruster\ 1 : \theta = 12,21°; \psi = 355.89° \\ &Thruster\ 2 : \theta = 12.17°; \psi = 4.12° \\ &Thruster\ 3 : \theta = 12,19°; \psi = 175.87° \\ &Thruster\ 4 : \theta = 12,22°; \psi = 184,15° \end{aligned} \tag{5.11}$$

Option B BOL:

$$\begin{aligned} &Thruster\ 1 : \theta = 14.06°; \psi = 0.65° \\ &Thruster\ 2 : \theta = 14.13°; \psi = 359.35° \\ &Thruster\ 3 : \theta = 14.17°; \psi = 180.55° \\ &Thruster\ 4 : \theta = 14.11°; \psi = 179.37° \end{aligned} \tag{5.12}$$

Option C BOL:

$$\begin{aligned} &Thruster\ 1 : \theta = 16.27°; \psi = 89.18° \\ &Thruster\ 2 : \theta = 16.29°; \psi = 270.79° \\ &Thruster\ 3 : \theta = 16.18°; \psi = 269.17° \\ &Thruster\ 4 : \theta = 16.23°; \psi = 90.84° \end{aligned} \tag{5.13}$$

Option D BOL:

$$\begin{aligned} &Thruster\ 1 : \theta = 18.81°; \psi = 127.57° \\ &Thruster\ 2 : \theta = 18.82°; \psi = 232.58° \\ &Thruster\ 3 : \theta = 18.79°; \psi = 307.44° \\ &Thruster\ 4 : \theta = 18.81°; \psi = 55.55° \end{aligned} \tag{5.14}$$

Option A EOL:

$$\begin{aligned} &Thruster\ 1 : \theta = 43,72°; \psi = 23.18° \\ &Thruster\ 2 : \theta = 43,67°; \psi = 337,11° \\ &Thruster\ 3 : \theta = 43,71°; \psi = 202,8° \\ &Thruster\ 4 : \theta = 43,74°; \psi = 516,8° \end{aligned} \tag{5.15}$$

Option B EOL:

$$\begin{aligned} &Thruster\ 1 : \theta = 35.48°; \psi = 47.81° \\ &Thruster\ 2 : \theta = 35.46°; \psi = 312.19° \\ &Thruster\ 3 : \theta = 35.32°; \psi = 227,78° \\ &Thruster\ 4 : \theta = 35.32°; \psi = 132.21° \end{aligned} \tag{5.16}$$

Option C EOL:

$$\begin{aligned} &Thruster\ 1 : \theta = 36.33^{\circ}; \psi = 57.32^{\circ} \\ &Thruster\ 2 : \theta = 36.31^{\circ}; \psi = 302.72^{\circ} \\ &Thruster\ 3 : \theta = 35.93^{\circ}; \psi = 237,32^{\circ} \\ &Thruster\ 4 : \theta = 35.95^{\circ}; \psi = 482.71^{\circ} \end{aligned} \tag{5.17}$$

Option D EOL:

$$\begin{aligned} &Thruster\ 1 : \theta = 39.21^{\circ}; \psi = 66.32^{\circ} \\ &Thruster\ 2 : \theta = 38.78^{\circ}; \psi = 294.12^{\circ} \\ &Thruster\ 3 : \theta = 38.11^{\circ}; \psi = 245.79^{\circ} \\ &Thruster\ 4 : \theta = 38.32^{\circ}; \psi = 114.12^{\circ} \end{aligned} \tag{5.18}$$

#### 5.2.2.3 8 Thrusters 2 DOF

The third configuration analyzed involves the use of 8 thrusters. In order to be able to do the analysis, it is assumed that the four redundant thrusters are used together with the four nominal thrusters. To ensure the failure tolerance, in this case it would be necessary to mount eight additional thrusters. The system has 2 degrees of freedom, the $\psi$ angle and $\theta$ angle, as show in figure 5.9. The analysis include two cases, since the use of all eight thrusters let to a positive margin already with the use of a complete thrust of 7 kN for each of the 3 ME. The option B is analyzed to achieve a margin of more than 400 Nm.

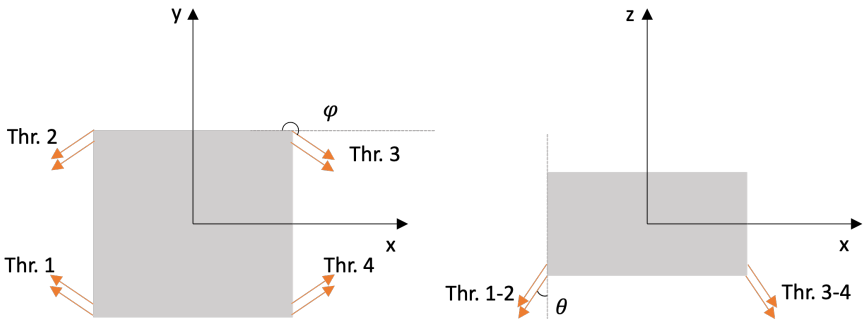



Figure 5.9: 8 Thr. 2 DOF Configuration Angles Reference Frame

The optimization procedure obtained the results presented in table 5.6, the corresponding angles are listed from 5.19 to 5.22. The margin increment is graphed in figure 5.10.

Table 5.6: 8 Thr. 2 DOF Torque Margin Results

| | ME thrust | BOL | EOL |
|---|---|---|---|
| Option A | 7 kN | 391.69 Nm | 777.92 Nm |
| Option B | 6 kN | 627.91 Nm | 900.78 Nm |

Option A BOL:

$$\begin{aligned} \theta &= 12^\circ \\ \psi &= 143^\circ \end{aligned} \tag{5.19}$$

Option B BOL:

$$\begin{aligned} \theta &= 16^\circ \\ \psi &= 150^\circ \end{aligned} \tag{5.20}$$

Option A EOL:

$$\begin{aligned} \theta &= 41^\circ \\ \psi &= 196^\circ \end{aligned} \tag{5.21}$$

Option B EOL:

$$\begin{aligned} \theta &= 42^\circ \\ \psi &= 194^\circ \end{aligned} \tag{5.22}$$

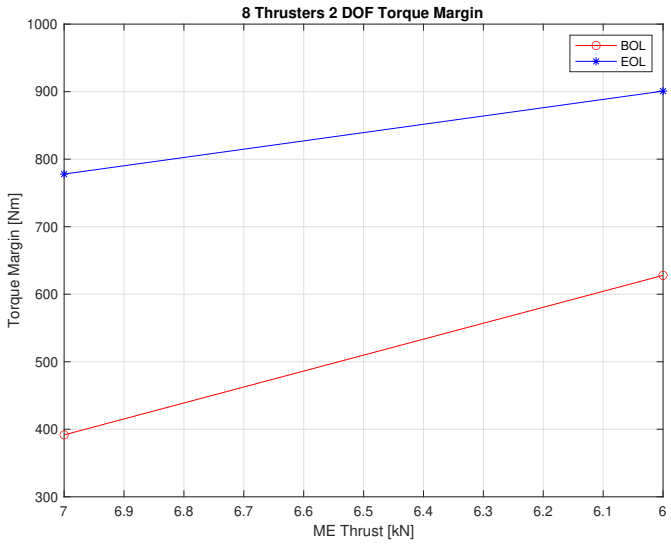



Figure 5.10: 8 Thr. 2 DOF Torque Margin

#### 5.2.2.4 8 Thrusters 3 DOF

The fourth configuration analyzed involves the use of the 4 main thrusters and the 4 redundant thrusters (as the previous), with 3 degrees of freedom, a $\psi_1$ angle for the 4 main thrusters, a $\psi_2$ angle for the redundant thrusters and the $\theta$ angle for both. The reference frame is shown in figure 5.11. The analysis include two cases, since the use of all eight thrusters let to a positive margin already with the use of a complete thrust of 7 kN for each of the 3 ME. The option B is analyzed to achieve a margin of more than 400 Nm.

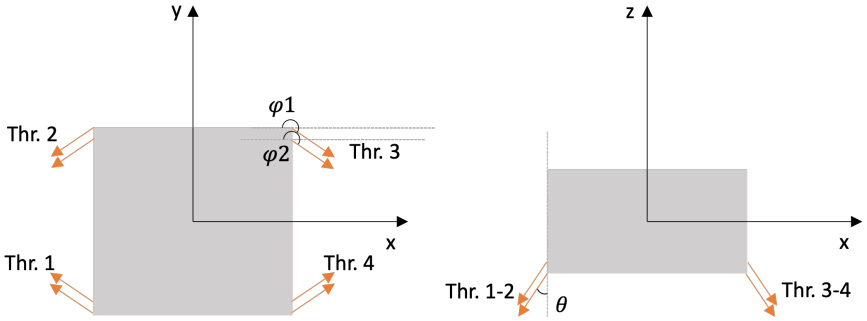



Figure 5.11: 8 Thr. 3 DOF Configuration Angles Reference Frame

The optimization procedure obtained the results presented in table 5.7, the corresponding angles are listed from 5.23 to 5.26. The margin increment is graphed in figure 5.12. The addition of a degree of freedom can at most improve or maintain unchanged the result obtained with the previous analysis. However, the result obtained for option B in BOL configuration show a decrease in margin while for option A the result is almost the same as the one obtained from previous analysis. Therefore it is clear that the optimization with ISOLE must have fall in a local minimum and was not able to find a better solution. While solutions for EOL configurations show an improvement of 4.01% for option A and 16.5% for option B.

Table 5.7: 8 Thr. 3 DOF Torque Margin Results

| | | BOL | EOL |
|---|---|---|---|
| Option A | 7 kN | 392.87 Nm | 809.15 Nm |
| Option B | 6 kN | 624.03 Nm | 1049.4 Nm |

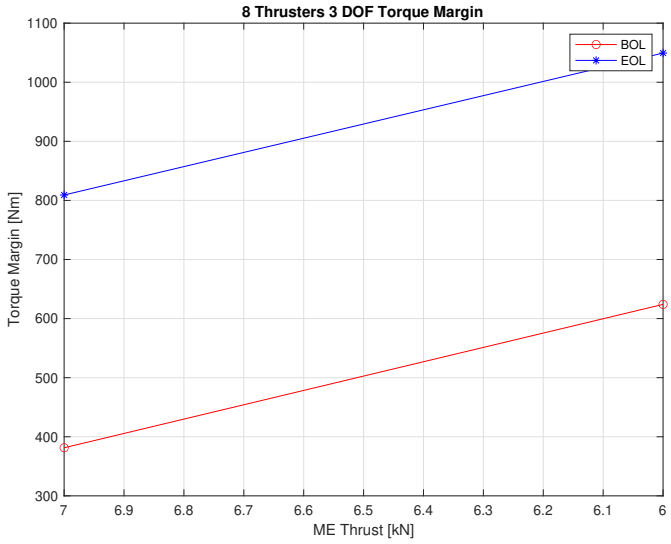



Figure 5.12: 8 Thr. 3 DOF Torque Margin

Option A BOL:

$$\begin{aligned} Thrusters\ 1,2,3,4 &: \psi_1 = 140°, \theta = 12° \\ Thrusters\ 5,6,7,8 &: \psi_2 = 155°, \theta = 12° \end{aligned} \tag{5.23}$$

Option B BOL:

$$\begin{aligned} Thrusters\ 1,2,3,4 &: \psi_1 = 130°, \theta = 16° \\ Thrusters\ 5,6,7,8 &: \psi_2 = 160°, \theta = 16° \end{aligned} \tag{5.24}$$

Option A EOL:

$$\begin{aligned} Thrusters\ 1,2,3,4 &: \psi_1 = 185°, \theta = 38° \\ Thrusters\ 5,6,7,8 &: \psi_2 = 191°, \theta = 38° \end{aligned} \tag{5.25}$$

Option B EOL:

$$\begin{aligned} Thrusters\ 1,2,3,4 &: \psi_1 = 230°, \theta = 45° \\ Thrusters\ 5,6,7,8 &: \psi_2 = 180°, \theta = 45° \end{aligned} \tag{5.26}$$

Other analyses were carried out by the company with the aim of obtaining a control margin of more than 500 Nm. The margin sought was obtained for the EOL and BOL configurations with a configuration with 10+10 thrusters (main + redundant). The method used was different, however, because it was decided to

work only in 2 dimensions, never using more than 2 DOF. While maintaining the thrust levels of the nominal MEs, this work has achieved an acceptable result with a solution that has 8 main thrusters and eventually 8 redundant. The increase in degrees of freedom has shown an increase in the margin in 2 configurations, almost no change in margin for 1 configuration and an unexpected decrease in another configuration. It is important to note that the increase in degrees of freedom together with a meta-heuristic analysis of the problem does not guarantee that it has been found the optimal solution.

However the tool, although still in an early stage of the development, has been proven able to manage real problems with a certain level of complexity.

# Chapter 6

# Conclusions

Numerous application problems encountered in science can be modeled as global optimization problems. Most real-world optimizations are highly nonlinear and multimodal, under various complex constraints. It is therefore important to develop and study numerical methods and tools that are able to identify the overall optimum of the function to be analyzed. This thesis is focused on the study of metaheuristic optimization, in particular in the usage of the island model paradigma coupled with the migration operator. The work I have done for this master's thesis has been carried out within the company Thales Alenia Space and addresses the need of the Guidance Navigation Control group to have an optimization tool available to be used for the study in different project phases. The objectives of this work were to:

- Review the state of the art of hybrid optimization systems that implement evolutionary algorithms and the evolutionary algorithms used with particular regard to the island model.
- Further develop a hybrid global optimization tool, implemented in MATLAB® script language, that uses evolutionary algorithms for a single-objective problem. In case of multi-objective functions and constraints, they are included in the single objective function using weight. The tool have to be modular, easy to use and without dependencies from the MATLAB® optimization toolbox.
- Include in the tool:
  - a global mutation mechanism;
  - the possibility to choose different cloning topology;
  - the availability of the Particle Swarm Optimization algorithm;

  - The capability to use also an approximate function together with the user defined one.

- Apply the developed tool for the ESA European Large Logistic Lander (EL3) RCS thrusters' orientation optimization problem. Studying various configurations with different degrees of freedom.

After almost one year work the development of the tool has been advanced: it can be described as a multi-population algorithm that exploits three evolutionary algorithms DE, ES and PSO, running in parallel. The populations exchange information between the islands during the iterations, in order to achieve a greater efficiency and try to achieve a smaller computational cost for the analysis. The new features implemented are:

- The modularity of input and output. These are now editable outside the MATLAB® script. The user that can decide how to perform the analysis, including the number of islands, the algorithm type and the number of individuals for each one, which results will be printout display.

- A PSO algorithm.

- The Mass Mutation mechanism, which can act on the single island or on the whole archipelago. It is used to re-initialise the 98% of the population when an aggregation degree is reached for the whole population, it has been introduced to avoid to stuck on local minimum.

- The possibility to choose between 3 cloning mechanisms, always carried out in synchronous mode.

- The skeleton of an algorithm that combines PSO with neural networks (ANN). And the possibility to create a global approximation function using ANNs that are trained with the information obtained from the other algorithms during the analysis. However, the results obtained so far are only preliminary and the thesis focused on other topics.

The tool has been tested on a series of benchmark functions and on ESA European Large Logistic Lander (EL3) RCS thrusters' orientation. A comparison between the use of individual algorithms and a combined use of them has been performed to show is effectiveness. The benchmark functions where used in order to compare the tool results on well-known problems. The results are:

- No one of the simple algorithms works well on all the problems.

- The Mass Mutation mechanism increase the probability of a convergence to the global optimum.

- On average with hybrid algorithms is possible to obtain a better efficiency without knowing which is the best optimiser for the problem. However a tuning improvement must be carried out for the algorithm ES that seems to slow the others.

- Eventually the optimum will be achieved more slowly, with a small increase in computational cost, but in some cases the computational cost is even decreased. It is important to remember that the number of function evaluations available for each run of the tool is constant. So the computational cost is increased, but not proportionally to the rise of the population number.

The study conducted on the orientation of EL3 RCS thrusters required the development of an approximate solution method. That's because in most real applications of EAs, the computational complexity due to fitness function evaluation is a prohibiting factor. It was then possible to apply the developed tool into the studying of 2 configurations, 4 and 8 thrusters, and different combinations of degrees of freedom for the orientation, with the aim of obtaining an acceptable margin in the lander control. The tool, working with 8 degrees of freedom, for the configuration with 4 thrusters, was able to find a correlation between the DOF. The torque margin results are very similar to those of the same problem dealt with 2 degrees of freedom. While analyzing the configuration with 8 thrusters the increase in the degrees of freedom led to a consistent improvement of the solution in only 2 cases out of 4. This study case was fundamental to understand what the potential of the tool can be, it has been proven able to manage real problems with a certain level of complexity and it may be the subject of further improvements.

Evolutionary Algorithms are usually quite versatile, the use of this tool on various problems and obtaining good results, even on a real problem like EL3, is a further proof. To improve tool's results on high non-linear problems, future work can focus on algorithms' parameters tuning and the analysis of different algorithms and topologies which can be added easily thanks to the modularity of the tool.

# Bibliography


[1] N. Hadi, K. S. Tey, Y. I. B. I. Mohd, N. Morteza, B. S. Rosli, and G. Adbullah, "Hybrid Metaheuristics for QoS-Aware Service Composition: A Systematic Mapping Study," *IEEE Access*, 2021.

[2] F. Biscani and D. Izzo, "A parallel global multiobjective framework for optimization: pagmo," *Journal of Open Source Software*, vol. 5, no. 53, p. 2338, 2020.

[3] B. Francesco and I. Dario, "pagmo." `https://esa.github.io/pagmo2/`.

[4] I. M. R. Sentinella, "Development Of New Procedures And Hybrid Algorithms For Space Trajectories Optimisation," *Doctoral Thesis, Politecnico di Torino*, 2008.

[5] M. S., Cahon ans N. and T. E. G., "ParadisEO: A Framework for the Reusable Design of Parallel and Distributed Metaheuristics," *Journal of Heuristics*, 2004.

[6] C.-P. Erick, "Efficient and Accurate Parallel Genetic Algorithms," *Springer Science+Business Media*, pp. 1–13,135–145, 2001.

[7] R. M., I. D., and B. F., "On the impact of the migration topology on the Island Model," *Parallel Computing*, 2010.

[8] F. d. V. Francisco, P. José Ignacio Hidalgo, and L. Juan, "Parallel Architectures and Bioinspired Algorithms," *Springer*, pp. 101–171, 2012.

[9] D. L.C.W., "Global optima without convexity," *Technical report, Numerical Optimization Centre, Hatfield Polytechnic*, 1978.

[10] D. Marco, B. Mauro, and S. Thomas, "Ant Colony Optimization," *IEEE COMPUTATIONAL INTELLIGENCE MAGAZINE*, 2006.

[11] K. J. and E. R., "Particle Swarm Optimization," *Neural Networks, IEEE International Conference*, 1995.

[12] E. Soren, K. Pascal, and G. Lino, "A Generic Particle Swarm Optimization Matlab Function," *American Control Conference*, 2012.

[13] S. Yuhui and E. Russell, "A Modified Particle Swarm Optimizer," *Department of Electrical Engineering, Indiana University Purdue University Indianapolis*, 1998.

[14] M. Ai-Qin, C. De-Xin, and W. Xiao-Hua, "A Modified Particle Swarm Optimization Algorithm," *Natural Science, Vol.1, No.2, 151-155*, 2009.

[15] T. Ioan Cristian, "The particle swarm optimization algorithm: convergence analysis and parameter selection," *Information Processing Letters*, 2002.

[16] M. Mitchell, "An Introduction to Genetic Algorithms," *A Bradford Book The MIT Press*, 1999.

[17] F. Xiaopeng, "Engineering design using genetic algorithms," *Iowa State University Capstones, Theses and Dissertations*, 2007.

[18] G. David E. and D. Kalyanmoy, "A Comparative Analysis of Selection Schemes Used in Genetic Algorithms," *University of Illinois at Urbana-Champaign*, 1995.

[19] L. Josè Allen, G. Nuno, P. Henrique, and R. Agostinho, "Fitness Function Design for Genetic Algorithms in Cost Evaluation Based Problems," *Conference Paper*, 1996.

[20] R. K. Bhattacharjya, "Introduction to Differential Evolution," *Indian Institute of Technology Guwahtai*, 2005.

[21] S. Rainer, "On the Usage of Differential Evolution for Function Optimization," *Conference: Fuzzy Information Processing Society*, 1996.

[22] S. Rainer, "Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces," *International Computer Science Institute*, 1995.

[23] H. Nikolaus, A. Dirk V., and A. Anne, "Evolution Strategies," *Publication*, 2015.

[24] B. Christian and R. Andrea, "Hybrid Metaheuristics: An Introduction," *Studies in Computational Intelligence*, 2008.

[25] R. Gunther R., "A Unified View on Hybrid Metaheuristics," *Institute of Computer Graphics and Algorithms Vienna University of Technology, Vienna, Austria*, 2006.

[26] A. Fatima Zahra and A. Said, "A hybrid neural network model based on improved PSO and SA for bankruptcy prediction," *National School for Computer Science and Systems analysis, Marocco*, 2019.

[27] D. Gaurav and K. Amandeep, "A Hybrid Algorithm Based on Particle Swarm and Spotted Hyena Optimizer for Global Optimization," *Springer Nature Singapore*, 2019.

[28] B. T.C., "The distributed genetic algorithm revisited," *Proceedings of the Sixth International Conference on Genetic Algorithms*, p. pp. 114–121, 1995.

[29] W. Darrell, R. Soraya, and H. Robert B, "The Island Model Genetic Algorithm: On Separability, Population Size and Convergence," *Department of Computer Science*, 1998.

[30] J. Vesterstrom and R. Thomsen, "A Comparative Study of Differential Evolution, Particle Swarm Optimization, and Evolutionary Algorithms on Numerical Benchmark Problems," *Proceedings of the IEEE International Conference on Evolutionary Computation*, vol. 2, pp. 1980–1987, 2004.

[31] E. S. Agency. `https://www.esa.int/Science_Exploration/Human_and_Robotic_Exploration/Exploration/European_Large_Logistics_Lander`.